Programs Offered

Post Graduate Programmes (PG)

- Master of Business Administration
- Master of Computer Applications
- Master of Commerce (Financial Management / Financial Technology)
- Master of Arts (Journalism and Mass Communication)
- Master of Arts (Economics)
- Master of Arts (Public Policy and Governance)
- Master of Social Work
- Master of Arts (English)
- Master of Science (Information Technology) (ODL)
- Master of Science (Environmental Science) (ODL)

Diploma Programmes

- Post Graduate Diploma (Management)
- Post Graduate Diploma (Logistics)
- Post Graduate Diploma (Machine Learning and Artificial Intelligence)
- Post Graduate Diploma (Data Science)

Undergraduate Programmes (UG)

- Bachelor of Business Administration
- Bachelor of Computer Applications
- Bachelor of Commerce
- Bachelor of Arts (Journalism and Mass Communication)
- Bachelor of Arts (General / Political Science / Economics / English / Sociology)
- Bachelor of Social Work
- Bachelor of Science (Information Technology) (ODL)

AMITY UNIVERSITY ONLINE
Amity Helpline: (Toll free) 18001023434
For Student Support: +91 - 8826334455
Support Email id: studentsupport@amityonline.com | https://amityonline.com

Product code
81030112000586

Human Computer Interaction

AMITY

Human Computer Interaction

AMITY UNIVERSITY ONLINE

(C) Amity University Online

Human Computer Interaction

(c) Amity University Online

AMITY UNIVERSITY ONLINE

© Amity University Press

All Rights Reserved

No parts of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior permission of the publisher.

SLM & Learning Resources Committee

Chairman : Prof. Abhinash Kumar

Members : Dr. Ranjit Varma
Dr. Maitree
Dr. Divya Bansal
Dr. Arun Som
Dr. Sunil Kumar
Dr. Reema Sharma
Dr. Winnie Sharma

Member Secretary : Ms. Rita Naskar

Published by Amity University Press for exclusive use of Amity Directorate of Distance and Online Education, Amity University, Noida-201313

(c) Amity University Online

Contents

Page No.

Module - I: Introduction to Interaction Design 01

1.1 Interaction between Humans and Computer Systems

1.1.1 Introduction to HCI

1.1.2 Input/Output Channels: Sight, Hearing, Touch, Visual and Angle

1.1.3 Memory: Sensory, Short Memory and Long Term Memory

1.1.4 Reasoning and Problem Solving

1.2 Human Computer Interaction

1.2.1 Devices: Part 1

1.2.2 Devices: Part 2

1.2.3 Devices: Part 3

1.2.4 Memory

1.2.5 Processing and Networks

1.2.6 Understanding Users: Personas and Scenarios

1.2.7 Models

1.2.8 Ergonomics

1.2.9 Styles

1.2.10 Elements

1.2.11 Interactivity

1.2.12 Paradigms

Module - II: Design and Software Process 91

2.1 Interaction Design Basics

2.1.1 Design

2.1.2 Process of Design

2.1.3 Screen Design and Layout

2.1.4 Universal Design Principles

2.1.5 Iteration and Prototyping

2.2 HCI in Software Process

2.2.1 The Software Life Cycle

2.2.2 Usability Engineering

2.2.3 Iterative Design and Prototyping

2.3 Design Rules

2.3.1 Standards

2.3.2 Guidelines

2.3.3 Golden Rules and Heuristics

2.3.4 HCI Patterns

2.3.5 Task Centered System: Design and User-Centred Design Prototyping

(c) Amity University Online

Module - III: Evaluation **142**

3.1 Evaluation in Human Computer Interaction (HCI)

3.1.1 Goals of Evaluation

3.1.2 Evaluation through Expert

3.1.3 Evaluation through Users: Styles of Evaluation and Experimental Evaluation Design

3.1.4 Evaluation through Users: Observational Techniques and Query Techniques

3.1.5 Choosing an Evaluation Method

3.2 Design principles and Visuals

3.2.1 Color Theory, Typography, and Imagery

3.2.2 Iconography

3.2.3 Designing for Different Screens

3.2.4 Responsive Web Design

Module -IV: Models and Theories **177**

4.1 Interaction Design in Human Computer Interaction (HCI)

4.1.1 Cognitive Model

4.1.2 Communication and Collaboration Model

4.1.3 Task Analysis and Decomposition

4.1.4 Knowledge Based Analysis

4.2 Human Computer Dialogues

4.2.1 Dialog Notation

4.2.2 Dialog Semantics

4.2.3 Standard Formalisms

4.2.4 Interaction Models

4.2.5 Hypertext and Multimedia

4.2.6 World Wide Web

4.3 Advanced HCI Technologies

4.3.1 Augmented and Virtual Reality

4.3.2 Natural Language Processing

4.3.3 Biometric Interfaces

Module - V: Research Frameworks **216**

5.1 Overview of Research Framework

5.1.1 Speech Interfaces

5.1.2 Information Visualisation

5.1.3 Ubiquitous Computing

5.1.4 Case Studies

5.2 Ethical Considerations and Future Trends

5.2.1 Ethical Issues in HCI

5.2.2 Emerging Trends

(c) Amity University Online

Module -I: Introduction to Interaction Design

Notes

Learning Objectives

At the end of this module, you will be able to:

- Understand basics of HCI
- Define Input/Output Channels
- Understand users: Personas and Scenarios
- Define Prototyping and Iterative Design
- Analyse Styles and Elements

Introduction

Interaction Design (IxD) holds significant importance in the realm of Human-Computer Interaction (HCI), actively contributing to the development of user interfaces and experiences characterised by intuitiveness, effectiveness and user satisfaction. HCI, as a multidisciplinary field, encompasses the design and utilisation of computer technologies, with a primary focus on the intricate interplay between humans and computers.

The user can communicate with the system in a variety of ways. Batch input is one extreme, when the user feeds the computer all the data at once and walks away to finish the work. Although there is contact between the user and the computer using this approach, many tasks are not effectively supported. Virtual reality applications and other highly interactive input paradigms, such direct manipulation, are at the other extreme. In this case, the user is always giving instructions and getting feedback. We are thinking about these kinds of interactive systems.

Let's now examine the interaction, or communication, that occurs between the user and the system. In addition, let's examine several interaction models that help us recognise and assess interaction components as well as the organisational, social and physical factors that set the scene for them.

1.1 Interaction between Humans and Computer Systems

The challenge of human-computer interaction yields great rewards. Ensuring the effectiveness, efficiency, user-friendliness and enjoyment of interactive computer systems is crucial for people and society to fully reap the rewards of computation-based technologies. The complexity arises from the intricate interplay of human, machine, algorithmic, task, social, aesthetic and economic restrictions and associated trade-offs. The reward can be found in the creation of virtual worlds for entertainment and social interaction, medical devices that enable a surgical team to conceptualise, locate and monitor a complex neurosurgical operation, responsive and efficient government services, ranging from online licence renewal to the analysis of parliamentary testimony, or smart phones that can recognise their location and understand limited speech. Designers of interaction construct and integrate interaction in both virtual and real-world environments.

Although it is a diverse specialisation in many domains, human-computer interaction is inextricably linked to computer science as a subfield. The majority of interactive computer systemis used for human purposes and communicate with people in human environments. Although the idea that computer science is only the study of algorithms is a good way to introduce some rigour, it can also cause one to overlook important limitations that are

© Amity University Online

Notes

necessary for the creation of well-designed interactive computer systems. Engineering students are often taught that narrow design optimisation that fails to adequately consider contextual considerations is a primary source of failure. Due to their complexity, human users and their environments constitute significant aspects of the design problem that cannot be ignored. In actuality, user interaction occupies the majority of the program code in the majority of interactive systems. Inadequate consideration for people and task context not only results in subpar user interfaces but also poses a risk to entire systems.

The challenge lies in understanding and designing the human and contextual aspects of a system with the same level of rigour as other system components. This means moving beyond generalisations like "know the user," which are accurate but do not provide a way to accomplish or assess the knowledge. Though challenging, this cannot be avoided and can actually advance. Applied psychology, industrial engineering, ergonomics, human factors and man-machine systems are some of the new academic disciplines that have emerged as a result of the necessity to consider human elements of technical systems over time. The newest of them is human-computer interaction, which is more complicated in some respects due to the variety of user groups and applications, the penetration of social and cognitive boundaries and the focus on interaction. A series of recommendations about the structure of a human-computer interaction field are derived from the experiences of other human-technical disciplines.

First, the activity takes place in design. Even while "usability analysis" is vital, it cannot form the main foundation of an effective human-computer interaction science. Notably, usability analysis is not creative, occurs too late and has insufficient degrees of freedom. Design is fueled by an awareness of limitations, a keen sense of the design space and an in-depth grasp of the materials of the design—that is, the machine, the task and the user. The Xerox Star and the Apple Lisa/Macintosh are two examples of iconic designs in human-computer interaction. While usability analysis played a significant role in their creation, user interface designers, who were in charge of both design and implementation, used generative principles to create these designs instead of usability analysis.

Second, while "user-centered design" receives a lot of attention, "task-centered design" is what we should be focusing on. Allocating responsibilities between humans and robots and creating their interaction both depend on an understanding of the goal and context of the system. The only way to reconcile the technological and human problems is to determine what a human-machine system should be able to perform and what limitations should be placed on it. The requirement for task-centered design highlights the need for task analysis techniques as a key component of system design.

Third, the basic curriculum for human-computer interaction should be designed to teach both implementation and analytical techniques within the same subject. Professionals in the field who are limited to evaluating just and not designing or building are at a disadvantage. A handicap exists for builders who are unable to think critically about the systems they create, or who are unaware of how others absorb information and how their designs are used in social contexts. Naturally, experts in various aspects of human-computer interaction will exist, but a shared foundation is necessary for the discipline to succeed.

Ultimately, a discipline is a collection of techniques for carrying out a certain task. A field needs to produce outcomes that others besides the original creators can learn from and apply to accomplish goals. A field organically develops over time from a collection of point results to a collection of techniques to a collection of facts, broad abstractions, procedures and ideas. Compaction of knowledge—that is, breaking down the results into methods and theories—is actually necessary for a discipline to be cumulative; otherwise,

(c) Amity University Online

Notes

it becomes fad-driven and a collection of an almost unteachably enormous set of weak results.

The most practical approaches and theories are generative theories, which calculate some insightful attribute that limits the design space of a system based on task analysis. Task analysis, approximation and calculation in a formula. For instance, we can anticipate that the illusion of animation would start to falter if a graphics system is unable to update the display more quickly than ten times per second. The architectural ramifications of this limitation applied backwards include how to ensure the required display rate in the presence of fluctuating computing load. One may design against it.

1.1.1 Introduction to HCI

When designing computers and similar devices, it is important to keep in mind that users with specific activities in mind will want to use them in a way that is consistent with their day-to-day work. In order to accomplish this, system designers must be able to conceptualise jobs for end users and convert that understanding into an executable system. However, there is an issue with attempting to impart the idea of creating computers for human use. Every designer is a person and they are almost certainly also consumers. So designing with the user in mind, is that not intuitive?

Given that everyone is aware of what makes a decent interface, why is it still necessary to teach it? Because of this, designers typically learn about human–computer interface (HCI) later in their education, if at all. The scenario we began with demonstrates that this is a false perspective; designing reliable, consistent systems that can handle a wide range of user negligence is neither straightforward nor simple. The design of the interface should be created in tandem with the rest of the system; it cannot just be patched in at the last minute. It should not only put on a "beautiful face," but also encourage individuals to complete the chores they genuinely desire to accomplish and overlook their careless errors. As a result, we must think about how HCI fits into the design process.

Creating systems that are easy to use is becoming more and more of a legal concern rather than just a question of marketing or even benevolence towards the end user. Employers are required by national health and safety requirements to provide their workers with computer systems that are not only safe but also functional. For instance, when designing, choosing, commissioning, or altering software, employers must make sure they fulfil the following requirements outlined in EC Directive 90/270/EEC, which has been integrated into the laws of member countries:

- that it satisfies the requirements of the task;
- that it is user-friendly and, when necessary, adjusts to the user's skill level;
- that it offers performance feedback;
- that it presents information in a manner and at a speed that is customised for the user; and
- that it complies with the "principles of software ergonomics."

Employers and designers cannot afford to disregard the user any more.

HCI

Although it has only become widely used since the early 1980s, the phrase "human-computer interaction" has origins in earlier, more recognised fields of study. Beginning in factories at the turn of the 20th century, systematic research on human performance was conducted with a focus on physical labour. The Second World War served as a catalyst for

© Amity University Online

Notes

research into human-machine interaction as both sides competed to develop increasingly potent weaponry. As a result, researchers became increasingly interested in the field and the Ergonomics Research Society was established in 1949.

Ergonomists' primary focus has always been on how the physical attributes of devices and systems impact user performance. Human Factors takes these problems into account along with other cognitive problems. Ergonomics is the standard term in the United Kingdom, while Human Factors is used in English-speaking portions of North America. The names are sometimes used interchangeably. User performance in the context of any system—computer, mechanical, or manual—is the focus of these two fields. With the growing use of computers, a growing number of researchers focused on the interaction between humans and computers, exploring the theoretical, psychological and physical components of this process. The term "human-machine interaction" was initially used for this research, but it was later changed to "human-computer interaction" to reflect the specific interest in computers and the makeup of the user base!

Information science and technology research has also had an impact on the advancement of HCI. Once more, the former is an antiquated field that deals with information administration and manipulation within organisations; it predates the invention of technology. The organisation and work environment have been significantly impacted by the arrival of technology, which has had a dramatic effect on how information may be stored, accessed and utilised. Systems analysis has historically focused on how technology affects the workplace and how to adapt it to the demands and limitations of the position. HCI is likewise concerned about these problems.

As we shall see, human-computer interaction (HCI) crosses many fields, but computer science and systems design are where it has to be acknowledged as a primary focus. While it can be a speciality for all other disciplines, it is an integral aspect of systems design; for all other disciplines, it is merely a vital input. According to this viewpoint, human-computer interaction (HCI) entails the creation, testing and assessment of interactive systems within the framework of user tasks and jobs.

But when we discuss human–computer interaction, we don't always picture one person using a desktop computer. A single user, a team of users working together, or a series of users in an organisation, each handling a different aspect of the task or process, can all be referred to as a "user." Whoever is attempting to complete a task with technology is the user. Any type of technology that can be classified as a computer includes embedded systems, process control systems, large-scale computer systems and standard desktop computers. Humans and other non-computerised components may be a part of the system. Any direct or indirect communication between a user and a computer is referred to as an interaction. Direct engagement entails a conversation with control and feedback as the task is being completed. Intelligent sensors managing the surroundings or batch processing are examples of indirect interaction. The user's interaction with the computer to achieve a goal is what matters most.

Definition and Importance

UX design is a word that has become extremely popular worldwide. What does it signify, though, really? Even though many businesses view UX design as crucial to product development, the phrase itself sometimes seems like simply another catchphrase bandied about at startup gatherings. User Experience, or UX Design, is the term used to describe all parts of a product as they are experienced by users. This includes landing pages, websites, the product itself, communities, services and more.

(c) Amity University Online

Notes

However, being aware of what UX stands for does not equate to truly comprehending the intricacies that make it function. To truly master UX Design, one must first have a thorough understanding of both what it is and—perhaps more importantly—what it is not. So continue reading to learn about the real duties of a UX designer, best and worst practices and how to begin your UX journey.

Any interaction a user has with a product or service is referred to as user experience (UX) for short. For instance, the way the product appears, how its components affect the user, how it feels and how the user engages with it. In turn, UX Designers employ UX Design as a technique to make a given product's user experience as simple, seamless and pleasurable as feasible.

Providing the user with an experience that is relevant, efficient and intuitive is the ultimate goal of UX design.

A thorough understanding of users—their needs, values, talents and limitations—is a key component of UX design. It also considers the company's business objectives and ambitions.

The method and technique used to create and execute a product that would enable beneficial and pertinent interactions with people is known as user experience (UX) design. All facets of product development and design are included in user experience design (UX design), often known as user-centric or user-centered design. These include the following:

- packaging and branding;
- installation and setup; and
- operation, usability and performance.

To put it briefly, UX design assists designers in producing useful and practical products. The six main aspects that UX design addresses are as follows:

- Utility: The user's practical need must be satisfied by the product.
- Desirability: The product needs to employ visually appealing and coherent aesthetics, such as graphic design.
- Practicality: The product needs to have an intuitive user interface and a well-known feature set.
- Availability: Everyone must be able to use the product, including those with disabilities.
- Sturdiness: It is imperative that the product has adequate support to enable consumers to swiftly and easily resolve issues.
- Authenticity: The product needs to be reliable; for example, it should come from a well-known brand or have meticulous security.

Although the phrase "UX design" is frequently linked to the design of software or digital services, it can also be easily applied to any tangible product or structure.

Why is UX design important?

An important factor in drawing in and keeping users or customers is the user experience.

Think about any product, digital or physical. Users will simply quit using the product and go to a competitor's product if they don't need it, don't enjoy using it, or can't utilise it adequately without experiencing undue trouble. This could have a negative impact on the company's income and reputation.

(c) Amity University Online

Notes

On the other hand, a well-made, dependable product that meets a defined need and provides excellent performance and support can increase revenue, improve a company's reputation and strengthen customer loyalty to the company and its brand.

In fact, most businesses depend on the revenue generated by their products or services from their consumers, therefore providing a positive user experience can be critical to their survival. For project managers and other product stakeholders, disregarding or cutting corners when it comes to UX challenges, such as usability testing or user acceptability testing, can be a dangerous move.

Lastly, any product design should approach UX design as a dynamic component. User experience strategies are constantly updated and changed in response to changing market conditions, competitive pressures, user expectations and enabling technology. As a result, decisions taken about UX design today are not set in stone. Periodically reviewing UX design decisions is commonly accomplished through the use of user research and user experience testing.

Key Concepts: UX Design, Usability, Interaction Models

The goal of user experience design is to give customers a satisfying and enjoyable interaction with a good or service. It includes a number of elements, including as general enjoyment, usability, accessibility and aesthetics. UX design's essential components include:

User-Centered Design: Throughout the design process, give users' requirements and preferences first priority.

Usability: Make sure the product is simple to use, effective and gives customers what they want out of it.

Information architecture is the process of arranging and structuring data to facilitate user navigation and make sense to the user.

Visual Design: Develop aesthetically pleasing and unified user interfaces that support the brand and increase interaction from users.

Creating interactive prototypes allows you to test and refine design concepts before putting them into practice.

User research: Gather information about user preferences, behaviours and problems.

Accessibility: Create inclusive products that are usable by people with a range of abilities.

Wireframing: To establish the fundamental organisation and design of the user interface, create low-fidelity models of it.

Create meaningful and simple interactions between users and the product through interaction design.

Usability:

Focusing especially on a product's usability and efficacy, usability is a crucial element of UX design. It entails evaluating the degree to which users can accomplish tasks, the degree of intuitiveness of the design and the degree of user satisfaction overall. Usability guidelines include of:

Learnability: The product should be easy for users to grasp and operate right away.

Efficiency: Tasks should be completed by users with the least amount of time and effort.

(c) Amity University Online

Notes

Memorability: After a break, users should be able to recall how to operate the product.

Error Prevention and Recovery: Errors should be minimised in the design and users should be able to quickly correct any mistakes they make.

Satisfaction: Users should have a pleasurable and fulfilling experience all around.

Interaction Models:

Users' interactions and the system's response are described by interaction models. Various application kinds and user behaviours are compatible with different interaction paradigms. Typical models of interaction are as follows:

The frameworks or patterns known as interaction models specify how users interact with a system and how the system reacts to those interactions. The structure and communication flow of user-system engagement are defined by these models. Certain interaction models might be needed for certain application kinds and user scenarios.

Command-Line Interface (CLI): Commands are typed by users to communicate with the system, more difficult to understand but effective for seasoned users.

Interface Graphical (GUI):

- Buttons, menus and icons are examples of graphical elements that users interact with on the system.
- Beginners may find it more intuitive, but for particular activities, speed may be compromised.

Interface for Natural Language:

Similar to voice-activated devices, users converse with the system in normal english, seeks to imitate speech in humans to make it easier to use.

Touch-Based Communication:Using touch gestures, which are popular on tablets and mobile devices, users communicate with the system, intuitive and simple to utilise in some situations.

Interaction using Augmented Reality (AR):Users engage with digital components superimposed over the physical environment, involves making motions, giving vocal instructions, or moving physically.

Product and service designers may create visually appealing, efficient and user-friendly products and services by combining usability principles, appropriate interaction models and UX design in an effective and efficient way.

Evolution of HCI

Over the course of several decades, Human-Computer Interaction (HCI) has undergone major technological advancements, paradigm shifts in design and notable milestones. The following is a synopsis of the major phases in the development of HCI:

1. Early Computing (1940s-1950s):
   - ❖ The earliest computers were primarily used for scientific and military purposes.
   - ❖ Interaction was limited to punch cards and early command-line interfaces.
2. Mainframe and Terminal Computing (1960s-1970s):
   - ❖ The introduction of mainframe computers led to the development of early interactive systems.

© Amity University Online

Notes

- Users interacted through text-based terminals and command languages became more standardised.

3. Personal Computing (1980s):
   - The advent of personal computers brought computing into homes and offices.
   - Graphical User Interfaces (GUIs) were introduced, starting with the Xerox Star and popularised by the Apple Macintosh.
4. Mouse and Pointing Devices (1980s):
   - The mouse and other pointing devices became integral for navigating GUIs.
   - Apple's Macintosh and Microsoft's Windows contributed to the widespread adoption of graphical interfaces.
5. Emergence of the World Wide Web (1990s):
   - The World Wide Web became widely accessible, introducing new interaction paradigms.
   - Web browsers and the hyperlink model changed how users accessed and interacted with information.
6. Ubiquitous Computing (1990s-2000s):
   - Ubiquitous Computing aimed to embed computing capabilities into everyday objects.
   - Pervasive technologies, such as smartphones and wearables, began to redefine user interactions.
7. Mobile Computing (2000s-Present):
   - The rise of smartphones and tablets brought about a shift in HCI towards mobile platforms.
   - Touchscreens, gestures and mobile apps became central to user experiences.
8. Social Media and Collaboration (2000s-Present):
   - Social media platforms transformed how users interacted and shared information.
   - Collaboration and communication tools, such as video conferencing, gained prominence.
9. Gesture-Based Interfaces (2010s-Present):
   - Gesture-based interfaces, popularised by devices like the Microsoft Kinect, allowed users to interact with computers through natural movements.
   - Virtual Reality (VR) and Augmented Reality (AR) systems expanded possibilities for immersive interactions.
10. Voice and Conversational Interfaces (2010s-Present):
    - Voice recognition and natural language processing enabled interactions through voice commands.
    - Virtual assistants like Siri, Alexa and Google Assistant became widespread.
11. Explainable AI and Ethical Considerations (Present-Future):
    - HCI is increasingly concerned with making Artificial Intelligence (AI) systems understandable and ethical.
    - Designing interfaces that promote transparency, trust and inclusivity is a growing focus.

(c) Amity University Online

Notes

As technology develops and designers look for new approaches to make user-centered, accessible and intuitive interactions, HCI continues to evolve. The industry is still changing, adapting to new technology and changing user needs.

1.1.2 Input/Output Channels: Sight, Hearing, Touch, Visual and Angle

Input and output—the sending and receiving of information—are how people engage with the outside world. During a computer interaction, the user provides input to the computer in response to receiving information that the computer has output; in other words, the user's input becomes the computer's input and vice versa. Because of this potential for confusion, we will intentionally blur the lines between input and output and instead focus on the channels that are involved. This blurring makes sense because, although while a given channel's principal function in an interaction may be as an input or output, it's quite likely that it also serves the other purpose. For instance, sight can be used to send information to the computer by, say, focusing on a specific spot on the screen while utilising an eyegaze system, or it can be used solely to receive information from the computer.

Humans primarily receive information through their senses and they output it through the motor control of their effectors. The five senses include taste, smell, touch, hearing and sight. The first three are the most crucial to HCI out of all of these. Although they might be useful in more specialised systems (smells to indicate malfunction, for example) or augmented reality systems, taste and smell do not currently play a significant role in HCI. It is also unclear whether they could be exploited at all in general computer systems. Nonetheless, touch, hearing and vision are crucial.

In a similar vein, the limbs, fingers, eyes, head and voice system are among the many effectors. The major means of interacting with a computer is through the use of fingers for typing or mouse control, along with some voice, eye, head and body position.

Consider utilising a mouse and keyboard with a personal computer (PC). The program you're using features windows, menus and icons in a graphical user interface. When using this system, the main way you get information is by looking at what's on the screen. On the other hand, you might also hear information. For instance, a multimedia presentation might have voice commentary, or the computer might "beep" at you to alert you to an error. Touch also has a role; you may feel the keys move and hear the "click" or sense how the mouse is oriented, which gives you important feedback about what you've done. Using your hands, you input data into the computer by clicking keys or dragging the mouse. In this case, sight and hearing are not directly involved in transmitting information, but they can be used to receive information from a third party (a book, for instance, or another person's speech) that is subsequently sent to the computer.

Vision

Despite a variety of physical and perceptual restrictions, human vision is a highly complicated function and the ordinary person's major source of information. Visual perception can be loosely split into two stages: the actual physical taking in of external stimuli and the mental processing and interpretation of those stimuli. Certain items are invisible to humans due to the physical characteristics of the eye and visual system, yet images can still be created from partial data thanks to the interpretive powers of visual processing. It is important for us to comprehend both phases because they have an impact on what a human can and cannot see visually, which has an immediate impact on how computer systems are designed. After discussing the eye as a physical receptor, we will turn our attention to the processing required for fundamental vision.

© Amity University Online

Notes

The Human Eye

Light is the foundation of seeing. Light is received by the eye, which then converts it into electrical energy. When an object in the world reflects light, the image of that object is focused upside down on the back of the eye. It is converted into electrical signals by the receptors in the eye and sent to the brain.

We'll take a closer look at the several crucial parts of the eye (refer to the Figure below for more information). Light is focused onto the retina, which is located on the back of the eye, by the cornea and lens at the front of the eye. Rods and cones are the two types of photoreceptors found in the light-sensitive retina.

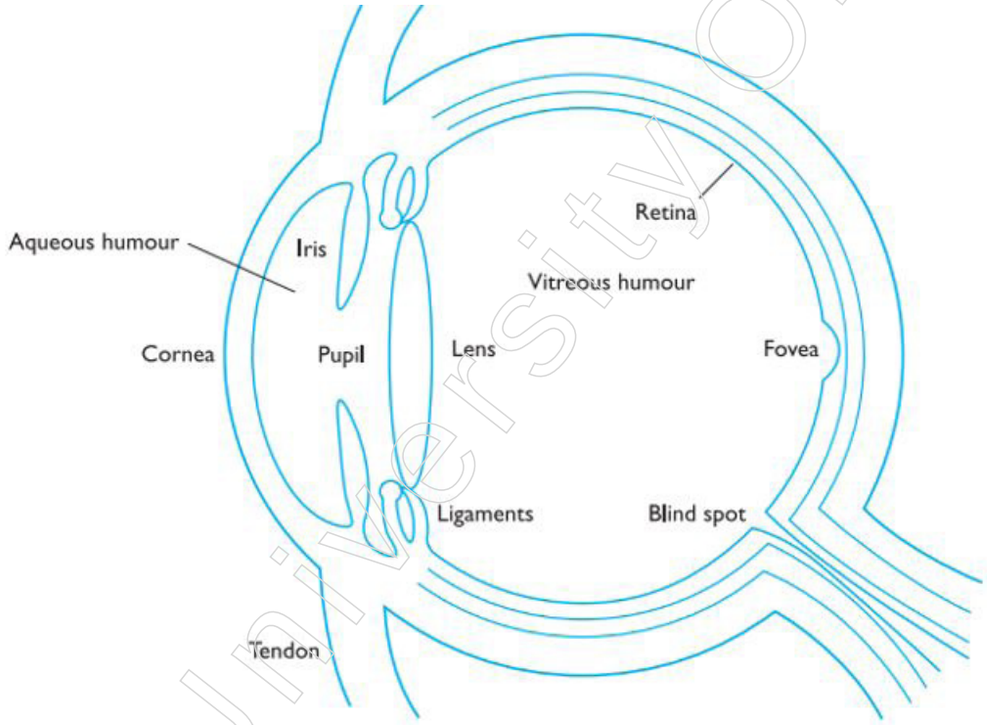


Figure 1: The human eye

Rods are extremely sensitive to light, which enables us to see in dimly lit environments. They are susceptible to light saturation and cannot resolve minute details. The rods have been active and are saturated by the sudden light, which explains why humans experience temporary blindness when we go from a darker room into sunlight. Because the rods restrict them, the cones also stop working. As a result, we are momentarily completely blind. Each eye has about 120 million rods, most of which are found along the borders of the retina. For this reason, rods rule peripheral vision.

The second class of receptors in the eye are called cones. They can withstand more light since they are less light-sensitive than the rods. Cones come in three varieties, each of which is sensitive to a certain light wavelength. Colour vision is made possible by this. The fovea, a tiny portion of the retina where images are fixed, is home to the majority of the eye's 6 million cones.

There is a blind patch where the optic nerve enters the eye, despite the retina being mostly covered in photoreceptors. Although the blind area lacks rods or cones, our visual system makes up for this, preventing us from noticing it under typical conditions.

Ganglion cells are another type of specialised nerve cells found in the retina. The early detection of pattern is carried out by X-cells, which are concentrated in the fovea, while the

(c) Amity University Online

Notes

early detection of movement is carried out by Y-cells, which are more widely dispersed in the retina. Because of the location of these cells, we are able to experience movement even though we may not be able to discern changes in pattern in our peripheral vision.

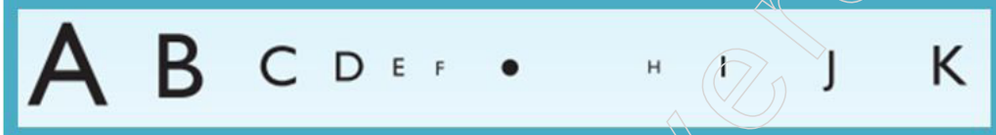
Visual Perception

While an understanding of the fundamental structure of the eye helps to explain the physical mechanics underlying vision, visual experience is more complex than this. Filtered information is fed to processing elements via the visual system, enabling humans to distinguish between colours, distinguish relative distances and recognise cohesive scenes. In order to create efficient visual interfaces, it is important to understand how humans perceive size, depth, brightness and colour. We will discuss some of the possibilities and limitations of visual processing later.

DESIGN FOCUS

Getting noticed

The extensive knowledge about the human visual system can be brought to bear in practical design. For example, our ability to read or distinguish falls off inversely as the distance from our point of focus increases. This is due to the fact that the cones are packed more densely towards the center of our visual field. You can see this in the following image. Fixate on the dot in the center. The letters on the left should all be equally readable, those on the right all equally harder.



This loss of discrimination sets limits on the amount that can be seen or read without moving one's eyes. A user concentrating on the middle of the screen cannot be expected to read help text on the bottom line.

However, although our ability to discriminate static text diminishes, the rods, which are concentrated more in the outer parts of our visual field, are very sensitive to changes; hence we see movement well at the edge of our vision. So if you want a user to see an error message at the bottom of the screen it had better be flashing! On the other hand clever moving icons, however impressive they are, will be distracting even when the user is not looking directly at them

Figure 2: Design focus

Perceiving Size and Depth

Assume that you are atop a hill. On the peak next to you, there are some rocks, some sheep and a tiny tree. A farmhouse with outbuildings and agricultural trucks sits on the slope. There's someone walking towards the top of the track. There is a tiny market town in the valley below.

Even when describing such a scenario, distance and scale are the most common concepts. These factors are simply taken into account by our visual system through easy interpretation of the images it receives. Even if two items seem to be very different in size to us, we are nevertheless able to distinguish between them. In actuality, we can determine distances with this information.

Then how does the eye distinguish between relative distances, size and depth? We need to think about how the image appears on the retina in order to comprehend this. The object's reflected light causes the retina to display an image that is upside-down, as we mentioned in the preceding section. A visual angle is used to specify the image's size. The calculation of the viewing angle is shown in the figure below.

© Amity University Online

Notes

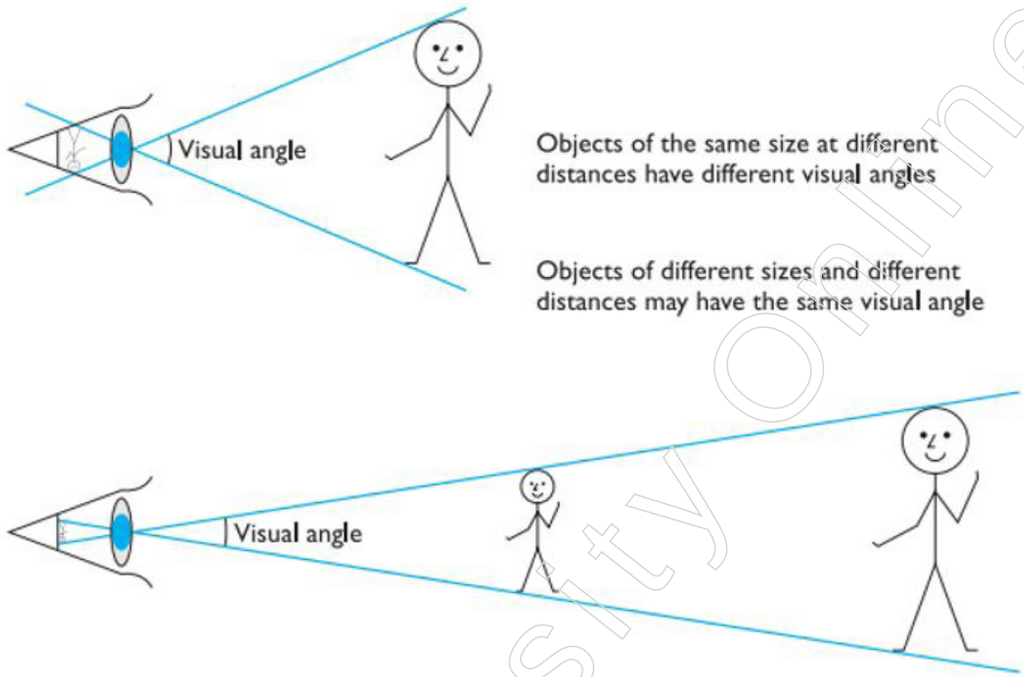


Figure 3: Visual angle

The visual angle of an object is determined by drawing two lines: one from the top of the object to a central point on the front of the eye and another from the bottom of the object to the same point. The object's size and distance from the eye both have an impact on visual angle. Consequently, the larger object will have a larger viewing angle if two objects are at the same distance. Likewise, when two identical things are positioned at dissimilar lengths from the eye, the object that is farther away will have a smaller visual angle. The visual angle shows how much of the object occupies the field of view. The measurement of the visual angle is expressed in either degrees or minutes of arc, where one degree corresponds to sixty minutes of arc and one minute to sixty seconds of arc.

So how does our sense of an object's size change depending on its visual angle? First of all, we cannot see an object at all if its viewing angle is too small. The capacity for fine detail perception is known as visual acuity. Visual acuity can be measured using a variety of methods, the majority of which are part of routine eye exams. For instance, if a line has a visual angle of 0.5 seconds of arc, a person with normal eyesight can detect it. Line spacing can be seen between 30 and 60 seconds of visual arc. These stand for the limitations of the optical acuity of humans.

Does the object's visual aspect influence how big we perceive it, assuming we are able to perceive it at all? We may anticipate that we would perceive an object as smaller as it grows farther away because its visual angle decreases with distance. In actuality, regardless of changes in an object's visual angle, our perception of its size stays unchanged. In other words, even when someone moves away from you, you still perceive their height as constant. The law of size constancy states that variables other than the eye angle affect how large something seems to us.

Our sense of depth is one of these variables. There are several cues that we might use to ascertain the relative positions and distances of the objects we see if we go back to the hilltop scene. When two things overlap, the partially covered object is seen as being farther away because it is in the backdrop. In a similar vein, the dimensions of an object within our field of vision indicate its distance.

(c) Amity University Online

Notes

Familiarity is a third clue; if we anticipate an object to be a particular size, we may estimate its distance appropriately. This has been used humorously in advertising: in one, a man is seen leaving a bottle in the foreground of an advertisement for beer. He trips over the bottle as he moves—it's a massive one in the backdrop, in fact!

Perceiving Brightness

The sense of brightness is a second part of visual perception. Actually, brightness is a personal response to light levels. Luminance, or the amount of light an object emits, has an impact on it. The amount of light that strikes an object's surface and its reflecting qualities determine how bright an object is. Luminance is a measurable physical property that may be assessed with a photometer. brightness and contrast are related because contrast is a function of both an object's and its background's brightness.

Brightness is a subjective response, however it can be defined as the luminance difference that produces a barely perceptible change in brightness. Still, variations in brightness are also compensated for by the visual system itself. The rods dominate vision in low light. Because the fovea has fewer rods than the other regions of the eye, objects in low light are easier to see in peripheral vision and harder to fixate upon. Normal lighting makes the cones dominant.

Increasing brightness leads to an increase in visual acuity. This could provide as justification for employing a bright display. Nevertheless, flicker also increases with luminance. When a light is turned on and off quickly, the eye will see it as always being on. However, the light is interpreted as flickering if the switching frequency is less than 50 Hz. At luminances above 50 Hz, flicker is perceptible. Peripheral vision is also where flicker is more noticeable. This implies that a display will appear to flicker more the larger it is (and, thus, the more peripheral vision it occupies).

Perceiving Color

The perception of colour is a third component that we must take into account. Hue, intensity and saturation are the three basic components of colour that are typically recognised. The light's spectral wavelength determines its hue. Reds have long wavelengths, greens are medium and blues are short. The average person can distinguish between about 150 distinct hues. The brightness of a colour is its intensity and its level of whiteness is its saturation. We are able to distinguish between about 7 million various colours by adjusting these two. But the number of colours that a person without training can recognise is much smaller—roughly ten.

Because the cones in the eye are sensitive to various wavelengths of light, colour is perceived by the eye. Cones come in three varieties, each of which is sensitive to a distinct colour (blue, green and red). The fovea has the best colour vision, while the periphery, where rods are more common, has the worst. It should be mentioned that blue acuity is decreased because cones, which are sensitive to blue light, only occupy 3-4% of the fovea.

Lastly, it's important to keep in mind that 1% of women and 8% of men are colour blind, with the most prevalent symptom being the inability to distinguish between red and green.

The Capabilities and Limitations of Visual Processing

We've already covered some of the strengths and weaknesses of the human visual processing system when we looked at how humans view images. On the other hand, we have mostly focused on low-level perception. The entire image is transformed and interpreted during visual processing, starting with the light that strikes the retina. As we've already mentioned, expectations have an impact on how we see something. For instance,

(c) Amity University Online

Notes

no matter how far away an object is from us, we will always perceive it to be that size if we know it to be that size.

The movement of the image on the retina that happens as we move and when the item we see moves is adjusted for by visual processing. The image that we see is stable even though the retinal image is moving. Similarly, despite variations in luminance, people interpret an object's colour and brightness as constant.

Uncertainty can be resolved by utilising this capacity to understand and take advantage of our expectations. Take a look at the picture in Figure below, for instance.

Figure 4: An ambiguous shape?

What do you perceive? Now consider Figure below

Figure 5: ABC

and Figure below.

Figure 6: 12 13 14

The object occurs in a scenario that makes it easy for our expectations to distinguish between it being a B or a 13.

It can, however, also produce optical deceptions. Take a look at Figure below, for instance. Which line is the longest?

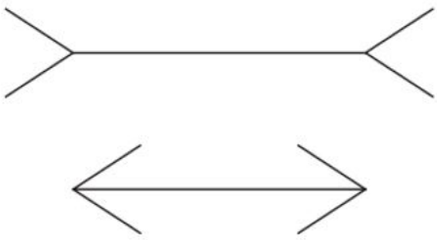


Figure 7: The Muller–Lyer illusion – which line is longer?

(c) Amity University Online

Notes

When this is shown to most people, they will usually respond that the top line is longer than the bottom. The length of the two lines is actually the same. The top line looks to be a concave edge and the bottom line to be a convex edge, which could be the result of a misapplication of the law of size constancy. For this reason, the former is scaled to appear larger and appears farther away than the latter. The Ponzo illusion is comparable (Figure below).

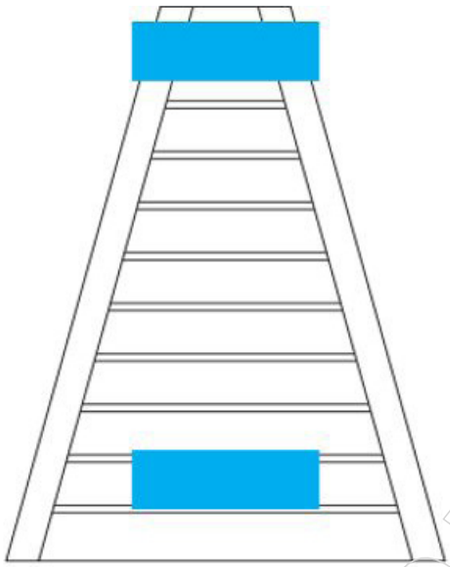


Figure 8: The Ponzo illusion – are these the same size?

Even though both lines in this instance have the same length, the distance effect makes the upper line appear longer. These tricks show us that our sense of scale is not entirely accurate.

The proofreading illusion is another one produced by our expectations offsetting an image. Quickly read the text in the figure below. What is stated there? The majority of readers who read this quickly would understand it correctly, but upon closer examination, the words "the" are repeated in the second and third lines.

The quick brown

fox jumps over the

the lazy dog.

Figure 9: Is this text correct?

These are only a few instances of how the visual system makes adjustments—and occasionally makes excessive adjustments—to enable us to see our surroundings.

Reading

Thus far, our focus has been on the overall perception of images. But text perception and processing is a unique situation that matters to interface design, which always calls for some textual display. It is with an examination of reading that we will close this section. The process of reading goes through numerous phases. The word's visual pattern on the page is first noticed. After that, it is decoded using a language internal representation. The last phases of language processing work with phrases or sentences and involve syntactic and semantic analysis.

© Amity University Online

Notes

The first two phases of this process and how they affect interface design are of particular interest to us. The eye performs quick movements known as saccades while reading, which are followed by fixations. Fixation periods, which make up around 94% of the total amount of time elapsed, are when perception takes place. Regressions are when the eye moves over the text both forward and backward. Increased complexity in the text will result in more regressions.

An adult reads 250 words per minute on average. Given that trials have demonstrated that words may be recognised just as quickly as single characters, it is doubtful that words are scanned sequentially, character by character. Instead, word shape is used to identify words that are familiar. This means that reading accuracy and speed are negatively impacted when word shape hints are eliminated, such as by capitalising words.

Legibility of text is determined by how quickly it can be read. According to experiments, typical font sizes ranging from 9 to 12 points are equally readable when lines are spaced proportionately apart. Likewise, lines with a length of 2.3 to 5.2 inches (58 to 132 mm) are equally readable. It has been demonstrated, nevertheless, that reading from a computer screen is slower than reading from a book. Numerous factors, such as a greater line length, fewer words per page, orientation and the page's experience with the medium, are thought to be responsible for this. Naturally, proper textual interface design can lessen these effects.

Finally, a note regarding the use of contrast in visual display: dark characters on a light screen, or negative contrast, offer higher luminance and, thus, increased sharpness than positive contrast. Thus, legibility will rise as a result. It will, nevertheless, also be more likely to flicker. Based on empirical data, it appears that negative contrast displays are more accurate and are therefore preferred in practice.

Hearing

Although hearing is frequently regarded as second only to sight, we frequently underestimate the quantity of information that our ears can perceive. Take a moment to close your eyes and listen. What auditory sensations are you aware of? From where do they originate? Why are they being made? One can hear the drone of a jet overhead, birdsong, machinery operating at a neighbouring site and cars going by on the road outside as one can sit at my desk. However, one can also able to determine the direction of the noises and their approximate distance. One can therefore determine the direction a car is travelling in and that it is passing on a certain road close to my house based on the sounds one can hear. One can aware that construction is underway in a specific area and that a specific kind of bird is perched in one of my garden's trees.

Our world may be understood in great detail through the use of our auditory system. However, how does it operate?

The Human Ear

Similar to how light initiates eyesight, sound waves or air vibrations initiate hearing. These vibrations are detected by the ear, which then passes them along to the auditory nerves via a number of processes. The outer ear, middle ear and inner ear are the three parts that make up the ear.

The portion of the ear that is visible is the outer ear. It consists of two parts: the auditory canal, which is used to send sound waves to the middle ear and the pinna, the structure that is affixed to the sides of the head. There are two functions for the outer ear. It shields the delicate middle ear from harm in the first place. Wax in the auditory canal

(c) Amity University Online

Notes

keeps dust, debris and overly curious insects from getting to the middle ear. It also keeps the temperature of the middle ear steady. Second, some sounds are amplified by the pinna and auditory canal.

The cochlea connects the middle ear, a tiny cavity, to the inner ear and the tympanic membrane, often known as the ear drum, to the outer ear. The ossicles, the tiniest bones in the body, are located inside the hollow. The ossicles, which transfer the vibrations to the cochlea and ultimately the inner ear, vibrate in response to sound waves that travel through the auditory canal and strike the ear drum. The inner ear is filled with a denser cochlean liquid than the air-filled middle and outer ears, which explains why this "relay" is necessary. It would be difficult for sound waves to go from the air to the liquid immediately. The sound waves are focused and intensified as they are transmitted through the ossicles.

The inner ear's liquid-filled cochlea receives the waves. Delicate hair cells called cilia are found inside the cochlea. When the cochlean liquid vibrates, these cilia bend and release a chemical transmitter that triggers impulses in the auditory nerve.

Processing Sound

We now know that sound is caused by variations in air pressure. We can distinguish it from other things thanks to a variety of features. The frequency of a sound is its pitch. A high frequency results in a high pitch, whereas a low frequency yields a low pitch. The amplitude of the sound determines how loud it is; the frequency stays constant. Timbre is related to sound type; sounds can be produced by different instruments and have varied timbres even though they have the same pitch and volume. Because of the time delay between the sound reaching the two ears and the decrease in strength brought on by the sound waves reflecting off the head, we may also determine the location of a sound.

The frequency range that the human ear can detect is roughly 20 Hz to 15 kHz. At low frequencies, it is less precise, but at high frequencies, it can detect changes in frequency of less than 1.5 Hz. distinct frequencies induce distinct regions of the auditory system's neurons to fire at different rates, which results in different rates of nerve impulse firing.

By filtering some of the noises we hear, the auditory system helps us focus on what's important by blocking out background noise. As seen by the cocktail party effect, when we can distinguish our name yelled over a packed, noisy room, we are selective in what we hear. We cannot distinguish between distinct sounds, though, if they are too similar in frequency or too loud.

As we've seen, sound has a tremendous capacity to communicate information. It is typically limited to alerts and warning noises in interface design, rarely being utilised to its fullest capacity. Multimedia, which could include sound effects, voice commentary and music, is an exception. But even without focusing on the sound source, the ear can distinguish very minor differences in sound and can identify familiar sounds. This implies that sound may be incorporated into interface design more often, for example, to communicate system status information.

Touch

Our final and third sense to discuss is touch, often known as haptic perception. Imagine living without this sense, despite the fact that it is frequently considered less significant than sight or hearing. Touch gives us important information about the surroundings. It can serve as a warning since it alerts us when we come into contact with hot or cold surfaces. Additionally, it gives us input when we try to lift something, like an object. Just pick up a

(c) Amity University Online

Notes

glass of water as an example. The action would be slower and less accurate if we could just see the glass and not sense its form or when our hand made contact with it. Users of several virtual reality games report having this experience: they see the computer-generated things they must manipulate, but they are not physically able to touch them.

Observing these individuals can provide both entertainment and educational value! For this reason, touch is a crucial kind of feedback and this holds true while utilising computer systems as well. One crucial aspect of pressing a button is feeling the button depress. Furthermore, we should be mindful that, while haptic perception is generally not a primary source of information, it can be extremely significant for those whose other senses are compromised. Braille interfaces, for example, might serve as these users' main information source during an engagement. Thus, we shouldn't undervalue the significance of touch.

Unlike the senses of sight and hearing, the sense of touch is not localised. The skin serves as a conduit for impulses. Three different types of sensory receptors are found in the skin: mechanoreceptors react to pressure, nociceptors respond to severe pressure and thermoreceptors respond to heat and cold. In terms of human–computer interaction, we are more interested in the last one.

Mechanoreceptors come in two varieties and they react differently to various pressures. As the skin is indented, quickly adapting mechanoreceptors react to the pressure. Additionally, these receptors respond to increased pressure faster. But if you apply pressure continuously, they quit reacting. Mechanoreceptors that gradually adapt react to pressure that is applied repeatedly.

While these receptors are found throughout the body, certain regions are more sensitive or acuous than others. The two-point threshold test can be used to assess the sharpness of various body parts. Hold two pencils so that the points are approximately 12 mm apart. Try to feel two points when you touch the points with your thumb. If it's not possible, shift the points slightly apart. Measure the distance between two locations that you can feel. The sensitivity decreases with increasing distance. This test can be repeated on other body parts. The measurement on the forearm should be approximately ten times that of the finger or thumb. The most acute senses are those of the fingers and thumbs.

Kinethesis is the second facet of haptic perception, which is knowledge of one's own body's and limbs' positions. The joints' receptors are to blame for this. Once more, there are three varieties: positional receptors, which only react when a limb is in a stationary position; slowly adapting, which reacts to both movement and static position; and quickly adapting, which reacts when a limb is moved in a certain direction. Performance and comfort are both impacted by this perspective. For instance, feedback from the keyboard and knowledge of the relative positions of the fingers are crucial for touch typists.

1.1.3 Memory: Sensory, Short Memory and Long Term Memory

Ever engage in a game of memory? Each player is supposed to recite a list of objects and add one more at the end. The items are all somewhat linked, despite the numerous variations: "I went to the market and bought a lemon, some oranges, bacon," or "I went to the zoo and saw monkeys and lions and tigers," and so forth. People are removed from the game when items are missed or recalled out of order as the list gets longer. The last person standing is the winner. These kinds of games depend on our capacity to store and recall data, including seemingly random objects. This is what our memory system is meant to do.

Indeed, memory plays a major role in many of our daily activities. Our memory not only retains all of our factual knowledge, but also our understanding of acts or processes.

(c) Amity University Online

Notes

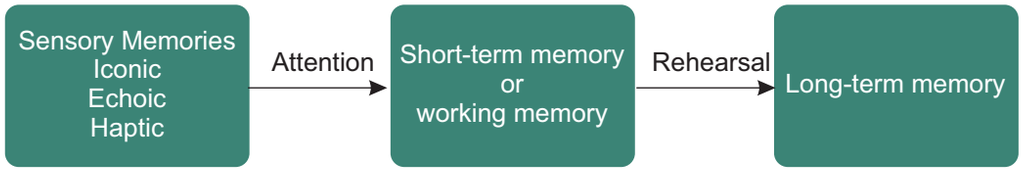


Figure 10: A model of the structure of memory

It enables us to use language, repeat behaviours and apply newly learned information from our senses. Through the preservation of memories from our prior experiences, it also contributes to our feeling of identity.

But how is our memory functioning? How can we retain random lists like the ones created during the memory game? Why is it that some people can recall things better than others? What occurs when we forget, too?

To address concerns like these, we must comprehend some of the capacities and constraints of human memory. The second component of our information-processing model of the human is memory. But as we have already mentioned, this divide is oversimplified because memory is connected to every processor level, as we will show. Keeping this in mind, we will look at the organisation of memory and the processes that occur inside the system.

Most people agree that there are three different kinds of memory or memory function: long-term memory, working memory, or short-term memory and sensory buffers. Whether there are three distinct systems or several roles within a single system is a topic of debate.

Sensory Memory

The stimuli that are perceived by the senses are filtered by the sensory memory. Every sensory channel has a corresponding sensory memory: haptic memory for touch, iconic memory for visual stimuli and echoic memory for aural stimuli. New information that arrives on these channels constantly overwrites these memories.

We can move a finger in front of the eye to show that iconic memory exists. Is it visible at more than one location at once? This suggests that the image persists even after the stimulus is eliminated. When moving sparklers create a lasting image during fireworks displays, a similar effect is most noticeable. Information only hangs out in iconic memory for about half a second.

In a similar vein, our capacity to determine the direction of sound is proof positive that echoic memory exists. This is because both ears are receiving the information. We must store the stimulus in the interim though, as this information is acquired at different times. Information can be briefly "played back" thanks to echoeic memory. When reading, have you ever been asked a question by someone? When you ask them to answer the question again, you find that you already know the answer. Echoic memory is demonstrated by this event as well.

Attention transfers information from sensory memory into short-term memory, limiting the number of stimuli that are viewed as interesting at any particular moment. The ability to focus attention is the ability to choose one competing concept or stimulus from among several. It is evident that humans possess the ability to selectively focus our attention, selecting to focus on one item over another. This results from our sensory and brain processes' limited capacity. We would be overburdened if we did not pay attention to the stimuli that are reaching our senses on a selective basis. We have the ability to select which stimuli to focus on and our arousal—that is, our degree of interest or need—influences

© Amity University Online

Notes

this decision in part. This explains the phenomenon of cocktail parties that was previously noted: we can focus on one topic above background noise, but if our name is mentioned, we can decide to move our attention to a different conversation taking place across the room. Sensory memories swiftly overwrite and lose information, or transfer it to a more permanent memory bank.

Short-term Memory

Working memory, also known as short-term memory, serves as a "scratch pad" for momentary information recall. It is used to store data that is only seldom needed. For instance, mentally do the multiplication 35 × 6. Most likely, you performed this computation in steps, like 5 × 6 and then 30 × 6, adding the results; or you used the formula 6 = 2 × 3 to compute 2 × 35 = 70 and then 3 × 70. We need to save the intermediate steps for later usage in order to do computations like this one. Alternatively, think about reading. You must keep the first part of the statement in mind while you read the remainder of it in order to understand it. These two jobs both require short-term memory.

Access to short-term memory happens quickly—about 70 milliseconds. But it also degrades quickly, so information can only be stored there for a short while—roughly 200 milliseconds.

The capacity of short-term memory is likewise restricted. Memory capacity can be measured using one of two main approaches. The first is figuring out how long a sequence is that can be recalled sequentially. The second lets you recall things at will and in any order. Based on the initial assessment, the typical individual can recall 7 ± 2 numbers. Experiments by established this. Give it a shot. Examine the following series of numbers:

265397620853

Now, record as much of the sequence as you are able to recall in writing. Did you get everything correct? If not, how many numbers were you able to recall? If you can recall five to nine numbers, your digit span is average.

Try the following order now:

44 113 245 8920

Did that come back to you more readily? The numbers are chunked or clustered here. The 7 ± 2 rule can be generalised to mean that humans can recall 7 ± 2 informational units. Thus, information can be chunked to boost short-term memory capacity. Because short-term memory is limited, there is an innate tendency to organise information into manageable chunks in order to make the best use of the memory. Closure is the state in which a chunk successfully forms. The need to finish or close tasks that are stored in short-term memory can be explained by this process in general. A subject runs the risk of losing track of what she is doing and making mistakes as a result if she doesn't do this or is distracted from doing so.

The above-mentioned chunk sequence likewise utilises pattern abstraction; it is composed to resemble a UK phone number for ease of memory. The first few numbers may even be known to us as Leeds' dialling code and the UK's international code—bits of information. Patterns can be helpful memory aids. For instance, most people would find it difficult to recall the following chunk order:

HEC ATR ANU PTH ETR EET

The sequence is simple to remember, though, if you note that the line "the cat ran up the tree" appears after the final character is moved to the first position.

(c) Amity University Online

Notes

recollection of the final words offered is superior to recollection of words presented in the middle, according to data from tests where individuals were free to recall terms. The recency effect is the term for this. The recency effect is mitigated, though, if the subject is given an alternative job to do in between the presentation and recall, such counting backwards. It has no effect on the memory of the other terms. This implies that extraneous information interferes with short-term memory recall. Nonetheless, there is some evidence for the existence of distinct long-term and short-term memories given that this interference had no effect on memory of prior items. The long-term store, which is immune to the recency effect, houses the early products.

It is not always the case that interference hinders short-term memory recall. Baddeley gave his subjects a task that required them to comprehend sentences while simultaneously memorising six-digit digits. They had to respond to questions based on statements, like "A precedes B: is AB true or false?" Surprisingly, this did not cause interference, indicating that short-term memory is composed of multiple channels, including articulatory and visual channels, rather than being a unified system. Interference only arises when tasks use the same channel; sentence processing used the visual channel, while memorising numbers used the articulatory channel.

As a result of these discoveries, Baddeley put out a working memory model that included several components as well as a central processing executive. This is shown in the figure below.

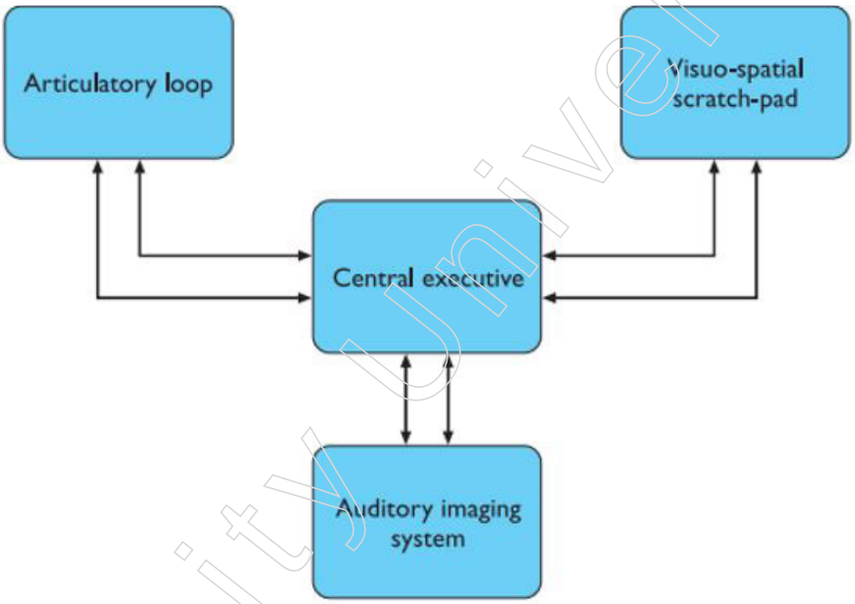


Figure 11: A more detailed model of short-term memory

Long-term Memory

If our working memory, or "scratch-pad," is short-term memory, then long-term memory is our primary resource. All of the things we "know," such as factual data, experiential knowledge and behavioural norms, are kept here. It is distinct from short-term memory in some important aspects. Its capacity is enormous, if not infinite. Second, its access time, which is about a tenth of a second, is comparatively slow. Thirdly, if forgetting happens at all in long-term memory, it does so more slowly. These differences offer more proof that memory is divided into multiple components.

Information is meant to be stored in long-term memory for a long time. Working memory stores information there through rehearsal. In contrast to working memory, long-

© Amity University Online

Notes

term memory does not deteriorate over time; after minutes, it remains unchanged from hours or days later.

Long-term Memory Structure

Semantic memory and episodic memory are the two categories of long-term memory. Our remembering of experiences and events in a sequential order is known as episodic memory. We are able to piece together the precise events that occurred at a certain moment in our lives using this recollection. Conversely, semantic memory is an organised repository of knowledge that we have learned, including facts, ideas and abilities. We can acquire new facts or concepts from our experiences because the information in our episodic memory gets derived into our semantic memory.

Inference, relationship representation and information access are all made possible by the structure of semantic memory. Semantic memory can be conceptualised as a network as one model. Classes link items to one another and parent classes' attributes may be inherited by child classes. We refer to this model as a semantic network. For instance, our understanding of dogs could be kept in a network like the one depicted in the figure below.

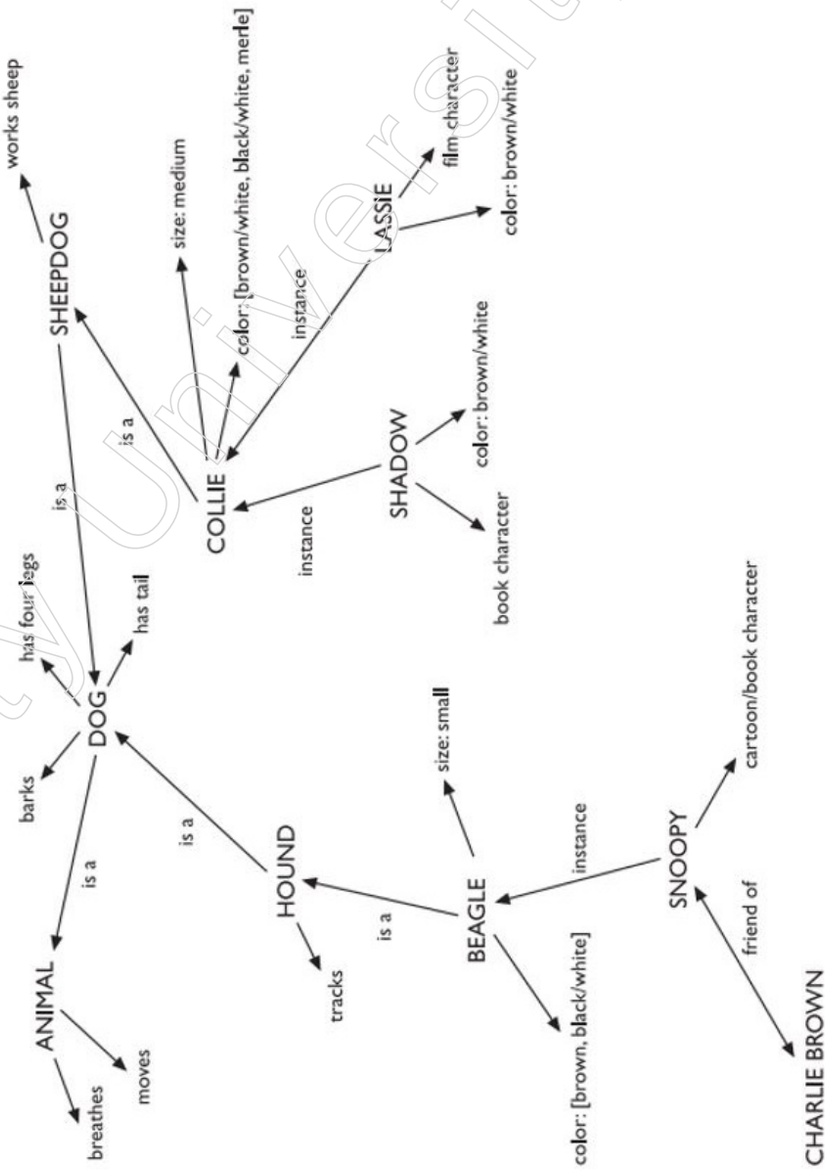


Figure 12: Long-term memory may store information in a semantic network

(c) Amity University Online

Notes

While breed-specific characteristics might be saved with each individual breed, overall canine data is kept at a higher level. This lets us make broad generalisations about certain instances. For example, even though we may not have been informed that Shadow the sheepdog has four legs and a tail, we can nevertheless deduce this information from our basic understanding of dogs in general and sheepdogs in particular. Also take note of the links inside the network that lead to other knowledge areas, such as cartoon characters. This demonstrates the association-based rganization of our knowledge.

Collins and Quillian have shown the validity of semantic networks as a model of memory rganization. The subjects' response times were timed as they were questioned about various attributes of similar objects. Using examples from our own network, fewquestions that were asked, such as:

- "Can a collie breathe?,"
- "Is a beagle a hound?,"
- "Does a hound track?" ().

Subjects took longer to respond to queries like "Can a collie breathe?" than to questions like "Does a hound track?" even though the answers to such questions might seem clear. It is claimed that this is because information is stored at the most abstract level in the former scenario, requiring participants to search further through the memory structure to get the answer.

Several alternative memory architectures have been put out to explain how humans store and represent various kinds of information. Since each of these stands for a distinct area of knowledge, it is possible to think of the models as complementary rather than antagonistic. The connections and interconnections that exist between individual things in memory are represented by semantic networks. They do not, however, enable us to model the depiction of more intricate things or events, which might consist of several different components or actions. Data structures are created by arranging information using structured representations like frames and scripts. These structures have slots where attribute values can be added. Frame slots may hold variable, fixed, or default data. When the slots are filled with the correct values, a frame is instantiated. Networks made up of frames and scripts can be used to depict hierarchically ordered knowledge.

Going back to the 'dog' domain, the knowledge might be represented using a frame system like the one in Figure below. Fixed slots have attribute values that are predetermined; default slots show the typical attribute value, which can be modified in specific instantiations (e.g., the Basenji does not bark); and variable slots allow you to fill them with specific values in a specific instance. Procedural knowledge can also be contained in slots. A slot may be linked to operations or actions that are carried out, for instance, each time the slot's value is altered.

Figure 13: A frame-based representation of knowledge

Notes

Semantic nets are expanded by frames to incorporate hierarchical, structured data. They clearly convey the relative significance of each piece of information by representing knowledge pieces in this manner.

Scripts make an effort to simulate the portrayal of conventional knowledge about circumstances. Take a look at this sentence:

John took his dog to the surgery. After seeing the vet, he left.

We may fill in a good deal of detail by drawing on our knowledge of the activities of veterinarians and dog owners. The animal was not well. The animal was inspected and given care by the vet. Before departing, John paid for the treatment. We are less inclined to interpret the language in the other way, which would imply that John did not remain long enough to speak with the veterinarian since he immediately disliked him!

Script for a visit to the vet

Entry conditions:	*dog ill* *vet open* *owner has money*	Roles:	*vet examines* *diagnoses* *treats* *owner brings dog in* *pays* *takes dog out*
Result:	*dog better* *owner poorer* *vet richer*	Scenes:	*arriving at reception* *waiting in room* *examination* *paying*
Props:	*examination table* *medicine* *instruments*	Tracks:	*dog needs medicine* *dog needs operation*

Figure 14: A script for visiting the vet

This default or stereotyped information is represented by a script, which enables us to completely understand only a portion of the description or cue. A script consists of several parts that can be filled with relevant data, just as slots:

Entry conditions	Conditions that must be satisfied for the script to be activated.
Result	Conditions that will be true after the script is terminated.
Props	Objects involved in the events described in the script.
Roles	Actions performed by particular participants.
Scenes	The sequences of events that occur.
Tracks	A variation on the general pattern representing an alternative scenario.

Figure above shows an example script for a visit to the veterinarian.

The last kind of knowledge representations we store in our memory is procedural knowledge, or our understanding of how to carry out a task. The production system serves as a popular paradigm for this. Long-term memory stores condition-action rules. When a criterion in one of these rules is met by information that enters short-term memory, the action is carried out. As an illustration, two manufacturing rules may be

IF dog is wagging tail

(c) Amity University Online

Notes

THEN pat dog

IF dog is growling

THEN run away

The second rule's requirement is met if we encounter a dog that is growling, in which case we turn tail and flee. (By the way, not recommended!)

Long-term memory processes

Much has been said about the composition of memory, but what about the operations that it employs? Long-term memory is involved in three primary processes: knowledge retrieval, forgetting and information storage. We'll talk about each of these individually.

First, what is the method by which information is stored in long-term memory and how can it be made better? Rehearsing helps transfer information from short-term memory to long-term memory. Information is stored in long-term memory by practice or repeated exposure to a stimulus.

There are several approaches to optimise this procedure. Ebbinghaus used himself as a subject in a number of memory experiments. He compared his memory minutes, hours and days after the learning process in order to assess his capacity to acquire and repeat nonsense syllables. He found that the amount of time spent learning was directly correlated with the amount of knowledge acquired. The entire time theory is the name given to this. On the other hand, research by Baddeley and colleagues indicates that learning time works best when it is spread out over time. In one trial, for instance, postal workers were taught to type and those whose instruction was broken up into one-hour sessions each week outperformed those who learned for two or four hours each week (albeit the former obviously took longer to finish). This is referred to as the practice effect distribution.

Repetition alone, though, is insufficient to effectively retain knowledge. It is harder to remember information that has no meaning. This is demonstrated by the fact that remembering a group of words that represent concepts is harder than remembering a set of words that represent objects. Give it a shot. First, give list A a try and see how well you can recall the words.

List A: Faith Age Cold Tenet Quiet Logic Idea Value Past Large

Now try list B.

List B: Boat Tree Cat Child Rug Plate Church Gun Flame Head

Given that you could see the items in the second list, it was probably easier to remember than the first.

Even easier to memorise are sentences. Bartlett conducted studies on the retention of significant information (as opposed to meaningless material, like that employed by Ebbinghaus). He had participants learn a story about a foreign culture and then retell it in one such experiment. He discovered that when subjects recounted the event, they would substitute words that meant something to them for foreign terms and ideas. The stories were successfully adapted for the target culture. This has to do with how long-term memory is semantically structured; familiar and relevant information can more readily be integrated into memory by connecting it to preexisting structures.

Given that familiarity, structure and concreteness aid in the learning process, what leads us to forget or lose the information we have learned? The two primary theories of forgetting are interference and decay. According to the first theory, knowledge stored in long-term memory may eventually be forgotten. Ebbinghaus came to the conclusion that

© Amity University Online

Notes

information in memory declined logarithmically—that is, it was lost quickly at first and later more slowly—after conducting studies with nonsense syllables. This leads to the conclusion that the older memory trace will last longer if two are equally strong at a given moment, according to Jost's law.

According to the second theory, interference causes information to be lost from memory. Older information is lost when we obtain newer knowledge. We call this kind of intervention retroactive. One prominent illustration of this is the fact that learning your new phone number makes it harder to remember your old one when you change phone numbers. This is so that the previous association is hidden by the new one. But occasionally, the remnants of the past seep through and obstruct the flow of fresh data. We refer to this as proactive inhibition. When you find yourself driving to your former home instead of your current one, this is one example of this.

Emotional variables also have an impact on forgetting. When presented emotive and non-emotive words in studies, people found the former to be more difficult to remember in the short term but easier in the long run. This observation aligns with our understanding of selective memory, in fact. Positive information is easier for us to recall than negative, which is why nostalgia for the "good old days" and intensely emotional experiences are more common than unremarkable ones.

Whether we truly forget anything or if it's just getting harder to recall some things from memory is up for debate. Since it is hard to demonstrate that humans forget things, this question is essentially meaningless. In fact, the appearance of forgetting something could simply be the result of being unable to locate it! Evidence does, however, seem to indicate that we might not fully lose knowledge from our long-term memory. Proactive inhibition, to start, shows that past knowledge can be recovered even after it has been "lost" via interference. The "tip of the tongue" feeling, on the other hand, suggests that although some information is there, it cannot be adequately accessed. Thirdly, knowledge might be recognised but not remembered, or it might be remembered only when prompted.

This brings up information retrieval, the third function of memory. Here, recall and recognition are the two categories of knowledge retrieval that need to be distinguished. Recall involves reproducing information from memory. Recognising that the material has been seen before is made possible by the way it is presented. As the information is given as a cue, recognition is the less complicated cognitive action.

Retrieval cues, on the other hand, let the person retrieve information from memory more rapidly, which can help with recall. Using categories is one of these cues. In an experiment, participants were asked to recall word lists that were randomly arranged and some that were categorised. It was simpler to remember words associated with a particular category than the other terms. Allowing participants to classify their own word lists while they are learning enhances recall even further. Take into consideration the list of words below, for instance:

child red plane dog friend blood cold tree big angry

Create a tale that connects the words now, utilising as many detailed details as you can. Now make an effort to remember every word you can. Was this exercise more straightforward for you than the last one when the phrases had no connection?

Vibrant imagery is frequently used as a trigger to aid in memory retention. It is well known that when someone describes a scene to them, they frequently picture it. On the basis of their visualisation, they can then respond to inquiries. In fact, when given a description of a scenario, participants tend to embellish it with more details. Think over the description below and picture the scene:

(c) Amity University Online

Notes

The crowd's cacophony was overpowered by the engines' roar. People stood up and excitedly waved their hands despite the intense heat. After the flag dropped, they took off. The automobile quickly separated from the group and accelerated frantically around the curve. It cornered and its wheels came off the ground for a moment. As the sun descended straight, its shimmering paint caught the light. The driver had a laser-like focus while holding the wheel. Drops of sweat were visible on his forehead.

What colour is the car, without turning back to look at the passage?

You have a mental image of the scene, complete with the colour of the car, if you could respond to that question. In actuality, the description makes no mention of the car's colour.

1.1.4 Reasoning and Problem Solving

We have talked about how data enters and exits the human body as well as how it is stored. At last, we examine its manipulation and processing. This is possibly the most intricate aspect that distinguishes humans from other artificial and natural information-processing systems. There is little evidence to suggest that animals can use information in quite the same way as humans, despite the fact that it is obvious that they acquire and store it. Similar to this, artificial intelligence has led to the creation of machines with restricted vision and information storage. However, only a small domain may make use of that knowledge.

Humans, on the other hand, can reason and solve issues using information, even in situations where the knowledge is incomplete or unavailable. Though we might not always be able to recognise the processes we employ, we can always recognise the end results of these processes—our thoughts. Human cognition is conscious and self-aware. We can also solve difficulties that we have never seen before and think about things that we have never experienced. How is this carried out?

Knowledge requirements for thinking might vary. Certain cognitive tasks have strict guidelines and limited knowledge requirements. Others demand a large amount of expertise across several fields. For instance, calculating a subtraction involves only a modest amount of knowledge from a limited domain, but comprehending newspaper headlines necessitates knowledge of politics, society, prominent characters and global events.

Reasoning

The process of using our existing knowledge to make new deductions or conclusions about an area of interest is known as reasoning. Reasoning comes in three flavours: abductive, inductive and deductive. Though we employ all of these forms of reasoning in our daily lives, there are some notable differences between them.

Deductive Reasoning

Using deductive reasoning, the stated premises lead to the logically necessary conclusion. As an illustration,

If it is Friday then she will go to work

It is Friday

Therefore she will go to work.

It is crucial to remember that this is the logical inference from the premises; it need not match our definition of truth. Thus, for instance,

If it is raining then the ground is dry

© Amity University Online

Notes

It is raining

Therefore the ground is dry.

is a completely reasonable deduction, despite the fact that it goes against what we know to be true about the world.

Thus, deductive reasoning is frequently used incorrectly. Considering the premises

Some people are babies

Some babies cry

It will be assumed by many that "Some people cry." Given that we are not informed that all infants are humans, this argument is in fact false. Thus, it stands to reason that the crying babies are the non-human ones.

Human deduction is at its worst here, where validity and truth collide. People's incorporation of their prior information into their thinking process is one explanation for this. This is for a valid purpose. It enables us to take shortcuts, resulting in productive but instructive interpersonal communication. In our interactions with one another, we presuppose a certain level of shared knowledge, which enables us to evaluate the conclusions and inferences made by others. Should the preference be for validity over truth, then all premises must be stated clearly.

Inductive Reasoning

The process of induction involves extrapolating knowledge about examples we haven't seen from cases we have seen. For instance, we can assume that all elephants have trunks if we can prove that every elephant we have ever seen has one. Naturally, this inference is untrustworthy and can only be proven to be incorrect, not true. We only need to provide an elephant without a trunk to refute the inference. No matter how many elephants with trunks we have seen or are aware exist, the next one we see might not have one. For this reason, we will never be able to prove it. Our best option is to collect data to back up our inductive conclusion.

We continuously employ induction as a helpful way to learn about our surroundings, despite its unreliability. Although it is impossible for us to witness every elephant that has ever existed or will ever exist, we do have some knowledge about them that, for all intents and purposes, we are willing to rely on. Most of this knowledge has been gained by induction. We are better at using positive than negative data, therefore even if we saw an elephant without a trunk, we wouldn't be inclined to change our opinion that "all elephants have trunks." Wason's original experiment serves as an example of this. Four cards are dealt to you, as shown in the figure below. One side of each card features a letter and the other a number. To determine whether the adage "If a card has a vowel on one side, it has an even number on the other" is true, which cards would you need to pick up?

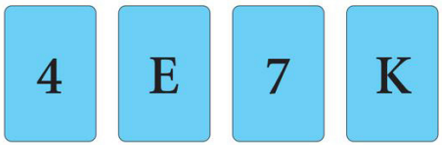


Figure 15: Wason's cards

Comparing the E and the 4. Is this how you usually respond to things? Was it yours? This, however, only makes use of affirmative data. Actually, in order to determine whether the assertion is true, we must look for countervailing evidence. If we can locate a card with an odd number on one side and a vowel on the other, the statement has been refuted.

(c) Amity University Online

Notes

Thus, we have to examine E and 7. (It makes no difference what is on the other side of the other cards; the statement simply states that all vowels have even numbers, not that all even numbers have vowels.)

Abductive Reasoning

Abduction is the third category of reasoning. Abduction ties a fact to the behaviour or condition that gave rise to it. We use this process to come up with explanations for the things we see. Let's say, for instance, that we are aware that Sam consistently drives too quickly after drinking. We can assume Sam has been drinking if we witness her driving too quickly. Naturally, this is also questionable because there could be other explanations for her erratic driving, such as being summoned to an emergency.

It is evident that individuals do draw conclusions in this manner, despite the fact that it is unreliable and they cling to these conclusions until they are refuted by evidence. This may cause issues when utilising interactive systems. Unless there is proof to the contrary, the user will assume that an action causes an event if the occurrence consistently follows the action. Confusion and sometimes even mistakes arise when the occurrence and the action are, in fact, unrelated.

Problem Solving

Problem solving is the process of applying our existing knowledge to solve an unknown task, whereas reasoning is the process of drawing new conclusions from what is already known. Human issue solving is typified by our capacity to modify our knowledge base in order to address novel circumstances. Nonetheless, solutions frequently appear to be unique and imaginative. There are several perspectives on how individuals resolve issues. The first is the Gestalt perspective, which dates to the first part of the 20th century and holds that solving problems requires both insight and knowledge reuse. Although this has been mostly replaced, the issues it sought to answer are still relevant and its impact can be observed in more recent studies. The issue space theory, which holds that the mind is a restricted information processor, was another significant hypothesis that was put forth by Newell and Simon in the 1970s. Subsequent iterations of this theory built upon the previous one and made an effort to reframe Gestalt theory in terms of ideas related to information processing. We shall examine each of these viewpoints in brief.

Gestalt Theory

Gestalt psychologists were responding to behaviourists' assertion that problem solving consists of trial and error or replicating pre-known answers. The Gestalt school believed that this theory was insufficient to explain how people solve problems. Rather, they argued that solving problems is both fruitful and procreative. Though productive issue solving entails insight and problem rearrangement, reproductive problem solving relies on prior experience, as asserted by behaviourists. Reproductive problem solving may, in fact, make it more difficult to find a solution since one may "fixate" on the issues that are well-known and fail to consider new perspectives that could provide a solution.

Gestalt psychologists provided experimental data to support their theories. Apes were spotted by Kohler combining sticks to reach food outside of their cages, which he claimed was evidence of apparent intelligence. Since the apes had previously been in the wild and might have been utilising prior information, it was challenging to confirm this.

Other studies looked at how people solve problems. Maier's pendulum problem is a well-known illustration of this. The issue was that two strings were hanging from the ceiling in the room where the subjects were being held. There were additional items in the room

© Amity University Online

Notes

as well, such as extensions, poles and pliers. The assignment was to knot the string fragments together. But it was impossible to grasp both at once because they were too far apart. Few people decided to use the pliers' weight as a pendulum to "swing" the strings together, despite the fact that many solutions were put forth by the subjects. However, this answer became apparent to the subjects when the experimenter brushed against the string, causing it to move. This, according to Maier, is an illustration of constructive restructuring. The individuals were able to gain understanding and view the issue from a different angle thanks to the movement of the string. The experiment also demonstrates fixation, as the volunteers were initially blind to everything other than the function or application of a pair of pliers.

Gestalt theory describes human problem solving in an appealing way, but it does not offer enough structure or proof to back up its claims. It doesn't define insight or specify when restructuring takes place, for example. On the other hand, information-processing theory was made possible by the departure from behaviourist ideas.

Problem Space Theory

According to Newell and Simon, the problem space is where problem solutions should be focused. Problem states make up the problem space and creating these states through the use of lawful state transition operators is the process of solving problems. People utilise the operators to transition from the problem's original state to its goal state. Because these problem spaces might be very large, heuristics are used to choose the right operators in order to accomplish the desired outcome. The means-ends analysis heuristic is one such tool. In a means-ends analysis, the objective state and the initial state are compared and an operator is selected to minimise the difference between the two. Consider the following scenario: you wish to relocate your desk from the room's north wall to the window as you rearrange your workspace. When you first arrive, you think the desk is against the north wall. Having the desk by the window is the ideal situation. The location of your desk is the primary distinction between these two. You can move objects using a variety of operators, such as carrying, pushing, dragging and so on. But you realise that something has to be light to transport and your desk weighs a lot. Making the desk lighter is your new subgoal as a result. For this, your operators might have to take out drawers, among other things.

The fact that Newell and Simon's model works within the limitations of the human processing system means that it is constrained in its ability to search the issue space by the short-term memory's capacity and the retrieval speed of information. This is a crucial aspect of the model. Experience helps us solve problems more quickly inside the problem space framework by enabling us to efficiently select operators and organise the problem space.

The General Problem Solver model developed by Newell and Simon, which is based on their theory, has primarily been used for problem solving in well defined domains, such as solving puzzles. Although the challenges may not be well-known to you, the problem description clearly outlines the information needed to solve them, as does the anticipated solution. In real-world situations, defining the objective or locating the knowledge needed to address the challenge may be challenging. These kinds of issues necessitate a deep understanding of the domain; for instance, solving a programming problem requires familiarity with both the language and the domain in which the program is used. In this case, a key to overcoming the issue might be to explicitly define the purpose.

Analogy in Problem Solving

Analogy is a third component of problem solving. Here, the way individuals resolve unique difficulties is of interest to us. Analogical mapping—the process of applying information from a related, well-known domain to a new problem—is one approach to

(c) Amity University Online

Notes

solving this. Operators from the known domain are moved to the new one after similarities between the two are identified.

Analogous stories have been used to examine this process. Gick and Holyoak presented the following issue to their subjects:

A cancerous tumour is being treated by a doctor. He needs to shoot it with high-intensity rays in order to destroy it. They will, however, also obliterate the surrounding healthy tissue of the tumour. The tumour will persist even if he reduces the intensity of the rays. How is the tumour destroyed by him?

The way to solve this issue is to aim low-intensity radiation at the tumour from several angles that converge. In this manner, the tumour receives a combined dose of high intensity radiation from the rays while the healthy tissue receives innocuous low intensity radiation. Researchers discovered that just 10 percent of participants arrived at this solution on their own. But when they were told this similar tale and informed it would be helpful, this increased to 80%:

A fortress is being attacked by a general. Since the roads are mined to detonate if a lot of troops cross them, he is unable to send all of his men in at once. For this reason, he divides his men into smaller units and dispatches them via several routes.

Despite this, it appears that individuals frequently overlook similar information unless it is conceptually related to the area of concern. Many individuals were unable to draw the connection when they were not instructed to use the narrative. But when the story was made semantically close to the issue—for instance, a general employing rays to destroy a castle—the number of people who noticed the comparison increased.

The Gestalt theory of constructive restructuring and understanding is reflected in the usage of analogies. An old problem is solved with an old knowledge base.

Skill Acquisition

So far, every approach to problem solving that we have thought of focuses on solving unknown problems. But most of the time, the issues we deal with are not entirely novel. Rather, we become proficient in a specific domain area over time. However, how does one acquire such a skill and how does it improve our ability to solve problems? By examining the differences between novice and expert behaviour in specific domains, we can obtain insight into the functioning of competent behaviour as well as the acquisition of skills.

Chess playing is a domain that is frequently explored. Because it is simply represented in terms of problem space theory, it is especially appropriate. The starting state of the board is the opening position; the objective state is when one player checkmates the other; and the operators to move states are legitimate chess movements. Examining competent behaviour in the framework of the issue space theory of problem solving is thus feasible.

Among others, DeGroot, Chase and Simon's studies of chess players yielded some intriguing findings. Chess masters' behaviour was compared to that of less skilled players in every trial. The first finding was that players rarely looked ahead more than six movements (and sometimes much fewer) while selecting their next move. Masters spent less time to decide and made better decisions, but they did not weigh more options than less skilled players.

What, therefore, distinguishes skillful from less competent behaviour in chess? Chess masters seem to be able to recall board setups and the advantageous moves that go along with them. Mastery players are far more adept at recreating the board than less experienced players when given actual board locations to recall. Nonetheless, the

(c) Amity University Online

Notes

groups of players performed just as poorly at reconstructing the locations when presented with random configurations, which were unknown. Therefore, it appears that skilled players "chunk" the board configuration so that they can remember it more quickly. More experienced gamers employ bigger chunks, which helps them retain more information.

Competent computer programmers also exhibit this behaviour. Additionally, because they have the structures necessary to assemble the right portions, they can reconstruct programs more skillfully than beginners. They obtain blueprints that serve as code to address certain issues. They will remember and apply that specific plan when they run into that issue in a different program or domain.

The way various problems are organised is another way that skilled and less talented problem solvers can be distinguished from one another. Novices frequently categorise situations based on flimsy criteria like the items or traits that both of them share. Experts, however, show a deeper comprehension of the issues at hand and classify them based on fundamental conceptual commonalities that may not be immediately apparent from the problem descriptions.

All of these variations result from the expert's superior knowledge encoding; thorough tuning of information structures allows for precise and efficient retrieval. However, how can this occur? How is a skill like this learned? Anderson's ACT* model is one skill acquisition model. ACT* distinguishes three fundamental skill levels:

- The learner interprets problem-related facts using general-purpose rules. This requires a lot of memory access and is slow.
- The student creates rules according to the assignment.
- To improve performance, the regulations have been adjusted.

There are general mechanisms given to explain the changes between these levels. One way to go from the first to the second is by proceduralisation. It substitutes variables with predetermined values and eliminates the portions of the rule that require memory access. Conversely, generalisation is a mechanism that advances from the second to the third level. From the particular situations, it generalises to the general characteristics of those cases. A general-purpose rule is created by condensing the commonalities among the rules.

Examples are the greatest way to explain these. Let's say you are taking a cooking class. At first, you could have a few clear mental images of foods as well as a rough guideline for how long a dish should bake. The rule can be instantiated by obtaining data from memory.

```
IF cook[type, ingredients, time]
THEN
    cook for: time
cook[casserole, [chicken,carrots,potatoes], 2 hours]
cook[casserole, [beef,dumplings,carrots], 2 hours]
cook[cake, [flour,sugar,butter,eggs], 45 mins]
```

Your knowledge gradually becomes proceduralised and you develop certain guidelines for every situation:

IF type is casserole

AND ingredients are [chicken,carrots,potatoes]

THEN

chook for. 2 hour

Notes

```
IF type is casserole
AND ingredients are [beef,dumplings,carrots]
THEN
   cook for: 2 hours
IF type is cake
AND ingredients are [flour,sugar,butter,eggs]
THEN
   cook for: 45 mins
```

Lastly, by drawing generalisations from these rules, you can create general-purpose rules that take advantage of their similarities:

```
IF type is casserole
AND ingredients are ANYTHING
THEN
   cook for: 2 hours
```

A great deal of knowledge is used in the initial stage. The second phase depends on established protocols. The final phase exemplifies proficient conduct. It may really become routine for such behaviour, making it challenging to identify explicitly. Consider an activity that you are proficient at, like riding a bike or driving a car. Try to explain to someone the precise steps you take to accomplish this. This will be really challenging for you. In actuality, specialists frequently need to mentally practice their actions to pinpoint exactly what they do. Although such expert behaviour is effective, it could lead to mistakes if the activity's setting changes.

1.2 Human Computer Interaction

We must comprehend both sides of the relationship if we want to comprehend how people engage with computers.

What goals are we attempting to accomplish when interacting with computers? Think about what transpires when we communicate with one another: we are either giving or receiving information from another. Most of the time, the information we get is in response to what we have just told them and we may react accordingly. Therefore, communication of information occurs during interaction. The same ideas apply to electronic computers as well: interaction is a process of information transfer that occurs between the user and the machine.

A Typical Computer System

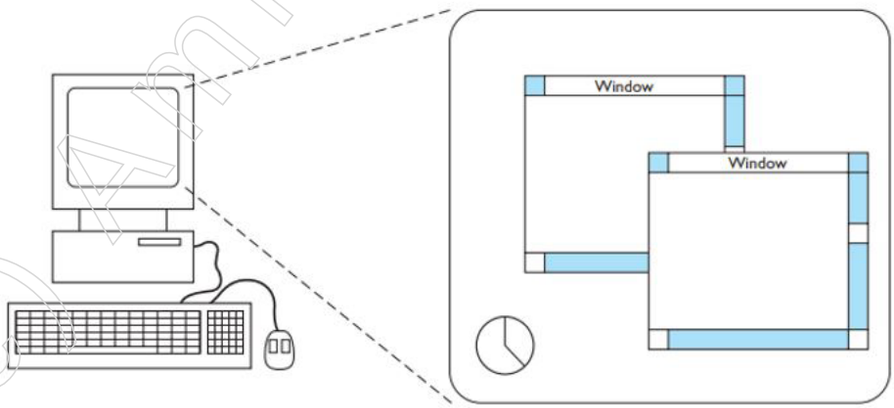


Figure 16: A typical computer system

(c) Amity University Online

Notes

Examine the standard PC configuration depicted in the figure below. A colour screen, a keyboard, a mouse and the actual computer "box" are all present. Alongside it is the screen layout. Examining the interface allows us to understand how its features relate to the devices that are being utilised.

Levels of Interaction – Batch Processing

Batch data entry was the method used in the early days of computers to enter information into the computer in big quantities. The user would merely place a stack of punched cards onto a reader, hit the start button and come back a few hours later to utilise the machine minimally. Even with pre-prepared electronic files or maybe machine-read forms, this still happens today. For some types of applications, such as entering survey findings or printing pay checks, it is unquestionably the most suitable option.

When batch processing is used, conversations happen over several hours or days. On the other hand, interactions on a normal desktop computer system take seconds or even fractions of a second (or, in the case of slow web pages, minutes!). This shift in interactive pace was significantly responsible for the growth of the area of Human–Computer Interaction.

Richer interaction – everywhere, everywhen

The era of computers is upon us! In order to automate shopping, provide email access in your kitchen, or just to call for repair when necessary, information appliances are adding internet access or specialised systems to the refrigerator, microwave and washing machine. We always have smartcards and WAP phones with us and our homes are visible to the public through webcams and security systems that keep an eye on us. Is the computer system shown in Figure above truly typical, or is it more like the one shown in Figure below?

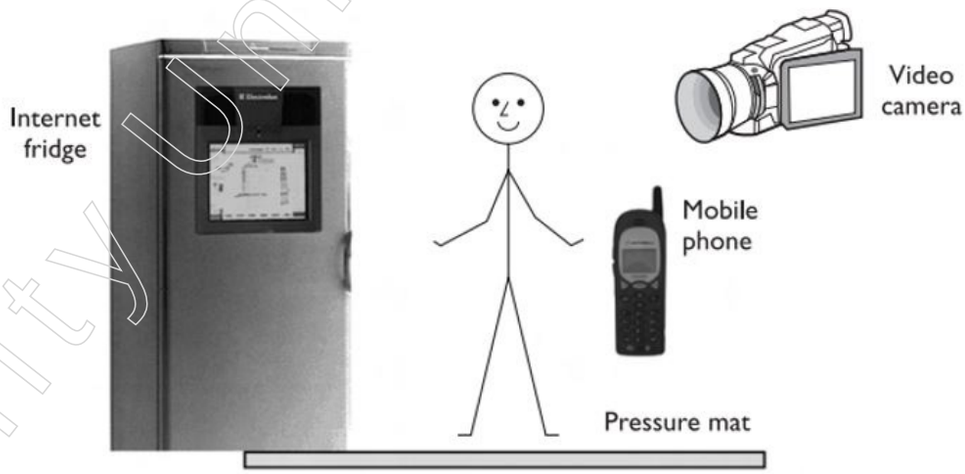


Figure 17: A typical computer system?

1.2.1 Devices: Part 1

Text Entry Devices

Entering text is one of our primary computer tasks, whether we're writing a book, creating an office memo, composing a thank-you note on your birthday, or just sending an email to a buddy. The most common way to enter text is using a standard keyboard, however there are a few alternatives available, such as phone keypads, keyboard layouts

(c) Amity University Online

Notes

and "chord" keyboards that need certain finger combinations to enter letters. Beyond speech recognition and handwriting, there are more drastic options.

The Alphanumeric Keyboard

One of the most widely used input devices in use today is still the keyboard. It is used to enter commands and textual data. The first six letters of the top row of alphabetical keys, QWERTY, are used to identify the great majority of keyboards, which share a standard layout. Although there are other keyboard layouts that offer some advantages over the QWERTY layout, they have not been able to overcome the QWERTY keyboard's enormous technological inertia. These options come in two flavours: chord keyboards and 26 key layouts. The alphabetic keys are rearranged in a 26-key configuration, which either uses easier procedures or places the most often used letters under the fingers with the strongest dexterity. We will talk about two 26 key layouts, alphabetic and DVORAK, in addition to QWERTY, as well as chord keyboards.

The QWERTY Keyboard

On a QWERTY keyboard, the letter and digit layout is fixed (see to Figure below); nevertheless, non-alphanumeric keys differ throughout keyboards. Key assignments on British and American keyboards, for instance, differ; specifically, the pound sign (£) appears above the number three on the UK keyboard, whereas the dollar sign ($) appears above the same on the US keyboard. The usual arrangement may also differ in how brackets, backslashes and other elements are arranged. Additionally, accented letters are present on several national keyboards and the traditional French layout arranges the capital letters in distinct positions (AZERTY begins on the top line).

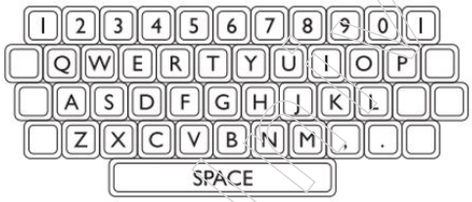


Figure 18: The standard QWERTY keyboard

But typing is not best done with the keys arranged QWERTY. The mechanical typewriter era is the source of the keyboard's current arrangement. A key hit propelled an arm in the direction of the carriage, leaving an impression on the paper that became the letter on the head on the ribbon. The solution to this was to arrange the keys so that common combinations of consecutive letters were placed at different ends of the keyboard, meaning that the arms would typically move from alternate sides. If two arms flew towards the paper in quick succession from nearly the same angle, they would often jam. A fascinating anecdote about the key arrangement is that a salesman's ability to type the word "typewriter" rapidly was crucial for impressing prospective clients—that is, the letters are arranged on the upper row!

The initial mechanical limitations no longer apply to the electric typewriter or the computer keyboard, although the QWERTY keyboard layout is still widely used. This is due to sociological reasons: the management is unable to accept a temporary decline in performance while the new skills are learned and the large pool of skilled typists would be hesitant to relearn their trade. Additionally, there is a significant investment in the present keyboards, which would all need to be phased out or replaced at a high expense, necessitating the need for workers to be skilled on both keyboards. It's likely difficult to overcome the technical stagnation that has occurred as entire communities have taken to keyboarding.

© Amity University Online

Notes

Ease of Learning – alphabetic keyboard

The alphabetic keyboard, in which the letters are placed alphabetically across the keyboard, is among the most obvious layouts to be created. Contrary to expectations, this layout does not make it easier for inexperienced typists to operate. It would seem that since this layout has no intrinsic advantages, studies have demonstrated that it is not speedier for adequately educated typists. Furthermore, it seems that the alphabetic layout has relatively little effect on typing performance, even for inexperienced or infrequent users. Some portable electronic personal organisers utilise these keyboards, maybe because they appear to be easier to use than QWERTY keyboards. Furthermore, it deters users from trying to employ their touch-typing abilities on a tiny keyboard, avoiding complaints about how difficult it is to use!

Speech Recognition

Although speech recognition has shown promise throughout the years, it is still only employed in extremely specific scenarios and is still a relatively new field in text entering. Since we are accustomed to this type of engagement, there is an innate excitement for being able to communicate with the computer and having it obey our directions. Although reports of successful recognition rates of over 97% have been made, this is still unacceptable because it translates to one misspelt letter per thirty words or roughly every six words! Also take note of the fact that this performance is typically only reported when a limited list of command phrases is used. With natural language's inherent imprecision, ambiguity and pauses, attempting to develop such systems to the point of comprehending it would only lead to a host of other issues that haven't been fully resolved, not even for keyboard-entered natural language. Furthermore, the system's performance will decline with each new speaker unless it is taught and adjusted for the differences in speech patterns among the speakers. In addition to background noise, strong accents, colds and emotions can also make it difficult to recognise someone. This brings up the practicality of using noise in an office setting: if everyone in an open-plan office were to talk on their phones, the noise level would significantly increase, posing challenges in addition to the possibility that background noise could lead to mistakes. Maintaining confidentiality would likewise be more difficult.

Speech technology, in spite of its drawbacks, has found use in certain areas, including telephone information systems, accessibility for the disabled, hands-occupied situations (particularly in the military)and RSI sufferers. The first is employing software that was initially intended for keyboard use as a substitute text entry device in an environment. The second is to completely rework a system to maximise its advantages and minimise any potential drawbacks. Lastly, it can be applied in situations when keyboard-based input is neither feasible or feasible. Speech technology is now making headway in the later, more extreme domains.

Positioning, Pointing and Drawing

The capacity to point at something on the screen and alter or do other tasks with it is a fundamental feature of most modern computing platforms. Such gadgets have a long history, especially in computer-aided design (CAD), where the main tasks are sketching and positioning. Using pointing devices, one can either manipulate a pointer on the screen or point, position and select items directly. Although the ability of drawing with a mouse differs greatly from that of drawing with a pencil, many pointing devices can also be utilised for free-hand drawing. Although desktop computers still utilise mice the most, the growing popularity of laptops and mobile devices is posing problems for the mouse. Yes, this laptop that has a touchpad instead of a mouse is being used to type these words.

(c) Amity University Online

Notes

The Mouse

The mouse, which is the small device with a tail linking it to the computer in our basic computer system, is now a crucial part of most desktop computers offered today. It's a little, palm-sized box with a weighted ball inside that rotates inside the housing when the box is moved over the surface. Tiny rollers in contact with the ball sense this rotation and change the potentiometer values accordingly. You might be able to see these rollers if you periodically remove the ball to clear the dust. Changes in the ball's position can be immediately correlated with the potentiometers' changing values. To detect both horizontal and vertical motion, the potentiometers are oriented in distinct orientations. The computer receives the relative motion data using a wire that is fastened to the box, or wirelessly or infraredly in certain situations. The computer uses this data to move the cursor, or pointer, around the screen. The word "mouse" comes from the whole arrangement's tendency to resemble a rodent, with the wire serving as the tail and the box as the body. Apart from its ability to detect motion, a mouse usually contains one, two, or three buttons on top. These are employed to make a choice or to take action. Because they implement distinct actions for a single and a double button click, single-button mice typically provide functionality that is comparable to multi-button mice. A "double-click" occurs when you quickly hit the button twice in a row. Multi-button mice typically assign one function to each individual button.

The mouse moves planarly across the desktop and is an indirect input device since it has to be transformed to match the desktop's horizontal orientation to the screen's vertical alignment. While up-and-down movements on the screen are accomplished by moving the mouse away from the user, left-and-right movements are directly mapped. The ball can be physically pulled up from the desktop and replaced in a new position without moving the cursor; the mouse just shows the relative movement of the ball within the housing. Although this has the benefit of requiring less actual space for the mouse, it is less user-friendly for inexperienced users. Users have minimal arm fatigue because the mouse is desk-mounted, but hand-eye coordination issues may arise due to the indirect nature of the medium. But one of the mouse's main advantages is that the pointer is small and easy to move without covering up the screen.

Optical mice

Optical mice work differently from mechanical mice. A light-emitting diode emits a weak red light from the base of the mouse. This is reflected off a special pad with a metallic grid-like pat tern upon which the mouse has to sit, and the fluctuations in reflected intensity as the mouse is moved over the gridlines are recorded by a sensor in the base of the mouse and translated into relative x, y motion. Some optical mice do not require special mats, just an appropriate surface, and use the natural texture of the surface to detect movement. The optical mouse is less susceptible to dust and dirt than the mechanical one in that its mechanism is less likely to become blocked up. However, for those that rely on a special mat, if the mat is not properly aligned, movement of the mouse may become erratic – especially difficult if you are working with someone and pass the mouse back and forth between you.

Figure 19: Optical mice

While most mice are managed by hand, others are not; trials have been conducted using a gadget known as the footmouse. It is a foot-operated device, as the name suggests, albeit it resembles an isometric joystick more than a mouse. Foot pressure applied to one side or the other of a pad moves the pointer. As a result, one can devote both hands to the keyboard. The footmouse is an uncommon gadget that has not gained widespread use!

It's interesting to note that foot pedals are widely utilised in mechanical equipment like automobiles, cranes, sewing machines and industrial controls, as well as in musical instruments like pianos, electric guitars, organs and drums. Thus, it is evident that this is

© Amity University Online

Notes

a fine idea in theory. With the exception of simulators and games, two factors appear to have restricted their employment in computer equipment. One is the usefulness of foot controls in the workplace: foot pedals on laptops would be very inconvenient and pedals beneath desks could be mistakenly pressed. The type of control being used is the second concern. In physical interfaces, one or more single-dimensional analogue controllers are often operated by pedals. It's possible that foot-operated controls could be more widely and successfully employed in more specialised interfaces.

Touchpad

Touchpads are generally 1-3 inches (50-75 mm) square, touch-sensitive tablets. They can be purchased separately to replace the desktop mouse and were first widely utilised in Apple Powerbook portable computers. They are currently found in many other notebook computers as well. Similar to a simulated trackball, they are operated by stroking a finger over its surface. Although it feels very different from other input devices, users rapidly become skilled with it as with all devices.

The cursor may need to be moved around the screen multiple times due to their small size. The software's acceleration settings, which connect the trackpad's movement to the screen's movement, can help with this. This changes according to movement speed instead of having a constant pad-to-screen ratio. Small distances on the screen correspond to the pad's movements if the finger goes slowly across it. The touchpad's cursor travels a considerable distance when a finger moves swiftly in the same direction. For instance, when writing this part on the trackpad being used, a very slow finger movement from side to side moves the cursor less than 10% of the screen's width. On the other hand, the cursor moves across the entire screen if the finger is moved very quickly from side to side.

1.2.2 Devices: Part 2

Joystick and Keyboard Nipple

As an indirect input device, the joystick is small in size. The joystick is a basic gadget that consists of a small palm-sized box with a stick or shaped grip extending out of it. Movements made with the stick move the screen pointer in similar ways. Joysticks come in two varieties: absolute and isometric. Since the location of the joystick in the base correlates to the position of the pointer on the screen, mobility is the most essential feature of the absolute joystick. When the pressure on the isometric joystick is released, it returns to its normal upright, centre position. The pressure on the stick in this system correlates to the velocity of the pointer. For obvious reasons, this kind of joystick is also known as the velocity-controlled joystick. Typically, the buttons are positioned on top of the stick or in front, resembling a trigger. Joysticks are often used in computer games because they are reasonably durable and reasonably priced. Their similarity to aeroplane joysticks and users' experience with them is another factor contributing to their dominance in the games business. Since planes are a popular theme for games, players are accustomed to using these joysticks for more obscure forms of entertainment.

Many laptop computers employ a smaller device that shares the same fundamental features to control the pointer. A single-key version known as the "keymouse" was used in some older systems. More frequently, a little rubber nipple that protrude into the keyboard's centre serves as a miniature isometric joystick. Although it is typically challenging for beginners to use, this appears to be connected to fine-tuning the speed settings. Similar to a joystick, the nipple regulates the speed at which objects travel across the screen, making it less precise than a mouse or stylus.

(c) Amity University Online

Notes

Touch-sensitive Screens (touchscreens)

Another way for the user to point and pick items on the screen is through a touchscreen. Unlike a mouse, however, touchscreens detect the user's finger or stylus directly on the screen, making them far more direct. They function in a variety of ways, including as through ultrasonic reflections, capacitance changes on a grid overlaying the screen, or the interruption of a matrix of light beams by the finger (or stylus). This is a direct device since there is no need for mapping because the user points at the exact object that is needed.

Because of its speed, the touchscreen doesn't need a specific pointing device. It works particularly well for choosing items from menus that are shown on the screen. Touchscreens are appropriate for usage in harsh conditions since the screen serves as both an input and an output device, eliminating the possibility of separate hardware being harmed by dirt. They have been effectively employed as the general public's interface to information systems and are also rather easy to use.

They do, however, have a number of shortcomings. Since the finger is a somewhat blunt instrument and might leave greasy stains on the screen, it is not always the best choice for pointing. This implies that correct sketching and the selection of small regions are highly challenging. In addition, reaching up to point at a vertical screen requires lifting one's arm, which may be quite taxing. Additionally, the screen must be roughly a metre away from the user in order to be viewed, which may be too close for comfort. Studies have indicated that a touchscreen works best when angled approximately 15 degrees upward from the horizontal.

Stylus and Light Pen

In order to prevent greasy screens and achieve more precise placement, touch-sensitive systems frequently come with a stylus. A little plastic stick that resembles a pen is used to point and draw on the screen as an alternative to simply pointing at it. Though some laptop computers also employ them, PDAs are where this is most common. The light pen is an older piece of technology that is employed in the same manner. When the pen is held up to the screen during a display scan, it senses a flash of light coming from the phosphor on the screen thanks to a cable connecting it to the screen. Because of its ability to address specific pixels, the light pen is far more accurate than a touchscreen.

For precise selecting and sketching, both a stylus and a light pen can be utilised; however, using both simultaneously with a keyboard can be cumbersome and tiresome when using upright screens. It's interesting to note that some PDA users with fold-out keyboards figure out how to text while holding the stylus by holding it outward between their fingers. A closed pen can be utilised in an emergency, but an unconnected stylus is readily lost.

In that there is an instantaneous interaction between the gadget and the item picked, styluses, light pens and touchscreens are all highly direct. On the other hand, the movements of the mouse, touchpad, joystick and trackball must all be mapped to the movement of the cursor on the screen.

Nevertheless, the direct devices have the drawback that when in use, pointing actually obscures the display, making it more difficult to use, particularly when quick sequence of intricate selections or actions is needed. This implies that the location of the user's hand must be considered while designing screens. For instance, you might wish to position menus lower on the screen than higher. Additionally, you might wish to provide left- and right-handed users with alternate layout options.

© Amity University Online

Notes

Digitising Tablet

The digitising tablet is a more specialised tool that can be used as a mouse replacement but is primarily used for freehand drawing. Certain extremely precise tablets are employed in specialised applications, such digitising data for maps and typically use a puck, a device that resembles a mouse.

The tablet can function in several ways to deliver positional information by measuring a device's position on a dedicated pad or tablet. Point contact between two separated conducting sheets is detected by the resistive tablet. It has the benefit of not requiring a specific stylus to function; a pen or the user's finger will do just fine. A tiny loop coil contained in a unique pen is used by the magnetic tablet to identify current pulses in a magnetic field. Similar tablets also function as capacitative and electrostatic ones. Similar to the above, the sonic tablet doesn't need a specific surface. A specialised pen emits an ultrasonic pulse, which is picked up by two or more microphones, which triangulates the pen's position. If necessary, this device can be configured to accept 3D input.

Digitising tablets come in various sizes and can achieve high resolutions. The resolution of cursor movement is affected by sample rates, which increase with increasing sampling rates. Since there is a mapping from the tablet's operating plane to the screen, the digitising tablet is an indirect device that can detect both relative and absolute motion. It can also be used for text input and handwriting can be read if character recognition software is available. The drawbacks of digitising tablets are that they take up a lot of desk space and can be difficult to operate if the keyboard is pushed to one side.

Eyegaze

With eyegaze systems, all you have to do is glance at the computer to operate it! Certain systems need you to wear special glasses or wear a tiny box mounted on your head; others are integrated into screens or appear as a small box beneath screens. Retinal reflection occurs when a low-power laser is directed into the eye. The eyegaze system tracks the reflected beam to ascertain the direction of gaze because the reflection varies with changes in the eye's angle. The system may be used to move the screen pointer and for other more specialised tasks once it has been calibrated, which is usually done by staring at a series of dots on the screen. Although Eyegaze is a highly quick and accurate instrument, the more precise models can be very pricey. Given that the eye does not follow straight lines, it works well for choosing but not for drawing. It can be challenging to discern between inadvertently glancing at something and purposefully staring at it in real-world situations.

Although these systems have been employed in the military, most notably to guide air-to-air missiles to their targets, they are beginning to be used in more peaceful contexts, such as for workers in locations where using their hands is impractical or for impaired users. Because of its cost and uniqueness, the eyegaze is not widely available and is typically limited to specialised applications. Because it is possible to track the exact location of the user's gaze, it is especially helpful in HCI evaluation. Eyegaze may find greater uses as costs come down and the technology is less obtrusive, particularly in the fields of augmented and virtual reality.

Display Devices

Without a display panel, the great majority of interactive computer systems would be unimaginable; yet, there are plenty of such systems, albeit typically in specialised applications only. Thinking beyond the box, devices like security alarms, vehicles and hi-fis all produce different outputs than what can be displayed on a screen, but displays are widely used in workstation and personal computer markets.

(c) Amity University Online

Notes

Let's take a closer look at the characteristics of bitmap screens, various screen technologies, enormous, strategically placed displays and a recently developed technology called "digital paper" as we go over the basic computer display.

Bitmap Displays – resolution and color

Bitmaps are the basis for almost all computer displays. That is, a rectangular grid of many coloured dots, or pixels, makes up the display. These pixels could only be available in grayscale, full colour, or black and white (like the tiny display on a lot of TV remote controls).

The video card in the computer stores the color—or, in the case of monochrome screens, the intensity—at each pixel. The name "bitmap" originates from screens that are limited to black and white, as each pixel can only retain information that is either on or off. There are more colour or intensity options when there are more bits per pixel. For instance, a pixel with 8 bits can have up to 256 different colours at any given time. The collection of colours is referred to as the colormap, which can be changed at any moment to generate a new set of colours. As a result, the system can actually display many more colours than those found in the colormap—just not all at once. Currently, the majority of desktop computers employ 24 or 32 bits per pixel, enabling nearly infinite colour possibilities; nevertheless, gadgets like PDAs and cell phones are frequently still monochromatic or have a restricted colour gamut.

The resolution of the screen is another crucial metric, in addition to the number of colours that may be shown at each pixel. In actuality, the term "resolution" is used to screens in a perplexing (and confusing!) way. Two figures need to be taken into account:

- The total amount of pixels is always at a 4:3 ratio on typical computer displays, meaning that it may be 1024 pixels wide by 768 pixels deep, or 1600 × 1200; on PDAs, however, this will likely be closer to a few hundred pixels in each direction.
- Pixel density is expressed in terms of pixels per inch. This density, in contrast to printers, barely changes between 72 and 96 pixels per inch.

A monitor, liquid crystal display (LCD) screen, or other display device may quote its highest resolution, but the computer may actually provide a lower resolution, which further complicates matters. For instance, the computer may only deliver 800 x 600 pixels to the screen, even though it has a resolution of 1200 x 900 with 96 pixels per inch. This usually means that the image is stretched across the screen surface of a cathode ray tube (CRT), resulting in a reduced pixel density of 64 per inch. Since an LCD screen cannot alter its pixel density, it will always have 96 pixels per inch but will not fill the entire screen; a black border will be added in its place. Certain LCD projectors will attempt to enlarge or decrease the number of pixels they have, which could result in one pixel being stretched to two or two pixels being "squashed" into one. This could cause display "artefacts" such uniform lines becoming alternatingly thick or thin or thin lines disappearing.

While lines at 45 degrees and horizontal and vertical orientations can be rendered precisely on bitmap displays, curves and lines at other angles exhibit "jaggies," or uneven borders, which are the result of attempting to approximate the line with pixels. There will inevitably be jaggies when utilising a single colour. Bitmap fonts exhibit similar effects. Using anti-aliasing techniques or high-resolution screens can help lessen the jaggies issue. By softening the edges of line segments, anti-aliasing blurs the discontinuity and lessens the visibility of jaggies.

Examine the two pictures in the figure below with slightly swollen eyes. Note the improved appearance of the second anti-aliased line. Though the screen resolution is obviously much higher, the same idea applies. This is possible because our brains are

Notes

always "improving" what we see by interpreting and transforming the unprocessed input from our eyes' rods and cones into something meaningful. Our eyesight is frequently blurry due to inadequate lighting, out-of-focus objects, or visual abnormalities. Our brain adjusts and straightens out hazy visuals. Anti-aliasing intentionally blurs the image to cause our brains to go through this processing, giving the impression that we are looking at an angled, smooth line.

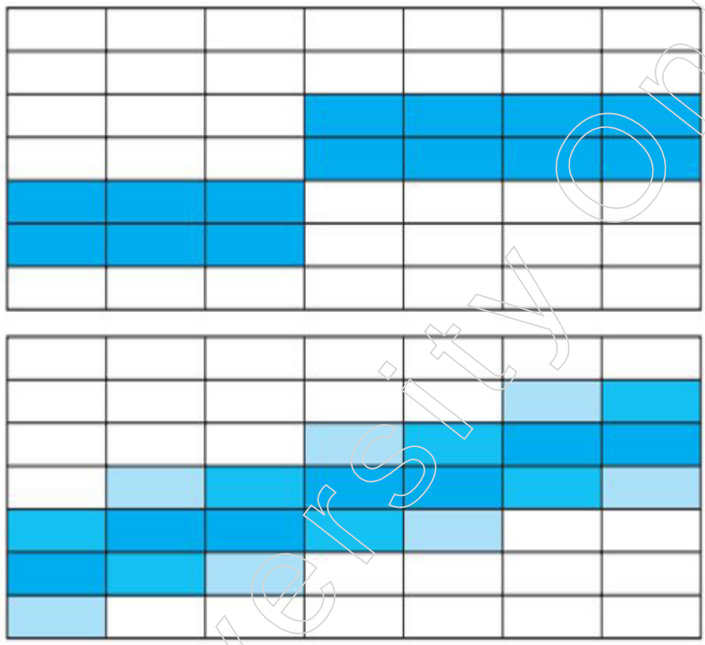


Figure 20: Magnified anti-aliased lines

Technologies

Cathode ray tube

Flat LCD screens are quickly replacing cathode ray tubes, which are still the most popular type of computer screen that resemble televisions as we write this. It functions much like a typical television screen. An electron cannon emits a stream of electrons that are focused and guided by magnetic fields. The phosphor is stimulated by the electrons as the beam strikes the phosphor-coated screen, causing it to glow (see Figure below). The electron beam is first scanned from top to bottom, then from left to right and finally it is flicked back to scan the following line. Approximately 30 Hz, or 30 times per second, is repeated for each frame; but, in certain cases, greater scan rates are employed to lessen screen flicker. Using interlacing, which scans all of the odd lines on the screen first and then the even lines, is another technique for eliminating flicker. Using a high-persistence phosphor, which shines longer when activated, also lessens flicker, but it also smeared the image, particularly in cases when there is a lot of motion.

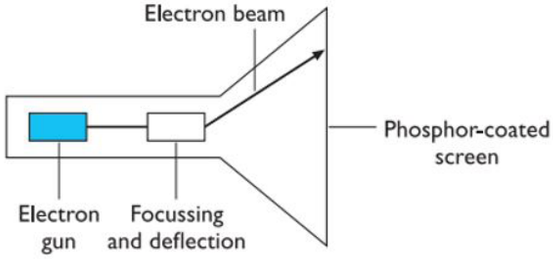


Figure 21: CRT screen

(c) Amity University Online

Notes

Grayscale can be shown on black and white screens by adjusting the electron beam's intensity; colour is produced through more involved techniques. To target the red, green and blue phosphors, three electron cannons are employed, one for each. When all of these colours are combined, numerous other colours can be created, including white. A shadow mask is used to focus these three phosphor dots into a single point, which is inaccurate and results in a lower resolution for colour screens than for comparable monochrome screens.

One more method for generating colour on the screen is to employ beam penetration. A unique phosphor changes colour in response to the intensity of the beam striking it.

The CRT is a low-cost display technology with great colour capability and quick enough response times for quick animation. It should be noted that animation refers more broadly to the use of motion in displays, including moving the pointer, opening windows, signalling processor-intensive calculations and other actions. It does not always imply tiny animals and humans running around on the screen. The price does, however, increase with screen resolution. CRTs are somewhat large due to the electron gun and focusing components behind the screen; however, more recent developments have produced flatter displays where the electron gun is positioned to fire parallel to the screen plane, bending the beam 90 degrees to strike the screen instead of directly at it.

Liquid Crystal Display

You have probably encountered the thin, flat plastic screens if you have ever used a notebook computer or personal organiser. Compared to conventional CRTs, these liquid crystal displays are smaller, lighterand require a lot less electricity. Another common name for these is flat-panel screens. They are also matrix addressable, meaning that individual pixels can be accessed without scanning and they don't have any known radiation issues.

A thin layer of liquid crystal is positioned between two glass plates in a design that is conceptually similar to the digital watch. The bottom plate is reflective and the top plate is transparent and polarised. Light from the outside is polarised, or it oscillates exclusively in one direction, after passing through the top plate. The cell appears white as a result of this passing through the crystal, reflecting off the bottom plate and returning to the eye. The crystal twists when a voltage is supplied to it through the conducting glass plates. As a result, it rotates the incoming light's plane of polarisation, preventing it from passing through the top plate and giving the active cell a black appearance. The LCD needs to be refreshed at the regular intervals, however flicker is typically undetectable because to the crystal's comparatively slow reaction. Because of their lower light intensity and less flicker than traditional CRT screens, LCDs are easier on the eyes and need less effort to operate.

It is now usual practice to replace a desktop computer's regular screen with one that uses this alternative technology. But because of their unique qualities—compactness, light weight and low power consumption—these screens have monopolised the market for notebook and portable computer systems, carving out a sizable niche for themselves.

The introduction of these screens opened up a sizable hitherto untapped market by enabling the construction of lightweight, compact computers. Riding the wave of technical advancement, these computers have allowed many people to work in new ways, giving them access to computers whether they are at home or away from the workplace. Once again, we can see that variations in hardware can significantly impact human–computer interaction. For example, working in a different place on a smaller system with different software clearly symbolises a different type of engagement.

The market for notebook computers was expanding, which encouraged investments in the development of LCD screen technology. Supertwisted crystals significantly increased

(c) Amity University Online

Notes

the viewing angle. LCD panels are increasingly seen in household televisions and even in personal DVD players due to improvements in response times.

Large Displays and Situated Displays

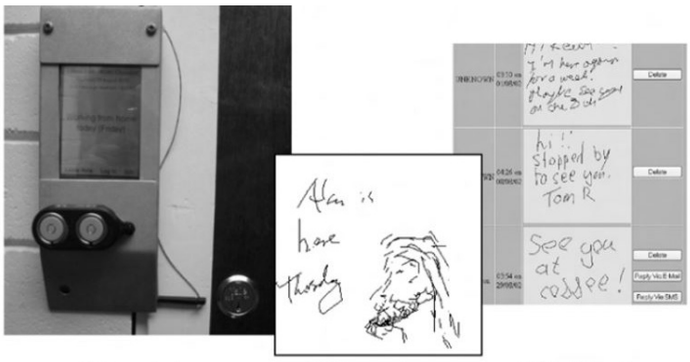
These days, displays are more than simply what you find on your laptop or desktop. A huge screen monitor comes in various varieties. Some make enormous flat bitmap screens using gas plasma technology. These function exactly like a regular screen, with the exception that they are larger and typically use the wide screen format of HDTV (high definition television), which has a 16:9 aspect ratio as opposed to the 4:3 of conventional TVs and monitors.

A video wall can be created by arranging multiple smaller displays, either CRT or LCD, together when very large screen areas are needed. These can show different images, or a single image from a computer or TV can be divided into multiple screens, each showing a fraction of the entire, creating a massive image as a result of software or technology. This method is frequently utilised in big concerts to show off the performers or video clips while they perform.

The enormous display that you have probably seen is probably a projector of some kind. These come in two varieties. Large red, green and blue projectors are common in very large lecture halls, particularly the older ones. To create a full-color image, each of them scans light across the screen. LCD projectors are commonly found in smaller lecture halls and intimate gathering spaces. These are similar to regular slide projectors, usually the size of a large book, except instead of a slide, there is a tiny LCD screen. After passing through the little screen, the lens focuses the projector's light onto the screen.

Projected displays have the drawback of frequently having the presenter's shadow move across the screen. In fixed lecture halls, back projection can occasionally be used to prevent this. A projector in a little space behind the lecture hall's screen projects a picture that is left-to-right flipped. The lecture hall can see the image projected on the rear thanks to the semi-frosted glass screen. They are less common than front projection because the projector's ability to handle a broad angle without distortion restricts the size of the picture, which is determined by the depth of the projection room behind.

Display screens can be utilised in a variety of public settings, not just for lectures and meetings, to provide information, connect spaces, or serve as message areas. Because they get their meaning from the context in which they are placed, these are frequently referred to as contextual displays. These could be very small or enormous screens that multiple people are supposed to view or interact with at the same time. An example of a small experimental display that is placed next to an office door and functions as an electronic sticky note is shown in the figure below.



(i) Door display (ii) Occupant's note (iii) Web interface

Figure 22: Situated door display

Notes

Digital Paper

The many types of digital paper are a very new and developing sort of "display." These are thin, flexible materials with electronic writing capabilities, similar to those seen on computer screens, but with the ability to retain content even when disconnected from an electrical source.

For this, several technologies are being researched. One entails covering the entire surface with microscopic black and white spheres. Each tiny sphere can be made to rotate from black to white thanks to electronics included into the substance. The ball maintains its original orientation after the electronic signal is cut off. Another method uses small tubes arranged side by side. Each tube contains a tiny reflecting spherical and a liquid that absorbs light. The sphere can be manipulated to shift towards or away from the top surface, turning a pixel white or black. Once the electronic signal is cut off, the sphere once more remains in its final location.

These will most likely be used for large, reprogrammable or slowly animating banners initially. This is the perfect application because it doesn't need tiny pixels or updates too frequently. The goal is to have programmable paper sheets that you can connect to your computer and receive a "soft" printout that you can modify later on as technology advances. Maybe in the future, you'll be able to have a "soft" book that looks and feels just like a real one, complete with turnable, skimmable pages, but whose cover and contents can be altered whenever you download a new book from the internet!

1.2.3 Devices: Part 3

Devices for Virtual Reality and 3D Interaction

You have to move around and engage with them in a three-dimensional environment. These can sometimes be used using standard desktop computer controls and displays, but they can also be operated with specialised hardware that allows you to see a 3D world in addition to moving and interacting with 3D objects.

Positioning in 3D space

3D virtual worlds are displayed by virtual reality devices. It is up to the users to manoeuvre through these areas and work with the virtual objects they encounter. Navigating involves more than just going to a specific place; it also involves selecting an orientation. Furthermore, when you grasp an object in actual space, you can twist and turn it in addition to moving it, like in the case of opening a door. As a result, when switching from mouse to 3D devices, two to six degrees of freedom—rather than just three—are typically involved.

Cockpit and Virtual Controls

Pilots of aircraft and helicopters are already required to manoeuvre in real space. Controls based on an aeroplane cockpit are used in many arcade games and more serious applications to allow users to "fly" across virtual space. It takes a lot of experience for people to be able to operate comfortably in such settings, though, because helicopter pilots are extremely proficient.

The controls are virtual in many PC games and desktop virtual reality (where the output is shown on a regular computer screen). This might be a more straightforward set of up/down, left/right buttons, or a simulation of the cockpit controls. These virtual controls are operated by the user with a standard mouse (or other 2D device). Keep in mind that this indicates two different levels of indirection. People's ability to not only learn how to utilise

(c) Amity University Online

Notes

such systems quickly, but also become expert in them is a testament to the adaptability of the human mind.

The 3D Mouse

Numerous gadgets function as three-dimensional equivalents of mice. The mouse can be rotated, tipped forward and backward and picked up in addition to being able to move it in three dimensions on a surface. With the ability to track its position (three degrees), as well as its up/down angle (called pitch), left/right orientation (called yaw) and degree of twisting about its own axis (called roll), the 3D mouse boasts a full six degrees of freedom (see Figure below). The position and orientation of the mouse are tracked by a variety of sensors, including mechanical joints where the mouse is mounted, which resemble an angle-poise light, magnetic coils and ultrasonic sensors.

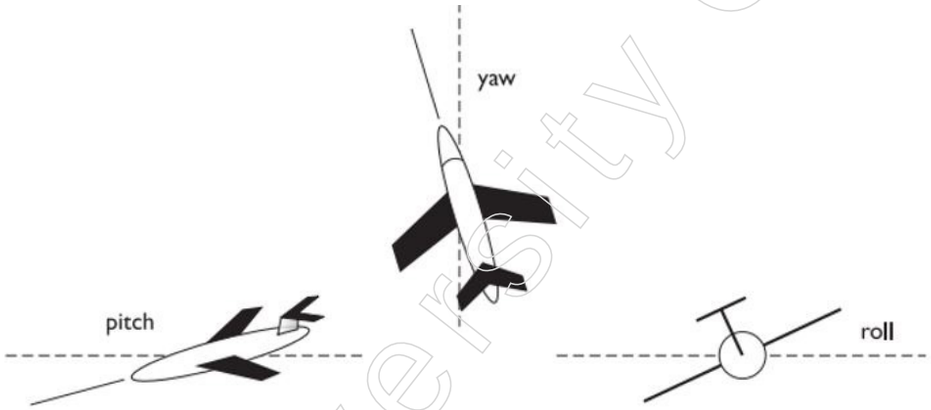


Figure 23: Pitch, yaw and roll

Holding the mouse in the air for extended periods of time might cause strain for users of the 3D mouse and most other 3D positioning devices. In the virtual world, putting down the 3D mouse could even be viewed as a negative action.

Dataglove

A key component of expensive virtual reality systems, the dataglove is a 3D input device. It measures the joint angles of the thumb and fingers by placing optical fibres along the fingers of a lycra glove. The fibre optic cable bends when the fingers do; a greater bend results in more light leaking from the fibre; the glove detects this decrease in intensity, which is correlated with the joint's degree of bend. Two ultrasonic sensors that detect 3D positional data and the angle of roll—that is, the degree of wrist rotation—are fastened to the top of the glove. Since most applications in use do not require such a thorough type of data input and those that do cannot afford it, such rich multi-dimensional input is now a solution in search of a problem. On the other hand, the availability of less expensive variants of the dataglove will stimulate the creation of more sophisticated systems that can fully utilise the dataglove as an input device. This technology has a lot of potential applications to help the disabled, but right now the cost is still the barrier.

One of the advantages of the dataglove is its ease of use, coupled with its potential for great power and expressiveness (it can deliver 50 times a second for 10 joint angles, 3D spatial information and wrist rotation degree). Its drawbacks include its exorbitant cost and its challenging usage when combined with a keyboard. This restriction, though, is ill-conceived. Consider a keyboard projected onto a desk, with software that recognises hand movements and determines whether or not virtual keys have been pressed. The dataglove has enormous promise; research is now being focused on two obvious areas, gesture detection and sign language interpretation, but less obvious uses are constantly emerging.

(c) Amity University Online

Notes

Virtual Reality Helmets

Some VR systems employ helmets or goggles that serve two functions: (i) showing the user's environment in three dimensions to each eye and (ii) tracking the user's head position. When we talk about output devices later on, we shall talk about the former. The output side is the main use for the head tracking. The user should be able to see various areas of the scene as their head moves around. However, some systems (much like the eyegaze systems) also use the user's head orientation to identify which items to manipulate and even the direction of movement inside the space. This is comparable to leading a horse in reverse. Using the reins, you can direct a horse's head in the direction you want it to go and the horse will follow your lead.

Whole-body Tracking

Aiming to give consumers the impression that they are actually in the virtual environment, several VR systems try to be immersive. In the real world, walking without looking where you're going is conceivable, however generally not a smart idea. You do not want your car to make an abrupt 90-degree turn if you are driving and you peek at something on the side of the road! For this reason, some VR systems try to track various body movements. A motorbike body that allows you to lean into curves is a feature of certain arcade games. Even stranger, tiny trampolines have been configured such that users can place weight on various areas of the trampoline to control their movement in a virtual environment. It is possible for the user to navigate through virtual space. At the extreme, image-processing methods or gadgets akin to the dataglove could be used to track the movement of the entire body. In the latter, each joint's location can be mapped by placing white dots at different sites on the user's body and tracking their positions using two or more cameras. While the final option may seem a bit limiting for those who value style, it does indicate the direction towards less invasive tracking methods.

3D Displays

Similar to how new types of input devices have been made possible by the 3D graphics utilised in virtual reality, more advanced outputs are also needed. Desktop virtual reality is displayed on a typical computer screen and effects like perspective, occlusion and shadows are used to create a 3D impression. This can work quite well and if your browser supports VRML (virtual reality markup language), you can even see 3D graphics online.

Seeing in 3D

To detect depth in the real environment, our eyes rely on a variety of cues. As each eye only sees a flattened form of the world, akin to a snapshot, it is actually pretty astonishing. Stereo vision, sometimes known as stereoscopic vision, is one significant result.

Our brain can determine an object's relative distance from another by using the slightly varied images that each eye sees since it is viewing the object from a different angle. This stereoscopic effect is not present in desktop VR. Still, there are a number of gadgets that can provide real stereoscopic views.

Any stereoscopic gadget begins with the creation of images from various viewpoints. This just entails figuring out the correct locations and angles that correspond to the normal distance between eyes on a human face, as the computer is already creating images for the virtual world. The user will be shown a perspective of the world through the eyes of a giant or a gnat if this distance is too much from the natural one!

Then, several methods are applied to guarantee that every eye sees the correct image. One way is to equip a set of goggles with two tiny screens. Each eye is then shown

© Amity University Online

Notes

a distinct image. The common perception of virtual reality (VR) is of a person wearing a helmet with what appear to be inverted binoculars protruding in front of them. These devices are still very bulky. Nonetheless, the gadgets may now be made to resemble regular spectacles in terms of weight and size because to the development of lighter and smaller LCDs.

An alternate approach is to use timed electrical signals to blank out each eye by connecting a pair of special glasses. Each eye sees a different image if this is synchronised with the frame rate of a computer monitor. Comparable methods employ polarised filters in front of the screen and glasses with various polarisations of the lenses. Both of these solutions work well by employing strategies that are comparable to the red-green 3D glasses that come free with some morning cereals. In fact, extensive 3D television broadcasts have tested the use of these red-green eyewear. Nonetheless, the 3D image quality obtained with the polarised and blanked eyeglasses is significantly superior.

The optimal scenario would be to be able to view 3D visuals on a specialised screen in the same way as a hologram or 3D television, as seen in the greatest science fiction films! However, there isn't yet a workable answer for this. One technique is to etch hundreds of prisms—small, vertical grooves—into the screen. The screen then appears to each eye as a series of alternating dots, providing for a stereo image with half the typical horizontal resolution. These screens aren't yet suitable for family watching, though, because of their extremely limited viewing angles.

Obtaining stereo images is not the complete story, in actuality. Not only do our eyes see differently, but each one also focuses on the object of interest at hand due to the tiny muscles in the eye's pupil changing its shape. The images that are displayed to the eye are produced with a fixed focus and frequently have an almost unlimited depth of field. This can be tiresome and confusing. Recent advancements have been made in the use of lasers to measure each eye's focus depth and modify visuals accordingly, much like eye tracking technology. It isn't currently used very much, though.

Simulators and VR Caves

Some virtual reality systems operate by immersing the user in an area with the virtual world displayed upon it, due to the challenges associated in delivering a complete 3D environment via head-mounted displays. Large flight simulators are the most prominent instances of this; inside, you enter a model of an aeroplane cockpit and virtual windows display the scenes that would be visible through the windows. Large screens are arranged such that the majority of your visual field is filled when using a motorcycle or ski simulator in a video arcade. While you are playing the game, your pals are still visible to you over your shoulder, but they are all around you.

Physical Controls, Sensors and Special Devices

In applications that use computers today, such interactive TV, in-car navigation systems, or personal entertainment, the mouse, keyboard and screen of the traditional computer system are no longer relevant or viable. These devices might sense your biosigns or the environment, have unique controls, use sound, touch and smell in addition to visual displays and have special displays.

Special Displays

In sophisticated systems, especially embedded systems, a variety of visual outputs are used in addition to the CRT screen. They can be analogue numerical value representations in the form of lights, gauges, or dials that indicate a particular system condition. Process

(c) Amity University Online

Notes

control systems use gauges and dials, while some computers include flashing light-emitting diodes (LEDs) on the rear to indicate the processor condition. Once you are thinking in this way, you can consider many different visual outputs that have nothing to do with the screen. The head-up display used in aeroplanes is one example of a visual display that has found a specific application. The pilot can't seem to look around the cockpit for information since he is too preoccupied staring forward. There are numerous things that must be understood, including information from aircraft status indicators, navigational data and tactical system data. A portion of this data is projected into the pilot's field of vision via the head-up display, placing it right in front of her. This facilitates the pilot's work by eliminating the need to examine massive information banks while paying little attention to external events. In order to prevent cluttering the head-up display, less critical information is typically displayed on fewer dials and gauges in the cockpit and they can be checked less frequently, at less stressful moments.

Sound Output

The output modality of auditory signals is another one to take into account. Auditory outputs, which are frequently intended to be used in tandem with screen displays, are not well understood since we do not yet know how to employ sound sensibly to achieve maximum impact and information transference. Sounds provide an essential degree of feedback in interactive systems in addition to expressing system output. One way to speed up interactive performance is to program keyboards to click every time a key is pushed. When a key is pressed on a telephone keypad, various tones are typically audible; one tone indicates that the key has been successfully pressed, while the other tone gives information specific to the pressed key. An easy-to-understand example of a gadget that benefits from audio feedback is a doorbell. We are left indecisive if we press it and get no response. If we didn't push it correctly the first time, should we try again, or did it ring but we didn't hear it? And will the residents of the house think we are being extremely impolite if we push it once more and it does ring? We experience discomfort and mild tension. In the same situation, we wouldn't enjoy interacting with a computer system as much and wouldn't perform as well if it were used in place of a doorbell. But, it's a straightforward issue that could be resolved with ease with a stronger original design that made use of sound.

Touch, Feel and Smell

Normal computer applications use fewer of our other senses, but you may have played computer games where the joystick or virtual steering wheel vibrated—possibly to indicate that a car was going to go off the track. In certain virtual reality applications, including the use of VR to 'practice' surgical operations in the medical field, the sensation of an instrument passing through various tissue types is crucial. Because these operations are simulated by devices with force feedback, the resistance provided varies according to the virtual operation's state. Haptic devices are these different types of force, resistance and roughness that affect our physical sensations.

Haptic devices are employed in specialised interfaces in the real world in addition to virtual ones. Electronic braille displays can use tiny vibrating pins or pins that rise or fall to create various patterns. In-car controls have been designed with force feedback in mind.

Actually, the vehicle provides a superb illustration of the effectiveness of tactile feedback. A little bump in the road causes the car to veer slightly off course, but you're probably going to notice it and correct yourself before you realise it. Your body has reflexes that cause your limbs to shift out of the way when you come into contact with anything unexpected or to slightly resist pressure in order to maintain them where you "want" them. Your lower brain is where these reactions happen and they happen quickly without

© Amity University Online

Notes

conscious thought. Haptic devices can therefore obtain very quick answers, although they are not entirely regulated. Effective use of this in design is possible, but caution is still advised.

Due of its reliance on minute variations between adjacent skin spots, texture is more challenging to manipulate. Furthermore, we typically feel textures when we move our fingers over a surface rather than just when we rest on it because most of our senses pick up on change rather than static stimuli. The necessary technology is only now starting to become accessible.

There is proof that one of the most potent memory signals is smell. Scents are used by a number of historical recreations, like York, England's Jorvik Centre, to evoke a sense of immersion in their static displays of bygone eras. Certain arcade games can produce odours; for example, you might smell burnt rubber when your racing car skids on the course. Both of these instances make use of a consistent scent in a specific area. Many attempts have been made to construct gadgets that would enable odours to be dynamically produced in response to online games or even websites. The technological challenge lies in the fact that our nostrils are made up of hundreds of distinct types of receptors that react to various chemicals in the air, rather than a limited range of basic aromas that are blended together (like salt, sweet, sour, bitter and savoury on our tongues). Similar to a printer cartridge with hundreds of ink colours, most scent-generating devices consist of a vast repertoire of tiny scent-containing capsules that are released in different amounts as needed. Although there doesn't seem to be a large market for these gadgets just yet, specialised niches could someday give rise to them.

Smell is a multifaceted sense that is intricate and has a unique capacity to elicit memories that cannot be quickly altered. These attributes could be useful in situations when having a broad sense of awareness and whereabouts is preferred. An investigation conducted at the Massachusetts Institute of Technology, for instance, looked into the application of a tiny battery of smell generators, which might be especially useful for background awareness and ambient displays.

Paper: Printing and Scanning

The paperless office was a common information technology motif a few years ago. Documents would be created, sent, read and filed online in the paperless office. Electronic data would only be transferred to paper when it left the office and was received by regular clients or by other businesses that were trailing behind in technology. The realisation that a wastepaper basket's floor space may cost hundreds of dollars in rent annually, coupled with the skyrocketing cost of real estate, gave rise to this notion. After a few years, most conventional paper files are now digital, but the dream of an entirely paperless workplace is waning. Wastepaper baskets are still found in offices and the unique computer tables that hold 14-inch colour monitors require additional floor space.

Printing

In fact, creating paper documents has become simpler thanks to computer technologies. Creating many copies of a letter or book to ensure it looks "just right" is a simple task. A printhead in older printers could only hold a limited number of characters. These included daisy-wheel and golf-ball printers in addition to the conventional line printer. Changing the typeface or size of text required replacing the printhead, which was an inconvenient and often messy task. However, for a long time, the daisy-wheel printer was the only way to provide excellent results at a reasonable cost. Daisy-wheels, however, are quickly becoming rare due to the decline in laser printer prices and the availability of more affordable, high-quality printers.

(c) Amity University Online

Notes

Every common printing technology, including screens, creates the picture on the paper as a collection of dots. This allows any character set or design to be printed, theoretically, with the dots' resolution serving as the only restriction. In terms of resolution, dots per inch (dpi) is used. Consider using a graph paper sheet and creating an image by connecting the dots at each line's crossing. The resolution in dpi is the number of lines per inch in each direction. This may cause some confusion for certain mechanical printers: the printed dots could be larger than the gaps and nearby print heads might not be able to print at the same time or might be shifted in relation to one another (forming a diamond-shaped grid as opposed to a rectangular one). Although the user is not greatly affected by these variances, they do indicate that, when two printers are used at the same nominal resolution, one printer's output seems better than the other because it has more skillfully handled the physical limits.

Dot-matrix, ink-jet and laser printers are the three most popular kind of dot-based printers. The resolution and quality of these are approximately arranged as follows: dot-matrix printers usually have a resolution of 80–120 dpi, which increases to roughly 300–600 dpi for ink-jet printers and 600–2400 dpi for laser printers. Digital photos can be printed in photo quality using ink-jet printers by adjusting the amount of ink and paper quality.

Printers differ in terms of resolution as well as price and speed. Office-grade inkjet or laser printers can print four to eight pages per minute on average. Characters per second (cps) is a common unit of measurement for dot-matrix printers; typical rates can be 200 cps for draft and 50 cps for letter-quality output. This actually translates to about one page each minute. These are the fastest that printers can go for plain text; for graphics, they can go much slower.

Even monochrome laser printers are significantly more expensive than colour ink-jet printers. But the ongoing expenses for consumables could easily outweigh this one-time expense. High-quality paper must be used with both jet and laser printers, which require replacement of their special-purpose parts (print cartridges, toner and print drums) every few thousand sheets. It could be more challenging to locate recycled paper in grades that work with laser printers.

Fonts and Page Description Languages

Certain printers have the ability to operate in a mode that prints characters in a single font, a la typewriter. Another scenario—which is theoretically straightforward—is when you wish to print a bitmap image. There is no need for additional interpretation because the printer receives the dots to print. In actuality, though, it is rarely that easy.

Numerous printed documents are significantly more intricate; they frequently include text that is bolded, italicised, underlined and composed in a variety of fonts and sizes. Line drawings, digital photos andimages created with "paint" software—including the all-pervasive "clip art"—are all included in the text. At times, the computer handles all the work, creating the appropriate bitmap to send to the printer from the page image. As an alternative, the printer can get a description of the page. At the most basic level, this will involve commands to adjust the font size and print position on the page.

PostScript is the most widely used page description language that can be supported by more advanced printers. This is a printing-oriented version of programming language. Along with some unique ones, it contains some common programming elements like scaled bitmaps, complex character and font handling and pathways for drawing lines and curves. The theory is that a page's description is much shorter than its corresponding bitmap, which cuts down on how long it takes to transmit the page to print. An A4 laser printer page at 300 dpi requires 8 Mbytes in bitmap format; it would take 10 minutes to deliver this along a

(c) Amity University Online

Notes

typical serial printer wire! Even though it usually takes less than ten minutes, the printer's computer must understand the PostScript program in order to print the page, which is a barrier for a lot of print jobs.

A specific font with size and shape is used while printing text. Points (pt) are used to measure text size. The point, which is approximately 1/72 of an inch, is a printer's measure. A font's height and point size are correlated: a 12-point font has roughly six lines per inch. A font's name, such as Times Roman, Courier, or Helvetica, indicates its shape. The Courier font has a typewritten shape, while the Times Roman font is similar to the type found in many newspapers, including The Times.

Courier is a fixed-pitch font
Times Roman is a variable-pitch serif font
Minion is also a variable-pitch serif font
Gill Sans is a variable-pitch sans-serif font
A mathematics font: αβξ±π∈ ∀∞⊥≠ℵ∂√∃

Figure 24: Examples of different fonts

Certain fonts—like Courier—have fixed pitch, meaning that every character has the same width. An option would be a variable-pitched font, like Gill Sans or Times Roman, in which some characters, like the'm,' are broader than others, like the 'i'. Serif or sans-serif fonts are distinguished by another feature. Fine, short cross-lines that resemble the ends of strokes in cut stone letters characterise serif fonts. The strokes of a sans-serif font are square-ended. Furthermore, there are fonts designed to resemble Gothic or cursive script, Greek letter fonts and mathematical symbol fonts.

Scanners and Optical Character Recognition

Electronic documents are printed by printers and then placed on paper; scanners do the opposite. They begin by converting the image to a bitmap, but they can then immediately return the page to text by using optical character recognition. The image that needs to be converted could be a printed picture, a photograph, or a hand-drawn image.

The two primary types of scanners are handheld and flat-bed. The paper is placed on a flat glass plate and scanned using a flat-bed scanner, which turns the entire page into a bitmap. A variation of the flat-bed is used in multi-function devices (printers, fax machines, copiers), where the pages to be scanned are pulled through the device. A modest stack of sheets can be put in a feed tray on many flat-bed scanners so that everything scans automatically without the need for user participation. The image is pulled over by hand using handheld scanners. A bitmap strip is produced as the head moves across an area that is read in. The scanner is able to determine the size of the image by measuring the speed at which it is being pushed thanks to a roller at the ends. Usually just 3 or 4 inches (80 or 100 mm) wide, the scanner might even be as small as a big pen (mostly used for scanning individual text lines). This means that in order to create a full page image, at least two or three strips must be "glued" together by software. This is a challenging operation because the strips will overlap, may not be perfectly parallel to one another and may have issues with brightness and contrast. Small pictures, like photographs, can be read in a single pass for desktop publishing, provided that one direction is less than the scanner's width.

A light beam is directed at the page by scanners, which then record the colour and strength of the reflection. Similar to photocopiers, certain colours may look darker on a monochrome scanner due to the colour of the light that is shone. For instance, a red image will fully reflect the light and not appear on the scanned image if the light is pure red.

(c) Amity University Online

Notes

Similar to printers, scanners have varying resolutions, typically ranging from 600 to 2400 dpi. As with printers, it is important to carefully interpret the quoted resolution. Many, like certain digital cameras, feature a scanhead with a lesser resolution but digitally interpolate extra pixels. Colour scanners frequently offer monochrome modes for copying in grayscale or black and white, however monochrome scanners are normally found only in multifunction machines. Typically, scanners can produce images with up to 256 grayscale or RGB (red, green, blue) colour levels. When a grayscale image needs to be converted to pure monochrome (such from a printed page), it can threshold the image, turning all pixels that are darker than a specific value black and the remaining pixels white.

In desktop publishing (DTP), scanners are often used to read in hand-drawn images and photos. This implies that cutting and pasting tasks can be completed electronically instead of using actual glue. A range of image modification software tools can also be used to rotate, resize and alter the photos in other ways. These tools are getting more and more powerful, making it simple to do complicated picture changes. These can be as simple as applying edge-detection and special effects filters, or as sophisticated as colour correction and image fusion. Photomontage effects that are not achievable with conventional photography or paper techniques can be achieved via the utilisation of numerous layers. To ensure that the scanned colours match the printed colours, considerable processing is usually required even in cases where the scanned image will just be reproduced again as part of a larger publication. Special film scanners are available that can scan colour slides or photographic negatives for film photos. Of course, scanning is not required if the photos are digital.

Document storage and retrieval systems represent another application area where paper documents are scanned and saved on a computer instead of—or occasionally in addition to—a file cabinet. Maintaining paper data comes at a huge cost, while electronic storage can be more flexible, dependable and less expensive. As we will see later, storing a bitmap image is not the most space-efficient nor beneficial for access techniques, thus optical character recognition can be used in conjunction with scanning to extract the text from the document instead of the page image.

The method by which the computer is able to “read” the characters on the page is called optical character recognition (OCR). The ability to dependably read print has only lately become possible due to the large range of typefaces and print sizes; it is not as simple as matching the shape of a character to the image on the page. These days, OCR is actually a bit of a misnomer since the OCR software works on the bitmap image even though the paper is optically scanned. The software available today is capable of identifying “unseen” fonts and can even generate output in word processing formats while maintaining formatting features like italics, centering and superscripts.

Electronic publication for the internet and multimedia is another crucial field. In electronic publishing, the scanned image is intended to be viewed on screen, whereas in desktop publishing, the scanned image typically ends up back on paper (after editing). These pictures can be used as digital photos only, or they can be made interactive such that when you click on an area of the picture, relevant information is shown. A significant issue with employing electronic images is the abundance of visual storage formats. The fact that different computers can display varying numbers of colours and that different displays can display the same image very differently present another issue.

The popularity of the digital camera has grown due to the significance of electronic publication and the simplicity with which photos may be electronically altered for printing. A digital camera is equipped with a tiny light-sensitive chip that records images straight into memory, eliminating the need for film.

© Amity University Online

Notes

1.2.4 Memory

Computer memory can be thought of as operating at several levels, similar to human memory, with the memory that has faster access usually having less capacity. We can categorise these into short-term and long-term memories (STM and LTM) by drawing an analogy with human memory, however this analogy is not very strong because a computer's STM has a far larger capacity than seven things! The terms primary and secondary storage are more frequently used to refer to the various stages of computer memory.

The user interface designer is not directly interested in the specifics of computer memory. Nonetheless, there are significant limitations on the type of interface that may be built due to capacity and access method limitations. After going over some fairly basic information, we'll put the raw memory capacity into context by looking at the kinds of things that may be stored. We'll also look at how technological advancements have given designers greater freedom to create interfaces that work better.

RAM and Short-term Memory (STM)

The registers on the computer chip are located at the lowest level of computer memory, however they primarily affect the computer's overall speed and have no effect on the user. The majority of data that is currently in use is stored in random access memory (RAM) on silicon chips. The precise access times, power consumption and features of various RAM types vary. Information can be accessed at a rate of about 100 Mbytes (million bytes) per second, with typical access times of the order of 10 nanoseconds, or one hundred millionth of a second. Modern personal computers typically have 64–256 Mbytes of storage.

The majority of RAM is volatile, meaning that when the power is switched off, its contents are gone. On the other hand, a lot of computers have a small quantity of non-volatile RAM, which may use a small battery to help it remember its data. In a huge computer, this may be used to store setup data, but the entire memory will be in a pocket organiser. Although non-volatile RAM is more expensive and should only be utilised in extreme circumstances, the gap is closing as more notebook computers adopt very low-power static RAM. Although non-volatile RAM should be categorised as LTM by strict analogy, the key point we want to make is the difference between STM and LTM in a conventional computer system.

The differences grow increasingly hazy with PDAs since the battery power ensures that the device is never fully off, meaning that RAM memory essentially lasts indefinitely. Additionally, some employ flash memory, a type of silicon memory that is in between standard RAM and fixed content ROM (read-only memory) chips. Although writing to flash memory is comparatively sluggish, once it is written, it keeps its contents even in the absence of power. On PDAs, these are occasionally referred to as silicon discs. Digital cameras usually store their photos on flash media and to transfer data, compact flash-based devices are plugged into a laptop or desktop computer's USB port.

Disks and Long-term Memory (LTM)

The LTM for the majority of computer users comprises of discs, potentially along with tiny backup tapes. One crucial aspect of user security is having backups and the right software to create and recover them. But our focus will be mostly on storage types that affect interactive computer users.

Magnetic and optical discs are the two primary types of technology utilised in discs. The most widely used storage media, hard (or fixed) discs and floppy discs, include magnetic coatings similar to those on audio cassettes, which are used to store data.

(c) Amity University Online

Notes

Floppy discs typically have capacities of 300 kbytes to 1.4 Mbytes, but since they are detachable, you can have as many on your desk as you have space for. Less than 40 Mbytes to several gigabytes (Gbytes), or several thousand million bytes, can be stored on a hard drive. There are two access times for discs to take into account: the time it takes to locate the correct track on the disc and the time it takes to read the track. The former, which usually takes on the order of 10 ms for hard drives, predominates in random readings. After the track is located, the transfer rate is quite high—possibly several hundred kilobytes per second. There are also other kinds of large detachable media that fall in between removable hard discs and floppy discs; these are particularly crucial for multimedia storage.

Laser light is used by optical discs to read and occasionally write data to the disc. Although there are other high capacity specialised optical media, the CD-ROM—which uses the same technology as audio compact discs—is the most widely used. Although they cannot be written to, CD-ROMs have a capacity of about 650 megabytes and are ideal for distributed software, multimedia and published information like online reference books. As a type of write-once read-many (WORM) technology, recordable CDs are more adaptable than blackboards in that data can be written on them, but only once at a time, as the name implies, at any position. They are obviously highly helpful for creating extremely secure audit information and for backup purposes. Lastly, there are fully rewritable optical discs; nevertheless, they are still mostly used for archival rather than dynamic storage because the rewrite time is usually significantly slower than the read time. Several CD-ROM readers and writers are also capable of reading DVD files, which were first created to store movies. Since optical media are more durable than magnetic discs, using a jukebox arrangement—in which numerous optical discs can be automatically brought online as needed—is simpler. This can provide many hundred terabytes of internet capacity. Nonetheless, some big capacity shops are switching to giant disc arrays since magnetic disc capacities have increased more quickly than the fixed standard of CD-ROMs.

Understanding Speed and Capacity

So how does the user benefit from the different capacities and speeds? We can enumerate some typical capacities for our typical personal computer system as shown in the table below.

Table 1: Typical capacities of different storage media

	STM small/fast	LTM large/slower
Media:	RAM	Hard disk
Capacity:	256 Mbytes	100 Gbytes
Access time:	10 ns	7 ms
Transfer rate:	100 Mbyte/s	30 Mbyte/s

First, documents come to mind. book, which is roughly 2 Mbytes (220,000 words) and would hardly fill up 256 Mbytes of RAM. (This size, 2 Mbytes, is unformatted and devoid of images; the whole data files are much larger in actuality but still well within main memory's capability.) The Bible, to use a more well-known work, would require roughly 4.5 Mbytes. This would vanish from a hard drive and still only use 2% of main memory.

On a smaller PDA, though, it might appear constrictive. As long as you don't want to upload your whole collection online, this makes the RAM appear to be in good shape. Nevertheless, a dictionary and thesaurus are often included with word processors and using the same one across many products is not normal practice. It is common to find each application using tens or even hundreds of megabytes of disc space in addition to help files and the program itself; it is not difficult to fill a few gigabytes of disc at all!

© Amity University Online

Notes

Comparably, while most (but not all) single programs can run in 256 Mbytes of RAM, windowed systems will quickly use up several megabytes when running multiple apps at once. Operating systems deal with this by switching the full program onto disc or paging unnecessary portions of programs out of RAM. This has a major impact on interaction but has less effect on the program's ability to work logically. In the event that you choose a window and the associated application is now switched out onto the disc, it needs to be switched back in. On many systems, this results in a noticeable and annoyance-inducing delay that can be significant.

The von Neumann bottleneck that exists between the main memory and the disc is the cause of the swapping delays. Although there is a lot of data in the machine's RAM, it is not where it is intended to be. The disk's transfer rate limits the path between them, making it very sluggish. While operating system-related swapping may be hard to eliminate, an interactive system designer can prevent some of these issues by carefully considering the storage and transfer locations of data. For instance, the program may transport information in an inefficient manner. Let's say the user requests to view a document. Just enough is read in order for the first page to be displayed; the remaining text is read in idle periods, as opposed to reading the entire thing before allowing the user to continue.

Going back to the documents, our system's capacity appears even less amazing if they are scanned as bitmaps (and not read using OCR). Assume that a paper measuring 11 by 8 inches (297 by 210 mm) is scanned at 1200 dpi using an 8 bit grayscale (256 levels) preset. One billion bits, or around 128 Mbyte, make up the image. Thus, 800 pages might be stored on our 100 Gbyte disc.

Things get considerably worse when we use video. Assume that we wish to save moving video at a frame rate of 25 frames per second, with 12 bits per pixel (4 bits for each primary colour, or 16 levels of brightness).

While this is by no means a high-quality image, 10 Mbytes are required per second, or around 400 kbytes per frame. One decent movie, or roughly three hours of video, will fit on our disc. Refocusing on still images, high-quality digital cameras often capture 2 to 4 mega pixels at 24 bit colour, which translates to 10 Mbytes of raw, uncompressed image. This means that not all of your vacation photos will fit into your primary memory!

Storage Format and Standards

Text and bitmap graphics are the most often stored data kinds by interactive programs; the use of music and video is growing. This topic examines the absurd variety of file storage formats. Database retrieval will be covered in the following subsection.

The ASCII (American standard code for information interchange) character codes are the fundamental standard for text storage. They assign an internationally recognised 7 bit code (decimal values 0–127) to each standard printable character and several control characters. This allows text to be stored in an 8 bit byte or transmitted as 8 bits including parity. Numerous systems extend the codes to include mathematical symbols, international letters like "ae," and line-drawing characters between 128 and 255. The UNICODE standard is a 16-bit expansion that supports a significantly wider range of characters, including the Japanese Kanji character set.

Modern documents consist of more than just characters, as we have already seen. There are many fonts used in the text and formatting elements like page headers, footers and centering are included. Formatted text storage is generally vendor specific because almost all applications have their own file formats. The fact that many vendors try to hide their file formats or update them often to prevent their products from working with those of

(c) Amity University Online

others doesn't help either. Rich text format (RTF), which includes formatting information like style sheets, is the most widely used shared format, aside from raw ASCII. Even yet, if an application can import or export RTF, it might only be a condensed representation of the complete document style.

RTF prioritises appearance since it views the document as prepared text. Another way to think of documents as structured things is like this: a book has chapters that are made up of sections, subsections, paragraphs, sentences, words and characters. Although they are rarely utilised in reality, there are ISO standards for document interchange and structure that might be used, in theory, to move data across packages and sites. Standard generalised markup language, or SGML for short, is a language that can be used to store structured text in a sufficiently extensible manner, much as PostScript is used to describe the printed page. You have the ability to build your own structures (the SGML definition itself) and create documents based on them. A simplified form of SGML called XML (extensible markup language) is currently widely utilised for web applications.

There is an almost infinite variety of formats available for bitmap storage. The information contained in the saved image must include the image's size, bit count per pixel and maybe a colour map in addition to the image's individual components. A "hot-spot" that can be used as a cursor may also be present in an icon. You may understand why there are issues if you explore every possible way to encode these features or to leave them implicit, as well as every possible combination of these various encodings. And that's all before we've even thought about how compression works! In actuality, an entire software sector exists that creates programs for format conversion.

There is no simple answer as to which storage standard is ideal given the variety (or rather lack thereof) available; but, if you are developing a new word processor and are about to choose how to save the document on disc, stop and consider your options before defining yet another format.

Methods of Access

Typically, unique key fields with corresponding indexes are used to access databases. Before the system can locate the information, the user must know the key. An excellent illustration of this is a phone book. If you know someone's name (the key), you can find out their phone number, but you can't discover the name next to the number.

Numerous computer systems' interfaces make this clear. Frequently, when you get in touch with a company, they can only assist you if you provide them with your customer number or last order number. Dependence on a single key and index is a poor decision that significantly reduces the system's usability. Actually, you can locate a record with only a partial set of information because the majority of database systems support multiple keys and indices. Thus, these issues can be avoided with a minimal amount of forethought.

There are good reasons to avoid indexing on an excessive number of things. One must weigh the expense of storage against ease of use when adding additional indices, as this increases database size. But unless in extreme cases, this is no longer a valid argument due to ever-increasing disc sizes. Regretfully, after attending lectures on algorithmic efficiency, computer scientists tend to be frugal when it comes to storage. Security and privacy are two more good reasons to limit the fields you index. For instance, phone companies usually maintain an online index that, when a subscriber's phone number is entered, returns the subscriber's name and address; however, this information is kept private to safeguard the privacy of their clients.

(c) Amity University Online

Notes

It's a common misconception that dictionaries are exclusively helpful to spellers. Poor spellers are unable to look up words since they do not know what a word looks like. In general, an application can assist the user by matching misspelt versions of keywords, not just in spelling packages. Do what I mean (DWIM), which is utilised in a number of Xerox PARC's experimental programming environments, is one illustration of this. The system prompts the user with the closest right name if a command name is misspelt. This is less of a problem with menu-based systems, but it is conceivable to do the same with file selection, for example. An other significant example of this concept is the Soundex system for indexing words, particularly names. Soundex locates words that sound similar when given a key. For instance, it would locate MacCleod if given McCloud. These are all instances of systems that are meant to be forgiving; generally speaking, one should try to allow for user error. There are, once more, certain exceptions to this rule. For example, you don't want the automated teller machine (ATM) at a bank to grant cash when the PIN is nearly exact!

Some databases may restrict the length of text strings that can be contained in records, or they may require that the length be predetermined. In such cases, the database severely limits interface applications in which text plays a significant role. On the other hand, unformatted and unstructured material is the main focus of free text retrieval systems. By maintaining an index of every word in every document, these systems enable you to request specific documents that contain the terms "human" and "computer." While they are much slower, programs like variations of the UNIX "grep" command provide some of the same features by rapidly searching a list of files for a certain term. Naturally, the most common method for using search engines to find content on the web is to use free text search.

1.2.5 Processing and Networks

Moore's law

Everyone knows that computers just get faster and faster. However, in 1965 Gordon Moore, co-founder of Intel, noticed a regularity. It seemed that the speed of processors, related closely to the number of transistors that could be squashed on a silicon wafer, was doubling every 18 months – exponential growth. One of the authors bought his first 'proper' computer in 1987; it was a blindingly fast 1.47 MHz IBM compatible (Macs were too expensive). By 2002 a system costing the same in real terms would have had a 1.5 GHz processor – 1000 times faster or 2^{10} in 15 years, that is 10 × 18 months.

There is a similar pattern for computer memory, except that the doubling time for magnetic storage seems to be closer to one year. For example, when the first edition of this book was written one of the authors had a 20 Mbyte hard disk; now, 11 years later, his disk is 30 Gbytes – around 2^{10} times more storage in just 10 years. The effects of this are dramatic.

If you took a young baby today and started recording a full audio video diary of every moment, day and night, of that child's life, by the time she was an old lady her whole life experience would fit into memory the size of a small grain of dust.

Figure 25: Moore's law

About 100 million instructions are processed per second by computers that execute interactive programs. It may seem like a lot, but just like memory, it can run out quickly. It's true that one of the authors' earlier programs "hung," and every attempt to debug it was unsuccessful. Subsequent calculations revealed that the program's completion time would have exceeded the universe's known age! Not all malfunctions have to be so spectacular to take down a system.

(c) Amity University Online

Notes

Think of one drawing system that the authors are familiar with, for instance. You depress the mouse button at one end of the line, drag it and then release it at the other end—but not too quickly—to create a line. The line starts halfway down if you don't press down the button and then hold your hand steady for a brief period of time! Even a small delay of a fraction of a second can have catastrophic consequences for tasks requiring the user's hand-eye coordination.

Effects of finite Processor Speed

As we can see, processing speed has a significant impact on the user interface. Considering these consequences is necessary while creating an interactive system. Two types of errors are caused by processing speed: excessively slow and excessively quick processing!

One instance of the former was seen above. This was a functional error because the software executed the incorrect command. The mouse button depressing and release locations should be represented by lines drawn by the system. The program makes a mistake, though, in that it does not quickly sufficiently detect the mouse's location after realising the button is down, which could lead to the user moving the mouse before the start position is recorded. This is not a design flaw, but rather an implementation issue with the system. To be fair, though, information from lower tiers of the system software might not be provided to the programmer in a suitable manner.

A second flaw resulting from sluggish processing is when the program appears to be operating well, but the feedback is processed too slowly, producing odd interface effects. The system buffers user input—that is, it stores keystrokes, mouse clicks and mouse movements—to prevent errors of the first kind. Sadly, this causes other issues of its own. Cursor tracking is one instance of this kind of issue that occurs with character-based text editors. The user presses the cursor-left key in an attempt to move backwards on the same line and fix an issue. The user releases the key when the pointer reaches the desired location. Regretfully, the cursor overshoots because the system is processing a few additional cursor-left keys because it is running behind schedule in responding to the user. Again overshooting, the user tries to fix this by clicking the cursor-right key. Usually, the only way the user can determine if the buffer is empty is to interact with the system very slowly and watch the cursor move after each keystroke.

Icon Wars is a related issue that arises on Windows-based platforms. When a user clicks on an icon or menu, nothing happens; the computer is either sluggish or busy. The user then tries again, clicking and all of a sudden all of the buffered mouse clicks are interpreted, causing a blur of menus and windows to flash on the screen. This time, the issue is more with the response's variability than its excessive slowness—it happens quickly enough when it does. These issues are usually caused by program switching in and out of main memory, which causes delays.

Moreover, an interaction style that works best on one system might not work as well on a slower one. More specifically, mouse-based interfaces cannot allow more than a microsecond to pass between an action and feedback; otherwise, the immediacy necessary for a successful interaction is lost. It might be necessary to use a more traditional, command-based interface if certain responses are not provided.

While it is immediately clear that a system's poor response time can cause issues for the user, it is less clear why this isn't always the best course of action. There are certain exceptions to this rule, though, such the requirement that the user be able to read and comprehend the system's output. For instance, a spreadsheet demonstration disc was previously provided to one of the writers. Regretfully, the demo machine was evidently slower

(c) Amity University Online

Notes

than the author's equipment—not by much, at most half as fast—but nonetheless noticeable. Nothing remained on the screen long enough to read as the demo went by in a blur.

Text display on many high-resolution monitors is plagued by a similar issue. Older character-based terminals would either redrew from the top or scrolled new text from the bottom of the screen, but bitmap screens would frequently "flash" the new page without indicating which way the cursor was moving. Finally, consider the rate at which the cursor flashes. This is typically done at a fixed frequency, so changing the processor's speed has no effect on the screen display. However, an LCD panel's more persistent nature makes a rate too quick for a CRT screen, which could cause the cursor to go invisible or turn slightly grey.

To some extent, the solution to these issues is simpler: instead of operating at the speed that the machine will allow, the designer can specify predetermined delays based on user desire and medium. The designer must comprehend the constraints of the computer system and factor them in throughout the entire design process in order to prepare for the first issue, which is inadequate speed.

Limitations on Interactive Performance

An interactive system's speed can be restricted by a number of factors:

- Computation bound

  This is uncommon for an interactive program, but it can happen when you use find and replace in a big document, for instance. Long delays shouldn't occur during user interactions and the system should be built to give the user a sense of how the work is going. When starting a lengthy process, attempt to indicate how long it will take. It might also be useful to indicate the stage the process is at when processing is ongoing. One way to accomplish this is to have a counter or slowly filled bar on the screen that shows how much work has been done, or you can change the pointer to show that processing is happening. Many systems indicate "this may take some time: continue (Y/N)?" after they have been processing data for a while. Naturally, by the time it states this, the procedure might already be almost complete!

- Storage channel bound

  As said, interactive performance might be hampered by memory access speed. We talked about one method of lessening this effect—laziness. It is also conceivable to trade off memory for processing speed if the system is only powered by memory and there is an abundance of raw computing power. For instance, compressed data requires compression before storing and decompression upon retrieval, but it requires less space to store and reads more quickly. Therefore, longer processing times are caused by faster memory access. If data is written more frequently than it is read, a method that is costly to compress but relatively easy to decompress can be selected. While speed of browsing is crucial for many interactive systems, people are willing to tolerate delays when updating information.

Graphics Bound

This is the most typical bottleneck for a lot of contemporary interfaces. It's common to misjudge how long tasks that seem like straightforward interface actions actually require. Programs running on the same hardware might vary greatly in performance and sometimes good coding can shorten the time required for typical graphical operations. To manage many of the most popular graphics tasks, the majority of PCs come with a dedicated graphics card. This frees up the main processor to perform other tasks, such modifying documents and other user data, while still being optimised for graphical operations.

(c) Amity University Online

Notes

Network Capacity

Most PCs are connected via networks. Using shared files on a distant computer can be the most basic example of this. Performance when accessing such files may be limited by the network speed rather than the memory.

Networked Computing

Since modern computer systems are far more powerful than they were a few years ago, a desktop computer can do high-performance interaction on its own without the need for outside assistance. But more frequently than not, we use computers connected to one another via networks rather than using them as standalone devices. This offers further advantages by enabling remote resource access for the desktop computer and communication between other parties—as long as they are linked to the same network.

Increased processing power and memory are only part of the story, as the benefits of enabling people to have much more extensive, faster and easier access to information are extremely significant for individuals, groups and institutions. As a result, these networks are intrinsically far more powerful than the individual computers that comprise them.

Even without any extra access to outside peripherals or information sources, such networked systems have an impact on interaction. Networks can function over great distances at times and information transmission may take some noticeable time. This might impact the system's response time and, ultimately, the type of interaction. If the user is not informed of what is happening, there could be a discernible lag in response. In that case, he might conclude that his instruction has been misplaced or disregarded and try again. Users of these systems often experience poor performance and dissatisfaction due to this lack of feedback, which can be mitigated by making more appropriate use of the desktop computer's ability to notify users of network activity.

An additional consequence is that human-machine contact shifts from being a closed loop to an open one. There could be numerous users interacting with the machine at once and their activities could influence how the system responds to you. A single central system that is accessed by numerous users will become slower; database updates made by one user may cause another user doing the identical query at a little different time to receive different results.

Because of its dispersion, distribution and multi-user access, a networked computer system has changed from being completely predictable and deterministic—under the user's complete control—to being nondeterministic, meaning that a single user is unaware of many significant changes that are occurring to the system as a whole. Because these systems go against the principles of consistency, useful feedback and predictable reaction, they present a unique set of challenges. However, because networked systems provide more power and flexibility, they are likely to be around for a long time, thus these problems must be carefully considered during the design process.

1.2.6 Understanding Users: Personas and Scenarios

Successful and user-centric product design requires a thorough understanding of people. Personas and scenarios are two essential instruments for reaching this comprehension. Let's take a closer look at these ideas.

Personas:

Definition:Personas are made-up characters designed to represent various user types within a specific user group, behavioural pattern, or demographic. They represent the traits, requirements, objectives and actions of a particular user base.

© Amity University Online

Notes

Components of Personas:

- Age, gender, occupation, education and other pertinent factors are included in the category of demographics.
- Goals and Motivations: Outlines the user's objectives as well as their underlying driving forces.
- Challenges and Pain Points: Describes the potential roadblocks or challenges the user may face.
- Behaviours and Habits: Describes the user's usual methods of interacting with goods and services.
- Personal Anecdotes and Quotes: Adding firsthand remarks or anecdotes adds a more human touch.

Benefits of Personas:

Empathy Building: By humanising people, personas help design teams better understand their needs.

Keep User Needs in Mind: This aids in setting feature priorities and design choices according to user specifications.

Communication Tool: Facilitates stakeholders and team members to have a common understanding.

Creating Personas:

Research: To learn more about the demographics, habits and preferences of actual users, conduct user research.

Segmentation: Create separate persona groups by grouping people with comparable demands and characteristics.

Persona Development: Using data from research, develop thorough personas for every category.

Validation: To guarantee correctness, validate personas through user testing and feedback.

Scenarios:

Definition:Stories or tales called scenarios explain how customers interact with a product or service. They give a more thorough knowledge of the user journey by outlining the context, objectives and activities users take in a given scenario.

Components of Scenarios:

- User Goals: Specifies the user's objectives in a particular circumstance.
- Context: Describes the user's surroundings, gadgets and any pertinent history to set the scene.
- Actions: Describes the precise actions the user takes to reach their objectives.
- Outcome: Indicates the outcome—positive or negative—of the user's activities.

Benefits of Scenarios:

- User-Centric Design: Directs the process by maintaining the user and their interactions in the real world at the centre.
- Contextual Understanding: Offers a comprehensive perspective on how customers interact with a good or service in different situations.

(c) Amity University Online

Notes

- Problem identification: Indicates possible weak points or places where the user experience could be enhanced.
- Developing Decisions: Supports the process of developing well-informed design choices grounded in practical user interactions.

Creating Scenarios:

- User Research: Learn about actual users' requirements, behaviours and difficulties by interviewing them.
- Determine Use Cases: Ascertain particular scenarios in which users would engage with the product or service.
- Development of Scenarios: Write stories that depict the user's path in each use case that has been discovered.
- Feedback and Iteration: Use user testing and feedback to validate scenarios and make appropriate iterations.

Integration of Personas and Scenarios:

- Thorough Understanding: Personas give a broad overview of user segments, whereas scenarios give a more in-depth look at particular interactions within those segments.
- User-Centric Design: By focusing on actual user demands and behaviours, personas and scenarios work together to support a user-centric design approach.
- Holistic Design Approach: Personas guide the development of plausible scenarios, guaranteeing that design choices take into account the various needs of various user types.
- Validation: Through user testing, personas and scenarios can be verified, improving our understanding of users as we move on with the design process.

Throughout the design process, personas and scenarios are essential tools for comprehending users. While scenarios offer a contextualised story of user interactions, personas create a humanised portrayal of user segments. Design teams may create goods and services that genuinely connect with consumers by incorporating these technologies, which will increase user pleasure and overall success.

Design Thinking Principles

Originally, the goal of design thinking was to teach engineers how to solve issues creatively, just like designers do. Stanford University mechanical engineering professor John E. Arnold was among the first to write about design thinking. He authored "Creative Engineering," the book that delineated the four domains of design thinking, in 1959. From that point on, the domains of science and design engineering started to adopt design thinking as a "way of thinking."

Design thinking gained popularity in the 1980s with the emergence of human-centered design and the 1990s with the founding of the design firm IDEO. By the early years of the twenty-first century, design thinking had found its way into the corporate sector. The d.school at Stanford University started offering design thinking courses in 2005 as a method for fostering social and technological innovation.

In fact, a lot of the strategies and tactics employed in design thinking are taken straight out of the toolkit of designers.

"Design thinking" is a human-centered innovation approach that combines the needs of people, the possibilities of technology and the prerequisites for productive business operations.

© Amity University Online

Notes

Before concepts are approved for implementation, designers conduct multiple approaches, merge them into new concepts and prototypes and test those with potential consumers.

Working with organisations or individuals, resolving difficult social problems or developing new products, devising strategies for gaining a competitive edge, or improving day-to-day operations are just a few of the difficulties that can benefit from the use of design thinking.

For instance, you're trying to figure out how to make your company's customer experience initiatives better, but you're not exactly sure where to start. Rather than starting from scratch when brainstorming potential solutions, consider applying design thinking principles like empathy and user research!

History of Design Thinking

A human-centered approach to innovation is design thinking. It brings together the best aspects of two worlds: technology, which it pulls quick prototyping and testing from and the humanities, where we find empathy and creativity.

Problems are opportunities in the context of design thinking. It's a method of working through problems by quickly iterating, prototyping and testing your concepts before settling on a single, effective solution. It inspires you to consider people when creating new designs—what will it look like? How will it feel? To what extent is it user-friendly?

This strategy isn't limited to big companies or well-known brands; small firms can also benefit greatly from it!

Types of Design Thinking

Design Thinking is an innovative process that prioritises design from the perspective of the human. It is a humanistic approach to innovation that solves issues via empathy and ingenuity. Any industry may use design thinking to address issues, generate new possibilities and enhance people's lives.

Process of Design Thinking

1. Identify the problem
2. Discover the need
3. Create solutions
4. Test the solution
5. Evaluate the solution

Principle 1 — Empathize with your Users

The capacity to comprehend and experience another person's emotions is known as empathy. It is essential to design thinking and a basic component of human connections. Imagining what others might desire or need in an effort to see things from their perspective is what is meant by empathy, not having compassion or feeling sorry for someone.

Seeing people in their natural settings—the grocery store checkout queue, the family cooking supper while their children are on their laps—and asking questions like "How do they feel?" are great ways to develop empathy. "What would improve this experience?" and "What are they attempting to achieve?"

Principle 2 — Define a Meaningful and actionable problem statement

(c) Amity University Online

Notes

This principle deals with specifying and honing the issue that you are attempting to resolve. Before beginning to solve a problem, you must identify its significance. If you don't, you risk tackling the incorrect problem or wasting time on unrelated activities that don't advance your main objective.

A solution statement includes:

What we think is wrong (problem)

How we plan on fixing it (action)

Principle 3 — Ideate by questioning presumptions and coming up with concepts for novel fixes.

Throughout the design thinking process, you will question presumptions in order to consider your issue from various angles. You'll make an effort to identify unmet demands and generate fresh concepts for creative fixes.

In addition to coming up with a solution, the objective is to come up with other options that can be compared to see which one works best. For this reason, we insist on asking better questions rather than discussing problem definitions in order to avoid getting caught up in preconceived notions or prior knowledge.

Principle 4 — Prototype to start creating solutions

Making and testing a model of your solution is the process of prototyping. It serves as a means of user input and design testing to ensure the expected functionality is achieved. Before investing excessive effort in creating something that might not be workable or practical, prototyping enables you to determine whether your idea is workable and feasible.

Prototypes should ideally be inexpensive to produce, but they also need to be sufficiently good to ensure that users grasp your goals without becoming lost or dissatisfied. An effective prototype will demonstrate functionality without requiring a great deal of explanation, which makes it simple for all parties engaged in the process—including potential users—to rapidly embrace an idea.

Depending on their objectives, designers employ several techniques for prototyping:

Some designers, for instance, use paper prototypes so they can rapidly test out various concepts and fast go through several iterations in short amounts of time.

Some designers might be more interested in digital prototypes created with programs like Adobe XD or Sketch. In other situations, when engaging with stakeholders who lack the technical knowledge of UX processes that user research professionals possess, even basic sketches can be sufficient.

Principle 5: Test those solutions in the real world.

A prototype, in the opinion of many designers and developers, is ready to be shown to users. That being said, there are numerous approaches to testing your solution on actual people.

When developing a website or application intended for a third party, such as an internal system, conducting end user interviews to test your design might aid in identifying issues prior to launch. Before going on to digital media like websites or apps, you may also try out prototypes on paper by creating sketches or wireframes of your ideas and exhibiting them around.

This will enable input from people who might not be acquainted with the inner workings of computers, but who can comprehend what would make sense in terms of functionality

© Amity University Online

Notes

and usability once they see something concrete, like a prototype, instead of simply words expressing concepts.

Prototyping and Iterative Design

Designing efficient and user-friendly human-computer interaction (HCI) solutions requires prototyping and iteration. Whether you are developing a software system, game, app, or website, you must test and improve your concepts with actual people and their input. This post will teach you how to do user testing, use various prototypes and apply design concepts and methodologies to enhance your HCI design solutions.

1. Types of prototypes

   A prototype is an iteration of your HCI design solution that you may utilise for ideation, testing and assessment. Depending on your objectives and available resources, prototypes can range widely in terms of quality, usefulness and amount of detail. For instance, low-fidelity prototypes, such wireframes, sketches, or paper models, can be used to quickly and affordably explore and produce many concepts. High-fidelity prototypes can be used to validate and show off particular features and functionalities in an authentic setting. Examples of these include interactive mockups, simulations and working versions.

2. User testing

   The practice of getting actual or potential people to assess and comment on your prototype is known as user testing. In addition to measuring user happiness and performance, user testing can assist you in identifying and resolving usability issues as well as user needs, preferences and expectations. There are various methods to do user testing, including think-alouds, surveys, observations, interviews and usability tests. Throughout the whole design process, from the initial phases of prototyping to the latter stages of implementation and deployment, you should plan and carry out user testing repeatedly.

3. Design principles and methods

   Based on theory, best practices and user research, design principles and methodologies are a set of rules and resources that can assist you in improving your HCI design solutions. Design principles include things like affordance, simplicity, feedback, visibility and consistency. Personas, scenarios, user journeys, user stories and heuristic evaluation are a few instances of design methodologies. In order to assess, evaluate and improve your HCI design solutions, you should apply design principles and methodologies to your prototypes and user testing data.

4. Benefits of prototyping and iterating

   For your HCI design solutions, prototyping and iterating can have a lot of benefits, such encouraging creativity and innovation by exploring and testing out various concepts and options. By seeing problems early on and fixing them, this method can help lower risks and expenses by preventing time and resource waste. By including consumers in the design process, it can also boost user satisfaction and involvement. It can also improve quality and performance by making sure that your solutions are practical, pleasant and valuable for your users.

5. Challenges of prototyping and iterating

   Your HCI design solutions may also face difficulties when prototyping and iterating. To balance the breadth, quality and viability of your prototypes and iterations, time and resource management is crucial. The goal, target audience and context of your prototype and user testing should all be taken into consideration when selecting the type and level

(c) Amity University Online

of prototype you create. Choosing the right techniques, resources and measurements for your user testing and design assessment is essential to obtaining user input. Additionally, make sure that your prototypes and iterations are in line with your organisational and technical requirements and limits before implementing and delivering your solutions.

6. Tips and best practices

It is advised to begin prototyping and iterating as early in the design process as feasible to optimise the benefits for your HCI design solutions and to keep going until you achieve your intended result. To test different components and phases of your design, use a variety of prototype types and levels. Engage stakeholders and users in the prototyping and iteration process and solicit their input on a frequent basis. Make sure that your decisions and process are guided by design concepts and processes. Finally, properly record and explain your iterations and prototypes to the stakeholders, users and team.

1.2.7 Models

We have already seen how effective models are in assisting us in comprehending intricate systems and complex behaviour. The user and the system are the minimum number of participants in an interaction. As we've seen, both are complex and differ greatly from one another in terms of how they communicate and approach the work at hand. For the interaction to be successful, the interface must thus translate between them in an efficient manner. There are several potential sites of failure and reasons why this translation might not work. We can pinpoint the likely source of problems and gain a clear understanding of the interaction by using interaction models. They also give us a framework for contrasting various interaction philosophies and taking interaction issues into account.

We start by examining Norman's execution–evaluation cycle, which is the most influential model of interaction. Next, we examine a different model that builds on the concepts of Norman's cycle. These two models both characterise the interaction in terms of the user's objectives and behaviours. As a result, before going into detail about the models themselves, we will quickly go over the terminology that was employed as well as the underlying presumptions.

The Terms of Interaction

An interactive system's traditional goal is to assist a user in achieving objectives from a particular application domain. A domain defines a field of knowledge and proficiency in a particular practical endeavour. Authoring, graphic design and process control in a factory are a few instances of domains. A domain is made up of concepts that emphasise its key features. Some key ideas in the field of graphic design are geometric shapes, drawing surfaces and drawing tools. Tasks are actions taken to modify a domain's notions. A goal is the intended result of a task that is completed. One goal in the field of graphic design, for instance, is to develop a particular geometric shape on the drawing surface with precise qualities. Creating a solid red triangle in the centre of the canvas would be a related objective. An intention is a particular action needed to achieve the objective.

Task analysis is the process of determining the domain, goals, intentions and tasks that comprise the issue space for an interactive system's user. We may evaluate the interactive system intended to support them by applying our understanding of tasks and goals. Since the concepts utilised in the system's design and the user's description are different, we can refer to them as independent parts, the system and the user, respectively. Both the System and the User are specified using a language that may convey ideas pertinent to the application's domain. We shall refer to the language used by the system as the core

(c) Amity University Online

Notes

language and the language used by the user as the task language. While the task language explains the psychological characteristics of the domain important to the User state, the core language describes the computational characteristics of the domain relevant to the System state.

Although a computerised application is anticipated for the system, non-computer applications can also use the models. Another widely held belief is that we can only create single-user apps because we can differentiate between users and systems. This is untrue. Still, the focus is on the interaction as seen through the eyes of a single user. From this vantage point, additional users are considered to be a part of the system, such as those participating in multi-party conferencing.

The Execution–evaluation Cycle

Because it most closely resembles our intuitive perception of the connection between a human user and a computer, Norman's interaction model is arguably the most important in the field of human-computer interaction. A plan of action is created by the user and then carried out via the computer interface. Following the execution of the plan, or a portion of it, the user reviews the computer interface to assess the outcome and decide on next steps.

The two main stages of the interactive cycle are execution and evaluation. These can then be further separated into seven stages. The following are the phases in Norman's interaction model:

- Establishing the goal.
- Forming the intention.
- Specifying the action sequence.
- Executing the action.
- Perceiving the system state.
- Interpreting the system state.
- Evaluating the system state with respect to the goals and intentions.

Naturally, the user engages in activities at each level. The user first establishes a goal. This is the user's idea of the task, expressed in the task language and framed in terms of the domain. It is likely to be vague, therefore before the user can carry it out, it must be converted into a more definite intention and the real steps that will accomplish the aim. After the action sequence is completed, the user sees the system in a new state and interprets it according to his expectations. The user has accomplished his goal and the interaction has been successful if the system state matches it; if not, the user needs to create a new objective and start the process over.

Norman demonstrates this cycle with the straightforward example of turning on a light. As nightfall approaches, picture yourself relaxing with a book. You make the decision that you need more light, so you set out to obtain it. From there, you decide that you want to turn on the desk lamp and you outline the necessary actions—like reaching over and flipping the switch. You can ask someone who is closer to you to turn on the light for you if your intentions are different. Although the aim and behaviours are different, your goal remains the same.

Once the action is completed, you see the outcome—either the light is on or it isn't—and you interpret it using your understanding of the outside world. For instance, if the light is off, you might assume that the lamp is not plugged in to the mains or that the bulb has blown, in which case you would set new objectives to address this. If the light does turn on, you will assess the current situation using the initial objectives: is there now enough light?

(c) Amity University Online

Notes

In that case, the cycle is finished. If not, you might come up with a fresh plan to turn on the main ceiling light as well.

Norman illustrates why certain interfaces generate issues for their users with this interaction model. He talks about these in terms of the evaluation and execution gulfs. As we previously mentioned, the phrases used by the user and the system to refer to the domain and goals are different. Keep in mind that the system's language is referred to as the core language, while the user's language is termed the task language. The discrepancy between the actions the user formulates to accomplish the goal and the activities the system permits is known as the gulf of execution. The interaction will be successful if the user intends the actions that the system permits. Thus, the interface's goal should be to close this gap.

The difference between how the system state is physically presented and what the user expects is known as the evaluation gap. The gap in evaluation is minimal if the user can easily assess the presentation in relation to his objective. The interaction is less effective the more work the user must put in to understand the display.

The Interaction Framework

The interaction framework divides the system into four primary components, as illustrated in the figure below and includes the system explicitly in an attempt to provide a more realistic description of interaction. The system, the user, the input and the output are the four main parts of an interactive system that are represented by the nodes. Every element has a unique language. Apart from the languages we have previously introduced for the User's work and the System's core language, there exist other languages for the Input and Output components. Together, input and output make up the interface.

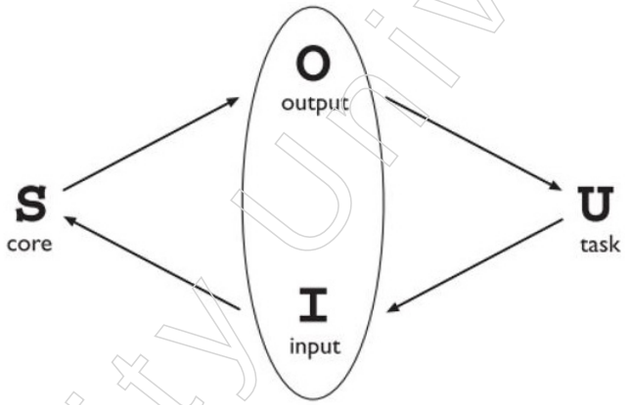


Figure 26: The general interaction framework

The interactive cycle consists of four steps that represent translations from one component to another and are shown by the labelled arcs in the following Figure, which illustrates how the interface sits between the User and the System. The interactive cycle starts with the user defining an objective and a task to accomplish that objective. Since the input is the sole means by which the user may control the computer, the task must be expressed in terms of the input language. The system performs operations by translating the input language into the core language. The cycle's execution phase is now over and the assessment phase has begun as the System changes as indicated by the operations. The User has to be informed of the new condition of the System. The output's notions or features are represented by the system attributes' current values. The evaluation phase and, thus, the interactive cycle, are subsequently concluded when the User examines the Output and evaluates the interaction's outcomes in relation to the initial objective. The

Notes

interaction involves four primary translations: presentation, performance, observation and articulation.

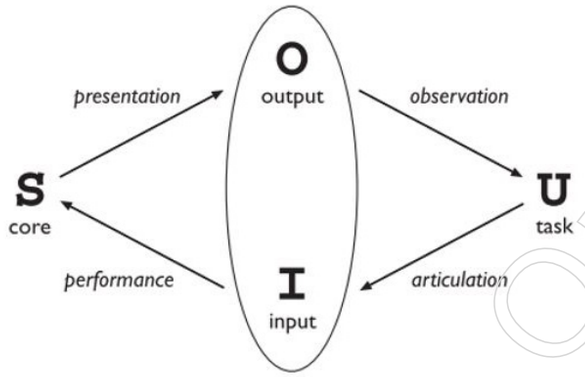


Figure 27: Translations between components

The intended task must be stated in the input language by the user in order to accomplish a particular goal. The tasks consist of the user's reactions, which must be converted into input stimuli. As previously said, the evaluation of this articulation is based on how well it covers tasks to input and how simple it is to complete the translation. The job is written in terms of specific psychological characteristics that draw the User's attention to the domain's salient elements. The work will be considerably easier to articulate if these psychological characteristics map onto the input language with clarity. Norman cited a huge room with overhead lights controlled by a bank of switches as an example of a poorly designed mapping. Often, it is preferable to have the lighting controlled so that only one area of the space is illuminated. The next puzzle we have to solve is which light goes on which switch. Usually, the end result is frustration and further trials. This occurs when an input language consisting of a linear row of switches that may or may not be oriented to match the room layout is used to articulate a goal (e.g., "Turn on the lights in the front of the room").

On the other hand, a well-mapped environment can be found in virtual reality systems, where input devices like datagloves are designed to facilitate articulation by turning the user's mental gestures into an action that can be executed right at the interface. Certain instructions related to file management can be expressed more easily when using direct manipulation interfaces, as those present on popular desktop operating systems like Windows and Macintosh. However, some operations are really difficult to accomplish with such an interface, including repeatedly changing files or starting programs whose icons are hidden.

The reactions from the input are converted into system stimuli in the following step. When evaluating this translation, it is interesting to see how many states of the system the translated input language can access straight from the System stimulus. For instance, some compact disc players' remote controls prevent the user from turning the player's power off; as a result, the player's off status cannot be accessed using the input language of the remote control. However, the power is typically controlled via a button on the compact disc player's panel. Because the user is not making the effort, it is less important how easy this translation is from input to system. Nonetheless, the programmer and designer may put in a lot of work. Here, the cost of implementation is taken into consideration when evaluating how easy the translation is.

The interaction's execution phase ends and the assessment phase starts when a state shift takes place within the system. The User has to be informed of the System's new state and the first step in doing so is to convert the System's reactions to the transition into

(c) Amity University Online

Notes

inputs for the Output component. Within the restricted expressiveness of the output devices, this translation of the presentation must maintain the pertinent system attributes from the domain. For this translation, expressiveness refers to the capacity of the Output to convey the System's domain notions.

For instance, when using a word processing program to write a paper, it is occasionally important to see both the text immediately surrounding the paragraph that you are presently writing and a larger context that is present throughout the entire work but is difficult to view on a single screen.

In the end, in order to assess what has transpired, the user must interpret the output. The output's reaction is converted into user stimuli that start the evaluation process. The final translation's coverage and convenience of use will be examined in the observation translation. For instance, it might be challenging to determine the exact time on an unmarked analogue clock, particularly if it is not orientated correctly. Determining the outcome of copying and relocating files within a hierarchical file system can be challenging when using a command line interface. It would be very impossible to develop a website using a markup language such as HTML without the ability to preview the result in a browser.

Assessing Overall Interaction

An interactive system's overall usability can be evaluated by using the interaction framework. Actually, every analysis that the framework suggests is contingent upon the task (or tasks) that the user is currently working on. This is not surprising, as we can only judge whether the tools we employ are sufficient when we try to complete a certain task inside a given domain. As an illustration, different text editors excel in different areas. One can select the text editor that is most appropriate for interacting with that particular editing task. If we had to pick just one, the best editor is the one that works best for the tasks that are completed most often. So, it is not entirely disheartening that we are unable to expand the interaction analysis past the parameters of a certain task.

1.2.8 Ergonomics

The study of the physical aspects of interaction, such as the layout and physical properties of the screen, the controls' design and the actual environment in which the interaction occurs, is known as ergonomics, or human factors. The main emphasis is on user performance and how the interface contributes to or takes away from it. Ergonomics will undoubtedly address system limitations and human psychology as it attempts to assess these interactional characteristics. Comprehensive coverage of this vast and well-established field—which is connected to but separates from HCI—would require a book unto itself. As an overview of the discipline, we look at some of the problems that ergonomics tries to solve here.

Arrangement of Controls and Displays

We have already discussed how perceptual and cognitive problems impact how we display information on a screen and give users control. Physical components of design are equally significant as these cognitive aspects. Control groups and display elements should be properly arranged to facilitate quick user access. When we examine a single user of a spreadsheet on a PC, this might not seem like a big deal, but when we consider safety-critical applications like air traffic control, aviation and plant control, it becomes crucial. Users are under pressure and confronted with an enormous array of displays and controls in each of these scenarios.

© Amity University Online

Notes

It is imperative that the physical arrangement of them be suitable in this case. In fact, going back to the less important PC program, ineffectiveness and annoyance might result from improper control and display arrangement. For instance, the command key to read articles from a newsgroup (y) is located right next to the command key to unsubscribe from a newsgroup (u) on the keyboard of one of the authors' electronic newsreaders. Newsgroups are regularly unintentionally removed as a result of this bad design. Even though this can be fixed, the user finds it frustrating and a waste of effort. As a result, it is clear that suitable layout matters for all applications.

The domain and application will determine which specific organisation this suggests, however potential organisations include the following:

- Displays and functional controls are arranged so that related functional components are grouped together;
- In domains where a specific task sequence is enforced, like aviation, sequential controls and displays are arranged to reflect the order in which they are typically used in an interaction; in contrast, frequency controls and displays are arranged based on how frequently they are used, with the most frequently used controls being the easiest to access.

The arrangement of the complete system interface with respect to the user's position is just as important as how the controls and displays are placed in relation to one another. Thus, the user should be able to observe all displays and reach all required controls without having to shift their body excessively. Crucial screens ought to be viewed at eye level. Lighting should be set up to prevent reflections and glare from warping displays. The distance between controls should allow the user enough leeway.

The Physical Environment of the Interaction

Ergonomics addresses both the physical aspects of the machine interface's structure and organisation as well as the design of the work environment itself. Where does the system get put to use? Who is going to use it? Are users going to be standing, sitting, or moving around? Once more, a lot will rely on the domain; in particular control and operational settings, this will be more important than in general computer use. The system's acceptance as well as the users' health and safety, however, may be impacted by the physical setting in which it is utilised. As such, it ought to be taken into account in every design.

The size of the user base is the first factor to take into account. This is obviously going to differ significantly. Nonetheless, in any system, the largest user shouldn't feel confined and the smallest user should be able to access all of the controls (this may even apply to a wheelchair user).

Specifically, it should be comfortable for all users to view important displays. The user should be seated for comfort and stability when using for extended periods of time. Back support should be provided by the seat. Should standing be necessary, there should be sufficient space for the user to manoeuvre and access all controls.

Health Issues

Even if we might not consider using a computer to be dangerous at first, we nevertheless need to consider the potential effects of our designs on users' health and safety. Other than the obvious safety dangers associated with poorly built safety-critical systems (such as aircraft crashes and nuclear plant leaks), a variety of factors may influence the adoption of general-purpose computers. Once more, the following physical environment elements have a direct impact on the effectiveness of the interaction and the user's performance:

(c) Amity University Online

Notes

Physical Position

Users should be able to see all displays and easily reach all controls, as we mentioned in the preceding section. Long durations of standing should not be required of users and back support should be offered when they are sitting. When a body component is to be used in a specific posture for extended periods of time (as when typing), it is recommended to offer support to facilitate rest periods.

Temperature

Temperature extremes can have a negative impact on performance and, in severe situations, health, even though most users can adjust to small changes in temperature without experiencing any negative effects. Experiments reveal that consumers' ability to focus is negatively impacted by temperature, either too high or too low.

Lighting

Once more, the working environment will determine the lighting intensity. However, sufficient illumination must be offered so that users can view the computer screen without straining their eyes. It's important to position the light source to prevent glare on the monitor.

Noise

Overexposure to noise can have negative health effects, including pain and, in extreme situations, hearing loss. It is important to keep noise levels in the workplace comfortable. This does not imply that there should be no noise at all. Users may find noise stimulating and it can offer the necessary validation of system activity.

Time

Controlling how long people spend on the system is also necessary. Pregnant women in particular may be at risk from CRT display overuse, according to certain theories.

The Use of Color

Given that both fields are concerned with human perception, ergonomics and human psychology are closely related. An example of an ergonomics concern is colour usage in screens. The amount of recognisable colours and the relatively poor blue acuity are two examples of the colour limitations of the visual system. A disproportionately large segment of the populace has colour vision impairment. Ergonomic principles result from each of these psychological occurrences; a few examples are covered here.

The display's colours should be as distinct as feasible and variations in contrast shouldn't have an impact on that differentiation. It is not appropriate to display important information in blue. When using colour as a cue, it's important to incorporate extra coding information in addition to colour.

Additionally, the colours used should match user expectations and accepted practices. The colours red, green and yellow are commonly linked to stop, go and standby, respectively. Hence, emergencies and alerts might be denoted by red, regular activity by green and standby and supplementary functions by yellow. It is not appropriate to break these customs unless there is a compelling reason to.

But it's important to keep in mind that colour conventions are influenced by culture. For instance, while most western cultures equate red with danger and caution, in Chinese culture colour represents luck and happiness. In many cultures, black is the colour of mourning, whereas in others, white is. It is especially crucial to consider the cultural connotations of colour when creating websites and systems for a worldwide audience.

(c) Amity University Online

Notes

Ergonomics and HCI

While separate from HCI, ergonomics is a vast field that coexists with it. It helps with HCI by pointing up limitations on how we design systems and by proposing precise, in-depth recommendations and standards. Because ergonomic aspects are typically well-established and known, hardware designs are standardised based on them.

1.2.9 Styles

A dialogue between the user and the computer can be thought of as interaction. The type of this dialogue can be significantly impacted by the interface design selection. Here, we present the most popular interface designs and discuss the various ways in which they affect user engagement. Several popular interface designs exist, such as.

- command line interface
- menus
- natural language
- question/answer and query dialog
- form-fills and spreadsheets
- WIMP n point and click

```
sable.soc.staffs.ac.uk> javac HelloWorldApp
javac: invalid argument: HelloWorldApp
use: javac [-g][-O][-classpath path][-d dir] file.java…
sable.soc.staffs.ac.uk> javac HelloWorldApp.java
sable.soc.staffs.ac.uk> java HelloWorldApp
Hello world!!
sable.soc.staffs.ac.uk>
```

Figure 28: Command line interface

Command Line Interface

Despite the availability of menu-driven interfaces, the command line interface (Figure above) was the first interactive dialogue style to be widely utilised and is still in widespread usage. It offers a way to directly communicate commands to the computer through the use of function keys, single characters, acronyms and full-word commands. Certain systems only allow communication via the command line, particularly when utilising telnet for remote access. These days, it is more frequently used as an addition to menu-based interfaces, giving seasoned users quicker access to the system's features.

Unlike menus, which are hierarchical, command line interfaces provide direct access to system functionality and can be used to apply many tools to the same data. This makes them quite powerful. They are also adaptable: the command can be applied to several objects simultaneously, which makes it helpful for repetitive tasks and it frequently offers a number of variables or arguments that will change its behaviour in some way. Nevertheless, acquiring and using this flexibility and power can be challenging. Since there is no indication in the command line as to which command is required, commands must be remembered. Thus, they are more appropriate for experienced users than for beginners. There is some relief from this issue when meaningful and consistent commands and abbreviations are used. Instead of using terminology from the technician's lexicon, the commands should be ones that the user is familiar with. Regrettably, commands are frequently ambiguous and differ between systems, which confuses users and raises the learning curve.

(c) Amity University Online

Notes

Menus

A menu-driven interface shows the user's options on the screen, which can be chosen with the mouse or by pressing the numeric or alphabetic keys. The selections are less taxing on the user because they are prominent and depend more on recognition than on memory. To help with identification, menu selections must still be meaningfully and logically arranged. Frequently, menus are arranged hierarchically and the necessary choice is not present at the top of the hierarchy. The sole hint for the user to locate the necessary choice is then provided by the grouping and labelling of menu items. These systems can be entirely text-based, with the menu options displayed as numbered choices (see Figure below). Alternatively, they may include a graphical element, where the menu is displayed within a rectangular box and selections are made by using the arrow keys to navigate the menu or by typing the first letter of the desired option.

```
PAYMENT DETAILS          P3-7

please select payment method:
  1. cash
  2. check
  3. credit card
  4. invoice

  9. abort transaction
```

Figure 29: Menu-driven interface

Natural Language

At first glance, using natural language to communicate with machines may seem like the most appealing option. People who are lost in a maze of menus or unable to recall commands may yearn for a computer that can comprehend instructions stated in simple language! Research and interest in natural language understanding, both for spoken and written input, are high. Unfortunately, natural language is particularly difficult for a machine to understand because of its ambiguity. There are several degrees of ambiguity in language. First, a phrase's syntax, or structure, might not be obvious. If the sentence is presented to us

The boy hit the dog with the stick

It's unclear to us if the dog is grasping the stick when the youngster hits it or if the boy is hitting the dog with it.

There may be ambiguity in a sentence's meaning even when its structure is evident from the words chosen. A sports pitch, a throw, a waterproof material, or even, informally, a region can all be referred to as "pitch." To filter through these ambiguities, we frequently rely on the context and our general knowledge. It is challenging to give the machine this information. The addition of ambiguity through the use of pronouns and relative words further complicates issues.

It appears doubtful that a general natural language interface will be accessible for some time given these issues. Still, it is possible to construct systems that comprehend limited parts of a language. The system can be given enough information in a known and restricted region to distinguish between terms. When utilising restricted natural language in interfaces, it's critical that users understand the system's limitations and don't anticipate too much comprehension.

© Amity University Online

Notes

While there is some success in using natural language in certain sectors, it is questionable if this truly qualifies as natural language. The user may experience frustration if they have high expectations for the computer and yet need to discover which language it can understand. On the other hand, the use of a broad natural language interface is likewise unclear. Due to language's inherent vagueness and imprecision, it is flexible and encourages innovation in expression. On the other hand, computers need exact instructions. If we were allowed to go wild, could we articulate our needs well enough to ensure a certain response? And even if we were successful, would the language we employed end up being a limited subset of natural language?

Question/answer and query dialog

A straightforward method for giving input to an application in a particular domain is the question and answer dialogue. The user is guided step-by-step through the interaction by asking a sequence of questions, most of which have multiple choice, yes/no, or code responses. Web surveys would be one instance of this.

These interfaces have limited capability and power, but they are simple to learn and use. As a result, they are suitable for casual or novice users as well as restricted domains (especially information systems).

On the other side, query languages are used to create queries that are used to extract data from databases. Although they employ language that sounds intuitive, they actually need to know the database structure and a certain syntax. In most cases, queries ask the user to provide one or more criteria for which the database should be searched, along with the attributes they want to have displayed. When there are just one or two attributes involved, this is simple. However, when there are several attributes, or when values of attributes need to be compared, it gets more complicated, especially if the user is only interested in one or the other attribute. The majority of query languages don't directly confirm requests, so the user's main source of validation is the search engine result. Consequently, some experience is needed to use query languages effectively. The internet search engine is one specific example.

The WIMP Interface

These days, a lot of popular interactive computer environments are instances of the WIMP interface style, also known as windowing systems. The majority of interactive computer systems in use today, particularly those that are PC and desktop workstation-focused, have WIMP as their primary interface style. WIMP stands for windows, icons, menus and pointers (or occasionally windows, icons, mice and pull-down menus). X Windows-based UNIX systems, MacOS for Apple Macintosh compatibles and Microsoft Windows for IBM PC compatibles are a few examples of WIMP interfaces.

1.2.10 Elements

The WIMP interface is named for its four main components: windows, menus, pointers and icons. In addition, a wide range of other interaction items and strategies—some more generic in nature and others tailored to particular uses—are frequently employed in WIMP interfaces. The majority of these components are shown in the figure below.

These components collectively, referred to as widgets, make up the user-system interaction toolbox of WIMP interfaces.

(c) Amity University Online

Notes

Figure 30: Elements of the WIMP interface – Microsoft Word 5.1 on an Apple Macintosh. Screen shot reprinted by permission from Apple Computer, Inc.

Windows

Windows are sections of the screen that operate independently of other portions of the screen. A window can typically be enlarged, moved and contain text or graphics. Multiple windows can be displayed simultaneously on a screen, enabling the simultaneous viewing of different tasks. As they go from one work thread to another, users can focus their attention on the many windows.

When two windows are overlaid, the rear window gets partially blocked and is refilled when it becomes visible. Windows that adjoin but do not overlap each other may also be tiled since overlapping windows can be problematic because they can hide important information. As an alternative, windows might be arranged in a cascading pattern, with each new window positioned beneath and to the left of the preceding one. This layout policy can be chosen by the user in certain systems, while it is fixed in others.

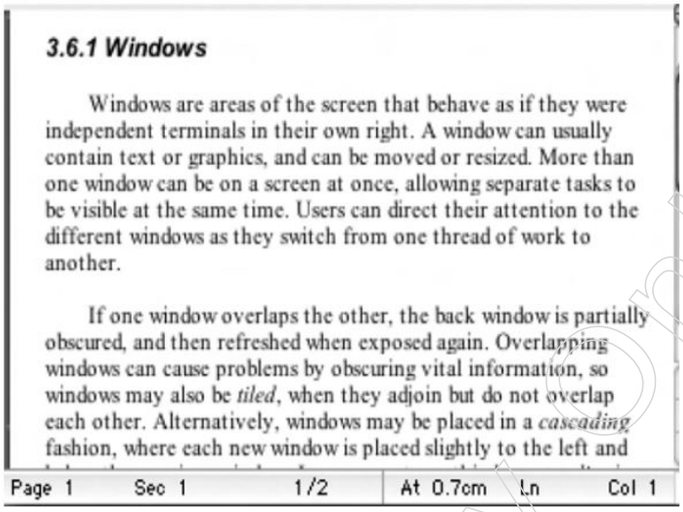
Windows typically come with a variety of accessories that improve their utility. One such attachment that lets the user shift the window's contents up and down or side to side is a scrollbar. As a result, the window acts like a true window into a much bigger universe, where dragging the scrollbars can reveal fresh information.

A window's title bar typically serves as an identifier for the user and it may have unique boxes in its corners to help with shutting, resizing, or making it as big as possible. You can see each of these in the figure below.

Furthermore, certain systems permit windows inside windows. For instance, each Microsoft Office program, such Word and Excel, has a window of its own and then each document has a window inside of it. It is frequently feasible to have various layout policies for the various application windows.

(c) Amity University Online

Notes



3.6.1 Windows

Windows are areas of the screen that behave as if they were independent terminals in their own right. A window can usually contain text or graphics, and can be moved or resized. More than one window can be on a screen at once, allowing separate tasks to be visible at the same time. Users can direct their attention to the different windows as they switch from one thread of work to another.

If one window overlaps the other, the back window is partially obscured, and then refreshed when exposed again. Overlapping windows can cause problems by obscuring vital information, so windows may also be *tiled*, when they adjoin but do not overlap each other. Alternatively, windows may be placed in a *cascading* fashion, where each new window is placed slightly to the left and

Page 1 Sec 1 1/2 At 0.7cm Ln Col 1

Figure 31: A typical window. Screen shot reprinted by permission from Apple Computer, Inc.

Icons

Windows have two options: they can be lowered to a very small representation or closed completely. A closed window is represented by a little image that is utilised; this representation is called an icon. Allowing icons allows many windows to open simultaneously on the screen, each of which can be clicked to expand to its full size. Iconifying a window is the process of shrinking it to its icon. A user can iconify the window holding a dialogue to temporarily stop following that particular thread of conversation. In addition to taking up less space on the screen, the icon reminds the user that he can return to the dialogue by opening the window. A few samples of icons from a standard windowing system (MacOS X) are shown in the figure below.

Figure 32: A variety of icons. Screen shot reprinted by permission from Apple Computer, Inc.

Other features of the system, like a trash can for discarded files or different user-accessible drives, applications, or features, can also be represented by icons. Icons come in a variety of shapes and sizes. They can be highly stylised or accurate depictions of the things they symbolise. Even random symbols may be used, but consumers may find it challenging to understand these.

Menus

The menu, an interaction method shared by many non-windowing systems as well, is the final primary characteristic of windowing systems. The system's available processes and services are displayed on a menu for the user to select from at any moment. Menus offer information cues in the form of a scannable, ordered list of operations. This suggests that useful and instructive titles for the menu's commands should be utilised.

(c) Amity University Online

Notes

The selected option is indicated by the pointing device. When the cursor advances to a menu item's location, it is typically indicated as a possible selection candidate by being highlighted (using inverse video or a comparable technique). Usually, selection necessitates an extra user action, like clicking a designated key on the keyboard or a mouse button that moves the pointer cursor across the screen. When a menu has too many things, it becomes inefficient. To solve this problem, cascading menus are used, in which selecting an item displays a menu next to it that allows you to narrow down your options. It is possible to use multiple levels of cascading menus.

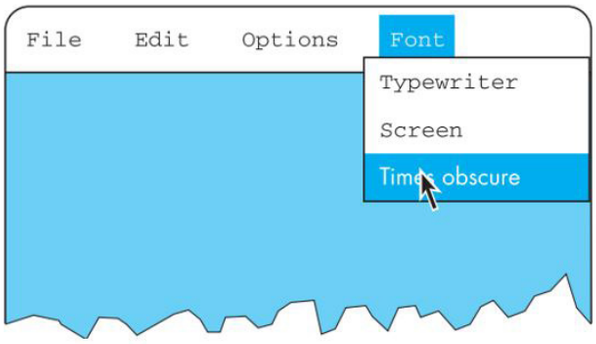


Figure 33: Pull-down menu

Because a menu bar and submenus can be dragged down or to the side of the main menu on demand, the user can always see it (Figure above). Menu bars are frequently positioned at the top of each window (as in Microsoft Windows) or at the top of the screen (as in MacOS). Other options include placing menu bars between the windows in the main "desktop" area or along one side of the screen. Websites employ the top, bottom and both sides of the screen as menu bar placements. As an alternative, the main menu can be buried and will appear on the screen upon request. These pop-up menus are frequently used to display context-sensitive options, including letting users look at the characteristics of specific on-screen items. When the mouse is depressed over the screen backdrop in certain systems, they can also be used to access more global actions.

By dragging the mouse pointer into the title bar area and clicking the button, pull-down menus can be dragged down from the title at the top of the screen. Fall-down menus function similarly, except instead of requiring the user to hit a button, they appear automatically when the mouse pointer reaches the title bar. Certain menus can be "pinned" to the screen and remain there until they are specifically asked to disappear, making them pin-up menus. When a specific area of the screen, perhaps indicated by an icon, is selected, pop-up menus show up. However, they are only visible while the mouse button is depressed.

Circular arrangement of the items is another method for selecting from a menu. Since the pointer is positioned in the circle's centre, getting to any of the options takes the same amount of time. This has the advantages that all things can have a bigger target area, making selection easier and all items have the same selection time because the pointer is equally spaced from all of them.

Buttons

Buttons are discrete, separate areas on a display that the user can pick to activate particular functions. Because these areas are intentionally designed to mimic the push buttons on a control panel, they are known as buttons. When a button is "pushed," a command is initiated; often, a small icon or textual label indicates what the command entails. In addition, buttons can be used to select options on a web form or to show status information, such as whether or not the current font in a word processor is italicised.

(c) Amity University Online

Notes

Toggle buttons of this type can be arranged in groups so that the user can choose one feature from a list of alternatives that are mutually exclusive, such the font size in points. Since the collection operates similarly to the vintage mechanical control buttons seen in automobile radios, these are known as radio buttons. A series of toggle buttons can be used to show the on/off status of a collection of options if they are not mutually exclusive, such as font attributes like bold, italics and underlining. Sometimes this kind of button collection is called a "check box."

Toolbars

Numerous systems feature a set of tiny, icon-adorned buttons that are positioned at the top or side of the window, providing frequently used functionality. This toolbar serves the same purpose as a menu bar, but more functions can be shown at once because the icons are smaller than the corresponding text. Although the toolbar's content is frequently set, users can typically customise it by selecting one of several predefined toolbars to display or by altering which functions are made visible.

Palettes

There are various modalities in which interaction can occur in application programs. The distinguishing feature of modes is that the way they perceive actions—like mouse movements or keystrokes—varies depending on the mode. For instance, keystrokes can be interpreted as actions to do file manipulation (command mode) or as actions to place characters in the document (insert mode) while using the basic UNIX text editor vi. If the person is unaware of the mode that they are in, problems can arise. Palettes are a way to show the user the available modes as well as the active mode. Typically, a palette consists of a group of icons that represent the different modes' respective purposes. An artist's palette for paint would be analogous to a set of icons in a drawing package, designating the colour or pattern of pixels used to fill in objects. Palettes can be made by the user using menus or toolbars in certain systems. Pull-down menus may allow the user to "tear off," converting the menu into a palette that displays the menu items. When it comes to toolbars, he might be able to pull them off of their typical spot and arrange them wherever on the screen. Tear-off menus are typically those that have a lot of graphics on them, such colour selection or line-style in a drawing program.

Dialog Boxes

Dialogue boxes are information windows that the system uses to alert the user to crucial information, such as a potential error or a warning to help them avoid making a mistake. As an alternative, they are employed to start a user-system subdialog for a single, highly specialised activity that is often integrated into a bigger one. For instance, the majority of interactive programs require the user to create a file that needs to be identified and kept in the filing system. A dialogue box can be used by the user to name the file and specify its location inside the filing system when the system or user wants to save it. The dialogue window will vanish after the save subdialog is finished. Dialogue boxes are used to factor out auxiliary task threads from the primary task dialogue, just as windows are used to separate the various threads of user-system dialogue.

1.2.11 Interactivity

It's simple to concentrate on the visually distinguishable elements of an interface, such as the buttons, menus and text spaces, while ignoring the dynamics of the system and how it responds to user input. It's important to keep in mind that what makes an interactive system interactive is its interactivity. This is seen in numerous HCI domains. In the case of an airline

(c) Amity University Online

Notes

reservation system, for instance, the system can reflect back its interpretation of what you said and seek confirmation as long as it can consistently recognise yes and no. However, the recognition rate for speech recognition is too low to allow transcription from tape. Speech-based interaction is simpler, while speech-based input is more challenging. Furthermore, the most fascinating advancements in information visualisation are those that allow users to engage with a visualisation in real time, adjusting settings and witnessing the result.

Another important factor in defining the "feel" of a WIMP environment is interaction. It seems that every WIMP system has almost the same components: dialogue boxes, windows, buttons, menus, pointers and so on. But these components' specific behaviours vary throughout contexts as well as within a single one. For instance, we've already spoken about how pull-down and fall-down menus behave differently. While these appear identical, fall-down menus are more likely to be accidentally accessed (which is not unexpected given that the windowing environments that make use of them have mostly stopped using them!). Actually, a pull-down menu appears when you click on the menu bar in Windows and it stays there until an item is picked or the option is aborted. This is one of the key ways that MacOS and Microsoft Windows differ from each other when it comes to menus.

The machine primarily dictated the order of interaction in earlier computer systems. When the computer was ready, you took action. WIMP environments allow the user to take charge, offering a plethora of options and frequently many programs at once. The preemptive interface sections are an exception to this rule, when the system takes control from the user for a variety of reasons (e.g., a malfunction or the requirement for information to proceed). Modal dialogue bubbles are the main illustration of this. It frequently happens that after a dialogue box opens, the program won't let you do anything else until it has been closed or dismissed. Sometimes this will just stop the app from working, but you can still use other apps to accomplish things. In other situations, you are left with no options until the dialogue window is finished. One particularly unpleasant instance is when a dialogue box appears and asks you a question, maybe just to confirm an action, but it conceals the information you need to respond!

Modal dialogue boxes are required in some situations, such as when a significant error has been discovered or for some types of educational software. Generally speaking, though, preemptive components should be used sparingly in order to give the user the greatest amount of choice possible.

Dealing with errors also requires interaction. Ensuring that the user or the system can identify when errors have occurred is one method of handling errors. Users are able to fix mistakes if they can recognise them. Thus, the contact is successful overall even in the event of faults. This capacity to recognise and fix errors is critical on both a large and small scale, involving keystrokes and button presses. For instance, if you have written a client and are anticipating a response, you can make a note in your diary for the day you anticipate hearing back. You should send a reminder or give them a call after the due date if the recipient fails to respond or the letter disappears in the mail.

1.2.12 Paradigms

An interactive system's principal purpose is to enable the user to accomplish certain goals in a given application domain; in other words, it needs to be functional. Therefore, the following two open questions are put to the interactive system designer:

1. How can an interactive system be developed to ensure its usability?
2. How can the usability of an interactive system be demonstrated or measured?

© Amity University Online

Notes

Using successful interactive systems as examples can help to address these issues because it is widely accepted that these systems improve usability and can thus be used as models for the creation of new products.

We think that compared to earlier generations of interactive systems, ours is a more usable design. Furthermore, we think that there is still much space for improvement in the future when it comes to creating more user-friendly solutions. As was previously said, significant advancements in computer technology have raised machine power and expanded the bandwidth available for human-computer communication. However, the impact of technology by itself is insufficient to improve its usefulness. The ability to accommodate and enhance human strength through the thoughtful and innovative application of technology has become increasingly important as our machines have grown in power. The majority of the time, interaction paradigms have relied on technology advancements and innovative ways to use them to improve interaction.

Let's examine some of the most significant historical developments in interactive design. What is noteworthy in this instance is that the methods and designs discussed are acknowledged as significant advancements in interaction, even though it can occasionally be challenging to come to a consensus around the cause of the achievement. Even more difficult to foresee in advance are the new paradigms. Exploratory designs have frequently given rise to new paradigms that are later recognised as having established a new foundation for subsequent design.

Paradigms for Interaction

Time sharing

The major developments in computing during the 1940s and 1950s came from new hardware technologies. Vacuum electron tubes were used in place of mechanical relays. Transistors took the place of tubes and integrated chips replaced transistors, boosting the overall computer capability by orders of magnitude. By the 1960s, it was clear that the rapid increase in computing power would be wasted if there wasn't a corresponding boom in creative thinking about how to use it. J. C. R. Licklider, who went on to head the Information Processing Techniques Office of the US Department of Defense's Advanced Research Projects Agency (ARPA), was one of the foremost proponents of research into human-centered technologies. Licklider aimed to fund a number of research institutes around the country to promote fresh perspectives on the optimal uses of the rapidly advancing computing technology.

The idea of time sharing, which allows a single computer to serve several users, was one of the most significant contributions to come from this renewed focus on research. The human (or more correctly, the programmer) was limited to batch sessions in the past. Complete jobs were sent to an operator on paper tape or punched cards and the operator would run each one separately on the computer. Programming became a fully interactive endeavour in the 1960s thanks to time-sharing systems, which also gave rise to a subculture of programmers known as "hackers"—single-minded experts of detail who enjoyed deciphering complexity.

Even though the initial interactive time-sharing systems were just intended to enhance the early hackers' programming skills, they represented a major advancement in computer applications intended for human use. A truly interactive exchange between programmer and computer was conceivable, as opposed to depending on a model of interaction as a pre-planned activity that produced a comprehensive set of instructions laid out for the machine to follow. With each user, the computer could now be perceived as a committed partner and

Notes

the enhanced information flow between the two made it possible for the human to work with the computer in a more flexible and responsive manner. In fact, genuine human–computer contact was now feasible with the introduction of time sharing.

Video Display Units

Researchers began experimenting with the idea of using a video display unit (VDU) to present and manipulate computer-generated visuals as early as the mid-1950s. These screens could be a better medium for presenting large amounts of strategic information for quick assimilation than paper printouts. Early display screen image applications were created for military use, most notably for the US Air Force's Semi-Automatic Ground Environment (SAGE) project. The potential of visual graphics was not realised, however, until 1962, when Ivan Sutherland, a young graduate student at the Massachusetts Institute of Technology (MIT), stunned the established computer science world with his Sketchpad program. According to Howard Rheingold's book Tools for Thought: A History of Computing,

Sketchpad allowed a computer operator to use the computer to create, very rapidly, sophisticated visual models on a display screen that resembled a television set. The visual patterns could be stored in the computer's memory like any other data and could be manipulated by the computer's processor. . . . But Sketchpad was much more than a tool for creating visual displays. It was a kind of simulation language that enabled computers to translate abstractions into perceptually concrete forms. And it was a model for totally new ways of operating computers; by changing something on the display screen, it was possible, via Sketchpad, to change something in the computer's memory.

Sketchpad illustrated two key concepts. Initially, computers were not limited to data processing. By allowing the user to see and work with many representations of the same information, they may increase their capacity to abstract away from specific levels of detail. Such abstractions were not restricted to representations in terms of obscure bit sequences stored in the back of a computer's memory. It would be better to make the abstractions genuinely visual. The information in the computer was made more palatable for human consumption in order to improve human connection. Rather than forcing a human to speak more like a machine, the computer was designed to speak a more human language. Second, Sutherland's efforts showed how significant a single creative mind's contribution—when combined with a relentless drive to carry an idea through to completion—could make to the entire computer history.

Programming Toolkits

Since the early 1950s, Douglas Engelbart has aimed to augment human problem-solving abilities using computer technology. When Engelbart was a graduate student at the University of California, Berkeley, he had the notion to teach people using computers. The idea that unsophisticated people may truly learn from a computer stood in sharp contrast to the prevalent belief among his peers that computers were intentionally sophisticated devices that were only accessible to those with high intelligence. In the 1960s, Engelbart's committed research team at the Stanford Research Institute attempted to realise the manifesto outlined in a 1963 paper.

> By 'augmenting man's intellect' we mean increasing the capability of a man to approach a complex problem situation, gain comprehension to suit his particular needs and to derive solutions to problems. . . . We refer to a way of life in an integrated domain where hunches, cut-and-try, intangibles and the human 'feel for the situation' usefully coexist with powerful concepts, streamlined terminology and notation, sophisticated methods and high-powered electronic aids.

Notes

Word processing and the mouse are two examples of concepts that Engelbart's team created at the Augmentation Research Centre, although they weren't widely adopted for commercial use until decades later. In the fall of 1968, he presented a live demonstration of his oNLine System (NLS, later dubbed NLS/Augment) to an enthralled audience of computer sceptics at the Fall Joint Computer Conference in San Francisco. The interaction strategies used in NLS are not as important to us at this time because many of them will be covered in a later section. The technique Engelbart's team used to build their incredibly inventive and potent interactive systems using the comparatively subpar technology of the 1960s is what matters in this case.

Engelbart described how people approach difficult intellectual tasks in a similar way to a carpenter who uses a good set of tools to create intricately beautiful woodwork. Having the appropriate toolbox was key to creating computing equipment that improved human problem-solving abilities. Taking this advice to heart, the programmers in his team focused on creating the tools necessary to create increasingly intricate interactive systems. Bootstrapping is the concept of creating computer system components that can be combined to create a more complicated system; it has been applied widely in the computing industry. Programming toolkits are powerful because they enable the fixed composition of tiny, well-understood components to generate larger tools. After these more complex tools are comprehended, the process can proceed with their composition with further tools.

Personal Computing

For individuals with strong computational abilities, programming toolkits offer a way to significantly boost productivity. Engelbart's vision, however, was not limited to those with computer literacy. Computing power aimed at the general public, whether or not they were computer literate, emerged in the 1970s. The graphical programming language LOGO for kids was one of the first examples of how the hacker's formidable tools could be made available to the average computer user. Seymour Papert, the inventor, set out to create a language that was simple enough for kids to learn. He created a computer-controlled mechanical turtle with colleagues from MIT and other places that traced their movement by dragging a pen along a surface.

A young child might easily imagine themselves "inside" the turtle and give it instructions to trace out basic geometric patterns like squares and circles. One may educate the turtle to draw increasingly complex figures by having the child/programmer type in English phrases like "Go forward" or "Turn left." No matter how powerful a system is, it will always be more powerful if it is easier to use. Papert illustrated this important principle for interactive system development by transforming the graphical programming language into a model that kids could comprehend and use.

Papert and Engelbart's contributions had a significant impact on Alan Kay. He saw that the success of a system like NLS depended on its ability to be just as user-friendly for inexperienced users as LOGO. His vision for the future of computing in the early 1970s was represented by compact, potent devices called personal computers that were intended for lone users. Kay collaborated with the initial group of researchers at the Xerox Palo Alto Research Centre (PARC) to integrate Smalltalk, a potent and straightforward visual programming environment, onto the rapidly emerging personal computing hardware. There is growing difficulty in differentiating between a workstation or personal computer and a mainframe as technology advances. Kay's concept of the ideal handheld personal computer, which he named the Dynabook, dates back to the mid-1970s and surpasses even current technology.

(c) Amity University Online

Notes

Window Systems and the WIMP Interface

Following the introduction of personal computers and its enormous commercial success, efforts to improve computing technology's usability were directed at the individual user who interacted with the computer to accomplish a task. Given their capacity for multitasking, humans often find themselves breaking from one task to focus on another related task while working on a task at hand. Such conventional working pattern does not apply to a personal computer system that requires the user to proceed through all of the tasks required to accomplish a goal, from start to finish, without any breaks. The personal computer needs to be just as adaptable as a human being when it comes to "changing the topic" in a conversation.

However, meeting the demands of an alternative user job is not the sole prerequisite. Since computer systems typically respond to input from the user, a meandering conversation started by the user can be easily accommodated. It gets harder for the user to keep track of the overlapping threads of activity when he participates in multiple plans of action over an extended period of time. Therefore, in order for the user to discern between each thread of dialogue, the computer dialogue partner must convey the context of each thread.

Using a display device to physically divide the presentation of the several logical strands of a user-computer discussion is one presenting method for accomplishing this dialogue partitioning. The common mechanism connecting these logically and physically distinct display regions is the window.

These days, WIMP interfaces—windows, menus, icons and pointers—are used often for interaction. With the release of the 8010 Star Information System by Xerox Corporation in April 1981, these interface devices made their debut in the commercial sector. However, a large number of the interaction strategies that underpin a windowing system were employed by Engelbart's group at Xerox PARC in the Alto experiment, which was an experimental forerunner to Star.

Hypertext

As we may think, Vannevar Bush, the US war effort's top scientific executive at the time, wrote an article for The Atlantic Monthly in 1945. During World War II, Bush oversaw a team of more than 6,000 scientists that significantly advanced scientific research. He noticed that one of the main consequences of these intensive research efforts was that it was getting harder and harder to stay up to date with the expanding corpus of scientific information that was published in the literature. He believed that those who were able to keep up with the ever-increasing influx of information would benefit the most from this technological revolution. In order to achieve this, he presented a novel and futuristic information storage and retrieval device known as the memex. It was built in 1945 using technology that was entirely existent at the time and was intended to increase human capacity to store and retrieve related knowledge by simulating our natural ability to form arbitrary associative links.

The memex was essentially a desk that could create and store enormous amounts of photographic copies of information that was documented. Apart from its massive storage capacity, the memex could maintain linkages between sections of various documents. This would make the information stored seem like a massive, linked web of data, much to how many people believe the human brain stores information. A tool that explicitly recorded that information would be extremely helpful in the setting of scientific writing, where it is frequently exceedingly difficult to keep track of the origins and interrelationships of the ever-growing corpus of research.

© Amity University Online

Notes

The memex's concepts had an equal impact on another follower, even though his hopes have not yet come true to the same degree as those of Kay and Engelbart. Another graduate student and dropout whose research interests were permanently altered by the invention of the computer was Ted Nelson. Nelson embarked on a lifelong quest to create Xanadu, a potentially revolutionary global publishing and information retrieval system based on the concept of interconnected, non-linear text and other media forms, after an unsuccessful attempt to create a machine language equivalent of the memex on early 1960s computer hardware. A conventional paper is read in a straight line, from start to finish. However, there are frequently suggestions or footnotes in that text that nudge the reader to stray into a more in-depth subject. This haphazard and related browsing task is not well supported by the linear information format. Bush's memex recommended keeping the real documentation's non-linear browsing pattern. In the middle of the 1960s, Nelson created the term "hypertext" to describe this non-linear text structure.

The first hypertext systems entered the market about twenty years after Nelson initially used the phrase. Hypermedia (or multimedia) is the term used to describe non-linear storage of all forms of electronic media, to represent the usage of such non-linear and associative linking systems for more than only textual information retrieval and storage.

The World Wide Web

The world wide web, also known as simply the web or WWW, is arguably the most important recent development in interactive computing. The web is based on the internet and provides a user-friendly, mostly graphical interface to information, concealing the underlying complexity of addresses, transmission protocols and remote data access.

All that is the internet is a network of computers connected by some kind of data connection, be it a high-bandwidth optical link or a sluggish phone line and modem. Common data transmission protocols (TCP/IP) and addressing systems (IP addresses and domain names) are used by all internet-connected machines to communicate with one another. In theory, this allows anyone to read anything from anywhere as long as it complies with the protocol. The web expands on this by utilising a global naming scheme (uniform resource locators, or URLs), a standard markup syntax (like HTML) for information page layout and its own layer of network protocol (http). Text, colour photos, videos, sound and—most importantly—hypertext links to other web sites can all be found on web pages. This means that anyone with access to a computer linked to the internet can "publish" hypermedia documents.

When Tim Berners-Lee was employed at CERN, the European Particle Physics Laboratory in Geneva, he came up with the idea for the World Wide Web in 1989 as a way to facilitate the global sharing of scientific data produced at CERN and to facilitate communication between physicists. The first text-based web browser was made available in 1991. Early in 1993, a number of graphical web browsers were released, the most notable of which was Mosaic, created by Marc Andreesen at the National Centre for Supercomputer Applications (NCSA) in Champaign, Illinois. This marked the beginning of the web's explosive expansion, which saw it take over internet traffic and alter people's perceptions of computers. Only Berners-Lee, among the other "heroes" of interactive computing mentioned in this section, has attained broad public recognition.

Advancement in HCI

The multidisciplinary discipline of human-computer interaction (HCI) is concerned with the development, assessment, and use of interactive computing systems for human use. Significant progress in HCI has been made over the years as a result of study, innovation in technology, and changing user needs. Among the noteworthy developments in HCI are:

(c) Amity University Online

Notes

Natural user interfaces (NUIs):

Natural user interfaces (NUIs) let people engage with computers using gestures, speech, touch, and gaze. By enhancing accessibility and intuitiveness of interactions, technologies such as voice recognition, motion sensing, and touchscreens have completely changed HCI.

Pervasive Computing:

Pervasive computing, also name for ubiquitous computing, is the integration of computer technology into commonplace items and surroundings. The Internet of Things (IoT) and wearable technology have proliferated as a result of this paradigm shift, enabling seamless and context-aware computing experiences.

Virtual Reality (VR) and Augmented Reality (AR):

Immersion and interactive experiences are now more possible thanks to AR and VR technologies. AR and VR applications are changing how people interact with digital material and settings in a variety of contexts, including gaming, entertainment, training, education, and healthcare.

Interfaces for Tangible Users (TUIs):

Through the direct interaction of physical things, TUIs allow users to manipulate digital information. Physical objects are integrated into the user interface. By bridging the gap between the real and virtual worlds, this method offers innovative paradigms for interaction that foster creativity, teamwork, and education.

Social Media Processing

The study of social computing looks at how technology may support community development, cooperation, and social connections. Users may now interact, share, and collaborate globally thanks to social media platforms, online forums, and collaborative technologies that have completely changed social dynamics and communication patterns.

Interfaces that are Personalised and Adaptive:

The creation of adaptive and personalised interfaces that customise content, functionality, and interactions to each unique user's preferences and traits has been made possible by developments in machine learning, data mining, and user modelling. Personalised experiences that are relevant and appropriate for the context improve user pleasure, engagement, and productivity.

Inclusive Design and Accessibility:

In order to guarantee that interactive systems are useable by individuals with a variety of needs, disabilities, and skills, accessibility and inclusive design concepts are becoming more and more important in HCI research and practice. In addition to helping those with disabilities, designing with accessibility enhances usability and user experience for all users.

Responsible & Ethical Design:

Professionals in HCI are becoming more and more concerned with the ethical consequences of technology use and design. This covers factors including algorithmic bias, data ethics, privacy, security, and digital well-being. The goals of ethical and responsible design are to reduce possible risks and encourage beneficial effects of technology on society.

(c) Amity University Online

Notes

Interfaces between many Devices and Platforms:

HCI has developed to meet the difficulties of creating seamless and consistent user experiences across many devices and interaction settings as a result of the proliferation of platforms and devices. With the help of cross-platform frameworks, adaptive layouts, and responsive design, designers can produce unified experiences that change depending on the screen size, input method, and device capabilities. The design and development of interactive systems are being influenced by these developments in HCI, which improve user experiences, boost productivity, and enable people to engage with technology in more significant and revolutionary ways. Future digital experiences and human-computer interaction will be greatly influenced by human-computer interaction (HCI) as technology develops.

Future of HCI

The future of Human-Computer Interaction (HCI) is marked by continued innovation, driven by advancements in technology, evolving user needs, and emerging societal trends. Here are some key directions and trends that are shaping the future of HCI:

Ubiquitous Computing and Internet of Things (IoT):

The integration of computing technology into everyday objects and environments will lead to the proliferation of smart devices and interconnected systems. HCI will focus on designing user experiences that seamlessly span across multiple devices and IoT ecosystems.

Wearable Technology:

There will be an increase in the use of wearable technology, including augmented reality glasses, fitness trackers, and smartwatches. In order to address the particular limitations and affordances of wearable technology, HCI will investigate novel interaction modes and design approaches.

AI and Machine Learning:

Personalised and adaptive interfaces that anticipate user demands, behaviours, and preferences are made possible by AI and machine learning, which will become more and more significant in the field of HCI. Predictive analytics, machine vision, and natural language processing will improve user experiences and automate repetitive chores.

Social and Collaborative Computing:

HCI will keep emphasising how to make digital environments more conducive to community creation, cooperation, and social interactions. Online communities, collaborative tools, and social media platforms will all change to facilitate deeper and more meaningful user interactions.

Ethical and Responsible Design:

Professionals in HCI will give ethical issues top priority when designing and developing new technologies. This entails tackling concerns like data ethics, privacy, security, algorithmic bias, and digital well-being to guarantee that technology maximises benefits while minimising potential drawbacks.

Accessible and Inclusive Design:

The design of inclusive and accessible interfaces that serve users with a range of needs, disabilities, and abilities will receive more attention in HCI. It will become essential to HCI practice to design for accessibility in order to make technology useful by all.

(c) Amity University Online

Notes

Human-Centered Design in AI Systems:

HCI will concentrate on creating human-centered AI that enhances rather than replaces human intelligence as AI systems become more commonplace. Building trust and cooperation between humans and machines will require creating visible, interpretable, and accountable AI systems.Cross-Cultural and Multilingual Interfaces:

HCI will address the challenges of designing interfaces for users from diverse cultural backgrounds and linguistic communities. Designing interfaces that are culturally sensitive, inclusive, and adaptable to different user preferences and expectations will be essential for global accessibility and adoption.

Sustainability and Green HCI:

Environmental sustainability will play a bigger role in HCI design and development. The development of eco-friendly materials, sustainable practices, and energy-efficient systems that reduce the environmental impact of digital technology will be guided by the concepts of green human-computer interaction.

In conclusion, human-centered design, technological innovation, and a dedication to moral, inclusive, and sustainable behaviours will define the future of human-computer interaction (HCI). HCI will continue to influence the development of interactive systems and enable people to interact with technology in more significant and revolutionary ways by embracing these trends and difficulties.

Summary

- Human-Computer Interaction (HCI) is a multidisciplinary field centered on designing computer systems that facilitate effective and satisfying interactions with users. It involves user-centered design, emphasising usability, accessibility and positive user experience.
- Sensory channels are crucial for creating a holistic and immersive user experience. Designing interfaces that effectively utilize these channels can enhance usability and engagement.
- Touch gives us important information about the surroundings. It can serve as a warning since it alerts us when we come into contact with hot or cold surfaces.
- Memory work together in the processes of encoding, storage and retrieval, influencing how individuals perceive, process and remember information. Understanding these memory systems is crucial in fields like psychology, cognitive science and human-computer interaction for designing effective interfaces and experiences.
- Ergonomics, also known as human factors or human ergonomics, is the scientific discipline that focuses on understanding the interactions between humans and their environments, as well as the design of tools, equipment, systems and workspaces to optimize human well-being and overall system performance.

Glossary

- HCI: Human Computer Interaction
- IxD: Interaction Design
- PC: Personal Computer
- WIMP: Windows, Icons, Menus, Pointer
- GUI: Graphical User Interface
- CRT: Cathode Ray Tube

(c) Amity University Online

Notes

- LCD: Liquid Crystal Display
- RTF: Rich text format
- ASCII: American standard code for information interchange
- NCSA: National Centre for Supercomputer Applications

Check Your Understanding

1. Which sensory channel is primarily associated with the input/output modality of viewing graphical information on a computer screen?
   a. Hearing b. Touch
   c. Sight d. Visual
2. What type of input/output channel is characterised by the perception of sound and is commonly used for providing auditory feedback in computer systems?
   a. Touch b. Visual
   c. Hearing d. Sight
3. The input/output channel related to the physical interaction with a touchscreen or touchpad on a device is referred to as:
   a. Sight b. Angle
   c. Touch d. Hearing
4. In the context of spatial input/output, which term is associated with the three-dimensional position and movement of input devices?
   a. Visual b. Angle
   c. Hearing d. Sight
5. What does HCI stand for?
   a. Human-Computer Integration b. Human-Computer Interface
   c. High-Contrast Imaging d. Human-Centric Interaction

Exercise

1. Write short notes on HCI
2. Define various Input/Output Channels.
   a. Sight b. Hearing c. Touch d. Visual
   e. Angle
3. Define various types of Memory:
   a. Sensory b. Short Memory c. Long Term Memory
4. Define various design thinking principles.
5. What do you mean by prototyping and iterative design?

Learning Activities

1. Describe how you would incorporate user feedback and iterate on the design of the task management application. Explain the importance of an iterative design process in improving the user experience.
2. Identify and describe two accessibility considerations that are crucial for ensuring a positive user experience for a diverse user base. Explain how you would address these considerations in your design.

(c) Amity University Online

Module - II: Design and Software Process

Notes

Learning Objectives

At the end of this module, you will be able to:

- Understand basics of interaction design
- Define HCI in software process
- Understand design rules
- Define iterative design and prototyping
- Analyse software life cycle

Introduction

Understanding is a key component of HCI, which is how people interact with technology. However, doing and creating things—design—represents a significant portion of HCI.

Let's now consider interaction design. Keep in mind that we are considering the interaction itself, not merely the design of interactive systems. An electric stapler was just purchased for an office. To staple papers together, you go to the stapler because it is heavy and connected to the mains electricity. When someone needed to staple something in the past, they would bring the stapler to their desk and store it there until someone else needed it. Letters might be written, printed, stapled, followed by another letter, written, printed, stapled and so on. Rather than writing and printing until you have a stack of items to staple, you write and print until you have to take the letters across the office to be stapled. The stapler has an impact on the entire interaction pattern.

Therefore, interaction design includes more than just creating an object, such as a computer program or physical gadget; it also involves determining how that object will impact people's daily tasks. In addition, the artefacts we provide to individuals include online help systems, manuals and tutorials in addition to these programs and devices. Sometimes we find that there's no need for a new system at all; in those situations, we might just recommend utilising the current tools in a different way.

For this reason, it might be more beneficial to consider creating interventions rather than systems or artefacts. The end result of a design exercise is our intervention to alter the current situation—ideally, for the better!

2.1 Interaction Design Basics

Interaction design (IxD) is a specialised field within Human-Computer Interaction (HCI) that concentrates on shaping user experiences with digital systems. Fundamental aspects encompass user-centered design, the five dimensions of design (visceral, behavioural, reflective, cultural and functional), affordances, signifiers, feedback mechanisms, mapping, consistency, usability principles, wireframes, prototyping, user testing and the incorporation of responsive and emotional design elements.

2.1.1 Design

What then is design?

The phrase "achieving goals within constraints" is a basic definition.

© Amity University Online

Notes

While it does not encompass all aspects of design, this helps to highlight certain key points:

Goals

What is the intended use of the design that we plan to create? For whom is it intended? For what reason do they desire it? When creating a wireless personal movie player, for instance, we might consider the needs of young, affluent consumers who want to see the newest films on the go, download them for free and maybe even share the experience with a few friends.

Constraints

What kind of materials are required? What guidelines do we need to follow? What is the potential cost? How long do we have to work on it? Exist any concerns about health and safety? Is it necessary for the personal movie player to be waterproof? Must we download films using the current video standards? Should we incorporate copyright protection?

Naturally, we can't always accomplish what we want to within the limitations. Thus, among the most crucial aspects of design may be:

Trade-off

Deciding which objectives or limitations can be eased in order to achieve others. For instance, we might discover that the most stable image while moving would come from an eye-mounted video display, similar to those used in virtual reality. But you couldn't show pals this and it may be risky if you were viewing a suspenseful scene in a movie while crossing the street.

In design, the most thrilling moments are frequently those that arise from a completely new concept that satisfies multiple seemingly incompatible requirements. Nonetheless, the more prevalent ability required in design is the ability to acknowledge conflict and select the best trade-off. The temptation is to concentrate on and optimise for a single purpose, then modify the design to only meet the restrictions and other objectives. Rather, the best ideas come from designers who are aware of the trade-offs and the variables that influence them. Ironically, the more extreme answers also become more obvious if you concentrate on the trade-off itself.

The Golden Rule of Design

A portion of the knowledge we require relates to the specific design problem's conditions and context. But there are also more general ideas to comprehend. Even though the designs we create could differ, the basic materials are frequently the same. This brings up the design golden rule:

Understand your Materials

This is evident in the case of a physical design. Compare two chairs: one with a wooden frame and the other with a steel frame. They differ greatly in that the steel frames are frequently made of thin L or H sections or tubular steel. Wooden chairs, on the other hand, have thicker, sturdy legs. A wooden chair created from a metal chair's design would break, while a metal chair made from a wooden chair's design would be too heavy to carry.

The two most obvious materials for human-computer interaction are the human and the computer. That is, we have to:

- understand computers – limitations, capacities, tools, platforms
- understand people – psychological, social aspects, human error

(c) Amity University Online

Notes

To err is human

To consider humans as "materials" may seem dehumanising, if not insulting. In actuality, the contrary is true: most designs treat physical materials better than people. This is especially clear in the case of failures.

The headlines read, "An industrial accident causes millions of pounds' worth of damage; an air crash claims a hundred lives; thousands of patients are recalled to the hospital after systematic mistreatment is discovered." The public investigations come to the following conclusion after a few months: human error in the use of technical equipment. Although "operator error" is commonly understood when the term "human error" is used, most disasters have intrinsic flaws in the installation or design of the human interface. Poor interfaces are difficult to use and might be slow. Inadequate user interfaces are costly and deadly.

Humans are fallible beings. This is human nature, not "human error," which is a pretext to hide behind in accident reports. We are not perfect, consistent beings; we frequently commit mistakes, omissions and slip-ups. A structure falls when a concrete lintel cracks. Does "lintel error" appear in the headlines? No. Because concrete lintels naturally fracture when stressed, it is the duty of the architect and engineer to make sure that a building only places the lintel under reasonable amounts of stress. In a similar vein, since human error is inevitable, systems should be built to minimise its repercussions and lower the probability that it will occur.

The standard approach to a confusing interface element is to add more text to the handbook. Contrary to concrete lintels, people may become "stronger" and are amazingly adaptive, so while improved training and documentation are vital, they are not a cure-all. Errors will arise from arcane or inconsistent interfaces under pressure.

When designing with physical materials, you must consider the potential sources and locations of failures in order to reinforce the structure, incorporate safety measures, or provide redundancy. Likewise, treating the human with the same care and attention as a piece of concrete or steel makes it clear that you also need to comprehend the ways in which human failures happen and design the remainder of the interface accordingly.

The Central Message – the user

It's common knowledge that the success of the many HCI techniques doesn't stem from their technical merits, but rather from the fact that they force designers to concentrate on the user. The fundamental principles of interaction design are to prioritise the user, maintain them at the centre and remember them at the conclusion.

2.1.2 Process of Design

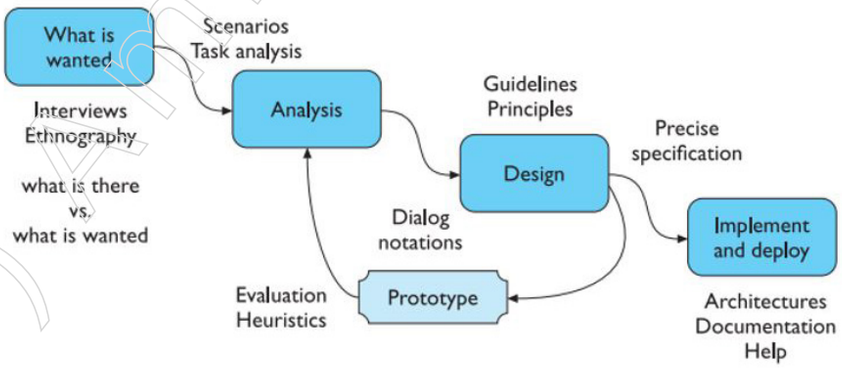


Figure 34: Interaction design process

Notes

It's common knowledge that the success of the many HCI techniques doesn't stem from their technical merits, but rather from the fact that they force designers to concentrate on the user. The fundamental principles of interaction design are to prioritise the user, maintain them at the centre and remember them at the conclusion.

Requirements – what is wanted

Identifying the precise needs is the first step. Finding out what is going on right now is frequently required as a prelude to this. For instance, how are films currently watched by people? What kind of household appliances do they utilise right now?

In HCI, a variety of methods are employed to do this, including videotaping, interviewing, examining the papers and tools they use on a daily basis and direct observation.

Analysis

To highlight important points and facilitate communication with following design stages, the findings from the observation and interview must be arranged in a certain manner.

Design

All of this has to do with design, but there's a crucial step when you go from wanting something to knowing how to get it done. To assist with this, a variety of regulations, guidelines and design principles can be applied.

Iteration and Prototyping

We cannot expect to get designs perfect the first time since people are complex. As a result, we must assess a design to determine its effectiveness and areas for improvement.

Implementation and Deployment

After we are satisfied with our design, it is time to make it and implement it. This will entail writing code, maybe building hardware, creating manuals and documentation and doing anything else needed to create a functioning system that can be shared with others.

It is simple to assume that identifying and fixing usability issues is the main objective, particularly during the iteration rounds. But as you work with actual designs, you quickly discover that the true challenge is not identifying flaws—that is simple—or figuring out how to correct them—that may not be too tough. Rather, the real question is: which usability issues are worth addressing? If you ever come across a system that appears to be flawless, it is most likely poorly designed. This isn't to say that the design is poor; rather, it is the result of excessive effort being put into the process. As with all trade-offs, it might be conceivable to come up with drastically different approaches that are inexpensive to adopt yet have a significant impact. But it's best not to plan, thinking that these inspiration bursts will happen when needed!

User Focus

The intended user or users must be the first consideration in any interface design effort. This is frequently expressed as:

know your users

This is sometimes phrased as "know thy user," even though it was originally written as "know the user," because it sounds almost like a command. Observe also the slight ambiguity regarding the user or users—a large portion of conventional user interface design has concentrated on a single user.

(c) Amity University Online

Notes

Consider a method for stock control. Is there simply one user in the system when the warehouse manager asks how many six-inch nails are in stock? For what reason did he do this? Maybe a salesperson is inquiring as to whether the organisation can complete an order for 100,000 six-inch nails in less than two weeks after being asked to deliver them. Therefore, the warehouse manager, the salesperson and the client are involved in the process of looking at the stock control system. While his supervisor is on vacation, the assistant warehouse manager must update the stock levels in order for the auditors to provide an appraisal of the company's assets, including the stock that is currently in hand.

Stakeholders are the numerous individuals who a system affects over time, either directly or indirectly. It goes without saying that you need to set parameters for who you should take into consideration when tracing the shaky connections between persons. This is mostly dependent on the type of systems being created, but it also calls for common sense.

How then do you learn about your users?

Who do they represent?

Naturally, the first thing you should do is identify your user base. Are they seasoned computer users or newbies, young or old? It might not be clear who the users are, as we saw with the stock control system, so you might need to reevaluate this question as you learn more about the system and its surroundings. It gets more difficult to respond to this question when creating generic software, like word processors, because there are a wide variety of users with varying needs and preferences. Many websites that have a diverse pool of potential visitors face a similar issue. It is probably preferable to think of multiple distinct users, either in addition to or instead of trying to think of a generic user with generic abilities and goals, even though it may be tempting to attempt.

Most likely not the same as you!

It is simple to create a system with your own interests and capabilities in mind, as this assumes you are the primary user. Designers frequently remark, "But it's obvious what to do." Maybe it's clear to her! The fact that men developers predominate in many software businesses does not assist with this. Despite the fact that people differ greatly, women tend to be more sympathetic than men.

Converse with them.

Asking them is usually the best course of action because it might be difficult to understand another person's thoughts. Open-ended conversations, systematic interviews about their work or personal lives, or involving the potential users completely in the design process are some ways to accomplish this. Participatory design is the term for the final one of these. It is feasible to gain a thorough understanding of users' demands and work context by incorporating them in the design process from the beginning. It goes without saying that this results in superior designs. But there's another motivational influence that might be just as significant as the design excellence. Engaging users helps them to "own" the design and, once it's implemented, turn them into advocates. Remember that a system needs to be utilised in addition to being practical and useable.

Additionally, people might be able to tell you about actual events rather than just the ways that organisations recommend they should be handled. Since the real behaviours frequently conflict with company policy, you will need to gain their trust before you can get people to tell you this. But rather than the official narrative, it is usually these impromptu techniques that enable organisations to function!

© Amity University Online

Notes

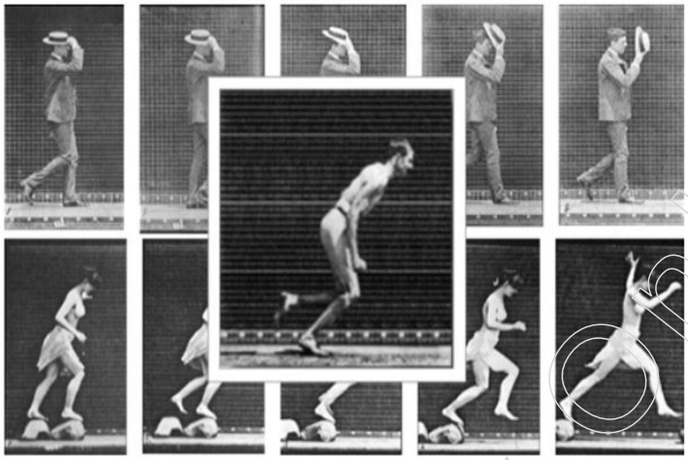


Figure 35: Eadweard Muybridge's time-lapse photography

Observe them.

While the information that individuals tell you is really important, it is not the complete picture.

Black-belt judo practitioners often give different answers when asked how they truly throw an opponent. When you walk, observe how your arms and legs move. It is more difficult than you may imagine! Even though humans have been running since the beginning of time, it wasn't until the 1870s, thanks to Eadweard Muybridge's groundbreaking time-lapse photography, that the true nature of human movement—including walking and running—became evident (see Figure above). Because it is so hard to reflect on intellectual pursuits, this becomes much more troublesome.

Any professional has extensive experience and is competent in their specialty. Even though a professor in the same field might not be able to accomplish anything, she is knowledgeable about the subject. These are distinct categories of expertise and understanding. People don't always know both, although they do sometimes. It's possible that the greatest painters and sports trainers are not the same people that make the best art critics or athletes.

It is crucial to pay attention to both what individuals say and do as a result. This could entail utilising a video camera or tape recorder, watching certain activities, or simply sitting down and making notes about their daily activities. It can be carried out informally or with the use of formal techniques like ethnography or contextual inquiry.

Users may occasionally be involved in this by being asked to do things like keep a journal or write down what they are doing when a 15-minute buzzer goes off. While it may seem similar to asking people what they do, the organised style enables them to provide a more precise response.

Examining the objects that people use and make is another method of learning about their activities. Take a look at an average office workstation. There are files, letters, papers, a computer, sticky notes and maybe a stapler. Some of these are informational, but they might just as well be in the filing cabinet and pulled out as needed if the information they contain were all that matters. It's not simply that Brian needs to reserve a table at a restaurant that's written on the sticky note on the edge of his screen that says "book table." He is reminded that something needs to be done because it is on his screen.

It is important to revisit and discuss the observations with the users in all of these observational methods rather than simply stopping at the observation. If they are shown,

(c) Amity University Online

Notes

they should be able to explain even if they had no idea what they were doing before. You will learn what they do and why from the observations.

Make use of your creativity.

It won't always be feasible to include as many consumers in your design exercise as you would like. It might be too expensive, difficult to schedule time with them (like a hospital consultant), or there might simply be too many of them (like the internet). You can attempt to imagine their experiences even if you are unable to involve actual people.

This is really risky right now! It would be simple to consider "this is what I would do if I were a warehouse manager." What matters is what the user would do, not what you would do if you were in their position. This calls for almost method acting. Consider managing a warehouse. What does he understand by the word "undo" on the menu?

Betty is 37 years old. She has been Warehouse Manager for five years and has worked for Simpkins Brothers Engineering for 12 years. She didn't go to university, but has studied in her evenings for a business diploma. She has two children aged 15 and 7 and does not like to work late. She did part of an introductory in-house computer course some years ago, but it was interrupted when she was promoted and could no longer afford to take the time. Her vision is perfect, but her right-hand movement is slightly restricted following an industrial accident three years ago. She is enthusiastic about her work and is happy to delegate responsibility and take suggestions from her staff. However, she does feel threatened by the introduction of yet another new computer system (the third in her time at SBE).

Figure 36: Persona – a rich description of Betty the Warehouse Manager

The persona is one technique that has shown to be quite effective in assisting design teams in creating user-focused designs. A persona is a detailed depiction of a fictional individual who embodies your primary user base. An illustration of Betty the warehouse manager's persona can be seen in the figure above. Multiple personae encompassing various jobs and target user types will be part of a design team's arsenal. The personae themselves will draw from observation, research of real users and other sources. The team might offer the question, "How would Betty react to this?" when a design solution is put out. Though this is crucial, the information is purposefully more than is strictly necessary. The team can only begin to envision Betty's behaviour when they begin to sense that she is a real person.

Scenarios

Rich narratives of interaction are what scenarios are for design. They are among the most adaptable and potent design representations, yet maybe being the most basic. Some examples are rather brief, such as "the user accidentally presses the "quit" button after intending to hit the "save" button, losing his work." Others are more concerned in setting the scene or providing background.

Brian would like to see the new film *Moments of Significance* and wants to invite Alison, but he knows she doesn't like 'arty' films. He decides to take a look at it to see if she would like it and so connects to one of the movie-sharing networks. He uses his work machine as it has a higher bandwidth connection, but feels a bit guilty. He knows he will be getting an illegal copy of the film, but decides it is OK as he is intending to go to the cinema to watch it. After it downloads to his machine he takes out his new personal movie player. He presses the 'menu' button and on the small LCD screen he scrolls using the arrow keys to 'bluetooth connect' and presses the 'select' button. On his computer the movie download program now has an icon showing that it has recognized a compatible device and he drags the icon of the film over the icon for the player. On the player the LCD screen says 'downloading now', with a per cent done indicator and small whirling icon.

(Amity University Online)

Notes

During lunchtime Brian takes out his movie player, plugs in his earphones and starts to watch. He uses the arrow keys to skip between portions of the film and decides that, yes, Alison would like it. Then he feels a tap on his shoulder. He turns round. It is Alison. He had been so absorbed he hadn't noticed her. 'What are you watching', she says. 'Here, listen', he says and flicks a small switch. The built-in directional speaker is loud enough for both Brian and Alison to hear, but not loud enough to disturb other people in the canteen. Alison recognizes the film from trailers, 'surprised this is out yet' she says. 'Well actually . . .', Brian confesses, 'you'd better come with me to see it and make an honest man of me'. 'I'll think about it', she replies.

Figure 37: Scenario for proposed movie player

An example of a situation for the personal movie player is shown in the above figure. Similar to the character, it may contain more detail than is necessary, but that detail adds to the events' realism. Although situations can be enhanced with drawings, simulated screen shots, etc., the figure only displays plain text. Storyboards are these drawings and illustrations that resemble the methods used in filmmaking to envision storylines.

In cases when the design incorporates tangible objects, the scenarios might function as a script to enact possible usage patterns. Consider a digital Swiss army knife, for instance, that has a tiny LCD screen and a toothpick for a stylus. The knife offers helpful advice from other Swiss army knife users and establishes a wireless connection with the internet through your phone. Get the two of them together at a party and you'll see how appealing this would be! Though it seems like a fantastic design concept, try acting out the use first. Use a Swiss army knife if you have one, or substitute a penknife if you don't. The LCD's suggestion indicates to "open the stone remover"; to do so, simply open the small LED that illuminates next to the right blade. It reads, "Now push the blade into the grommet's rubber." After completing this, search for the following instruction. One can see that you have a knife in your hand and your thumb is covering the screen. A voice interface might work better.

More information will allow you to ask questions like "what is the user intending now?" and "what is the system doing now?" when you have a detailed description of the user-system interactions. This can assist in confirming that suggested implementation architectures would function and that the design would make sense to the user.

Scenarios can also be utilised for:

- Communicate with others – users, clients, or other designers. Discussions about abstract concepts can lead to misunderstandings. It is much simpler to provide specific examples of application.
- Validate other models- It is possible to "play" a comprehensive scenario against a variety of more formal representations, like task models, dialogue models and navigation models.
- Express dynamics- While individual screenshots and images can give you an idea of a system's appearance, they cannot convey its behaviour.

Let's talk about how to characterise how people engage with a system. These are more intricate and entail hierarchies or networks. On the other hand, scenarios are linear; they show one possible path out of all the possible combinations.

Both positive and bad aspects of this linearity exist:

- Time is linear- Since time is a linear aspect of our existence, simple linear narratives are simpler for us to comprehend. We listen to and tell stories naturally.
- But no alternatives-In real interactions, decisions are made by systems as well as by individuals. These additional routes are not visible in a basic scenario. Specifically, it is simple to overlook the inadvertent actions that people might take.

(c) Amity University Online

Notes

Throughout the design process, scenarios are a valuable tool that can be used again and again. They aid in identifying goals, make suggestions for how users could interact with a potential design, verify that suggested implementations would function and produce test cases for the final assessment.

Navigation Design

We emphasized that the goal of design is the socio-technical intervention as a whole, not just a computer system or gadget. But as design develops, there comes a time when we do have to take these most concrete results into account.

Think of yourself as a word processor user. This will be something you perform with a purpose, in a specific social and physical setting. Now, though, our attention is on the computer system itself. You communicate on various levels:

- Widgets: You can use menus and buttons more effectively if you choose the right widgets and phrasing for each pick or action.
- Screens or windows: You must locate items on the screen and comprehend how the buttons are logically grouped.

Table 2: Levels of interaction

PC application	Website	Physical device
Widgets	Form elements, tags and links	Buttons, dials, lights, displays
Screen design	Page design	Physical layout
Navigation design	Site structure	Main modes of device
Other apps and operating system	The web, browser, external links	The real world!

- Navigation within the application: To grasp where you are in the interaction and what will happen when a button is pressed, you must be able to comprehend.
- Environment: Documents must be read by the word processor from disc; some may be located on distant networks. You switch between programs and might even copy and paste.

As the table above illustrates, you can view comparable levels in other application and device kinds.

There are certain distinctions. For instance, we have less control over how users access websites on the web, whereas buttons and displays on physical devices are always in the same arrangement regardless of their internal condition (though we may handle them differently).

In case you didn't already know, the first thing to do when thinking about an application's structure is to evaluate its intended usage.

- ❖ who is going to use the application?
- ❖ how do they think about it?
- ❖ what will they do with it?

This can therefore motivate the second step, which is to consider structure. There will be a framework to individual displays or device layouts, but that is for the next section. Here, we'll focus on two primary categories of issues:

- ❖ local structure

– looking from one screen or page out

Notes

❖ global structure

– structure of site, movement between screens.

Global structure – Hierarchical Organisation

Hierarchy is one approach to system rganize ion. Although functional boundaries, or distinctions between different types of items, are usually used to rganize this, other options include roles, user types, or even more esoteric divisions like educational system modules.

Logical groupings of screens, pages, or states are connected by the hierarchy. For instance, the high-level breakdown of a message system is provided in the figure below. In addition to being useful for aiding in design, this kind of hierarchy can also be rganize to rganize the system itself. This might mimic the layout of a web page or the menu system of a PC program, for instance.

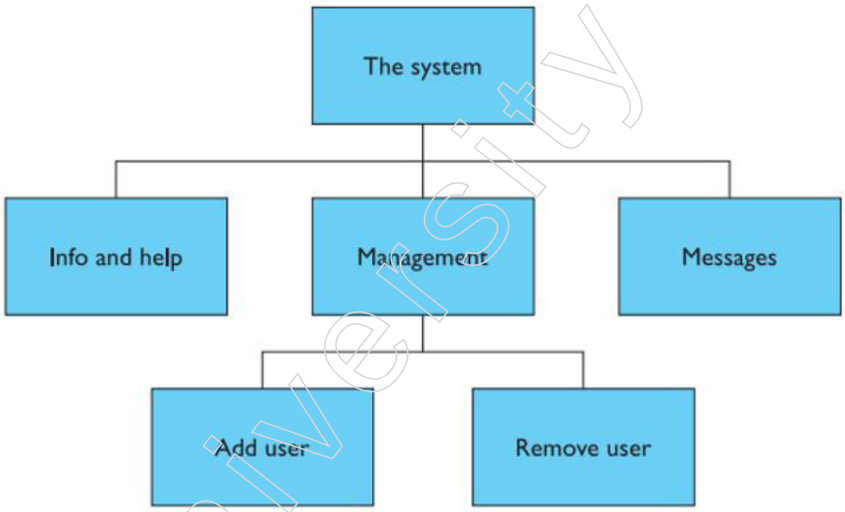


Figure 38: Application functional hierarchy

While ategorize any kind of information is challenging, there is evidence that some people find hierarchies to be more understandable than others. Various people may use various vocabulary and have distinct internal frameworks for their knowledge, which makes it challenging to ategori information or system functionality. This is one of the situations when having a thorough understanding of the intended users is crucial. It makes no sense to create a hierarchy that the designers are aware of but the users are not aware of, yet this is unfortunately all too often the case.

It Is unavoidable that not every user will grasp it precisely, no matter how well you think you have worded and ategorizedd things. This is because various users have different needs.

Additionally, there is evidence that navigating deep hierarchies is challenging; therefore, it is preferable to have wide top-level categories or to display multiple menu levels on a single screen or webpage. In this situation, Mille''s working memory capacity magic number of 7 ± 2 is frequently applied incorrectly. Many recommendations state that the number of options accessible at each level of the menu, or menu breadth, should be approximately seven. However, Mille''s finding does not apply to visual search; it only relates to working memory.

In fact, if the items are arranged so that the viewer can quickly locate the appropriate one, the ideal breadth can be rather large—perhaps 60 or more items for a web index page. Of course, more classification is needed in order to arrange the objects on the page. Here,

(c) Amity University Online

Notes

though, the classificatio''s naturalness—which may or may not depend on the use''s goal—is what matters most. An alphabetical list of all city names, for instance, would be quick to search for information on a certain city; but, a list arranged by area would be better suitable for other uses.

Global Structure – dialog

It might be sufficient to have a fully hierarchical structure in a static website or pure information system, possibly with next/previous connections between items in the same group. But i''s incredibly annoying to continually drill down from one area of the hierarchy to another for any system that requires action. There are usually methods to move from one place to another more rapidly. For instance, in a stock control system, it might be possible to navigate from a stock item to every order that is still pending for that item and from an order to the custome''s purchase record. These would all be directly accessible from one another even though they would be located in quite distinct areas of the applicatio''s hierarchical display.

In addition to these cross-links within hierarchies, there is clearly a flow of screens and commands that is unrelated to hierarchy when you get to specific interactions, such editing or removing a record. This pattern of interactions between a user and a system is called a““dialogu”” in HCI.

Take a look at this passage from a marriage ceremony:

Minister: Do you *name* take this woman…
Man: I do
Minister: Do you *name* take this man…
Woman: I do
Minister: I now pronounce you man and wife

Yo''ll see that while the broad structure of the ceremony is described, the names of the bride and groom are left blank. Thus, it provides a pattern of the partie'' interactions, but it is instantiated differently for every service. Human-computer dialogue is uniform, with specifics changing with each use of the program; there are general patterns of movement between windows in a PC application or between the major states of a device.

Remember that each scenario provided a single way to navigate the system. A complete system description must consider loops that return the system to the same screen as well as various paths through the system.

Using a network diagram that shows the main states or screens connected by arrows is a straightforward method. This may:

- show what leads to what
- show what happens when
- include branches and loops
- be more task oriented than a hierarchy.

The network architecture in Figure Application functional hierarchy below depicts the primary screens for adding or removing users from the messaging system. The basic flow between the states is depicted by the arrows. It is evident that accessing the““remove use”” or““add use”” screens is possible from the main screen. Presumably, you do this by clicking buttons or links, but w''ll leave the specifics of how they are shown to the screen designer. Additionally, we can observe that the system always goes back to the main screen after the““add use”” screen, but that there is an additional confirmation screen following the““remove use”” screen.

Notes

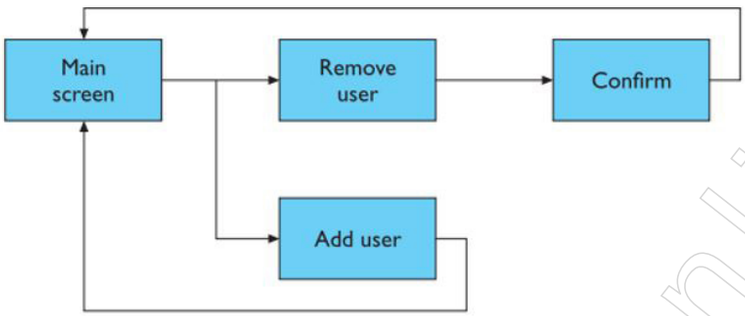


Figure 39: Network of screens/states

Wider still

"No man is an island, entire of itself," wrote Donne. This also applies to the products that we design. Each must take into account how it fits in with other applications and devices while designing a gadget.

There are various ramifications to this:

- Style issues:In order to guarantee consistency between programs, we should typically adhere to platform standards, such as the locations of menus on PC apps. For instance, we ought to utilise the conventional fast-forward, play and pause icons on our suggested personal movie player.
- Functional issues:A PC application must support cut and paste, read common file formats and allow file interaction.
- Navigation issues:In order to enable features like data embedding from one program into another or the ability to double-click an attachment symbol in a mail system to open the appropriate application, we might need to provide links between applications.

We face an additional challenge when it comes to links from other websites and applications that take users straight to the core of our website or web application, bypassing our "home page" and other pages. Furthermore, we have no control over any websites we link to, nor how their content may change over time.

2.1.3 Screen Design and Layout

Although we have discussed the many components of interactive apps, we have not discussed their assembly. It is frequently necessary for a single screen image to serve as both the interface for interacting with the system and a clear source of information.

The fundamentals of screen design are similar to those of other disciplines of interaction design:

Ask What is the user doing?

Think What information is required? What comparisons may the user need to make? In what order are things likely to be needed?

Design Form follows function: let the required interactions drive the layout.

Tools for layout

To assist us in recommending to the user acceptable methods to read and engage with a screen or gadget, we have a variety of visual tools at our disposal.

(c) Amity University Online

Notes

Billing details:
Name:
Address: ...
Credit card no:

Delivery details:
Name:
Address: ...
Delivery time:

Order details:

item	quantity	cost/item	cost
size 10 screws (boxes)	7	3.71	25.97
... ...	...	...	...

Figure 40: Grouping related items in an order screen

Grouping and Structure

Things should typically be physically grouped together if they make sense together intellectually. There could be several tiers of organisation involved here. For instance, a possible ordering screen design is shown in the Figure above. Take note of the spatial grouping of the billing and shipping data as well as the way they are separated both geographically and by a line from the list of things that are actually ordered. This illustrates the logical framework that follows:

Order:
- Administrative information
  - Billing details
  - Delivery details
- Order information
  - Order line 1
  - Order line 2
  - ...

Order of groups and items

Reexamining the figure above reveals that the screen appears to automatically advise reading or filling out the billing information first, then the delivery information and finally the individual purchase items. Is this the correct sequence?

Generally speaking, we must consider: what is the user's natural order? Normally, this should correspond with the screen's order. We should additionally configure the sequence in which the tab key navigates between fields on data entry forms and dialogue boxes.

Sometimes we could also wish to impose a specific order; for instance, we might want to make sure we remember the credit card information!

Decoration

Again looking at Figure above, we can see how the design employs boxes and a separating line to make the grouping evident. Other decorative aspects like font style and text or background colors might be utilised to accentuate groupings. Look at the microwave control panel in Figure below. See how the buttons differ in employing the foreground and background colors (green and gold) such that groups are related with one another. See also how the buttons are arranged out to break them into groups of similar function.

(c) Amity University Online

Notes

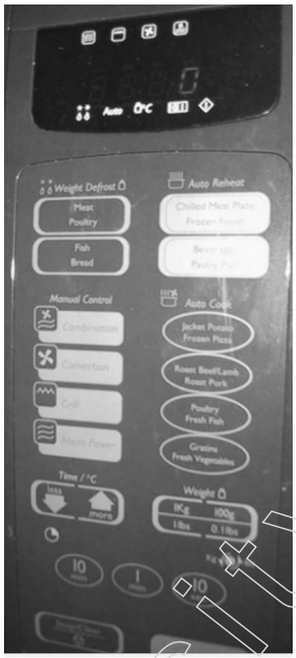


Figure 41: Microwave control pane

Alignment

List alignment is another crucial factor. Lists of text items should generally align to the left for users who read text left to right. On the other hand, for integers, numbers should typically be positioned at the decimal point or to the right. This is because the column's form, which resembles a minihistogram, then provides a magnitude indication. Names are among the more challenging items.

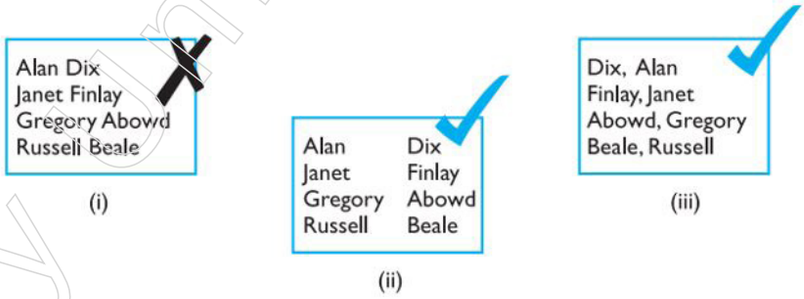


Figure 42: Looking up surnames

Take a look at list (i) in the above Figure. It is obvious that using just their surname to search for someone is difficult. Such lists should be organised in columns as in (ii) or with the forename and surname reversed as in (iii) to make things easier.

Lists with multiple columns need further attention. Text columns may have significant spaces between them because they must be large enough to accommodate the largest item. You can see how difficult it is for your eye to scan across the rows when you look at the example of this (i) in the figure below. To address this, there are a few different visual strategies: (ii) "leaders," which are dots that connect the columns; and (iii) soft tone greys or colours that are used behind rows or columns. Additionally, it could be worthwhile to deviate from other alignment guidelines at this point and align some text items to the right, as seen in (iv). This latter option might be a fine option if you were reading the numbers more often and the item names less frequently, but not if you needed to look up names all the time (which aren't arranged in this figure anyhow!).

(c) Amity University Online

Notes

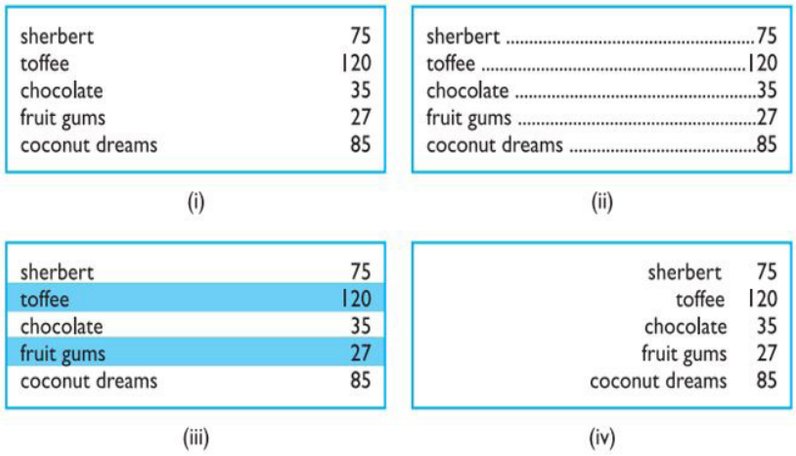


Figure 43: Managing multiple columns

White Space

The counter is the term used in typography to describe the space between letters. This is particularly significant in painting, where artists may concentrate just as much on the area between foreground features like buildings and people as they do on the actual elements. Giving the counters an equal weight in calligraphy and typography determines the word balance and in paintings, the counter shape frequently plays the most significant role in the composition. One can gain a general sense of the layout by focusing on the counter, or the space between the elements, rather than the "content" of a screen. When components that ought to be connected appear disconnected when you concentrate on the counter, there's a problem. Another useful trick for deflecting attention from the text and focusing on the overall structure is to subtly distort the screen with your eyes.

There are various uses for space. A few of them are displayed in the figure below. The coloured sections show text or visual elements that are continuous. As is frequently the case in report gaps between paragraphs or sections, we can observe in (i) that space is used to divide blocks. Complex buildings can also be made out of space. There are four distinct primary regions in (ii): ABC, D, E and F. Three further regions, A, B and C, are contained within one of these; A is categorised separately, followed by B and C collectively. Figure (iii) below shows how space is utilised to draw attention. Magazines commonly utilise this method to draw attention to a statement or image.

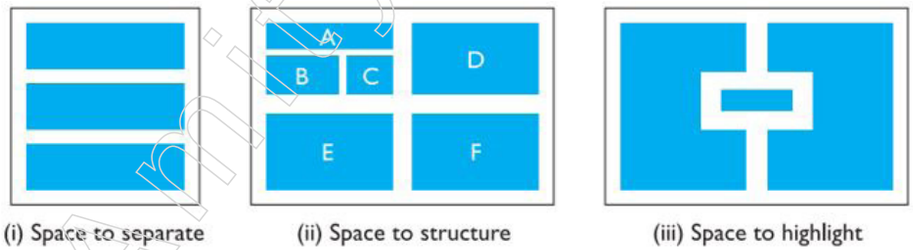


Figure 44: Using white space in layout

User Action and Control

Entering information

Forms-based interfaces and dialogue boxes have some of the trickiest and most challenging screen layouts. In every instance, the screen includes areas for the user to input data or make selections in addition to the information that is displayed to them. In

(c) Amity University Online

Notes

actuality, data entry fields share many of the same layout concerns as data presentation fields. Alignment remains crucial. Because the field names vary in length, it is very common to see the text entry boxes aligned in an uneven manner. In this case, field labels should be written using text that is justified to the right. Alternatively, if the field labels are to the left and above the field they relate to in a graphical interface, a smaller font can be used.

Knowing what to Do

While some screen elements are active and require you to fill them up or take action, others are passive and only provide you with information. Frequently, it's unclear even which elements are active, let alone what will probably happen if you interact with them!

Platform and company style guides are in place for this purpose, among others. Users will be able to identify menus and buttons when they see them if everyone designs them to look the same. Still, this is insufficient on its own. It's crucial that menu labels and symbols are easily comprehensible. Standards are useful once more for routine operations like print, delete and save. Broader principles must be followed for additional actions that are specific to a given system. When a button says "bold," for instance, does it indicate the condition of the system right now or what will happen when the button is pressed?

Affordances

These are particularly challenging issues in multimedia applications where an avant-garde and non-standard style may be purposefully adopted. How are users meant to be able to select where to click? According to the psychology of affordance, objects may indicate through their shape and other characteristics what can be done with them. For example, a handle may be used for pulling or lifting, while a button may be used for pushing. One can leverage these affordances to develop new interactive features. One can attempt to replicate the essential features of real-world items or outright duplicate them. It is imperative that you avoid representing a real-world object in a situation where its typical affordances are not applicable!

Additionally, keep in mind that affordances are culturally and historically specific and not inherent. The majority of computer-savvy people will click on an icon. They know that this is an affordance of such objects in the computing domain, not because they go about promoting pictures in art galleries. Similar to this, seasoned users might double-click if a single click is ineffective, but inexperienced users would never consider doing so because, well, double-clicking most actual buttons also turns them off!

Appropriate Appearance

Presenting information

Information can be presented on a screen in a variety of ways, depending on its type (text, numbers, maps, tables, etc.), the technology available to present it (character display, line drawing, graphics, virtual reality, etc.) and—most importantly—the intended application. Look at the window in the following figure. Finding recently updated files is somewhat challenging because the file listing is alphabetical, which is good if we want to seek up specific file details. Of instance, it would be challenging to locate a certain file if the list were arranged chronologically. Different representations are needed for different reasons. We might be thinking about scatter graphs, histograms, or three-dimensional surfaces for more complicated numerical data; outlines or organisation diagrams for hierarchical systems. But the idea of matching presentation to goal is true regardless of how complicated the data are.

(c) Amity University Online

Notes

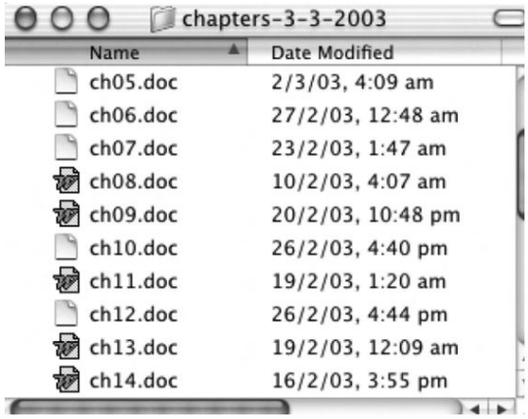


Figure 45: Alphabetic file listing

Long before computers, interactive systems, or human-computer interaction (HCI), the problem of presentation existed! Tufte's book is most likely the most reliable source on this topic. Although books and other static information presentations are its main aim, most design concepts apply equally.

One benefit we have when displaying information in an interactive system is that it is simple to let the user select between multiple representations, enabling the achievement of various objectives. Macintosh folder windows, as seen in the above figure, allow the user to reorganise the file list by clicking on a column header. This allows one to view the files according to certain criteria, such as name or date. This indicates that we can design for a variety of potential uses, but it is not a justification for disregarding the user's intent.

Aesthetics and Utility

Keep in mind that not all interfaces are good just because they are attractive. As with anything that is well-designed, an interface should ideally look good. It's true that eye-catching displays and effective graphic design can boost user pleasure and boost productivity.

However, there are situations when usefulness and beauty clash. An industrial control panel, for instance, is frequently composed of the discrete controls of multiple subsystems, some of which were purchased and some of which were designed by separate teams. The ensuing discrepancy in appearance could appear disorganised and indicate cleaning. Undoubtedly, a portion of this discrepancy could lead to issues. Numeric keypads in the style of a calculator and a telephone, for instance, might be combined. It would be simple to miskey while switching between these when under pressure. The operator can, however, benefit from the multiplicity of controls by being able to keep track of which controls go to which subsystem; this benefit must be maintained in any redesign.

Numerous "weildesigned" posters and multimedia systems also exhibit this tension between functionality and beauty. To make text visible, the background behind it should ideally have low contrast; nevertheless, this isn't always the case and graphic designers sometimes incorporate overly intricate and dramatic backdrops just because they look nice. Although the results are remarkable—possibly even award-winning—they are totally useless! Positively speaking, thoughtful use of aesthetic principles can also promote comprehensibility.

Localisation / internationalisation

If you work abroad, you may come across a word processing document where all of the menus and instructions are still in English, but the document's text and file names are in the

Notes

local tongue. Localisation, often known as internationalisation, is the process of adapting software for use in other languages and cultural contexts.

It is obvious that terminology must adapt and a lot of interface construction toolkits use resources to make this process simple. The program employs a resource identifier—typically only a number—instead of the text itself when referencing names of menu items, error warnings and other text. Separately, a basic database is built that associates these identifiers with certain words and expressions. Since a distinct resource database is created for every language, the program can be tailored for use in a given nation by selecting the relevant resource database.

But language translation is just the most basic aspect of internationalisation. Several stated guidelines for alignment and layout rely heavily on the use of a left-to-right, top-to-bottom language, like English and the majority of European languages. It goes without saying that this is quite different for other languages. Moreover, a lot of icons and pictures only have significance in a limited cultural context. It is unrealistic to expect that Anglo-American cultural norms and symbols will be understood by everyone, even with its seeming global dominion. The application of crosses 7 and ticks 3 is a prime illustration of this. These are associated with positive and negative opposites in Anglo-American culture, although in most of Europe, the two are interchangeable.

2.1.4 Universal Design Principles

The practice of creating items that can be utilised by as many individuals as possible in as many scenarios as feasible is known as universal design. In our situation, this implies creating interactive systems in particular that can be used on any technology platform by users of all skill levels. This can be accomplished by building systems with redundancy or assistive technology compatibility in mind. An interface that can be accessed both visually and audibly for commands is an example of the former; a website that offers text replacements for graphics so that screen readers can read it is an example of the later.

“The process of designing products so they can be used by as many people as possible in as many situations as possible” is how we describe universal design. What does that actually mean, though? Is it feasible to create anything that can be utilised by everyone? If yes, what kind of use would it serve? Wouldn’t the price be too high? In practice, we might not be able to make everything universally accessible or guarantee that every user will have the same experience with a product, but we can strive towards the goal of universal design and try to offer a comparable experience.

Despite its intimidating appearance, universal design doesn’t have to be difficult or expensive. As a matter of fact, if you look closely, you will notice that there are numerous instances of design that try to include user variety. Take note of the pavement the next time you cross the street. It may be possible to lower the curb to make it easier for wheelchair users to traverse. For those who are blind or visually impaired, the pavement close to the curb may have elevated ridges or bumps that make it easier for them to locate the crossing. Take note of how many contemporary structures have automatic doors that open upon arrival. or lifts that notify the user of the floor they have reached both audibly and visually. Furthermore, take note of how these designs benefit other users in addition to making the building, the lift and the crossing more accessible to those with disabilities.

People are less likely to miss their floor because they weren’t paying attention and shoppers with large bags or small children may enter the building and cross the road more quickly. Parents pushing strollers and travellers with wheeled luggage can also approach the building with greater ease. The main goal of universal design is to avoid excluding

(c) Amity University Online

Notes

anyone through your design decisions, but by considering these concerns, you will inevitably improve your design for everyone.

Seven broad global design principles were put up by a group from North Carolina State University in the USA in the late 1990s. These are equally applicable to the design of interactive systems and were meant to encompass all design domains. These ideas provide us with a framework for creating universal designs.

1. Principle one is equitable use: People of various abilities can benefit from the design, which is visually appealing to everyone. Nobody is stigmatised or left out. Access should be equal for all wherever feasible; if this isn't feasible, then equivalent use should be encouraged. Everyone should have access to security, privacy and safety measures as necessary.
2. Principle two is flexibility in use: Through a variety of usage options and flexibility to accommodate the user's speed, accuracy and preferences, the design accommodates a wide range of skill and preference.
3. Principle three is that the system be simple and intuitive to use, irrespective of the user's degree of concentration, linguistic proficiency, expertise, or experience. In addition to meeting the demands of users with varying language and literacy proficiency, the design must support user expectations. It should be arranged to make it easier to access the most crucial regions and shouldn't be overly complicated. It should, to the greatest extent feasible, offer guidance and criticism.
4. Principle four is perceptible information: Information should be effectively communicated through the design, irrespective of the user's ability or the surroundings. Presenting information in multiple ways is crucial; it should be done so in a variety of ways (e.g. graphic, verbal, text, touch). Important information needs to be highlighted and made evidently distinct from unimportant information. People with diverse sensory capacities should be able to access information via a variety of devices and methods, which should be supported by the presentation.
5. Principle five is tolerance for error: reducing the effect and harm brought about by errors or inadvertent actions. Hazardous conditions ought to be eliminated or made difficult to access. Warnings should be used to protect people from potential dangers. Users should be supported when performing tasks that call for focus and systems should fail safe from their point of view.
6. Principle six is low physical effort: Systems should be made as easy to operate as possible to reduce tiredness and physical strain. With appropriate operational effort, the user should be able to maintain a natural posture thanks to the system's physical architecture. Avoiding prolonged or repetitive activity is advised.
7. Principle seven requires size and space for approach and use: Any user, regardless of size, posture, or mobility, should be able to reach and use the system due to its positioning. Both seated and standing users should be able to see important features. Users should be able to reach every physical component with ease, whether they are sitting or standing. Systems ought to accommodate a range of hand sizes and give adequate space for the usage of assistive devices.

When thinking about universal design, these seven guiding principles provide a solid foundation. Naturally, not all of them are equally appropriate in every circumstance. For instance, while creating an information booth, principles six and seven would be crucial, but they wouldn't be as significant when creating word processing software. However, they offer a helpful checklist of things to think about for designers as well as instructions on how to apply each principle.

© Amity University Online

Notes

2.1.5 Iteration and Prototyping

Our initial design will probably not be flawless because human conditions are complicated and creators are fallible. Iterations of concepts are hence a common feature of practically all interaction design. Storyboards and paper designs are frequently the first things that colleagues and potential users see. Mockups of real devices or software tools like Shockwave or Visual Basic may be used later in the design phase to produce software prototypes.

These prototypes, whether they are software-based or paper-based, may all be assessed to determine their acceptability and areas for improvement. Formative evaluation is a type of assessment meant to enhance designs. In contrast, summative evaluation is done at the end to make sure the product meets quality standards.

Having an expert examine each screen individually to check for infractions using a set of guidelines—such as the one on "knowing where you are"—is one method. The other major strategy is to involve actual users in real-world environments, such as "in the wild," or in a controlled experimental setting.

Prototyping and iteration are therefore the generally recognised "best practice" methods for interaction design. But there are also significant drawbacks to prototyping that are rarely mentioned in the literature.

One example of what's referred described as a hill-climbing strategy is prototyping. Assume that you are someplace in the vast countryside. You begin walking uphill and continue in an ascetic manner as you can. At some point, you will arrive at the top of a hill. That's precisely how iterative prototyping operates: you start somewhere, assess it to see how to improve it, make changes to improve it and continue this process until there's no more that can be done.

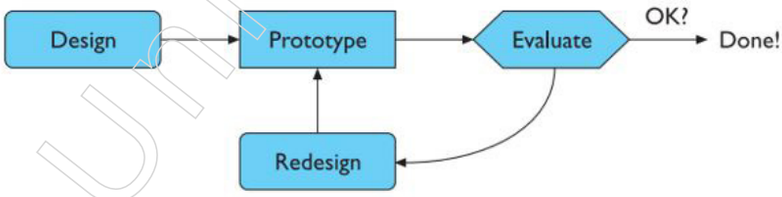


Figure 46: Role of prototyping

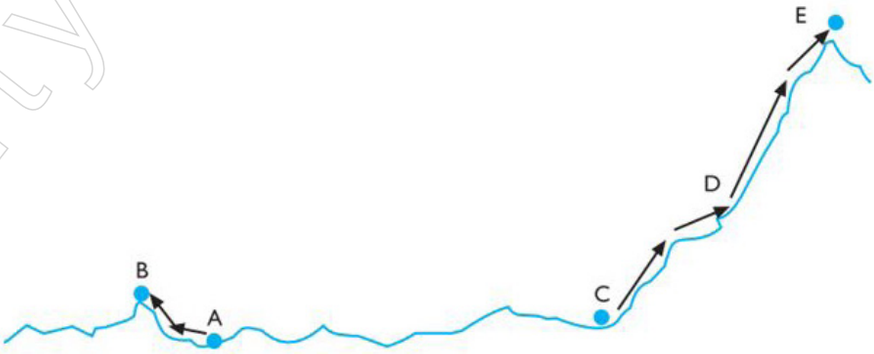


Figure 47: Moving little by little... but to where?

Climbing hills, though, isn't always effective. Assume that you begin your journey close to Cambridge, UK. If you continue uphill (which is quite difficult to determine because the terrain is so flat!), you will ultimately reach the summit of the Gog Magog hills, which is the closest point to you by around 300 feet. On the other hand, you might arrive at the summit of the Matterhorn if you started somewhere else. When using

Notes

hillclimbing techniques, you may always end up somewhere that is excellent locally but subpar when compared to farther away. The diagram above illustrates this: starting at A traps you at the local maximum at B, whereas starting at C allows you to go up via D and to the global maximum at E.

With interfaces, the issue of becoming stuck at local maxima is also conceivable. A poorly thought-out design concept could be the basis for something that is just a refined version of the original!

This indicates that for prototyping techniques to be effective, the following two requirements must be met:

1. To understand what is wrong and how to improve.
2. A good start point.

The first is obvious: until you know what has to be done to improve the design, you cannot iterate it. However, in order to prevent local maxima, the second is required. To reach the highest altitude possible, you would most likely reserve a flight to the Himalayas rather than Cambridgeshire.

A truly skilled designer may use experience and intuition to determine what an excellent first design would be. However, this understanding is challenging due to the intricacy of interaction design difficulties. Another strategy that is frequently used in graphical design is to start with multiple design concepts and eliminate them one at a time as they are refined. It's similar to parachuting ten people at random locations around the planet. It's possible that one of them will wind up close to a tall peak.

2.2 HCI in Software Process

We now focused on finding characteristics of interactive systems that are useful by using specific instances of paradigms that have proven effective. The objective of design is to offer dependable methods for the successful and repeatable design of interactive systems. For this reason, we need to look at the interactive system design process rather than just identifying paradigms. A few components of a user-centered design process were presented previously. Here, we build on that procedure by situating interactive system design inside pre-existing software development frameworks.

Software engineering is a sizable subfield of computer science that deals with the administrative and technical aspects of developing software systems. The software life cycle, which outlines the processes involved in developing a software system from inception to eventual replacement and phase-out, is one of the fundamental concepts of software engineering. Since this is not meant to be a textbook on software engineering, we are not going to go into great detail on every problem related to software engineering and its various life-cycle models.

The key point we want to make is that problems with HCI that impact interactive systems' usability apply to every stage of the software development life cycle. As a result, software engineering for interactive system design entails more than just tying in a new task with the cycle's already-existing operations. Instead, it uses methods that cover every stage of the life cycle.

2.2.1 The Software Life Cycle

One of the arguments made for software development is that, like electrical engineering, it ought to be regarded as an engineering discipline when it comes to developing hardware. Every engineering field is unique in that it involves the methodical

(c) Amity University Online

Notes

application of scientific methods to the creation of a final product. Therefore, one of the core components of software engineering is that it offers the framework for using methods to create software systems. An attempt to categorise the processes involved in software development is made with the software life cycle. In every development project, these tasks must then be scheduled appropriately and completed using the right methods.

When developing software, two primary stakeholders are taken into account: the end user who needs to utilise the product and the designer who has to create it. Customers and designers are usually groups of people, however some individuals may be both. Differentiating between the customer—the person who hires the design firm—and the customer—the person who will ultimately utilise the system—is frequently crucial. Different people can fill these two customer responsibilities.

It is possible that the group of people who discuss the desired system's features with the designer will never actually use the system. This frequently holds true for online apps in particular. Here, the group of people who communicate with the design team will be referred to as "customers," and those who will interact with the developed system will be referred to as "users" or "end-users."

Activities in the Life Cycle

Figure below provides a more thorough explanation of the life cycle activities. Each activity in the graphical representation flows easily into the next, akin to a cascade. Although the waterfall analogy does not perfectly capture the true nature of the relationships between these activities, it is a useful place to start when talking about the activities' logical flow. Next, we will go over the tasks included in this waterfall model of the software life cycle.

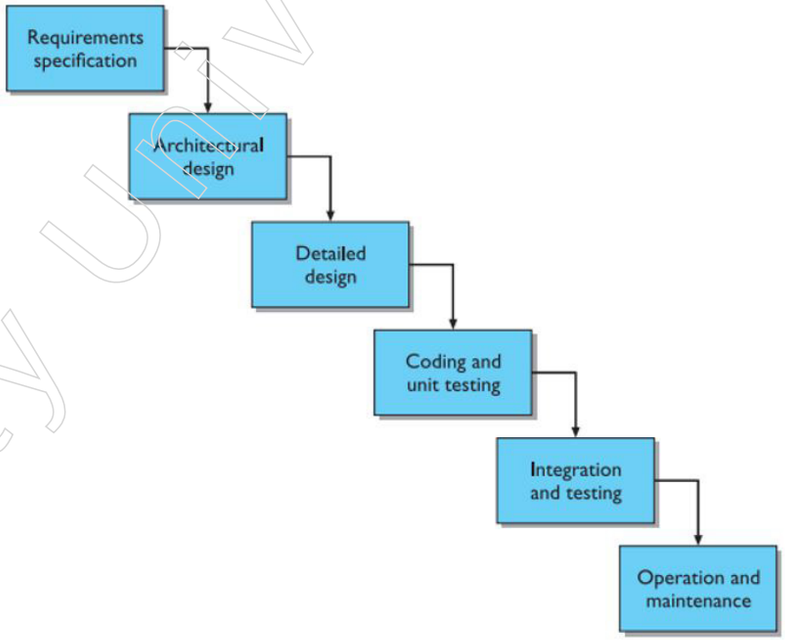


Figure 48: The activities in the waterfall model of the software life cycle

Requirements Specification

The designer and client attempt to describe the desired functionality of the final system in the requirements specification. On the other hand, figuring out how the system will deliver the desired services is the focus of subsequent tasks. Obtaining information from the client regarding the domain—or work environment—in which the finished product will operate is part of the requirements specification process. Aspects of the work domain encompass not

(c) Amity University Online

Notes

only the specific tasks that the software product needs to accomplish but also information about the environment in which it must function, including who it may impact and how the new product interacts with any older products it may be updating or replacing.

The process of defining requirements starts at the outset of product development. Even though the requirements are written from the customer's point of view, the software product must be able to implement them if the needs are to be met. Typically, requirements are first stated in the customer's native tongue. Software executable languages are less naturally occurring and more closely resemble mathematical languages where each word has a specific meaning, or semantics. One of the keys to effective development is the translation from the expressive but relatively ambiguous natural language of needs to the more precise but less expressive executable languages.

Architectural Design

As previously stated, the requirements specification focuses on the intended functionality of the system. The upcoming tasks focus on how the system delivers the services that users have come to anticipate. The first step involves breaking down the system at a high level into its component parts, which can then be independently built or imported from pre-existing software. Decomposition of an architectural design is carried out by it. It is not just focused on figuring out how the system is divided into its component parts and what services they each perform. It must also outline the resource sharing that will occur between various components as well as the interdependencies between individual components.

A designer can get help in creating an architectural description from details in the needs specification using a variety of structured methodologies (e.g., CORE, MASCOT and HOOD). We will note here that most of these techniques work well for capturing the functional requirements of the system, or the services the system needs to deliver in the work domain. However, they do not offer a quick way to capture other non-functional requirements, or aspects of the system that have to do with the way the system must deliver services instead of the actual services itself. Typical instances of non-functional needs include the system's efficiency, dependability, timeliness and safety features.

Detailed Design

By breaking down the system description, the architectural design enables the isolated development of distinct components that will eventually be combined. The designer needs to give a thorough enough description of the components that aren't ready for instant integration so that they can be developed in a programming language. The architectural design's description of the components is improved upon in the detailed design. The more in-depth explanation needs to maintain the behaviour that is suggested by the higher-level description.

Usually, there are multiple ways to modify the architectural element while still meeting the behavioural requirements. Selecting the optimal refinement frequently involves attempting to meet as many of the system's non-functional needs as you can. Therefore, in order to evaluate the design's qualities, the language used for the detailed design must permit some analysis of the design.

Coding and Unit Testing

In the Software Development Life Cycle (SDLC), coding and unit testing are crucial phases that contribute significantly to the development process. Let's delve into each of these phases:

(c) Amity University Online

Notes

Coding Phase:

1. Understanding Requirements: Developers need to have a clear understanding of the requirements specified in the design phase. They should be aware of what needs to be built and how it should function.
2. Selecting Technologies: Based on the project requirements, developers choose appropriate programming languages, frameworks, and tools for implementation.
3. Writing Code: Developers write code according to the design specifications. They follow coding standards and best practices to ensure readability, maintainability, and efficiency.
4. Version Control: Code is managed using version control systems like Git. This ensures that changes are tracked, and collaboration among team members is facilitated.
5. Documentation: Developers document their code, including comments and explanations where necessary, to aid understanding and future maintenance.
6. Code Review: Peer code reviews are conducted to ensure code quality, adherence to coding standards, and to catch potential bugs early in the development process.

Unit Testing Phase:

1. Writing Unit Tests: Developers write small, focused tests to verify the behavior of individual units of code (functions, methods, classes) in isolation. These tests cover various scenarios and edge cases.
2. Running Tests Locally: Developers run unit tests locally to ensure that their code behaves as expected and that any changes made do not break existing functionality.
3. Continuous Integration (CI): Unit tests are integrated into a CI/CD pipeline. Whenever code changes are pushed, CI systems automatically run unit tests to check for regressions.
4. Debugging: If unit tests fail, developers debug the code to identify and fix the issues. They may modify the code or the tests themselves to resolve the failures.
5. Code Coverage Analysis: Developers analyze code coverage reports to ensure that unit tests adequately cover the codebase. This helps identify areas that require additional testing.
6. Refactoring: After passing unit tests, developers may refactor the code to improve its structure, readability, or performance while ensuring that all unit tests still pass.

Importance:

- Early Bug Detection: Unit testing helps catch bugs early in the development process, reducing the cost and effort of fixing them later.
- Code Quality: Writing tests encourages developers to write modular, well-structured code, leading to higher-quality software.
- Regression Prevention: Unit tests provide a safety net against regressions when making changes to the codebase.
- Documentation: Unit tests serve as executable documentation, showcasing how different parts of the codebase are intended to function.
- Continuous Integration: Unit testing is an integral part of CI/CD pipelines, ensuring that only code meeting quality standards is integrated into the main codebase.

Integration and Testing

After a sufficient number of components have been evaluated and put into practice, they need to be integrated according to the architectural plan. To make sure appropriate

(c) Amity University Online

Notes

behaviour and responsible usage of any shared resources, more testing is conducted. To make sure the system satisfies their needs, it is also feasible to conduct some acceptance testing with the clients at this time. The integrated system must first be accepted before the product can be ultimately made available to customers.

The final system could also need to be certified in accordance with standards set by an outside body, like an aviation certification board. A European Health and Safety Act enacted in 1993 mandates that all firms furnish their employees with systems that are easy to use. Additionally, a standard (ISO 9241) defining the usability of workstations in office environments has been established by the worldwide standards organisation, ISO. When combined, ISO 9241 and health and safety laws give designers a strong incentive to consider the human-computer interface (HCI) implications of their designs.

Maintenance

Until a new edition of the product necessitates a complete redesign or the product is phased out entirely, any system work done after it is released falls under the category of maintenance. Because of this, maintenance takes up most of a product's lifetime. Correction of system errors found after release and modification of system services to meet requirements not met during previous development are examples of maintenance. As a result, as seen in the figure below, maintenance gives input to every other life cycle activity.

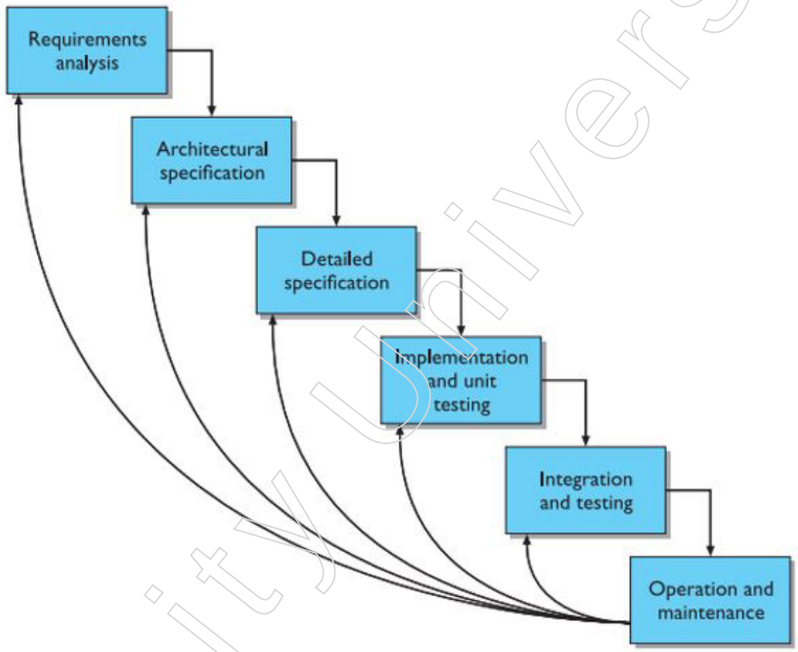


Figure 49: Feedback from maintenance activity to other design activities

Validation and Verification

The design needs to be reviewed at various stages of its life cycle to make sure it is complete, internally consistent and meets the high-level objectives that were agreed upon with the client. These examinations are known as verification and validation, in that order. Boehm draws a helpful line between the two, defining verification as creating "the thing right" and validation as building "the right thing." Throughout the design process, a variety of languages are utilised, from very rigorous and precise mathematical languages to informal natural language. Exercises involving validation and verification are challenging enough when done in a single language; when done across languages, they become far more challenging, if not impossible.

Notes

The majority of the time, a design's verification will take place within one life-cycle activity or in between two related activities. For instance, a payroll accounting system component's designer will be concerned with the accuracy of the algorithm used to calculate the taxes that are deducted from an employee's gross income. A general specification of the data that this component should have received as input and output from other sources should have been included in the architectural design. Further information will be introduced in the comprehensive description to help improve the general specification.

It's possible that the detailed design will need to alter the information representations and split a single high-level activity into multiple lower-level operations that can be put into practice later on. The designer must demonstrate that the refined description is legitimate within its language (internal consistency) and that it describes all of the specified behaviour of the high-level description (completeness) in a way that can be verified (relative consistency) before introducing these changes to information and operations.

Validating a design shows that the needs of the client are met across the different processes. Compared to verification, validation is a far more subjective process mostly due to the inconsistency between the requirements and design languages, which makes objective proof impossible. Validation against HCI requirements in interactive system design is sometimes called evaluation and it can be carried out by the designer alone or in conjunction with the client.

What defines a proof is a crucial question that concerns both validation and verification. The terminology employed in any design activity and the foundation for its semantics have been brought up numerous times. Argumentation and proof in the objective sense are possible in languages having a mathematical foundation. It is possible to support or disprove an argument based solely on a mathematical language using metrics that are widely accepted.

When a proof is fully supported by the conventions of the mathematical language, it's referred to as a formal proof. More typical is a rigorous proof, which is expressed in some mathematical language but depends on the reader's comprehension to acknowledge its accuracy without referring to the complete arguments, which are usually not given but can be. The degree of information omitted by the prover while still retaining acceptance of the proof distinguishes formality from rigour.

Verification proofs for designs can often be found within a language or between two languages with identical mathematical meanings. A design project's time constraints as well as the estimated cost implications of the various components typically determine which proofs are completed completely formally and which are only rigorously (if at all). The cost of evidence should go down as this field of study advances and automated tools help with the mechanical parts of proof.

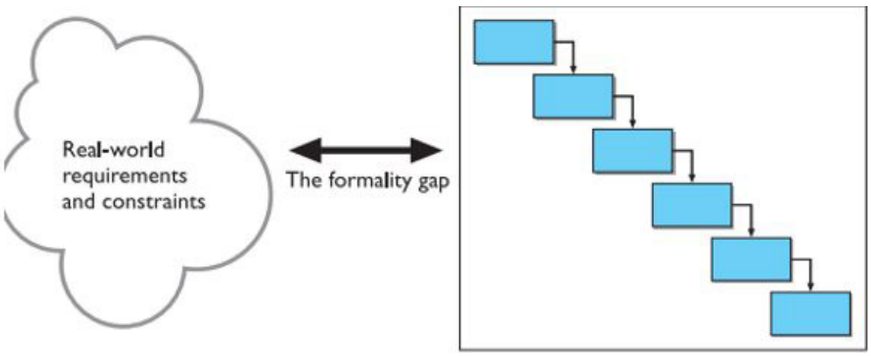


Figure 50: The formality gap between the real world and structured design

(c) Amity University Online

Notes

Proofs of validation are far more difficult because they nearly invariably require a translation across languages. Moreover, rather than the mathematical world, the underlying ambiguity of the real world is where consumer expectations originate. This eliminates the chance of rigorous or formal objective evidence. Rather, there is always going to be a jump from unstructured real-world scenarios to any formal, structured development approach. This inherent difference is known as the "formality gap," as seen in the following figure.

Because of the formality gap, validation will always depend to some degree on subjective methods of evidence. By effectively using real-world specialists to carry out specific validation activities, we can boost our confidence in the subjective proof. Since these specialists might not be trained in design, it's possible that they won't comprehend the design notations being employed. It is crucial that the design notations close the formality gap and clarify the statements that the expert can subsequently verify. Since the expert for interactive systems would possess knowledge from the cognitive or psychological domain, the design specification needs to be easily interpreted psychologically in order to be validated against the system's interactive requirements.

.Interactive Systems and the Software Life Cycle

The necessity to give massive software system development some structure in the 1960s and 1970s gave rise to the conventional software engineering life cycles. Back then, most large systems developed had to do with business data processing applications. These were batch-processing systems rather than highly interactive ones. As such, usability difficulties from the standpoint of the end user were not very essential. Since the introduction of personal computers in the late 1970s and their enormous commercial success and acceptance, the majority of modern systems developed today are far more interactive. A product's ability to function well for someone who isn't expected to know much about its design is essential to its success. The modern user is highly skilled in his profession, even though he may not be all that skilled in software creation.

The above-discussed life cycle for development shows the design process in a fairly pipelined sequence. The actual design process is actually iterative, even for batch-processing systems; work in one design activity influences work in any other activity, either before or after it in the life cycle. Although we can depict this iterative relationship as shown in the figure below, it does not significantly improve our comprehension of the interactive system design process. One can wonder if learning about the interactive system design process is worthwhile. Do the interactive components of a system really require that much design work to be worth our time and attention? An iconic IBM survey conducted in 1978 by Sutton and Sprague estimated that fifty percent of the designer's effort was devoted to creating the user interface's code. That conclusion is supported by a more recent and compelling survey conducted by Myers and Rosson, which was conducted in the 1990s. Therefore, it is well worth the work to offer methods and structure for comprehending, organising and enhancing the interactive design process! We will discuss aspects of interactive system design in this section that the typical software life cycle does not adequately address.

A principled approach to design, which knows what it is we want to generate from the start and structures our approach to design to achieve that aim, fits well with the traditional software life cycle. As we've already discussed, designers typically don't know all of the requirements for a system before they start. Figure 6.4 shows how iterations back to earlier stages can reflect discovery made in later activities. This acknowledges that the requirements capture activity was not carried out correctly. We argue—and there are many solid arguments to support this position—that it is not possible to determine all of the needs for an interactive system at the outset. This is a more serious allegation. As a result, in

Notes

order to figure out how to make systems more useful, they need to be constructed and user interactions need to be monitored and assessed.

We are unable to forecast how to design for maximum usability because our theories of the psychology and sociology of the human and human cognition—whether in isolation or in a group—are insufficient. Numerous studies have been conducted on human user models that enable the prediction of how well users would interact with interactive technologies.

Because there is a lack of predictive psychology theory, designers must watch real users engage with the generated product and gauge how well they perform in order to test certain usability features of their designs. The experiments should be as near to an actual interaction situation as possible in order to make the observations' results meaningful. This implies that the experimental system ought to closely resemble the ultimate product whose specifications the designer is attempting to determine! It is not worth it to experiment with rough estimations of the actual system, as John Carroll has pointed out, as this will produce observations whose conclusions will not necessarily apply to the real system. This is because the exact intricacy of the actual system might vitally alter its usefulness.

An early knowledge of the tasks the user wants to do is crucial for implementing a principled approach to interactive system design, which will be discussed in following sections. One issue with this assumption is that a user frequently doesn't know the tasks he will complete until he is comfortable with the technology he uses to complete them. Both the tasks he completes and the objects he uses to complete them are subject to the chicken-and-egg problem. For instance, an author would not have thought to employ a contracting and expanding outline tool to swiftly and readily experiment with a paper's structure while it was being typed, had word processors not been invented. How could a designer know to support such a task when creating the first word processor, if a typewriter just did not have the capability to do so?

Furthermore, not every task that a user completes using a system is one that the system's creator intended for it to be used for. Consider a graphics drawing program that divides an assembled image into discrete layers. To create graphical pictures that are whole objects—like a square or a circle, for example—that may be edited as such while maintaining their object identity, one layer is utilised. Paint is applied to images, which are merely collections of pixels, on the other layer. By alternating between the layers, the user can produce incredibly intricate images that combine painted scenery and objects. However, it is also feasible to conceal some areas of the image in one layer and reveal them in the other due to the intricate interaction between the overlapping images in the two levels. With such a feature, the user will be able to perform basic simulations, like demonstrating the shadowing effect when turning on and off a light. It is quite unlikely that the inventors of these graphics systems, whose purpose was to create intricate yet static images, had explicit support for such simulation or animation tasks in mind when they created them.

Finally, a drawback of the traditional software life cycle is that it discourages the application of strategies and notations that enhance the way users perceive interactive systems. The goal of validation and the formality gap were already covered. When the notation for an abstract design does not reflect the kind of knowledge the user needs to recollect in order to interact, it is exceedingly difficult for an expert in human cognition to forecast the cognitive demands that an abstract design would need on the intended user. The same is true when evaluating the timing behaviour of an abstract design that doesn't specifically state how the activities to be invoked will be timed or in what order. The formality gap will never completely disappear, but with the right help, specific notations can greatly increase the likelihood that non-functional requirements can be validated.

(c) Amity University Online

Notes

Agile:

Agile is an incremental, iterative software development methodology that prioritises client happiness, adaptability, and teamwork. Although a number of frameworks and approaches have been developed based on agile concepts, it is more of a philosophy or attitude than a set of particular behaviours. The following are Agile's main features:

Principles of Agile Manifesto:

1. Individuals and Interactions over Processes and Tools: Agile prioritizes people and their interactions over rigid processes and tools.
2. Working Software over Comprehensive Documentation: While documentation is important, delivering functional software is the primary measure of progress.
3. Customer Collaboration over Contract Negotiation: Agile encourages close collaboration with customers to understand and respond to their needs effectively.
4. Responding to Change over Following a Plan: Agile embraces change and adapts to evolving requirements, rather than sticking rigidly to a predefined plan.

Key Concepts and Practices:

1. Iterative Development: Agile projects progress through a series of iterations or sprints, with each iteration delivering a potentially shippable product increment.
2. Continuous Feedback: Regular feedback loops with stakeholders, including customers and team members, help validate assumptions, identify issues, and guide development efforts.
3. Cross-Functional Teams: Agile teams are typically cross-functional, comprising individuals with diverse skills necessary to deliver complete product increments.
4. Prioritized Backlog: Requirements are managed in a prioritized backlog, with the most valuable features and tasks at the top.
5. Daily Stand-ups: Daily stand-up meetings (or "daily scrums") keep the team synchronized, identify obstacles, and facilitate collaboration.
6. Retrospectives: Periodic retrospectives allow teams to reflect on their processes, identify areas for improvement, and make adjustments accordingly.
7. Adaptive Planning: Agile planning is flexible and adaptive, with plans adjusted based on feedback and changing priorities.
8. Test-Driven Development (TDD): Developers write automated tests before writing the code, ensuring that the code meets the specified requirements and is maintainable.
9. Continuous Integration and Delivery (CI/CD): Continuous integration and delivery practices ensure that changes are integrated into the codebase frequently and delivered to customers quickly and reliably.

Agile Methodologies and Frameworks:

- Scrum: Scrum is one of the most widely adopted agile frameworks, featuring roles (Scrum Master, Product Owner, Development Team), events (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective), and artifacts (Product Backlog, Sprint Backlog, Increment).
- Kanban: Kanban emphasizes visualizing workflow, limiting work in progress, and continuously improving the process.
- Extreme Programming (XP): XP focuses on engineering practices such as pair programming, test-driven development, continuous integration, and frequent releases.

© Amity University Online

Notes

Benefits of Agile:

- Faster Time-to-Market: Agile allows for quicker delivery of valuable software increments, enabling rapid response to changing market conditions.
- Improved Customer Satisfaction: Continuous feedback and collaboration with customers ensure that the delivered product meets their needs and expectations.
- Enhanced Flexibility: Agile practices enable teams to adapt to changing requirements and priorities more effectively.
- Higher Quality: Emphasis on testing, continuous integration, and iterative development results in higher-quality software with fewer defects.
- Better Team Morale: Agile principles promote collaboration, empowerment, and a focus on delivering customer value, leading to higher team morale and engagement.

DevOps

The goal of the DevOps set of techniques, values, and cultural tenets is to enhance coordination and communication between teams working on software development (Dev) and IT operations (Ops). To produce high-quality software fast and effectively, it focuses on automating and optimising the processes of developing, testing, deploying, and managing software applications. These are the major elements of DevOps:

Principles and Practices:

- Automation: DevOps advocates for automating repetitive tasks throughout the software development lifecycle, including code building, testing, deployment, and infrastructure provisioning.
- Continuous Integration (CI): Developers integrate their code changes into a shared repository frequently (often multiple times a day), triggering automated builds and tests to detect integration issues early.
- Continuous Delivery (CD): CD extends CI by automatically deploying code changes to production-like environments after successful testing, enabling frequent and reliable releases.
- Infrastructure as Code (IaC): Infrastructure configuration is defined and managed through code, allowing for automated provisioning, scaling, and management of infrastructure resources.
- Monitoring and Logging: DevOps teams implement monitoring and logging solutions to gain visibility into application and infrastructure performance, detect issues, and troubleshoot problems quickly.
- Collaboration and Communication: DevOps fosters a culture of collaboration and shared responsibility between development and operations teams, breaking down silos and promoting transparency.
- Feedback Loops: Continuous feedback loops enable teams to identify areas for improvement, iterate on processes, and enhance overall efficiency and quality.

Tools and Technologies:

- Version Control: Tools like Git enable teams to manage and collaborate on code changes efficiently.
- CI/CD Pipeline Tools: Jenkins, GitLab CI/CD, and CircleCI are popular tools for automating build, test, and deployment processes.
- Configuration Management: Tools like Ansible, Chef, and Puppet automate the provisioning and configuration of infrastructure resources.

(c) Amity University Online

Notes

- Containerization and Orchestration: Technologies such as Docker and Kubernetes facilitate containerization and management of applications and microservices.
- Monitoring and Logging Tools: Solutions like Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana), and Splunk provide visibility into application and infrastructure performance.
- Collaboration Platforms: Platforms like Slack, Microsoft Teams, and Atlassian Jira facilitate communication, collaboration, and project management within DevOps teams.

Benefits of DevOps:

- Faster Time-to-Market: DevOps practices enable rapid and frequent software releases, allowing organizations to respond quickly to market demands and customer feedback.
- Improved Quality: Automation and continuous testing help detect and address issues early in the development process, leading to higher-quality software.
- Increased Efficiency: Streamlined processes and automation reduce manual effort, enabling teams to focus on value-added tasks and innovation.
- Enhanced Collaboration: DevOps encourages cross-functional collaboration and shared ownership, fostering a culture of teamwork and accountability.
- Greater Stability and Reliability: Continuous monitoring and feedback loops help identify and address performance issues and failures proactively, improving system stability and reliability.

2.2.2 Usability Engineering

Explicit usability engineering goals have been incorporated into the design process as a means of achieving user-centered design. This technique was proposed by Nielsen at Bellcore and by Whiteside and colleagues at IBM and Digital Equipment Corporation. Engineering relies on interpretation in the context of specified goals, a shared meaning and a knowledge of how successful completion will be assessed. The focus of usability engineering is on precisely defining the standards by which a product will be evaluated for its usability.

Users' experiences with a product serve as the ultimate test of its usability. It makes sense to concentrate on the real user interface as here is where a user interacts with an interactive system directly—at the physical interface. This narrow focus runs the risk of ignoring the fact that a large portion of the effort done in interaction involves more than simply the visible aspects of the systems involved. In actuality, to arrive at relevant measurements, the entire system's functional architecture as well as the users' cognitive capacity should be observed. However, the applicability of usability engineering is limited since it is extremely difficult to extract metrics of activity beyond the physical behaviours that occur in the real world.

One of the key components of usability engineering in the context of the software life cycle is the inclusion of a usability specification as part of the requirements specification. This specification focuses on aspects of the user-system interaction that enhance the product's usability. It is proposed to use a variety of system attributes as barometers to assess usability. Six definitions are provided for each attribute, which together make up the usability specification. The table below is an illustration of a usability specification based on the methodology described by Whiteside, Bennett and Holtzblatt for the design of a video cassette recorder (VCR) control panel.

© Amity University Online

Notes

Table 3: Sample usability specification for undo with a VCR

Attribute:	Backward recoverability
Measuring concept:	Undo an erroneous programming sequence
Measuring method:	Number of explicit user actions to undo current program
Now level:	No current product allows such an undo
Worst case:	As many actions as it takes to program in mistake
Planned level:	A maximum of two explicit user actions
Best case:	One explicit cancel action

The principle of recoverability is selected as the specific usability attribute of interest in this instance. Recoverability is the capacity to accomplish a desired outcome following the identification of a mistake in an earlier engagement. There are two possible ways to go about the rehabilitation process: backward or forward. The interactive systems created by the current VCR design are infamously hard to use; redesigning a VCR offers a useful case study for usability engineering. When creating a new control panel for a VCR, the designer considers how the user might be able to fix a mistake he makes when attempting to set up the VCR to record a television show while he is away. One strategy the designer chooses to employ is giving the user the option to reverse the programming sequence, returning the VCR to its initial condition prior to the start of the programming task.

By defining the abstract backward recoverability trait in terms of a measurement notion and relating it to the actual product, it becomes more tangible. Thus, in this instance, backward recoverability is understood to be the capacity to reverse a mistaken programming sequence. No matter where the user is in the programming sequence, the measuring technique specifies how the attribute will be measured, in this example by the number of explicit user actions needed to complete the undo.

The agreed-upon standards for evaluating the product's performance based on the measuring technique are then provided by the final four entries in the usability specification. Regardless of whether the system is computer-based or not, the now level shows the value for the measurement made using the current setup. The planned level is the target for the design; the best case is the level that is agreed to be the best possible measurement given the current state of development tools and technology; and the worst case value is the lowest acceptable measurement for the task, giving a clear distinction between what will be acceptable and what will be unacceptable in the final product.

To find a good presently level in the example, the designers can refer to their past VCR products as well as those of their competitors. In this instance, it is found that no model as of yet permits an undo that would restore the VCR's previous state prior to the programming task. If a VCR has three different recording programs, for instance and you start a new program in the first program slot, the VCR forgets what was in there before and you can't get it back unless you remember what it was and reprogram it.

Table - 4: Measuring technique determination criteria

1. Time to complete a task
2. Per cent of task completed
3. Per cent of task completed per unit time
4. Ratio of successes to failures
5. Time spent in errors
6. Per cent or number of errors
7. Per cent or number of competitors better than it
8. Number of commands used
9. Frequency of help and documentation use

(c) Amity University Online

Notes

10. Per cent of favorable/unfavorable user comments
11. Number of repetitions of failed commands
12. Number of runs of successes and of failures
13. Number of times interface misleads the user
14. Number of good and bad features recalled by users
15. Number of available commands not invoked
16. Number of regressive behaviors
17. Number of users preferring your system
18. Number of times users need to work around a problem
19. Number of times the user is disrupted from a work task
20. Number of times user loses control of the system
21. Number of times user expresses frustration or satisfaction

The worst-case value is determined based on several factors. It should normally not be any lower than the current level. It appears that several usability attributes should offer worst case values that are better than the existing level because the new product should offer some improvement over the current situation. In any other case and unless the new system can be demonstrated to offer the same usability at a fraction of the cost, why would the client bother with it? According to the example's designers, the user would have to undo as many actions as he had taken to program in the error in order to have the bare minimum of an undo feature. Given that there is at least the option to undo, this is a definite improvement over the current state. An undo button on the control panel, which would essentially undo the prior non-undo operation, would be one approach to provide such a feature. The designers determine that, in terms of the number of explicit activities, the best situation would require only one, but they still think that the user should be able to perform a complete restoration of the VCR state in a maximum of two explicit user actions.

The measurement criteria listed in the tables above and below—which were adapted from Whiteside, Bennett and Holtzblatt—can be used to establish the best/worst case and planned/now level targets, as well as the measuring methodology for a certain usability feature. Usability metrics are measurements similar to those advocated by usability engineering.

Table 5: A usability specification's possible measurement level configurations

Set levels with respect to information on:
1. an existing system or previous version
2. competitive systems
3. carrying out the task without use of a computer system
4. an absolute scale
5. your own prototype
6. user's own earlier performance
7. each component of a system separately
8. a successive split of the difference between best and worst values observed in user tests

As previously mentioned, the ISO standard 9241 also suggests using usability specifications to specify needs. The following table provides examples of usability metrics arranged according to how they contribute to the effectiveness, efficiency and satisfaction of the three usability categories.

Notes

Table 6: Examples of usability metrics from ISO 9241

Usability objective	Effectiveness measures	Efficiency measures	Satisfaction measures
Suitability for the task	Percentage of goals achieved	Time to complete a task	Rating scale for satisfaction
Appropriate for trained users	Number of power features used	Relative efficiency compared with an expert user	Rating scale for satisfaction with power features
Learnability	Percentage of functions learned	Time to learn criterion	Rating scale for ease of learning
Error tolerance	Percentage of errors corrected successfully	Time spent on correcting errors	Rating scale for error handling

Problems with Usability Engineering

The key component of usability engineering is the early definition of clear usability metrics that can be used to evaluate a system once it is delivered. There is a strong case to be made that the only way we can consistently create more usable systems is by using empirical methods like the use of usability metrics. While measuring and observing user performance may be the gold standard for assessing usability, this does not imply that these metrics are the most effective means of creating a predictive design approach for usability.

Because usability metrics are based on measurements of extremely particular user activities in very specific settings, there is an issue with them. The designer can create objectives for measurable observations after she is aware of the situation and activities. But designers don't have access to this knowledge in the early phases of the design process. Consider the VCR usability specification we used as an example. When determining what levels of regression are acceptable and unacceptable for backward recovery, it is assumed that an undo button will be present. Actually, the designer previously implicitly assumed that mistakes will be made by the user when programming the VCR. If we could solve the source of the code errors, perhaps undo wouldn't be required.

It is important to acknowledge that usability engineering has an inherent constraint in that it just offers a way to meet usability requirements, not necessarily usability. Even yet, the designer must comprehend why a certain usability statistic improves usability for actual users. Once more, the inventor of the VCR example made the assumption that the undo process would be simpler with fewer explicit actions. Is it appropriate to make such an assumption?

2.2.3 Iterative Design and Prototyping

We previously brought up the fact that specifications cannot be finalised at the outset of the life cycle of an interactive system. Certain aspects of the proposed design can only be confirmed by building and evaluating them on actual users. Any incorrect assumptions that emerged from the testing can subsequently be incorporated into the design. This is the fundamental idea behind iterative design, a deliberate design process that cycles through multiple concepts, gradually improving the final result with each iteration in an attempt to address the inherent issues of insufficient needs specification.

(c) Amity University Online

Notes

The usability elements of the planned system are not the only ones that give rise to an iterative design philosophy due to design process issues. The issue pertains to requirements specification in general, making it a general software engineering problem that also involves administrative and technological concerns.

Prototypes are technical artefacts that mimic or animate some but not all of the features of the desired system; this is how iterative design is defined. Three primary methods exist for prototyping:

Throw-away: The model is constructed and evaluated. The actual prototype is thrown away; instead, the design knowledge acquired from this exercise is utilised to construct the finished product. The process of creating a final requirements specification from throwaway prototypes so that the rest of the design process can move forward is shown in the figure below.

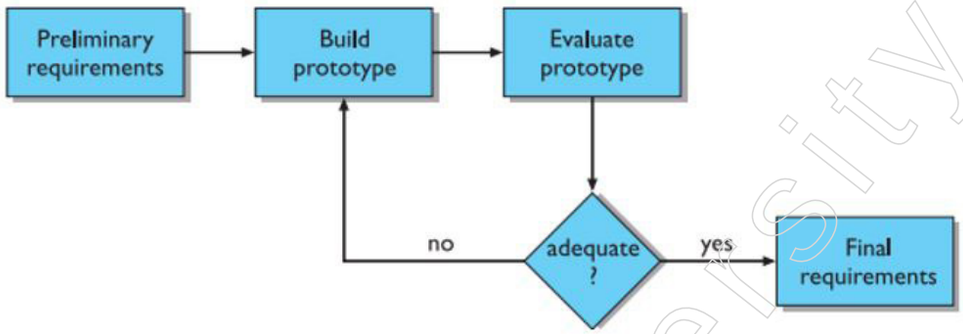


Figure 51: Throw-away prototyping within requirements specification

Incremental: One component at a time, the finished product is constructed from different parts. The finished system has a single overall design that is divided into separate, more manageable parts. After then, the finished product is released one component at a time in a sequence of releases. This is shown in the figure that follows.

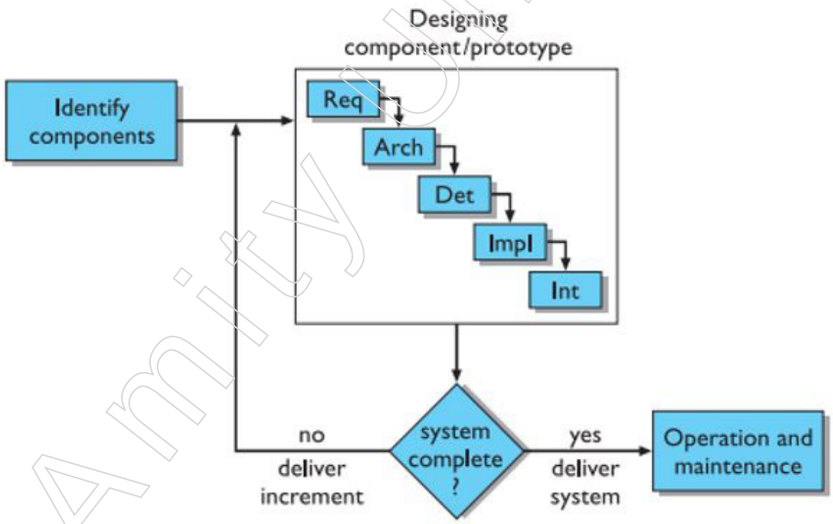


Figure 52: Incremental prototyping within the life cycle

Evolutionary: In this case, the prototype is kept and used as the foundation for the subsequent design iteration. As seen in the figure below, the real system in this instance is viewed as progressing from a very limited first version to its final release. Evolutionary prototyping is also a good fit for the changes that need to be made to the system as it operates and requires maintenance over its lifetime.

© Amity University Online

Notes

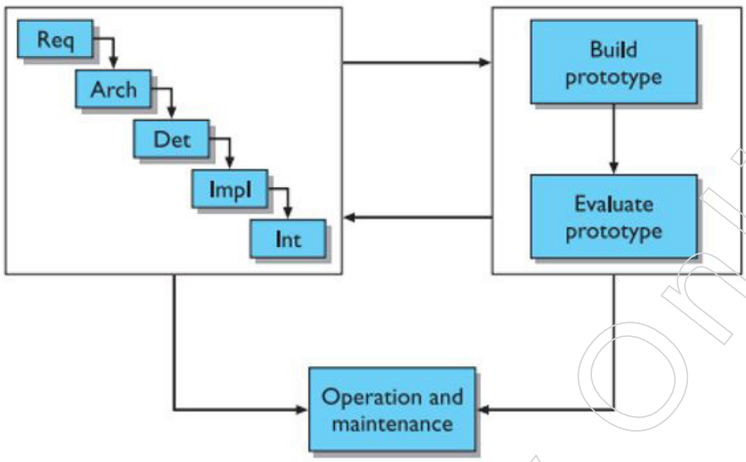


Figure 53: Evolutionary prototyping throughout the life cycle

The functionality and performance that prototypes offer in comparison to the finished product varies. For evaluation reasons, an animation of requirements may include limited functionality or no functionality at all, simulating only a portion of the interactive behaviour. On the other hand, complete capability may be offered at the price of other performance attributes like fault tolerance or speed. A prototype is significant regardless of its functionality because of the perceived realism it conveys. An interactive system prototype is used to test requirements by assessing their effects on actual users. Only when the evaluation conditions closely resemble those expected for the real operation can an accurate assessment of the final system's requirements be relied upon. However, achieving realism is expensive, thus assistance is needed so that a designer or programmer may rapidly and effectively produce a realistic prototype.

Sommerville identified a number of potential issues with management, including:

Time:It takes time to build prototypes and if they are disposable, that time could be better spent on other design-related tasks. Therefore, prototyping is only valuable if it is done quickly, which is why the phrase "rapid prototyping" is used. Rapid prototype development and modification should not be confused with hurried evaluation, though, since this could provide inaccurate findings and negate the main benefit of rganizat a prototype in the first place.

Planning: The majority of project managers lack the expertise required to properly schedule and budget for a design process that includes prototyping.

Non-functional features: Safety and dependability are two examples of non-functional aspects that are frequently the most crucial to a system and they are the same features that are given up in the process of creating a prototype. Response time, another element that is frequently sacrificed in a prototype, may be crucial for assessing a prototype's usability aspects and determining whether or not the product will be accepted. Prototype realism is a technical challenge that is akin to this problem.

Contracts: Many of these management and technological concerns have an impact on contractual agreements between the designer and the client, which frequently regulate the design process. An iterative design approach will still need paperwork that acts as the legally enforceable agreement because prototypes and other implementations cannot serve as the foundation for a contract. The outcomes of prototyping must be effectively translated into sufficient documentation. Changes can be made quickly during a rapid prototyping process, but the design process is not the same.

(c) Amity University Online

Notes

Techniques for prototyping

Here are some methods for creating quick prototypes that are currently available:

Storyboards

A storyboard, or schematic representation of the proposed system's external design without any associated system functionality, is perhaps the most basic idea of a prototype. The creation of storyboards doesn't require a lot of processing power; in fact, any computing resource can be used to mock one up. Storyboards have their roots in the film business, where a succession of panels loosely illustrates still pictures from a planned scenario to convey the overall idea. In the context of interactive system design, storyboards offer momentary views of the interface during specific phases of the interaction. Determining whether the design is on the right track can be done quite fast by analysing user or consumer perceptions of the storyboards.

With the use of contemporary graphical sketching software, storyboards may now be manually created on a computer. The finished system must be exhibited on a screen, therefore even while the graphic design achievable on screen might not be as intricate as that doable by a professional graphic designer, it is more realistic. Additionally, a sequence of images can be automatically sequenced to produce a basic yet effective animation. Traditional paper-based storyboards might not be able to capture the dynamic characteristics of the planned user–system interaction, but animation can. Storyboards, if they aren't animated, typically have scripts and annotations explaining how the interaction will happen.

High-level programming support

One example of a special-purpose high-level programming language that makes it simple for the designer to program specific interactive system capabilities at the price of other system properties, such as responsiveness or spatial efficiency, is HyperTalk. Programmers can associate functional behaviour with certain user inputs, like moving and clicking the mouse pointer over a button on the screen, using HyperTalk and many more languages of a similar nature. The challenge with interactive programming in the past was that it depended so heavily on the implementation that controlling even the most basic interactive behaviour required the programmer to have a thorough understanding of the hardware system. Programmers may now abstract from hardware details and consider more in terms of how input and output devices are regarded as interaction devices thanks to these high-level programming languages.

A user interface management system, often known as UIMS (pronounced "you-imz"), might be thought of as offering such high-level programming support, even though it isn't typically discussed in conjunction with such simulation environments. In interactive system design, the separation of application functionality and appearance is a commonly proposed conceptual model. After that, it is feasible to independently program the system's core functions and the user interface's behaviour. Therefore, a UIMS's role is to let the programmer to make the connection between the functionality that lies beneath the surface behaviour. The important thing to note in this case is that the implied separation provided by a UIMS permits the independent development of the interface's features from the underlying functionality. It is possible to quickly design and assess multiple interface prototypes to find the best one if the underlying system is already developed.

2.3 Design Rules

Giving designers the capacity to assess how their design choices will affect usability is one of the main issues that needs to be resolved in a user-centered design approach.

(c) Amity University Online

Notes

We need design guidelines, which are guidelines that a designer can adhere to in order to improve the software product's final usability. Based on the authority and generality of the rule, we can categorise these regulations along two dimensions. By authority, we mean a signal that specifies whether the rule is required to be followed in design or if it is merely recommended. When we talk about generality, we mean if the rule is applicable in a wide range of design scenarios or if it is specific to a smaller subset of scenarios. The degree of abstraction in rules varies as well; some are quite detailed, while others abstract away from the specifics of the design solution. Determining a design rule's roots is equally crucial. Several categories of design guidelines will be discussed. Abstract design guidelines with minimal authority and high generality are called principles. While guidelines often have a lesser authority and a more broad scope of application, standards are explicit design norms with a high degree of authority.

Psychological, cognitive, ergonomic, sociological, economic, or computational theory—which may or may not be grounded in empirical data—can provide support for design guidelines for interactive systems. It is not always the case that designers possess the appropriate training in psychology, cognitive science, ergonomics, sociology, business, or computer science to comprehend the implications of those ideas in the context of the design they are developing. The theory is put to use in practice through the design rules. A set of design guidelines will frequently contradict one another, making strict adherence to them all impractical. The designer can better grasp the trade-offs involved in adhering to or ignoring specific design guidelines by understanding the philosophy behind each one. A design rule's potential of conflicting with other rules and the necessity for the designer to comprehend its underlying theory usually increase with its level of generality.

We can distinguish between principles, standards and guidelines in general terms once more. Principles depend considerably more on a thorough comprehension of the human element in the interaction than they do on technology; they are generated from knowledge of the psychological, computational and sociological components of the issue areas. As a result, they are broadly applicable but less helpful for providing precise design guidance. Though guidelines are generally more technology-focused and less abstract, it is nevertheless vital for designers to be aware of the theoretical underpinnings that underpin them. Applying a standard won't require a designer to be as knowledgeable about the underlying idea. However, the accuracy or soundness of the underlying theory is more crucial because standards have a far larger degree of authority.

2.3.1 Standards

In order to guarantee that a wide community follows a set of design guidelines, national or international organisations typically develop standards for interactive system design. Specific standards may be applied to the software or hardware rganiza in the construction of the interactive system. Smith identifies the differences between software and hardware that impact the applicability of design standards to each:

Underlying theory: Hardware standards are predicated on knowledge of human factors, ergonomics and physiology, the outcomes of which are reasonably well known, constant and easily adapted to hardware design. Software standards, on the other hand, are derived from less developed, still-evolving theories in psychology or cognitive science that are difficult to understand in the context of software design. Therefore, whereas software standards would need to be more ambiguously phrased, standards for hardware can directly relate to a hardware specification and still reflect the underlying principle.

Change: Software is typically built to be far more versatile than hardware, which makes changes to it more costly and complicated. As a result, modifications to hardware

Notes

requirements happen less frequently than those to software. Standards are better suited for hardware than software since they are also more stable.

For these reasons, historically, hardware standards have been established before software standards by a particular standards rganization, such as the British Standards Institution (BSI), the International Organisation for Standardisation (ISO), or a national military agency. An Interim Defence Standard 00-25 on Human Factors for Equipment Designers, for instance, was released by the UK Ministry of Defence and is divided into 12 parts:

Part 1 Introduction

Part 2 Body Size

Part 3 Body Strength and Stamina

Part 4 Workplace Design

Part 5 Stresses and Hazards

Part 6 Vision and Lighting

Part 7 Visual Displays

Part 8 Auditory Information

Part 9 Voice Communication

Part 10 Controls

Part 11 Design for Maintainability

Part 12 Systems

The software design process is only addressed in the final one of these. There are 17 components in the international standard ISO 9241, Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs). The criteria for visual display, keyboard layout, workstation layout, environment, display with reflections, display colours and non-keyboard input devices are seven of these that deal with hardware concerns. Software-related topics are covered in seven sections: command dialogues, direct manipulation dialogues, menu dialogues, information display, user assistance, general dialogue concepts and form-filling dialogues. Nonetheless, standards addressing software-related topics are now being created. For instance, software ergonomics for multimedia user interfaces is covered in the draft ISO 14915 standard.

The following figure presents Instances of the terminology used in display standards. As we move from the hardware concerns in a UK defence standard for pilot cockpit controls and instrumentation via a German standard for user interface design of display workstations to a US military standard for display contents, take note of the growing generality and vagueness of the wording.

11.3 *Arrangement of displays*
11.3.1 Vertical Grouping. The engine display parameters shall be arranged so that the primary or most important display for a particular engine and airplane (thrust, torque, RPM, etc.) be located at the top of the display group if a vertical grouping is provided. The next most important display parameter shall be positioned under the primary display progressing down the panel with the least important at the bottom.

(a) A typical example of a military standard

(c) Amity University Online

Notes

5.1 *Subdivision of the display area*
In consideration of a simple, fast and accurate visual acquisition, the display area shall be divided into different sub-areas.
Such a division should be:

- Input area
- Output area
- Area for operational indications (such as status and alarms)

(b) From German standard DIN 66 234 Part 3 (1984), adapted from Smith [324]

5.15.3.2.1 *Standardization*
The content of displays within a system shall be presented in a consistent manner.

(c) From US military standard MIL-STD-1472C, revised (1983), adapted from Smith [324]

Figure 54: Sample design standards for displays

There is a usability specification component in ISO standard 9241 that is applicable to both software and hardware design. The definition of usability stated at the outset of that document is as follows:

Usability: The degree to which specific users accomplish specific objectives in specific settings with efficacy, efficiency and happiness.

Effectiveness: The precision and comprehensiveness with which specific users can accomplish specific objectives in specific settings.

Efficiency: The resources used in relation to the precision and entirety of the objectives attained.

Satisfaction: the ease of use and acceptability of the work system among those who utilise it and those who are impacted by it.

Such a description is crucial to the standard since it provides a way to define precise usability measurements.

The power of a norm to compel vast populations to follow it—the so-called authority we mentioned earlier—is what makes it strong. It should be remembered that the publishing of a standard by a national or international organisation does not always confer such authority. Several software design standards are really proposed as guidelines rather than requirements. A standard's (or guideline's, for that matter) authority can only be ascertained by looking at how it is actually used. Long before a formal standards document is released, certain software products (like the X windowing system) become de facto standards.

2.3.2 Guidelines

We have noted that the development of authoritative and precise standards is hampered by the incompleteness of the theories that guide the design of interactive software. Because of this, most design standards for interactive systems are more suggestive and broad in nature. Analysing the abundance of existing rules, we are interested in finding out how applicable they are to different stages of the design process.

To a certain extent, the rules can also be automated, offering a straightforward way to convert intricate design specifications into real implementation. Guidelines for interactive system design (sometimes called guidelines for user interface design) have been published in copious quantities. To illustrate the content of rules in that extensive literature, we will only provide a few instances here.

(c) Amity University Online

Notes

Huge catalogues of guidelines can be found in several books and technical studies. A famous example was a very general list that Smith and Mosier of Mitre Corporation, with sponsorship from the US Air Force's Electronic Systems Division, put up in 1986. The Smith and Mosier criteria fall under the following basic categories:

- Data Entry
- Data Display
- Sequence Control
- User Guidance
- Data Transmission
- Data Protection

The specific guidelines are contained in more detailed subcategories within each of these main categories. An illustration of the data found in the Smith and Mosier guidelines can be found in the figure below. The catalog's substantial cross-referencing and citations to published work supporting each guideline are remarkable features of this collection of recommendations. By utilising this structure, the Mitre Corporation has used the Smith and Mosier recommendations on a hypertext system that allows quick navigation of the network of guidelines to look up cross-references and citations.

1. Data Entry

1.1 *Position Designation*

1.1–1 Distinctive Cursor

For position designation on an electronic display, provide a movable cursor with distinctive visual features (shape, blink, etc.).

Exception When position designation involves only selection among displayed alternatives, highlighting selected items might be used instead of a separately displayed cursor.

Comment When choosing a cursor shape, consider the general content of the display. For instance, an underscore cursor would be difficult to see on a display of underscored text, or on a graphical display containing many other lines.

Comment If the cursor is changed to denote different functions (e.g. to signal deletion rather than entry), then each different cursor should be distinguishable from the others.

Comment If multiple cursors are used on the same display (e.g. one for alphanumeric entry and one for line drawing), then each cursor should be distinguishable from the others.

Reference Whitfield, Ball and Bird, 1983

See also 1.1–17 Distinctive multiple cursors
4.0–9 Distinctive cursor

Figure 55: Sample guideline from Smith and Mosier

Table 7: Comparison of dialog styles mentioned in guidelines

Smith and Mosier [325]	Mayhew [230]
Question and answer	Question and answer
Form filling	Fill-in forms
Menu selection	Menus
Function keys	Function keys

Notes

Command language	Command language
Query language	–
Natural language	Natural language
Graphic selection	Direct manipulation

Mayhew has put together a more recent, just as extensive list of general guidelines. This catalogue is one of the best resources for the experimental findings that support the particular guidelines, even if it is only available in book form, which restricts the ease of cross-referencing.

Dialogue styles, which in the context of these rules refer to the ways in which the user conveys input to the system, including how the system presents the communication device, are a primary concern for all of the general standards. Mayhew lists seven distinct dialogue styles, whereas Smith and Mosier include eight (see Table above for a comparison). The only actual distinction is that query languages aren't included in Mayhew's list; nonetheless, query languages are only special cases of command languages.

The majority of recommendations can be used to adopt any one of these dialogue styles on Its own. It's crucial to take into account the potential for combining several dialogue styles inside a single application.

Transitioning from general recommendations to automated and more targeted ones requires making assumptions about the computer platform the interactive system is built upon. Thus, for instance, there is a distinct difference between the actual rules—which presume certain Macintosh hardware and software—and the abstract guidelines, or concepts, in Appl''s Human Interface rules: the Apple Desktop Interface. The so-called programming philosophy that Apple wants designers to use while creating Macintosh apps is laid out in the abstract rules. Then, it is believed that the more specific guidelines represent more specific examples of that concept.

For instance, consistency is one abstract idea supported by the Apple guidelines:

2.3.4 Applications that work well are consistent both amongst and within themselves. Golden Rules and Heuristics

Thus far, we have examined various theoretical frameworks and comprehensive recommendations that can assist designers in creating more intuitive systems. However, adhering to all of these criteria necessitates a certain level of effort from the designer, whether it be in finding the right guidelines or in understanding the guiding concepts. Is there another easier way?

A variety of""golden rule"" or heuristics have been put out by proponents of user-centered design. These design guidelines are unavoidably""broad-brus"" guidelines that might not always apply to all circumstances, but they do offer a helpful summary or checklist of the most important aspects of design guidance. It is obvious that any designer who adheres to even these basic guidelines will create a system that is superior to one that does not.

Although there are other sets of heuristics, the most popular ones are Norma''s seven principles, Shneiderma''s eight golden rules and Nielse''s ten heuristics.

Shneiderman's Eight Golden Rules of Interface Design

The essential ideas of interface design are conveniently and concisely rganizatid by Shneiderma''s eight golden guidelines. Although their primary use is for design purposes, they can also be rganizad for system evaluation, much like Nielse''s heuristics.

Notes

1. Aim for uniformity in the usage of commands, vocabulary, action sequences, layout and other elements.
2. Give regular users the option to employ keyboard shortcuts, such as macros, unusual key combinations and abbreviations, to complete common, everyday tasks more rapidly.
3. Provide insightful feedback in proportion to the use''s activity level for each and every action.
4. Create dialogue boxes with yield closure in mind so that users can rganizate when the''ve finished a task.
5. Provide error prevention and straightforward error management so that users are ideally kept from making mistakes and, in the event that they do, are provided with understandable and helpful directions to help them get back on track.
6. Allow for simple action reversals to reduce anxiety and promote exploration because the user will feel less anxious knowing that he can always go back to the initial condition.
7. Encourage internal locus of control, which puts the user in charge of the system and makes it react to his actions.
8. By simplifying displays, combining multiple page displays and allocating time for action sequence learning, short-term memory burden can be rganizatd.

These guidelines offer a helpful abbreviation for the more comprehensive sets of ideas previously discussed. Similar to those guidelines, they must be evaluated for each unique circumstance and are not relevant to all possible outcomes. They are, nevertheless, generally helpful and using them will only benefit the majority of design tasks.

Norman's Seven Principles for Transforming Difficult Tasks into Simple Ones

The following seven principles are used by Norman to summarise user-centered design in his book, The Design of Everyday Things:

1. Use both knowledge in the world and knowledge in the head. When the information required to complete a task is externally accessible, either explicitly or through environmental limitations, people perform better. However, in order to become more efficient, professionals must also be able to rganizatioe routine duties. In order to help the user create a suitable mental model of what is happening, systems should offer the information required within the context and operate transparently.
2. Simplify the structure of tasks. Simple tasks are necessary to prevent complicated problem solving and an excessive amount of memory usage. Task structure can be made simpler in a variety of ways. One is to give the user mental support to assist them remember steps in a more difficult task. Another is to employ technology to give the user greater feedback and more work information. Automating all or a portion of the task is a third strategy, provided that it does''t negatively impact the use''s experience. The last method of simplification is to alter the tas''s nature to make it easier to complete. I''s crucial in all of this to maintain the use''s control.
3. Make things visible: close the gaps between execution and assessment. The user interface should allow them to easily see how their actions affect the system and should explain what the system is capable of doing.
4. Get the mappings right. System controls should be clearly mapped to user objectives. It should be obvious how user activities relate to system events. Thus, it ought to be obvious what affects what and to what extent. A slight movement has a small effect and a huge movement has a large effect. Controls, sliders and dials should reflect the task.

Notes

5. Exploit the power of constraints, both natural and artificial. Things in the world known as constraints make it impossible to accomplish anything other than the right action in the right way. A jigsaw puzzle, where the parts can only fit together in one way, is a basic example. In this case, the user is guided to finish the task by the physical limitations of the design.
6. Design for error. Since human mistake is inevitable, account for potential user errors and incorporate recovery mechanisms into the system.
7. When all else fails, standardize. Users should only need to learn arbitrary mappings once, provided that there are no natural mappings. Because critical controls are rganizationd, drivers can easily get into and operate a new car thanks to this rganizational n principle. While the windscreen wipers may occasionally be turned on by mistake, the essential controls (brake, clutch, steering and accelerator) remain constant.

Although Norma''s seven principles serve as a helpful synopsis of his user-centered design theory, The Design of Everyday Things should be read in its entirety to get the whole story.

2.3.4 HCI Patterns

Patterns have been effectively applied in software development to capture solutions to common programming challenges and they have their origins in architecture. They have been rganizad in web and interface design more recently.

An invariant solution to a recurring issue in a particular setting is called a pattern. Patterns offer a""solution statemen"" to the issues that designers encounter. Examples are the greatest way to explain this. Alexander, the person who first proposed the idea of patterns, suggests a pattern for designing homes called""Light on Two Sides of Every Room"" Here, the issue being discussed is that""People will always gravitate towards rooms with two sources of light when given the option, leaving unoccupied and empty rooms with only one source of light""

The suggested fix is to let natural light into each room from two directions:

Place windows in these exterior walls so that natural light enters every room from many directions. Orient each room so that it has at least two sides with an outside space.

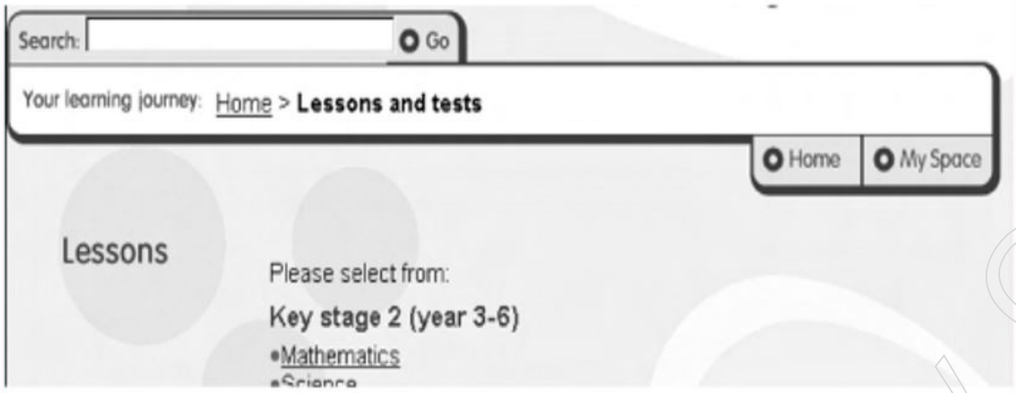
It should be noted that the solution makes no mention of the placement or angle at which these windows should be placed in relation to one another. The pattern would be completed in a room with windows on opposing walls, at right angles, or with a window plus a skylight. Only the invariant qualities of excellent design—the characteristics that are shared by every instance of the solution—are captured by patterns. The situation and the designe''s inventiveness will determine how the pattern is specifically implemented.

HCI patterns come in a variety of forms and those who are interested can refer to pattern collections and languages like the Pattern Gallery, which shows some of the different forms that can be employed in HCI patterns. As an example, Tidwel''s Common Ground collectio''s well-known phrase""go back to a safe plac"" is provided (see Figure below). Although patterns can also handle high-level concerns like cooperative groups or rganizationall structures, this pattern of interface is quite low-level. As you can see, the pattern not only identifies the issue and offers a solution, but it also provides a justification, explains how it originated, what situations it applies to and provides examples to highlight the pattern.

(c) Amity University Online

Notes

... NAVIGABLE SPACES or STEP BY STEP and a CONTROL PANEL are in place, which require the user to be able to move through the steps page by page.



From (www.learn.co.uk). An example of a toolbar that provides the option to return home and keeps track of the learning journey, providing the option to link back to a safe place.

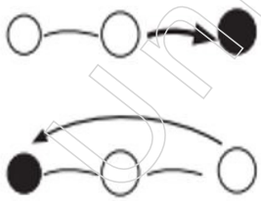
❖ ❖ ❖

It is easy to get lost in the tangle of links in a website.

People don't use the web like a TV or magazine. They use the web to find what they are looking for and then stop. They may select a wrong link and not be able to find their way back to something relevant. They do not always keep track of where they've been. They may forget where they have been if they are interrupted when using the site.

You are more likely to explore a website if you are sure that you can easily get out of an undesired state or space; that assurance engenders a feeling of security. Backtracking out of a long navigation path can be very tedious.

Therefore:
Always include a way back to a place that acts as a 'vantage point' from where you can reorientate yourself.



This pattern can be used in conjunction with GO BACK ONE STEP and CONTINUE TO NEXT STEP.

Figure 56: An example pattern 'go back to a safe place' adapted from Tidwell's Common Ground collection.

The pattern has references to other patterns as well, with the top references denoting the context in which it can be used and the bottom references denoting any other patterns that could be required to finish it. This creates a language by joining the patterns together. Patterns alone are not very useful, but by working through the hierarchy and these references, the user can be helped to create a fully realised design.

Several characteristics set patterns and pattern languages apart from other design rules when considered collectively:

- They are derived from actual practice rather than psychological theory and they encapsulate design practice and understanding of effective solutions.
- They encapsulate the fundamental characteristics of excellent design, which are that they advise the designer on what must be done and why rather than how to accomplish it.

(c) Amity University Online

Notes

- They reflect different levels of design expertise, from conceptual design to intricate widget design and from social and organisational difficulties to design knowledge at all.
- They are not impartial; rather, their reasoning embodies values. Alexander's appreciation of building is evident in the way he uses language. HCI patterns can convey principles of what constitutes a humane interface.
- Since the idea of a pattern language is generative, it can help in the creation of comprehensive designs.
- They can be utilised for communication between all stakeholders because they are often readable and intuitive.

In HCI representations, patterns are a relatively new addition and there are still a lot of unanswered research questions. For example, it is not evident how to identify patterns most effectively or how to build languages to take interaction's temporal concerns into account. The release of a comprehensive pattern language for web design, targeted at commercial designers, may, nevertheless, signal a turning point and lead to a broader use of the methodology in interface design.

2.3.5 Task Centered System: Design and User-Centred Design Prototyping

Task-centered system design is a kind of user-centered design that enables programmers to create and assess user interfaces based on actual tasks performed by users. This becomes a requirements analysis as part of the design (requirements are the activities to be accomplished). Prototypes can be examined and tasks can be used to develop detailed scenarios of how users should interact with the system as part of the evaluation process. You will be prompted with the following queries at every stage of the walkthrough: Does the individual know enough to do it? If not, a mistake was discovered.

What is a task?

Tasks are defined by the way they are completed. Tasks are human behaviours that help the system achieve a useful goal. User performance, not computer performance, is defined by task analysis.

TCD Principles

- Recognising user Needs and Goals: The first step in the design process is to identify user needs and goals. In order to learn more about the users, their work and their surroundings, research must be done.
- Task Identification: Following an understanding of the objectives and requirements of the users, the tasks that they must do are identified. This entails determining each task's objectives, the procedures necessary to finish it and the data and materials required.
- Efficiency and Effectiveness In Design: The system should be created to support the tasks that were determined in the previous phase in the most effective and efficient manner feasible. This entails taking into account elements like the interface's design, the information's organisation and the application of technology.
- Iterative Design Process: Task-centered system design is an iterative approach that involves several cycles of testing and assessment to fine-tune the design.
- User-Centered Design: Task-centered system design places a strong emphasis on user-centered design concepts. When building the system, this entails taking user satisfaction, accessibility and usability into account.

(c) Amity University Online

Notes

- Evaluation: A crucial component of task-centered system design is evaluation. User testing and evaluation of the system enables the collection of input and the implementation of changes aimed at improving the system.

TCD Phases

Typically, task-centered system design (TCD) entails the following stages:

Analysis: Information about the users, their tasks and their surroundings are gathered at this phase. This entails watching and evaluating the current system and workflows in addition to performing user research through questionnaires and interviews.

Design: The system is designed during this phase using the tasks that were determined during the analysis phase. To visualise the sy andl, this entails producing wireframes, mockups and prototypes in addition to drafting development specs and standards.

Implementation: The system is put into place after the design is finished. This entails designing and constructing the system in addition to doing user testing and evaluation.

Evaluation: To get user input and pinpoint areas that need work, the system is assessed during this phase. This covers usability as well as obtaining opinions via questionnaires and interviews.

To make sure the system keeps meeting user needs once it is implemented, maintenance is carried out. This entails fixing bugs, adding new features to the system and altering it in response to user input.

It's important to keep in mind that TCD is an iterative process and these stages could be repeated several times to make sure the system is as effectively designed to suit user needs as feasible.

Ensuring software quality is crucial for delivering reliable, maintainable, and user-friendly products. Here are some best practices to follow:

1. Clear and Measurable Quality Objectives:
   - ❖ Establish precise quality goals at the outset of the project.
   - ❖ Make sure your quality targets are quantifiable and in line with your company's aims.
   - ❖ Establish standards for judging the software's quality, taking usability, security, performance, and dependability into consideration.
2. Robust Requirements Management:
   - ❖ Collect all of the stakeholders' detailed requirements and record them.
   - ❖ Employ strategies such as use cases, user stories, and prototypes to efficiently gather requirements.
   - ❖ Make that requirements are full, consistent, and traceable throughout the development process by validating and verifying them.
3. Continuous Testing:
   - ❖ Put into practice a thorough testing plan that incorporates acceptance, system, integration, and unit testing.
   - ❖ Optimise the automation of testing procedures to provide comprehensive and uniform coverage.
   - ❖ To promote quality from the outset, apply strategies like behavior-driven development (BDD) and test-driven development (TDD).

© Amity University Online

Notes

4. Code Reviews and Quality Assurance:
   - ❖ Review code frequently to find errors, make sure coding guidelines are being followed, and impart expertise to other team members.
   - ❖ To find possible problems like code smells, vulnerabilities, and performance bottlenecks, use static code analysis tools.
   - ❖ To preserve code quality throughout the development lifecycle, put in place quality gates and peer review procedures.
5. Continuous Integration and Delivery (CI/CD):
   - ❖ Automate the build, test, and deployment processes by implementing CI/CD methods.
   - ❖ Make certain that each code update is automatically tested, compiled, and deployed to environments similar to production.
   - ❖ To securely roll out changes and keep an eye on how they affect users, employ feature flags and canary releases.
6. Performance Optimization:
   - ❖ Early in the development process, locate performance bottlenecks and address them.
   - ❖ To examine and improve code for efficiency and resource use, use profiling tools.
   - ❖ To make sure the program satisfies performance requirements under various circumstances, undertake load and stress testing.
7. Security Measures:
   - ❖ Throughout the development lifecycle, apply security best practices such as threat modelling, secure coding principles, and security testing.
   - ❖ Conduct routine security audits, such as code reviews, penetration tests, and vulnerability scanning.
   - ❖ Update third-party dependencies and keep an eye out for security holes in frameworks and libraries.
8. User Experience (UX) Design:
   - ❖ Incorporate usability and UX designers into the design process to produce interfaces that are simple to use and intuitive.
   - ❖ To improve the user experience repeatedly, test the system with actual users and collect feedback.
   - ❖ By adhering to accessibility standards and guidelines, you can guarantee accessible for people with impairments.
9. Documentation and Knowledge Sharing:
   - ❖ Keep thorough records of all your paperwork, including as design papers, user manuals, and requirements specifications.
   - ❖ To aid in comprehension and upkeep, provide useful comments, README files, and inline documentation in your code.
   - ❖ Encourage the team to share information by using collaborative tools, pair programming, and code reviews.
10. Continuous Improvement:
    - ❖ To find areas for improvement, review and analyse software quality data on a regular basis.

(c) Amity University Online

Notes

- ❖ Organise retrospectives to look back on previous endeavours, acknowledge accomplishments, and pinpoint lessons discovered.
- ❖ Promote a culture of ongoing education and development by holding seminars, training sessions, and knowledge exchanges.

Ensuring that software processes and products adhere to specified quality standards is known as software quality assurance, or SQA. The following methods and resources are frequently employed in software quality assurance:

Techniques:

Static Testing:

- Code Reviews: Peer review of source code, either manually or automatically, is done to find errors, enhance code quality, and exchange expertise.
- Static Code Analysis: Automated source code analysis to find possible problems including security flaws, coding standards infractions, and code smells.

Dynamic Testing:

- Unit Testing: Writing and executing tests for individual units of code to verify their correctness.
- Integration Testing: Testing the integration of different modules or components to ensure they work together as expected.
- System Testing: Testing the entire system as a whole to validate its functionality and performance against requirements.
- Regression Testing: Re-running tests on previously tested software to ensure that new changes haven't introduced defects.

Test Automation:

- Test Scripting: Writing automated test scripts to perform repetitive tests and validate software functionality.
- Test Frameworks: Using frameworks like Selenium for web applications or Appium for mobile applications to automate testing across different platforms.
- Continuous Testing: Integrating automated testing into the CI/CD pipeline to ensure that every code change is thoroughly tested before deployment.

Performance Testing:

- Load Testing: Testing the system's behavior under normal and peak load conditions to assess its performance and scalability.
- Stress Testing: Subjecting the system to extreme conditions to identify its breaking point and potential failure modes.
- Performance Profiling: Analyzing the performance of the software to identify bottlenecks and optimize resource utilization.

Security Testing:

- Penetration Testing: Simulating real-world cyber attacks to identify security vulnerabilities and weaknesses in the software.
- Security Scanning: Using tools like vulnerability scanners and security analyzers to identify security flaws in the codebase.
- Security Code Review: Manual or automated review of code to identify potential

© Amity University Online

Notes

security vulnerabilities such as injection attacks, authentication flaws, and access control issues.

Summary

- The goal of interaction designers is to craft user experiences that are not only intuitive and efficient but also deeply satisfying. This involves a continuous consideration of user needs and preferences throughout the entire design process. Regular user testing and an iterative design approach are indispensable for refining products, ensuring optimal usability and fostering user engagement
- The software life cycle, also known as the software development life cycle (SDLC), is a systematic process used by software developers and project managers to design, develop, test, deploy and maintain software applications. It encompasses various phases and activities to ensure the successful creation and evolution of software.
- HCI patterns contribute to creating a consistent and efficient user experience by leveraging solutions that have proven successful in various contexts. Designers and developers can adapt and customize these patterns to fit the specific needs of their applications.
- Iterative design and prototyping are essential components of the software development and design process, particularly in the context of Human-Computer Interaction (HCI) and user experience design. These approaches emphasize continuous refinement, user feedback and incremental development to create products that better meet user needs.
- Shneiderman's Eight Golden Rules emphasize principles that contribute to the creation of user-friendly and efficient interfaces. By adhering to these rules, designers can enhance the overall user experience and facilitate effective interaction with digital systems.

Glossary

- HCI: Human Computer Interaction
- IxD: Interaction Design
- PC: Personal Computer
- LED: Light Emitting Diode
- LCD: Liquid Crystal Display
- VCR: Video Cassette Recorder
- TCD: Task-centered system design
- Persona – a rich description of Betty the Warehouse Manager

Check Your Understanding

1. What is the primary purpose of the Requirements Gathering and Analysis phase in the software life cycle?
   a. Implementation of code
   b. Evaluating system performance
   c. Defining and documenting software requirements
   d. Deployment of the software
2. Which software development model follows a sequential and linear approach, with each phase dependent on the completion of the previous one?

(c) Amity University Online

Notes

a. Agile
b. Waterfall
c. Scrum
d. Spiral

3. What is the main goal of the Testing phase in the software life cycle?
   a. Writing code
   b. Deploying the software
   c. Identifying and fixing defects
   d. Gathering user requirements
4. Which software development methodology emphasizes adaptability, collaboration and customer feedback throughout the development process?
   a. Waterfall
   b. Agile
   c. V-Model
   d. Incremental
5. In the context of the software life cycle, what does “SDLC” stand for?
   a. System Development Lifecycle
   b. Software Design and Logic Control
   c. Sequential Development and Life Cycle
   d. Systematic Deployment and Life Control

Exercise

1. What do you meant by the process of design?
2. Define the software life cycle.
3. Explain iterative design and prototyping.
4. Define golden rules and heuristics.
5. Define various HCI patterns

Learning Activities

1. Imagine you are leading a team in the development of a new e-commerce platform. Outline the key stages of the Software Development Life Cycle (SDLC) that your team will go through to ensure the successful delivery of the e-commerce platform. Provide a brief explanation of each stage and highlight the activities and considerations relevant to each phase.
2. Design the user interface for a new mobile banking application. Outline the key steps and considerations you would take in the interaction design process to ensure a user-friendly and effective mobile banking experience.

(c) Amity University Online

Notes

Module - III: Evaluation

Learning Objectives

At the end of this module, you will be able to understand:

- Define various goals of evaluation
- How to Choose an Evaluation Method
- Analyse observational techniques and query techniques
- Understand design principles and visuals

Introduction

The purpose of evaluation is to evaluate our designs and test our systems to make sure they work as intended and satisfy user expectations. It is not appropriate to view evaluation as a stand-alone step in the design process, much less as an afterthought if time allows. Evaluation ought to take place at every stage of the design process and its findings ought to influence design changes. Analytical and informal procedures can and should be employed instead of thorough experimental testing, which is clearly not always viable to do consistently throughout the design.

3.1 Evaluation in Human Computer Interaction (HCI)

Prototyping approaches and evaluation are closely related in this regard because the latter helps to guarantee that the design is continuously evaluated. This has the benefit of allowing issues to be resolved before significant time and money are spent on the actual implementation because it is far simpler to alter a design early on in the process than it is to do so later.

We may distinguish broadly between evaluations conducted by usability experts or designers that do not directly involve users and evaluations that look at how the system is really used. While the latter typically calls for a functioning prototype or implementation, the former is especially helpful for evaluating early designs and prototypes. This is a broad distinction, though, because in reality, user feedback may be solicited during the early stages of design (e.g., through focus groups) and expert-based analysis can be used to quickly and affordably assess the usability of finished systems.

3.1.1 Goals of Evaluation

Three primary objectives of evaluation are to determine the scope and usability of the system's capability, gauge how well users interacted with it and pinpoint any particular issues.

The functionality of the system is crucial since it has to meet the needs of the user. Stated differently, the system's design should facilitate users in accomplishing their intended duties with more ease. This entails not just providing the necessary capability within the system, but also making it easily accessible to the user in terms of the steps required to complete the task. It also entails aligning the system's use with the user's expectations for the assignment. For instance, if a filing clerk is accustomed to obtaining a customer's file using their mailing address, the computerised file system must to have at least that same capability. This kind of evaluation may also involve tracking how well the user interacts with the system in order to determine how well it supports the task.

(c) Amity University Online

Notes

It is crucial to evaluate the user's experience and the interaction's impact on him in addition to the system design's functional capabilities. This entails taking into account factors including the system's usability, ease of learning curve and user happiness. In the case of systems intended for leisure or entertainment, it might also incorporate his delight and emotional reaction. It's crucial to pinpoint any parts of the design that put the user under too much stress—for example, by asking for too much information to be retained. These kinds of qualities are measured in many evaluations.

The identification of certain design issues is the evaluation's ultimate objective. These could be elements of the design that, when used in the way intended, provide unexpected outcomes or perplex consumers. Naturally, depending on the root of the issue, this has to do with the design's usability as well as functionality. But its unique focus is on pinpointing problem areas so that they can be fixed.

3.1.2 Evaluation through Expert

As we've mentioned, assessment needs to happen all the way through the design phase. Specifically, it is best to complete a system's initial evaluation prior to beginning any implementation work. As the design can be changed before significant resource commitments are made, costly errors can be avoided if the design itself is evaluable. Errors are usually more expensive to correct and, hence, less likely to be fixed the later in the design process they are found. However, conducting user testing on a regular basis during the design phase can be costly and gathering reliable feedback on the interaction experience from unfinished ideas and prototypes can be challenging. As a result, several techniques have been put forth to assess interactive systems using expert analysis. These rely on the designer or a human factors specialist evaluating the design to determine how it will affect the average user.

The main goal is to pinpoint any regions that, by disregarding acknowledged empirical findings or by breaking known cognitive principles, are likely to produce problems. These approaches are flexible assessment methods since they may be applied at any point in the development process, from storyboards and prototypes to full implementations. Because they don't need user input, they are also reasonably priced. Nevertheless, they only evaluate whether a system adheres to recognised usability standards; they do not evaluate how the system is actually used.

Let us examine four methods of expert analysis: model utilisation, heuristic evaluation and cognitive walkthrough.

Cognitive Walkthrough

Polson and colleagues first proposed and then refined cognitive walkthrough in an effort to bring psychological theory into the informal and subjective walkthrough method.

The code walkthrough, which is widely used in software engineering, is where the cognitive walkthrough evaluation method originated. A thorough examination of a series of steps is necessary for walkthroughs. The sequence in the code walkthrough is a section of the program code that is walked through by reviewers to verify specific features (such as adherence to coding style, standardisation of variable spelling versus procedure calls and verification of no violations of system-wide invariants).

The actions that an interface will ask a user to take in order to complete a task that is known to exist are referred to as the sequence of actions in the cognitive walkthrough. After that, the evaluators "step through" the action sequence to look for any potential usability issues. The primary goal of a cognitive walkthrough is often to determine how simple a system is to learn. To be more precise, learning via exploration is the main objective.

© Amity University Online

Notes

Experience has shown that a lot of users would rather pick up a system's capabilities through hands-on exploration than through thorough training or reading through an instruction manual. As a result, the walkthrough's tests pose queries that deal with this exploratory learning. In order to accomplish this, the assessors go through each activity step by step and offer a "story" explaining why the step is appropriate for a new user or not. Four items are needed to complete a walkthrough (henceforth, "walkthrough" refers only to the cognitive walkthrough; other walkthrough types are not included).

1) a system prototype or specification. It should be rather detailed, but it need not be exhaustive. Small details like a menu's wording and location can have a significant impact.
2) a description of the system task that the user is expected to complete. The majority of consumers should find this job to be representative.
3) a comprehensive, written list of all the steps required to do the assignment using the suggested system.
4) a description of the users' identities and the range of expertise and knowledge the assessors can infer from them.

With this knowledge, the assessors proceed through the action sequence (mentioned in item 3 above) to provide a critical assessment of the system and a convincing account of its usefulness. In order to do this, the evaluators attempt to respond to each of the four questions that follow at each stage in the action sequence.

1) Is the effect of the action the same as the user's goal at that point?Every action taken by the user will have a distinct impact on the system. Does this outcome match the user's current goal as well as their intentions? For instance, is "saving a document" what the user wants to do if the action's result is to do so?
2) Will users see that the action is available?Will users be able to see, for instance, the button or menu item that initiates the action? It is not being asked if they would realise that this is the button they desire. This is only a simple question about whether they can see it when they need to. When a menu item is tucked away in a submenu or a VCR remote control has a covered panel of buttons, for example, the answer to this question may be "no."
3) Once users have found the correct action, will they know it is the one they need?This answers the preceding query. A button or menu item being visible is one thing, but will the user be able to identify that it is the one he needs to finish his task? This question concerns whether the action's meaning and consequence are evident, as opposed to the preceding one, which focused on the action's appearance.
4) After the action is taken, will users understand the feedback they get?Will the user be aware that he has completed the right activity, assuming you now assume that he did? Will the comments made be enough to verify what has truly occurred? The cycle of interaction between execution and evaluation is now complete. Users require relevant feedback to assess whether they have achieved their aim.

Recording the cognitive walkthrough is essential in order to maintain track of the design's strengths and areas for development. For the walkthrough, it is therefore a good idea to create some common evaluation forms. The information in items 1-4 of our first list above, along with the walkthrough's date, time and evaluators' names, would all be listed on the cover form. Next, a separate standard form is filled out answering each of the four questions in our second list above for each activity (from item 3 on the cover form). Any response that is negative for any of the questions related to a specific action needs to be recorded on a different report sheet for usability issues.

(c) Amity University Online

Notes

The system being created (and, if applicable, its version), the date, the evaluators and a thorough explanation of the usability issue should all be included in this problem report form. Indicating the problem's severity—that is, whether the assessors believe it will happen frequently and how detrimental it will be to users—is also helpful. Since it's not always possible to address every issue, this information will help the designers prioritise what needs to be fixed.

Heuristic Evaluation

A heuristic is a broad concept, guideline, or rule of thumb that can be applied to direct design decisions or to assess decisions that have already been made. Jakob Nielsen and Rolf Molich created heuristic assessment, a technique for organising system criticism that makes use of a number of broadly applicable, rather straightforward heuristics. Heuristic evaluation is helpful for assessing early design since it can be applied to a design specification. However, it can also be used to storyboards, prototypesand operational systems. As a result, it's a flexible and reasonably priced method. For this reason, it's frequently seen as a cheap usability method.

Heuristic evaluation's basic tenet is that several assessors independently analyse a system to identify possible usability issues. It is crucial that these assessors be numerous and that their assessments be carried out independently. According to Nielsen's experience, it is usually adequate to have three to five evaluators; at five, approximately seventy-five percent of the total usability issues are found.

A collection of ten heuristics is offered to help the assessors identify usability issues. The heuristics are associated with guidelines and principles. Heuristics relevant to the given domain can be used to supplement these as necessary. Thus, "awareness of other users" could be added as a heuristic, for instance, if the system is for synchronous group discussion. Other guidelines could be applied, even though Nielsen suggests using these 10 since they offer the best coverage of the most typical usability issues.

Every assessor evaluates the system and records any deviations from these heuristics that might point to a possible usability issue. Additionally, the evaluator determines the level of severity for each usability issue based on four criteria: the problem's frequency, its ease of resolution for the user, whether it will persist over time and how seriously the issue will be taken. On a scale of 0–4, these can be added together to create an overall severity rating:

0 = I don't agree that this is a usability problem at all

1 = Cosmetic problem only: need not be fixed unless extra time is available on project

2 = Minor usability problem: fixing this should be given low priority

3 = Major usability problem: important to fix, so should be given high priority

4 = Usability catastrophe: imperative to fix this before product can be released (Nielsen)

These are Nielsen's ten heuristics:

1. System status visibility Users should always be informed about what is happening by providing pertinent feedback in a timely manner. For instance, indicate how long and how much of a system operation is finished if it will take some time.
2. System compatibility with the actual world Instead of using terms that are specific to the system, the user should be able to understand the words, phrases and concepts that the system uses. Observe norms found in the real world and arrange the material in a logical and natural sequence.

Notes

3. User autonomy and liberty Users frequently select system functions by accident, so it's important to have a designated "emergency exit" so they can quickly quit the undesirable condition without having to go through a lengthy dialogue. Encourage "undo" and "redo"
4. Standards and consistency The meaning of words, circumstances, or acts should always be clear to users regardless of the context in which they are used. Observe the norms and customs of the platform.
5. Preventing errors Make mistakes tough to commit. Careful design that keeps a problem from happening in the first place is preferable to excellent error messages.
6. Identifying as opposed to remembering Show options, actions and objects. Information shouldn't need to be remembered as the user moves between sections of the dialogue. When necessary, the system's operating instructions should be clearly displayed or accessible.
7. Efficiency and adaptability of use Permit users to customise recurring activities. Accelerators, which are invisible to the beginner user, can frequently accelerate the expert user's engagement to the point where the system can accommodate both novice and expert users.
8. Design that is both elegant and simple Information that is rarely needed or irrelevant should not be included in dialogues. Each additional piece of information in a dialogue reduces the relative visibility of the pertinent pieces of information by competing with them.
9. Assist users in identifying, diagnosing and fixing issues Error messages should accurately describe the issue and offer a workable remedy. They should not use codes.
10. Assistance and records It could be required to offer assistance and documentation because not all systems can be operated without instructions. Any such material ought to be compact, easy to search, task-specific and provide a list of specific actions that need to be taken.

After every assessor has finished their individual evaluation, all of the issues are gathered and the mean severity ratings are computed. After that, the design team will decide which ones are most crucial and will get priority attention. While heuristic evaluation is a valuable method for assessing interface usability, it can face several challenges:

- Expert Availability: Recruiting and retaining usability experts can be challenging, especially if there are budget constraints. Hiring or contracting experienced evaluators may require financial resources that are not available.
- Time Constraints: Conducting a thorough heuristic evaluation takes time, as each evaluator needs sufficient time to review the interface independently and provide detailed feedback. Tight project schedules may not allow for the extended time required for this process.
- Availability of Resources: Access to appropriate tools and resources for conducting heuristic evaluations, such as usability evaluation software or specialized testing environments, may be limited due to budget constraints.
- Logistical Challenges: Coordinating the logistics of heuristic evaluation sessions, including scheduling meetings, distributing evaluation materials, and consolidating findings, can be challenging, especially if evaluators are located in different geographical locations.
- Limited Scope: Due to budget or time constraints, it may not be feasible to evaluate the entire interface comprehensively. This limitation can result in a narrower focus or less thorough evaluation of certain aspects of the interface.

(c) Amity University Online

- Bias and Subjectivity: Heuristic evaluation relies on the judgment and expertise of evaluators, which can introduce bias and subjectivity into the evaluation process. Without sufficient time and resources for training and calibration, consistency among evaluators may be compromised.
- Scope Creep: Budget and time constraints may lead to scope creep, where the evaluation expands beyond the initial scope, resulting in increased costs and delays.

Model-Based Evaluation

Modelling is a third expert-based strategy. Design definition and evaluation can be integrated into one framework using certain cognitive and design models.

As an illustration, the GOMS (goals, operators, methods and selection) model can be used to filter certain design possibilities and forecast user performance with a given interface. In a similar vein, lower-level modelling methods, such the keystroke-level model, forecast how long users will need to complete simple physical tasks.

Evaluation at the design stage also involves design techniques, such as design rationale. Design justification offers a framework for assessing different design possibilities. Informed decisions can be made regarding the design by looking at the criteria connected to each alternative and the proof offered to support these criteria. Evaluation of dialogue sequences for issues like complexity, inaccessible states and circular dialogues can also be done using dialogue models. Before being implemented, models like state transition networks might be helpful in assessing dialogue designs.

Using Previous Studies in Evaluation

There is an abundance of experimental data and empirical support for both experimental psychology and human–computer interaction. While much of it deals with more general difficulties and may be applied in a range of contexts, some of it is domain-specific. The usability of various menu types, the recall of command names and icon selection are a few examples of these problems.

A last method of expert review takes advantage of this heritage, substantiating (or disproving) some features of the design with prior findings. Repeating trials over and over is costly; this can be avoided with a thorough evaluation of pertinent material by a professional. It should be mentioned that results from experiments cannot be assumed to hold in all situations. As a result, the reviewer must carefully choose the evidence, taking into account the experimental design selected, the participant population employed, the analyses carried out and the assumptions made. An experiment evaluating a help system's usability with novice participants, for instance, might not accurately assess a help system intended for experienced users. Therefore, both the similarities and differences between the experimental situation and the proposed design should be considered in the review. This is the reason this is an expert review: knowledge of the subject is necessary to guarantee that the right assumptions are made.

Another method of evaluating a system's usability is called Model-Based Evaluation (MBE), which entails modelling user tasks, behaviours, and interactions. Nevertheless, MBE may encounter difficulties with funding, scheduling, and logistics, just like heuristic evaluation:

- Expertise and Training: Proficiency in both usability engineering and human-computer interaction (HCI) is necessary for the development of precise and efficient MBE models. It can take a lot of time and money to train assessors in the design and interpretation of models, particularly if specialised knowledge or software tools are needed.

© Amity University Online

Notes

- Resource Availability: Budgetary restrictions may restrict the availability of suitable resources and tools for developing and modelling user models. It could cost a lot of money to purchase specialised software for simulating virtual environments or user behaviour.
- Time Constraints: It might take a lot of time to create detailed models of user tasks and interactions, especially for complex systems or domains. Tight project timelines could not permit the considerable time needed to properly construct and evaluate these models.
- Data Collection: Observation, interviewing, and data analysis are some of the logistical issues associated with gathering information to support the development of realistic user models. The extent or magnitude of data collection activities may be constrained by a lack of time and resources.
- Model Validation: Ensuring the efficacy of MBE requires validating the precision and dependability of user models. However, this procedure can call for a great deal of testing and improvement, which could be resource-intensive and impractical given a limited budget or amount of time.
- Integration with Development Process: Logistically, integrating MBE into the software development lifecycle could be difficult, especially if there are already workflows, tools, or methods in place. Coordinating and carefully structuring development procedures to suit MBE practices may be necessary.
- Scope and Complexity: Time and money constraints may limit the breadth and intricacy of user models, which could lead to simplistic depictions of tasks or user behaviour. This may jeopardise MBE's reliability and efficacy as a usability assessment technique.

3.1.3 Evaluation through Users: Styles of Evaluation and Experimental Evaluation Design

So far, the methods we've looked at focus on analysing a system or design by the designer or another qualified evaluator, as opposed to conducting user testing. Though helpful in sifting and improving the design, these methods cannot take the place of genuine usability testing conducted with the system's intended users. We will examine several methods for user participation in evaluation in this section. These include physiological monitoring techniques, such as skin conductance and heart rate monitoring, observational approaches, empirical or experimental methods and query techniques.

In later stages of development, when the system has at least a functional prototype in place, user participation in evaluation is more common. This could be anything from a simple functioning prototype to a completely working system, or even just a simulation of the system's interactive capabilities without any of its underlying functionality (like the Wizard of Oz technique). But several of the techniques covered can also help with early design phases, including requirements capture, where user surveys and observation are crucial.

Styles of Evaluation

We will first distinguish between two different evaluation styles: laboratory-based and in-workplace, or "in the field," evaluations. Only then will we look at some of the strategies that might be evaluated with users.

Laboratory Studies

Users are removed from their regular work environments to participate in controlled tests in the first kind of evaluation studies, which are frequently conducted in a specialised

(c) Amity University Online

Notes

usability laboratory (though the "lab" could just be a silent room). There are some pros and cons to this strategy.

It is not possible to duplicate the advanced audio/visual recording and analysis capabilities, two-way mirrors, instrumented computers and other equipment found in a well-equipped usability laboratory in a work setting. The participant also works in a setting free from disruptions. But the unusual scenario and lack of context (file cabinets, wall calendars, books, or interruptions, for instance) could lead to an exact recording of an event that never happens in the actual world. Seeing multiple people working together on a task in a lab setting is particularly challenging because interpersonal communication depends so much on context.

However, there are other circumstances in which laboratory observation is the only viable choice, such as when the system must be installed in a hazardous or isolated area, like a space station. Additionally, a laboratory may be able to satisfactorily complete some extremely limited single-user tasks. Lastly and possibly most frequently, we might want to compare different designs in a controlled environment or purposefully alter the context to find issues or detect less usual practices. Laboratory investigations are suitable for these kinds of assessments.

Field Studies

To assess a system in action, the second kind of evaluation involves sending the designer or evaluator into the user's workspace. This strategy also has advantages and disadvantages.

Field observation is challenging due to high ambient noise levels, increased activity and frequent disruptions like phone calls. Nonetheless, because of how "open" the setting is, you will be able to see connections that would have gone unnoticed in a lab study—that is, interactions between people and systems. The user is seen in his "natural environment," and the context is preserved. Furthermore, some tasks—those are challenging even in the field—cannot be studied in a lab, such as those that take days or months to complete.

Overall, since field observation enables us to examine the interaction as it happens in real-world settings, it should be chosen above laboratory research. Even little disruptions matter because they reveal actions like storing and restoring state while working on a job. We must constantly operate at a modest distance from the natural scenario, a kind of Heisenberg uncertainty principle, because even in field observations, the participants are likely to be impacted by the analyst and/or recording equipment.

This is a generalisation, of course; as we have shown, there are situations where laboratory testing is both required and beneficial. Controlled studies, which are typically need to be carried out in a laboratory, are particularly helpful for evaluating particular interface aspects. From an economic perspective, we must compare the expenses of bringing one or more participants out of the workplace and into the laboratory vs the costs of setting up recording equipment in the field and perhaps upsetting the actual work environment. Any study must balance the higher expenses and difficulties of field investigations against the loss of contextual information, as this balance is not at all evident.

Empirical Methods: Experimental Evaluation

Conducting a controlled experiment is one of the most effective ways to assess a design or a feature of a design. This offers factual proof in favour of a specific assertion or theory. It can be applied to the analysis of many problems at various levels of detail.

Every experiment has an identical structure. An attribute of participant behaviour can be measured to help the evaluator select a hypothesis to test. Different experimental

Notes

settings are taken into consideration, with the exception of some controlled variable values. Any variations in the behavioural metrics are ascribed to the various circumstances. Many elements that are crucial to the experiment's overall dependability are included in this basic form and their careful consideration during experimental design is required. These consist of the selection of participants, the testing and manipulation of variables and the testing of the hypothesis.

Participants

Any experiment's ability to be successful depends on its participant selection. The goal of participant selection in evaluation trials should be to reflect the projected user group as nearly as feasible. The best scenario would entail conducting experimental testing on real consumers, however this isn't always feasible. In the event that participants are not actual users, they ought to be selected based on their age and educational attainment in relation to the target user base. Both their general computer experience and their familiarity with the systems under test, as well as their task domain expertise, should be comparable. Undergraduate computer science students make up the participant group, hence they are not a good representation of the intended user demographic for evaluating an interface meant for broad usage.

The sample size selected is a second concern regarding the participant set. A lot of the time, this is decided by practical factors like participant availability or resource scarcity. However, taking into account the experiment's design and the selected statistical techniques, the sample size must be sufficiently large to be regarded as representative of the population.

It is estimated by Nielsen and Landauer that a single participant in a usability test will identify approximately one-third of the problems and that testing with more than five participants yields little useful information. This may be true for observational studies in which the primary goal is to identify usability difficulties, but such small sample sizes make it impossible to draw firm conclusions about the scope of usability issues. It is definitely advised to double this amount if a controlled experiment is to be conducted and statistical analysis is to be done on the outcomes.

Variables

To test the hypothesis, controlled variables are manipulated and measured in experiments. There are two primary categories of variables: those that are measured (the dependent variables) and those that are "manipulated" or altered (the independent variables).

The components of the experiment that are changed to create several comparison conditions are known as independent variables. The quantity of menu items, the help level, the interface style and the icon design are a few examples of independent variables in evaluation tests. There are several possible values for each of these variables and each value that is applied in an experiment is referred to as the variable's level. Therefore, menus with five, seven and ten items might be taken into consideration in an experiment that aims to determine whether search speed increases as the number of menu items reduces. The quantity of menu items, the independent variable in this case, has three levels.

Multiple independent variables may be present in more intricate experiments. For instance, we might surmise from the experiment above that the user's response time is influenced by both the quantity of menu items and the commands they select from the menu. There are two independent variables in this instance. In order to explore every possible combination of three menu size levels and two command name levels, six experimental conditions would be needed if there were two sets of command names, or two levels.

(c) Amity University Online

Notes

Conversely, dependent variables are those that are quantifiable in the experiment and whose values are "dependent" upon modifications made to the independent variable. This would be the menu selection speed in the case above. The dependent variable needs to be measured in some manner, influenced by the independent variable and, to the greatest extent feasible, independent of external variables. The amount of time needed to finish a task, the amount of mistakes made, user preference and the calibre of the user's performance are typical choices for dependent variables in evaluation trials. Of course, it is easier to measure some of these in an objective manner than others. The more subjective measurements, however, can be used in comparison to preset scales and are often crucial to take into account.

Hypotheses

A hypothesis is a forecast for how an experiment will turn out. It is stated that a change in the independent variable will result in a change in the dependent variable. It is phrased in terms of the independent and dependent variables. The experiment's goal is to demonstrate the accuracy of this prediction. The null hypothesis, which claims that there is no difference in the dependent variable between the levels of the independent variable, must be refuted in order to achieve this. The following statistical measures yield values that can be compared at different significance levels. A significant finding indicates that the measured differences would not have happened by chance at the specified degree of certainty (i.e., the null hypothesis is false).

Experimental Design

An experiment needs to be carefully designed in order to yield results that can be trusted and applied broadly. The participants, the independent and dependent variables and the hypothesis are only a few of the elements that the researcher needs to take into account when creating the design. Selecting the hypothesis, or determining precisely what you are attempting to establish, is the first stage in the experimental design process. By identifying what you are intending to manipulate and what change you expect, you will have clarified the independent and dependent variables. You should reconsider your hypothesis if it fails to specifically specify these variables. You should now also think about your participants: how many are accessible and do they accurately reflect the user group?

Selecting the experimental approach that you will employ is the next stage. Within-subjects and between-subjects are the two primary approaches. Every participant in a between-subjects (or randomised) design is put into a separate condition. There are two conditions at least: the control, which is the same as the experimental condition but for this manipulation and the experimental condition (where the variable has been changed). This control makes sure that any differences that are measured are the result of the modification. Naturally, the number of independent variables and the number of levels that each variable can take will determine whether or not there are more than two groups.

A between-subjects design has the advantage of controlling any learning impact that might arise from having each user perform under a different condition after they have completed one under the other. The drawbacks are that larger sample sizes are needed and that large differences between the groups may obliterate any findings. Individual variations among users may also skew the outcomes. Ensuring that all participants are representative of the population through thorough selection and group matching can help address these issues.

Within-subjects (or repeated measures) is the second experimental design. Every user performs differently in this situation. Transfer of learning effects may occur with this design, but they can be mitigated if users complete the requirements in a different order.

© Amity University Online

Notes

For instance, group A might complete the first condition before moving on to the second, whereas group B would complete the second condition before moving on to the first. Since fewer users are needed, within-subjects is less expensive than between-subjects and it can be especially useful when learning is involved. Additionally, the likelihood of effects resulting from participant variance is reduced.

The available resources, the likelihood or controllability of learning transfer and the perceived representativeness of the participant group will all influence the choice of experimental design. When there are multiple independent variables, creating a mixed design with one variable put between groups and one within groups is a common compromise. Referring back to our menu design example, the participants would be divided into two groups, one for each command set and would be required to perform in three different conditions, which would correspond to the three conceivable levels of menu item count.

We must choose how to analyse the data we capture after deciding on the hypothesis we are attempting to test, the variables we are examining, the participants we have at our disposal and the best design. There are several statistical tests accessible and the experiment's success depends on the test selection. Different tests make different assumptions about the data and findings may not be valid if the wrong test is used.

Statistical Measures

Examining the data and storing the data are the first two guidelines for statistical analysis. While a quick look at a graph, histogram, or table of findings might be more enlightening, it is simple to do statistical tests without any guidance. Examining the data, in particular, can highlight outliers—single data points that stand out significantly from the rest. Outliers frequently arise from transcribing errors or unusual occurrences unrelated to the study. For instance, we observe that one person completed a task three times longer than the others. Upon further investigation, we find that the subject had the flu on the day of the study. It seems obvious that including participant data would skew the findings.

It's crucial to save the data since we might wish to use a different analytic technique later. It happens all too often for an experimental to tabulate results or take some averages and then discard the original data. In the worst case scenario, the residual statistics may not be useful for statistical analysis and in the best case scenario, we are no longer able to link unusual outcomes to the original data—something we would like to accomplish, for instance, with outliers.

The kind of data we have and the questions we hope to answer influence the statistical analysis we choose. Standard tests can be applied in many scenarios, however it is good having an experienced statistician review critical results. Variables go into one of two categories: continuous or discrete. A discrete variable, such as the colour of the screen—it can be red, green, or blue—can only have a limited number of values or levels. A continuous variable, such as a person's height or the amount of time needed to finish a task, can have any value, however it may have an upper or lower bound. Positive continuous data—for instance, a response time cannot be negative—represent a specific situation. By grouping a continuous variable into classes, we can make it discrete. For instance, we could separate heights into short (6 ft (1.8 m)) and tall classes. In numerous interface studies, we will be contrasting different designs. Typically, the independent variable in these situations is discrete.

The measured variable, which is liable to arbitrary experimental fluctuation, is the dependent variable. If this variable is continuous, the random variation can have a unique shape. In the event that the data conforms to a known distribution, more advanced statistical tests can be applied. These kinds of tests are known as parametric tests and

(c) Amity University Online

Notes

the most popular ones are applied in cases when the variation is distributed normally. This indicates that the random errors will form the well-known bell-shaped graph (Figure below) if we plot a histogram of them. Fortunately, a good number of these tests have reasonable robustness, meaning they produce valid answers even in cases when the data exhibit non-normality. This implies that, in the early stages of analysis, you shouldn't worry too much about verifying normalcy.

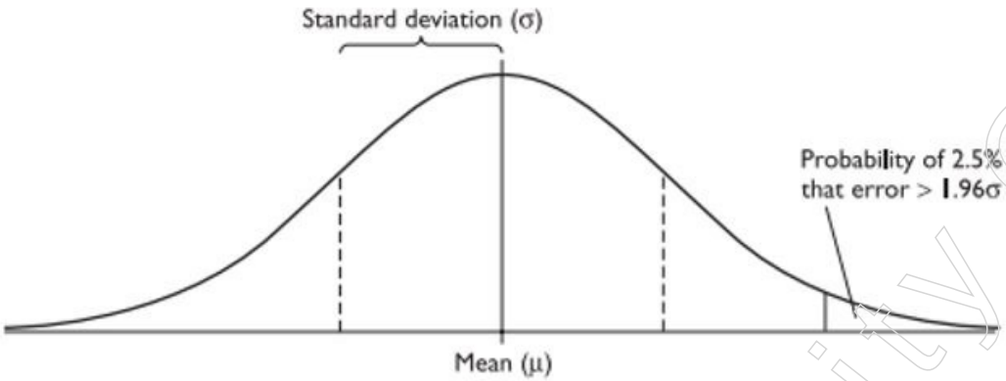


Figure 57: Histogram of normally distributed errors

There are methods for determining whether data are truly normal, but the reader should refer to a statistics book or a qualified statistician for these. On the other hand, statistics are generally considered to be normal if they can be viewed as the average or sum of numerous tiny independent effects. For instance, the total time required to do all of the smaller activities that make up a difficult task is the total time required to complete the complex activity. However, a subjective assessment of an interface's usability is abnormal. On rare occasions, data can be altered to approximate normality. The most popular transformation, applied to positive data with values close to zero, is the log-transformation. A log-transformation is frequently used by experimenters since it has minimal impact on data that are clustered far from zero. Nevertheless, this approach complicates the interpretation of the results and is not advised.

Often, non-parametric tests must be used when it is not possible to assume that the data are normally distributed. These statistical tests are typically based only on the data's ranking and don't make any assumptions about the specific distribution. That is, prior to analysis, every item in a data collection (57, 32, 61, 49, etc.) is lowered to its rank (3, 1, 4, 2). There is less chance of obtaining erroneous findings when using non-parametric tests since they are less presumptive about the data than parametric tests and are more resilient to outliers. They are not as strong as the matching parametric tests, though. This implies that a parametric test may find a difference that the non-parametric test would overlook if the identical set of data were used.

The contingency table is a third type of test in which data is categorised using many discrete qualities and the number of data items with each attribute combination is counted.

A few common tests are listed in the table below, arranged according to the kind of independent and dependent variables (discrete, continuous, or normal). The independent variable's normality is unaffected, however there is a specific situation where it is discrete and has only two values, as when comparing two systems. We are unable to cover every technique here, therefore you should consult a standard statistics work, such one on the reading list. The table is meant to serve as a guide only for you when selecting the test.

(c) Amity University Online

Notes

Table 8: Choosing a statistical technique

Independent variable	Dependent variable	
Parametric		
Two valued	Normal	Student's *t* test on difference of means
Discrete	Normal	ANOVA (ANalysis Of VAriance)
Continuous	Normal	Linear (or non-linear) regression factor analysis
Non-parametric		
Two valued	Continuous	Wilcoxon (or Mann–Whitney) rank-sum test
Discrete	Continuous	Rank-sum versions of ANOVA
Continuous	Continuous	Spearman's rank correlation
Contingency tests		
Two valued	Discrete	No special test, see next entry
Discrete	Discrete	Contingency table and chi-squared test
Continuous	Discrete	(Rare) Group independent variable and then as above

If a thorough and precise analysis provides the incorrect answer, it is useless. The following are some examples of queries one may pose regarding the data:

Does anything differ? Is one system better than another, for instance? Testing hypotheses is the term for methods that deal with this. Answers to this question take the form of "we are 99% certain that selection from menus of five items is faster than that from menus of seven items" rather than a simple yes or no.

To what extent is the difference? Selection from five items, for instance, takes 260 ms less time than selection from seven. Point estimate, which is frequently achieved via averages, is this.

To what extent is the estimate accurate? Selection, for instance, is 260 30 ms faster. Confidence intervals or measurements of variation like the estimate's standard deviation are the two types of statistical responses to this. Once more, the results are probabilistic: "We have 95% confidence that the response time difference is between 230 and 290 ms."

The first question mostly addresses the challenges of experimental design that we have examined. Nonetheless, the majority of the previously mentioned statistical methods, whether parametric or non-parametric, provide some sort of response to either one or both of the other queries.

Studies of Groups of Users

Thus far, we have examined the assessment of single-user systems through experimentation. Further issues arise from group system evaluation experiments. It should come as no surprise that groupware and group experimentation are more challenging than matching single-user research given the intricacies of human-to-human communication and group activity. Let's pretend for the sake of discussion that we are assessing a shared application that allows for video connections between participants and we'll talk about some of the issues that might come up.

The Participant Groups

Ten people are needed to plan, say, ten tests of a single-user system. Naturally, we will require 30 volunteers for the same number of experiments in an experiment with groups of three. Furthermore, group working experiments are frequently longer than their single-user counterparts since we need to give the group time to "settle down" and build

(c) Amity University Online

Notes

rapport. All of this implies increased participant disruption and perhaps increased expense reimbursement. It's not easy to find a time period that works for everyone when the equipment and participants are available. We strive to accommodate at least six persons, not including the experimenters themselves, as the workstations utilised in the experiment are frequently colleagues' personal computers. It should come as no surprise that reports of group work frequently only include three or four groups. While this is undoubtedly an issue for statistical analysis, it is not the main barrier.

The Experimental Task

Selecting an appropriate task is equally challenging. A range of task kinds, such as structured, creative, information-passing and so on, would be desirable to test. Additionally, the tasks must promote active cooperation because they either call for consensus or because they divide control and knowledge among the participants. Of course, the work also depends on the groupware system's characteristics; if it has multiple channels available, we want to promote widespread adoption. For instance, in a shared application with video, we are merely looking into video conferencing if the task can be completed (or at least made difficult) without using the program.

Creative assignments like "write a research proposal" or "write a short report on..." are generally successful because they force participants to come to a consensus and may require them to use the shared application to complete their final report. Tasks related to design are also employed. Users of the York Conferencer system, for example, were asked to remodel a bank layout in one experiment. The participants used a spatially structured electronic pinboard with an image of the current layout as the background, placing comments and ideas in relation to the characteristics they were referring to.

The purpose of decision games in management education is to assess and foster teamwork. They frequently rely more on teamwork than on personal talent to succeed. The desert survival challenge serves as an illustration of this, in which participants are informed that they have crashed in the desert. They are given a list of goods, such as a knife and plastic sheet, to prioritise for their survival. Together, the participants must create a single list; one knowledgeable person cannot "go it alone." Since the game of diplomacy incorporates both cooperative and adversarial elements, an electronic version has also been utilised.

Lastly, since participants are controlling many aspects of the model, time-sensitive simulated process control tasks necessitate a faster rate of engagement. A bottling factory simulation called ARKola [147] was utilised at Xerox PARC to study the significance of background noise in difficult cooperative control tasks. The selected task will frequently call for additional implementation work, which in the case of games could be substantial. This is undoubtedly a significant consideration while selecting an appropriate work.

Data Gathering

We could very well use many video cameras in addition to direct application logging, even in a single-user trial. In a group context, each member experiences this same thing. We are attempting to synchronise the recording of three keyboard records and at least six video sources for a group of three people. These could be dispersed among several offices or even locations, which would exacerbate the situation. It is obvious that there are serious technical issues. It is feasible to capture four images at once, one for each quadrant of the screen, but even this is not enough to accommodate the number of channels we would like.

One approach to get around this is to concentrate on each participant separately, recording the sounds they hear and the visual pictures that are being communicated as

© Amity University Online

Notes

part of the system (if there is a video connection). These can then be synced with additional video observations and the specific participant's keystrokes. We can therefore replicate the circumstances exactly as they seemed to the individual. We might not be able to decipher the other participants' activities from this tape, but at least we have a comprehensive record for one.

If enough recording equipment is available, each participant can have this done again. Fortunately, synchronisation between participants does not have to be as high as it would be for each person alone. Just set the recorders' clocks simultaneously and you won't have to worry about participants' sub-second accuracy. What matters is that each person's experience may be virtually recreated.

Analysis

We would like to see statistical differences between conditions of the experiment, in the real spirit of experimentation. Individual differences were a challenge in single-user tests, as previously observed. Group variation is, if anything, more pronounced. When put in randomly mixed groups, one will function democratically; another will have a dominant pair lead the debate; and a third will have a coordinator who screens the contributions of the other members. There is so much variance that findings may not always be statistically significant, even in cases when there are spectacular achievements in one condition and disastrous failures in another.

Consider a quantitative output quality metric as an illustration of this. Assume for the moment that we will almost definitely need to utilise non-parametric tests and that we have discovered that every group under one circumstance scored higher than every group under the other. To get even 5% significance, we would want a minimum of four in each condition (one tailed). In the unlikely event that one of the typically better groups underperformed and our results were only marginally worse, we would need at least five participants in each condition.

Now, this example just took into account one scenario and made the best-case scenario assumption. Generally speaking, we would anticipate a wider variation across groups within conditions and we might wish to test multiple conditions concurrently. It will take a sharp rise in our ten groups before we can expect statistically significant outcomes. But as we've seen above, there are serious issues even with getting 10 experimental groups together.

This problem has three potential answers. Initially, within-group experiments can be used, in which each group is subjected to a variety of situations. Naturally, there are the typical issues with this kind of analysis—transfer effects and the like—but there is also a greater possibility that the group effect will be eliminated. A microanalysis of characteristics such as the intervals between utterances is the second option. Since these measurements are more likely to fit a normal distribution, more potent parametric tests can be applied. They might also be more resilient to the significant social divides that exist between groups.

The third option is to do a more anecdotal study of the data, searching for noteworthy occurrences such as intriguing events or malfunctions. Of course, qualitative and quantitative methodologies can be applied to the analysis of experiments. In fact, any thorough anecdotal examination of the logs will reveal useful statistical analytical metrics. On the other hand, attempts at controlled experiments may not be successful and may essentially "waste" the groups used in the control if there is a restricted number of experimental groups. Selecting settings that are likely to yield intriguing results is necessary given the significant expenses associated with group-working studies, even in the event that statistical analysis is not feasible.

(c) Amity University Online

Notes

Field Studies with Groups

It goes without saying that there are issues when placing user groups in an experimental setting. We are essentially studying the process of group formation rather than that of a typical working group if the groups are randomly mixed. Removing individuals from their regular work setting can drastically change their working patterns, even in situations where an already-existing group is used. There might not be a "normal" workplace under a new system, therefore our only option is to create an artificial one. Even with a new system, though, we still have the option of creating a naturalistic environment or a "good" experiment. More qualitative sociological analysis is at tension with the traditions of experimental psychology.

One could argue that contextual analysis is the only way to study group work. Exiting the actual setting will change the study's focus and scope of work. Consequently, alternative methods from the social sciences—like ethnography—have gained popularity, especially when it comes to researching group interaction. In ethnography, interactions between individuals, their surroundings and one another are meticulously documented. The ethnographer makes an effort to maintain objectivity throughout the study process and does not force their point of view on what is seen. The experimental perspective, with its hypothesis testing, is substantially different from this.

3.1.4 Evaluation through Users: Observational Techniques and Query Techniques

Seeing users interact with a system is a common technique to learn about its real use. Typically, individuals are requested to accomplish a predefined set of tasks; but, if surveillance is taking place at their place of employment, they could be seen going about their regular business. The assessor observes and logs the users' behaviours (using many methods). Since simple observation does not often provide insight into users' decision-making processes or attitudes, it is rarely sufficient to assess how effectively the system fits users' expectations. As a result, users are requested to "think aloud" in order to clarify their actions. This section examines several methods for assessing systems through user behaviour observation.

Think Aloud and Cooperative Evaluation

Think aloud is a type of observation in which the subject is asked to explain what he is doing while it is being observed. He might explain what he thinks is happening, the reason for an action, or the goal of the activity.

The benefit of thinking aloud is its simplicity; it can offer helpful insight into interface issues and is relatively easy to implement (albeit it can be challenging to thoroughly analyse). It can also be utilised to watch real-time system usage. Throughout the design process, it can be evaluated on paper or, in the early stages, with simulated mock-ups. But depending on the tasks given, the information could be selective and frequently subjective. Observation can cause people to do activities differently, leading to a biassed perspective. Like the tale about the centipede who was asked how he walked, the act of articulating what you are doing frequently modifies the way you do it.

Cooperative evaluation [240] is a variant on think aloud where the user is urged to view himself as a collaborator in the evaluation rather than just as a participant in the experiment. The evaluator can ask the user questions (usually of the "why?" or "what-if?" type) if his behaviour is unclear, in addition to asking him to think aloud at the start of the session. If a difficulty develops, the user can also ask the evaluator for explanation. There are several benefits to this looser interpretation of the think aloud process:

© Amity University Online

Notes

- The evaluator can maximise the usefulness of the approach for detecting issue areas by clarifying points of uncertainty at the time they arise,
- As the procedure is less limited and hence easier to learn to utilise.
- The user is encouraged to criticise the system.

The efficacy of the recording technique and the analysis that follows determines the general utility of think aloud, joint evaluation and observation. A protocol is the documentation of an assessment session of this kind and there are several approaches to pick from.

Protocol Analysis

The following are some techniques for capturing user actions on record:

Paper and Pencil

This is rudimentary but inexpensive and it enables the analyst to record interpretations and unrelated occurrences as they happen. However, because of the analyst's writing speed limitations, obtaining detailed information is challenging. Although they can take some time to design, coding schemes for common behaviours that were created during preliminary investigations can significantly increase the rate of recording. Using a notebook computer for direct entry is an alternative to using paper and pencil, however it limits the analyst to their typing speed and eliminates the flexibility of paper for writing styles, rapid illustrations and spatial layout. It is advised to designate a specific note-taker, independent of the assessor, if this is the only recording device available.

Audio Recording

If the user is actively "thinking aloud," then this is helpful. But it might be challenging to capture enough detail to pinpoint precise activities in a later analysis and it can be challenging to align an audio recording with another type of procedure (such a handwritten script).

Video Recording

The benefit of this is that, if the subject stays in the camera's field of view, we can observe what the subject is doing. It can be challenging to select the best camera placements and viewing angles to capture enough detail while maintaining the participant's field of vision. As an alternative, one must request that the person remain motionless, which might not be suitable for researching typical behaviour! Two video cameras are usually used for single-user computer tasks: one to focus on the computer screen and the other to take in the user's hands and face. If the computer system is being logged, the previous camera might not be required.

Computer Logging

Setting up a system that automatically log keystrokes made by users is not too difficult, especially if this feature was taken into account from the beginning of the design process. When source code is unavailable for proprietary software, it can be more challenging (though some programs now come with built-in playback and logging features). Of course, computer logging simply provides information on what users are doing on the system, but this might be enough in some situations. Additionally, keystroke data are "semantics free" since they simply describe the lowest-level activities; they do not explain the rationale behind them or their structure, albeit minute pauses and gaps can provide information. The benefits of direct logging include low cost (apart from disc storage requirements), minimal interference and application in longitudinal research, which monitors one or more users

(c) Amity University Online

over weeks or months. Its technical issues include the fact that, in many cases, one must take precautions to restore the system's state (file contents, etc.) before replaying the logs and that the sheer volume of data can become overwhelming without automatic analysis.

User Notebooks

It may be possible to ask the participants to keep activity/problem logs. Evidently, this will be done at a very basic level, with records occurring, at most, every few minutes and, more frequently, hourly or less. Additionally, it provides us with "interpreted" records, which have benefits and drawbacks. The method is particularly helpful when conducting longitudinal research and when keeping a record of uncommon or occasional chores and issues.

In actuality, one employs a variety of recording techniques since they are complementary. Even in cases where we have more advanced audio/visual recording, we might yet maintain a written record of noteworthy occasions and situations. Similar to this, even in situations when a video recorder is being utilised, we may still employ separate audio recording because the majority of built-in video microphones are not as good as specialised audio recording. Furthermore, stereo audio recording can be utilised to identify noises that are not shown on screen. When utilising many sources, like as audio, video (×2) and keystroke recording, it can be challenging to synchronise them while playing back. An on-screen clock is something that most video recorders can overlay, which is helpful, but the best solution is to utilisespecialised equipment that can automatically synchronise the various sources, perhaps combining many video displays onto one screen. Regretfully, specialised laboratories are frequently the only places where one can find this kind of equipment.

Transcription is a big issue when recording audio or video. Taped dictation is not the same as typing a transcript from a recording. Usually, there will be muttered words, inarticulate noises and partial or broken sentences in the discourse. Furthermore, it will be necessary to annotate the transcript with the many voices (which might only be evident from context) and non-verbal cues like pauses, emphasis, equipment noises, phone ringing, etc. Correcting grammatically incorrect sentences and finishing mispronounced words are common tasks for an experienced audio-typist; yet, it could be challenging to type exactly what is said. According to some practitioners, using typists is still not a good idea because the analyst would miss a lot of subtleties that are not present in the written transcript. But if you want to type your own transcripts from tape, you should definitely take a touch-typing course.

Professional typists are not an option for video transcription; instead, the analyst must create custom notations to fit the specific needs of the situation because there is no set method for annotating video recordings. It is important to recognise the magnitude of this undertaking. Speaking with practitioners who have tens or hundreds of hours of footage recorded but have only thoroughly examined small portions is common. Naturally, the pieces were selected after a more thorough examination of the content, but any notion of thorough coverage is definitely eliminated.

Coding can be used to identify specific occurrences, but choosing an appropriate coding scheme and applying it consistently can be challenging, especially when multiple people are coding at the same time. Therefore, coding schemes should be tested by a variety of transcribers to be sure that they are being understood correctly for a given data set.

Automatic Protocol Analysis Tools

Protocol analysis, be it audio, video, or system logs, is laborious and time-consuming when done by hand. If there are multiple streams of data to synchronise, it becomes more difficult. Providing automatic analytic tools to assist with the process is one way to solve this issue. These provide a way to synchronise, modify and annotate system, audio and video logs for in-depth study.

(c) Amity University Online

Notes

A multimedia workstation with a direct connection to a video recorder hosts the EVA (Experimental Video Annotator) system [220]. The assessor may design a series of buttons denoting various actions. These could contain snapshots and timestamps, along with notes on mistakes and anticipated events. The evaluator uses the buttons to add notes to the video during a recording session. The user works at a workstation while being recorded, maybe with the use of audio, video and system logging. The multimedia workstation running EVA is used by the evaluator. A view of the user's screen and the live video record are displayed on the screen (see Figure below). The evaluator can take further notes in a text editor and utilise the buttons to mark noteworthy events as they happen. Following the session, the evaluator has the option to examine the tagged segments and search the data using these and the conventional video controls. It is possible to create links with other kinds of records, like audio files and system logs. Although a system like EVA lessens the workload associated with video analysis, it is not without issues. The evaluator may find it difficult to focus on the events themselves while labelling and annotating them. This can imply that occurrences are overlooked or labelled improperly.

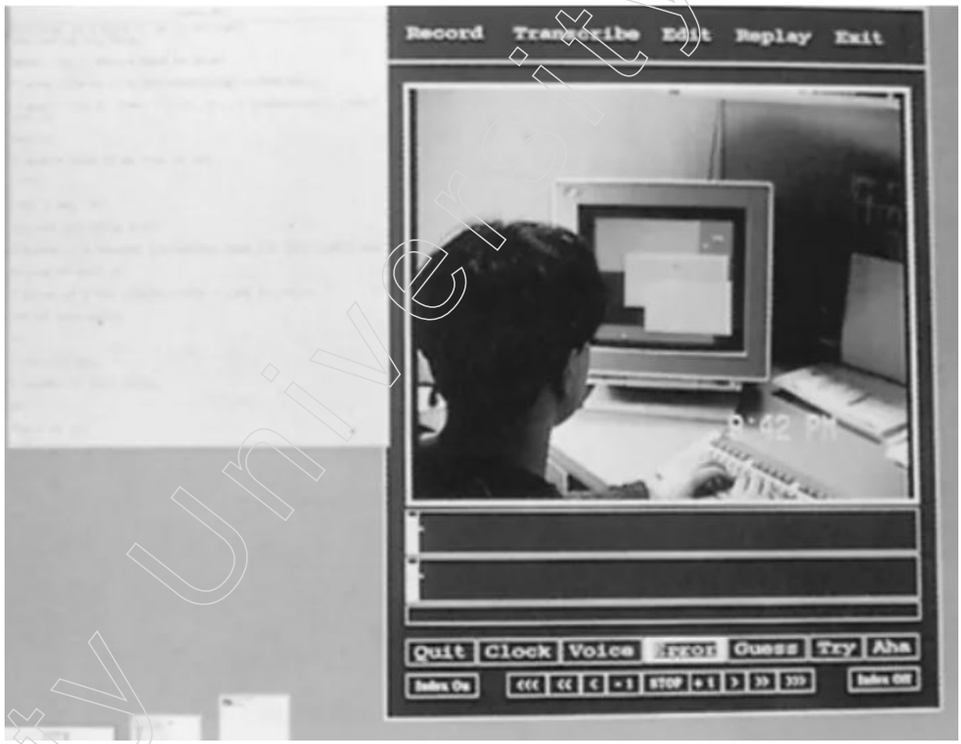


Figure 58: EVA: an automatic protocol analysis tool. Source: Wendy Mackay

Comparable capability to EVA may be found in commercial systems like Noldus' Observer Pro (www.noldus.com); portable versions are now available for use in field research.

Protocol analysis assistance is provided by a system included in the Workplace project at Xerox PARC [348]. Supporting the analysis of synchronised data from several data sources, including video, audio, notes and diagrams, is the primary focus here. The data streams are shown in an aligned manner to facilitate the comparison of each record at a specific interaction point. The alignment is accomplished by the use of hypertext connections and might be based on timestamps, events, or actions.

A third illustration is DRUM [223], which offers features for video tagging and commentary as well. The MUSiC (Measuring the Usability of Systems in Context/Metrics for Usability Standards in Computing) toolkit includes DRUM. This toolkit supports a

(c) Amity University Online

Notes

comprehensive evaluation methodology based on the application of usability metrics on analytic metrics, performance, user satisfaction and cognitive workload. DRUM is especially interested in performance measurement. In addition to DRUM, the technique offers a number of tools, such as databases, analysis software, surveys and manuals.

These kinds of systems are crucial as assessment instruments because they provide a way to manage the data gathered from observational studies and enable a more methodical approach to the analysis. This makes the evaluator's job easier and more insightful insights should come to light as a result.

Post-task Walkthroughs

Direct observation data frequently lack interpretation. We know the fundamental things that were done, but not much about why. The information might be inappropriately levelled, even in cases when the participant has been asked to think aloud during the exercise. The participant might respond, "and now I'm selecting the undo menu," for instance, but they wouldn't explain what went wrong for undo to be required. Furthermore, information on potential courses of action that are not taken is not included in a think aloud.

By reflecting back to the participants what they did during the event, a walkthrough aims to mitigate these issues. The participant hears the written or recorded transcript again and is given the option to respond or ask questions directly of the analyst. This can be done immediately, when the subject might genuinely recall the rationale behind specific behaviours, or later, when the responses are more likely to represent the subject's post hoc interpretation. (Indeed, interpretation is probably possible even in the first scenario.) A postponed walkthrough has the benefit of giving the analyst more time to formulate pertinent queries and concentrate on particular instances. The drawback is that the freshness is gone.

In certain situations, such as when the activity is excessively demanding or during a crucial task, it is not reasonable to expect the participant to speak throughout the actual observation. Under these conditions, obtaining a subjective perspective on the user's behaviour can only be achieved through the post-task walkthrough. There is also the idea that in order to obtain the most natural performance possible during direct observation, it is best to minimise non-task-related conversation. That's why the walkthrough is crucial once more.

Query Techniques

Asking the user directly about the interface is the basis of another set of evaluation methodologies. When obtaining specifics about a user's perspective of a system, query techniques can be helpful. They represent the idea that "asking the user" is the best way to learn how a technology satisfies user requirements. They can be employed more broadly to gather data regarding user tasks and requirements as well as in evaluation. The benefit of these approaches is that they directly obtain the user's perspective and could highlight problems that the designer had overlooked. They are also rather easy and inexpensive to administer. Nevertheless, the knowledge acquired is inevitably subjective and can represent a "rationalised" version of what happened rather than an entirely true one. Furthermore, if the user has not used the other designs, it could be challenging to obtain reliable input, which would restrict the amount of information that can be learned. Nonetheless, the techniques offer valuable adjunct information to other techniques. Questionnaires and interviews are the two primary categories of query techniques.

Interviews

One straightforward and organised method of obtaining information is through conducting user interviews regarding their interactions with interactive systems. The

© Amity University Online

Notes

flexibility to adjust the degree of inquiry to the situation and the evaluator's ability to go deeper into intriguing topics as they come up are two benefits of conducting interviews. often, an interview proceeds in a top-down manner, beginning with a broad inquiry concerning the work at hand and moving on to more probing inquiries (often in the form of "why?" or "what if?") to elucidate specifics of the user's response.

For high-level evaluation, interviews can be useful, especially when gathering data on user preferences, perceptions and attitudes. They might also highlight issues that the designer had not foreseen or that had not been noticed. They are a helpful tool for providing context for an event when used in conjunction with observation (see the post-task walkthrough).

To optimise effectiveness, the interview ought to be prearranged, with a predetermined list of key questions ready. These questions then provide the framework for each interview. This aids in narrowing down the interview's goal, which could be, for example, to delve deeper into a specific area of the exchange. It also aids in maintaining a foundation of consistency among user interviews. Having said that, the interview is not meant to be a controlled experimental approach, thus the evaluator is free to customise the form for each user in order to maximise the benefits.

Questionnaires

Questioning the user using a questionnaire is an additional approach. Compared to an interview, this method is obviously less flexible because the questions are predetermined and probably won't be as penetrating. On the other hand, it can be used to examine a larger sample size, requires less time to administer and allows for more thorough analysis. It can also be used to gather data about user requirements, preferences and experience at several stages of the design process, such as requirements gathering, task analysis and evaluation.

It is essential that the questionnaire be adequately constructed because the evaluator is unlikely to be actively involved in its completion. The assessor must first determine the goal of the questionnaire: what data is being sought? Making a decision about the analysis of the questionnaire replies at this point is also helpful. For instance, what precise, quantifiable input are you looking for on certain parts of the interface, or what impression does the user have of the interface?

There exist several question styles that can be used into the survey. Among them are the following:

General

These are the kinds of queries that aid in determining the user's history and position among the user base. They ask about things like location of residence, employment, sex, age and so forth. Along with that, they might ask about prior computer experience in the form of open-ended, multiple-choice, or scaled questions.

Open-ended

These ask the user to answer a question on their own, without prompting them, such as "Can you suggest any improvements to the interface?" They can only be seen as supplemental since they are hard to compare or analyse rigorously; instead, they are good for obtaining broad, subjective information. Additionally, responders who are pressed for time are inclined to overlook them! They might, however, point out mistakes or offer recommendations that the designer had not thought about. Asking the user for real information, such how many commands were performed, is a specific case of this kind.

(c) Amity University Online

Notes

Scalar

These gauge the user's level of agreement or disagreement with a given statement by asking them to rate it on a numerical scale. As an illustration,

Making amends for errors is not difficult.

Disagree 12345 Accept

The scale's granularity varies; a coarse scale (let's say ranging from 1 to 3) makes it obvious what each number means (disagree, neutral and agree). It does not, however, allow for different degrees of agreement and as a result, users can be inclined to respond neutrally to things with which they are mildly disagreed or agreed. The reverse issue arises with a very fine scale (let's say 1 to 10): it becomes harder to consistently interpret the numbers. It is inevitable that various users may perceive the scale in different ways. Thus, finding a middle ground is advised. One to five or one to seven scales have been applied with success. They are sufficiently fine-grained to enable consumers to distinguish between them while maintaining semantic clarity. It can be useful in illuminating the significance of intermediate scalar values. The majority of the time, odd-numbered scales are utilised, although if the "neutral" option is not desired, even-numbered scales (such as 1–6) can be utilised. This prevents fence sitting (unless one chooses 31 /2!), in which case it is allowed.

Multi-choice

The respondent is presented with a list of explicit options in this instance and they can be asked to choose as many or as few as suit. As an illustration,

How do you most often get help with the system (tick one)?
- Online manual ❑
- Contextual help system ❑
- Command prompt ❑
- Ask a colleague ❑

Which types of software have you used (tick all that apply)?
- Word processor ❑
- Database ❑
- Spreadsheet ❑
- Expert system ❑
- Online help system ❑
- Compiler ❑

Figure 59: Illustration

These are very helpful for learning about a user's prior experiences. This kind of specific instance occurs when there are just two options: "yes" or "no."

Ranked

These organise the things in a list and serve as a helpful way for users to express their preferences. As an illustration,

Please rank the usefulness of these methods of issuing a command (1 most useful, 2 next, 0 if not used).
- Menu selection ❑
- Command line ❑
- Control key accelerator ❑

Figure 60: Ranking

As we've mentioned, there are several uses for each of these question kinds. But, it is preferable to employ closed questions, such scalar, ranked, or multi-choice, as much

(c) Amity University Online

Notes

as possible to lessen the respondent's effort and so promote a high response rate among users. These lessen the work needed by giving the user options for responses. They also have the benefit of being simpler to dissect. Numerous techniques are available for analysing responses, ranging from factor analysis and correlation analysis to as simple as calculating percentages for each response.

Conducting a pilot research is advisable, regardless of the kind of questionnaire that is being prepared. This makes it possible to address any design issues with the questionnaire before it is sent to a possible hundreds of users! To ensure that the questions are understandable, the findings match expectations and the questionnaire can be utilised as intended, it should be tested on four or five users. Before the final version is distributed, a question can be rephrased (and retested) if users appear to be misinterpreting it.

Questionnaire distribution may potentially present issues. While it's crucial that the respondents reflect the user base as a whole, you also need to make sure that you can connect with as many available responders. Since the average questionnaire return rate is just 25% to 30%, a large number of them must be sent out in order to receive a respectable return. Ideally, a random portion of the user base should receive a questionnaire. Therefore, if the population includes all employees of a corporation, for instance, one could decide to send a questionnaire to every fourth individual on a personnel list that is arranged alphabetically. But these days, most questionnaires are delivered online. They can be sent by email, where possible responders can be chosen at random, or they can be sent via a website, where respondents can only be those who view the page, which may not be a representative sample. Since only those who choose to participate in the study are included, questionnaire respondents are actually self-selecting in any case. If the questionnaire is made to collect demographic data about each participant, it will be possible to infer from the responses whether or not the sample is representative.

3.1.5 Choosing an Evaluation Method

An interactive system can be evaluated using a variety of methods at any point throughout the design phase. How then do we determine which approaches are best suited to our requirements? There are no hard-and-fast rules here; each strategy has unique advantages and disadvantages that can be overcome with the right application. When choosing assessment methods, there are several considerations that must be made. In order for us to compare and select between the many approaches, these also offer a means of classifying them. Let's talk about these elements.

Factors Distinguishing Evaluation Techniques

We are able to discern a minimum of eight elements that differentiate various assessment methods, hence facilitating our decision-making process. These are the following:

- the stage in the cycle at which the evaluation is carried out
- the style of evaluation
- the level of subjectivity or objectivity of the technique
- the type of measures provided
- the information provided
- the immediacy of the response
- the level of interference implied
- the resources required.

Notes

Design vs. Implementation

The stage of the design process at which evaluation is necessary is the first factor that influences our choice of evaluation method. As was previously mentioned, it is ideal to incorporate some form of evaluation all the way through the design process. Evaluation of an implementation differs primarily from evaluation of a design in that the latter involves the assessment of a physical artefact. This needs to be something tangible that can be tested; it could be as simple as a paper mock-up or as complex as a full implementation. On the other hand, the evaluation of a design comes before this phase and aims to supply data that will support the creation of the physical artefact.

In general, assessment during the design phase should be quick, low cost and analytical; it may involve just design professionals. In contrast, evaluation at the implementation stage should be more thorough and involve users. Of course, there are certain exceptions to this: Through participatory design, users are involved at every stage of the process. Expert-based and analytical methods, including cognitive walkthrough, can be used to assess designs as well as implementations.

The best results will come from early review, whether it be of a design, early prototype, or mock-up, since issues may be readily fixed at this point. Changes are harder to come by the more a specific design is committed to in its implementation, regardless of the evaluation's recommendations. Ironically, late reviews frequently receive the greatest attention and resources. This is less profitable and ought to be avoided, even if it is plain that some user evaluation is necessary when the system is complete or nearly complete because some aspects (such system performance) will only be noticeable when the system is operational.

Laboratory vs. Field Studies

While retaining part of the naturalness of the user's surroundings, regulated testing and observation are made possible by laboratory investigations. While field studies do not permit control over user behaviour, they do preserve the latter. Both forms of evaluation should ideally be incorporated into the design process, most likely with field studies carried out after the new implementation and laboratory research dominated in the early phases.

Subjective vs. Objective

The objectivity of evaluation methodologies varies as well; while some mostly depend on the evaluator's interpretation, others offer comparable information to anyone doing the operation correctly. The more subjective methods, including think aloud and cognitive walkthrough, mostly depend on the evaluator's skills and knowledge to identify issues and comprehend the user's actions. When utilised properly, they can be quite effective and yield information that may not be accessible through more impartial means. However, it's important to acknowledge and steer clear of the issue of evaluator bias. Using many evaluators is one technique to reduce the chance of bias. Conversely, objective methods ought to yield dependable outcomes that are independent of the specific assessor's influence. One type of objective measure is a controlled experiment. These are free from prejudice and offer equivalent outcomes, but they might not identify the unforeseen issue or offer in-depth user experience feedback. It is ideal to employ both subjective and objective methods.

There are variations in the degree to which the outcomes rely on the user's subjective response. Requesting the user's thoughts, which is very subjective, is at one extreme; measuring physiological changes in the body, which are not under the user's control, is at the other.

© Amity University Online

Notes

Qualitative vs. Quantitative Measures

The kind of measurement that the assessment method offers is also crucial to take into account. The two primary categories of measuring are qualitative and quantitative. The former is typically numerical and therefore amenable to statistical analysis. The latter can offer crucial information that cannot be ascertained from statistics because it is non-numeric and hence more challenging to evaluate. The subjectivity or objectivity of the approach affects the sort of measure; subjective procedures typically yield qualitative measures, whereas objective techniques yield quantitative measures. But this isn't a strict guideline. Sometimes, by placing data into a scale or other comparable measure, it is feasible to quantify data that is actually qualitative in nature. One typical instance of this is found in surveys that employ a quantitative scale to collect qualitative data (like user preferences). This is also typical in experimental design, when dependent variables are employed and quantified, such as the user's performance quality.

Information Provided

There may be variations in the amount of information needed for an evaluation. At any point during the design process, an evaluator may need information that ranges from low-level details (like the best legible font) to higher-level details (like "Is the system usable?"). Certain assessment methods, like controlled experiments, are great at supplying detailed information since they can be set up to measure specific aspects of the interface. Questionnaire and interview procedures are useful for gathering higher-level information as they offer a more comprehensive picture of the user's perception of the system.

Immediacy of Response

The promptness with which evaluation approaches respond is another characteristic that sets them apart. Certain techniques, such think aloud, capture the user's actions during the actual engagement. Some, like the post-task walkthrough, depend on the user's memory of what happened. Users may perceive events based on preconceived notions, leading to bias in memory and reconstruction in such recollections. Also possible is incomplete recall. However, since the assessment process might really change how the user works, instantaneous techniques can also provide challenges.

Intrusiveness

The intrusiveness of the technique itself is related to the promptness of the reaction. Some methods, especially those that yield instantaneous measurements, are visible to the user throughout the interaction and may therefore impact the user's behaviour. The evaluator's sensitive actions can lessen this, but they cannot completely eliminate it. Aside from automatic system logging, most quick evaluation techniques are obtrusive. Sadly, this can only offer a limited amount of information and is challenging to understand.

Resources

The availability of resources is the last factor to take into account while choosing an evaluation technique. Equipment, time, money, participation, the evaluator's experience and context are among the resources to take into account. Resource constraints compel some decisions, such as the necessity of having access to a video camera in order to develop a video protocol (and likely editing facilities as well). Other choices, though, are less cut-and-dry. For instance, there can be financial or time constraints that require choosing between two alternative assessments. In these situations, the evaluator has to determine which assessment strategy will yield the most insightful and practical data for the system in question. It could be feasible to avoid conducting new tests by using the findings of previous research.

(c) Amity University Online

Notes

Certain methods, like the more formal analytical methods, depend more on the competence of the evaluator than do others. It might be more sensible to employ more straightforward heuristic techniques rather to ones that call on comprehension of user goal structures and other details if the evaluator's experience is limited.

Lastly, what can be done will depend on the setting in which evaluation can take place. It might not be feasible to access a system's intended users for practical reasons (if it is a generic system, for example). Alternatively, testing the system in its intended context (a space station or defensive system, for instance) might not be possible. It is necessary to use simulations in these situations.

A Classification of Evaluation Techniques

Classifying the evaluation procedures we have explored can be done by using the factors that were previously addressed. This enables us to determine which methods most closely match our needs.

The classification of analytical techniques is displayed in the table below.

Table 9: Classification of analytic evaluation techniques

	Cognitive walkthrough	Heuristic evaluation	Review based	Model based
Stage	Throughout	Throughout	Design	Design
Style	Laboratory	Laboratory	Laboratory	Laboratory
Objective?	No	No	As source	No
Measure	Qualitative	Qualitative	As source	Qualitative
Information	Low level	High level	As source	Low level
Immediacy	N/A	N/A	As source	N/A
Intrusive?	No	No	No	No
Time	Medium	Low	Low–medium	Medium
Equipment	Low	Low	Low	Low
Expertise	High	Medium	Low	High

The classification of experimental and query evaluation methodologies is displayed in the table below.

Table 10: Classification of experimental and query evaluation techniques

	Experiment	Interviews	Questionnaire
Stage	Throughout	Throughout	Throughout
Style	Laboratory	Lab/field	Lab/field
Objective?	Yes	No	No
Measure	Quantitative	Qualitative/ quantitative	Qualitative/ quantitative
Information	Low/high level	High level	High level
Immediacy	Yes	No	No
Intrusive?	Yes	No	No
Time	High	Low	Low
Equipment	Medium	Low	Low
Expertise	Medium	Low	Low

© Amity University Online

Notes

The classification of observational evaluation approaches is displayed in the table below.

Table 11: Classification of observational evaluation techniques

	Think aloud[1]	Protocol analysis[2]	Post-task walkthrough
Stage	Implementation	Implementation	Implementation
Style	Lab/field	Lab/field	Lab/field
Objective?	No	No	No
Measure	Qualitative	Qualitative	Qualitative
Information	High/low level	High/low level	High/low level
Immediacy	Yes	Yes	No
Intrusive?	Yes	Yes[3]	No
Time	High	High	Medium
Equipment	Low	High	Low
Expertise	Medium	High	Medium

- Assuming a simple paper and pencil record
- Including video, audio and system recording
- Except system logs

The classification of monitoring assessment methodologies is displayed in the table below.

Table 12: Classification of monitoring evaluation techniques

	Eye tracking	Physiological measurement
Stage	Implementation	Implementation
Style	Lab	Lab
Objective?	Yes	Yes
Measure	Quantitative	Quantitative
Information	Low level	Low level
Immediacy	Yes	Yes
Intrusive?	No[1]	Yes
Time	Medium/high	Medium/high
Equipment	High	High
Expertise	High	High

The classification is meant to be used simply as a general reference; some techniques do not easily fall into this category due to the wide range of applications for which they can be used.

3.2 Design principles and Visuals

One of the basic components of design is shape. They can be applied to express meaning and generate textures and patterns. Shapes come in a variety of sizes and proportions and can be either geometric or organic. They aid in content organisation, draw attention to key details and add visual appeal.

The art of typography involves placing typefaces in a way that makes written language

(c) Amity University Online

Notes

understandable, readable and visually appealing. It entails selecting typefaces and figuring up their leading (the distance between lines), kerning (the space between letters), tracking (the space between groups of characters)and size, colour and line spacing. In order to create a message that is clear and easy to read while maintaining the desired tone, typography is essential.

Another crucial component of design is colour, which creates feelings and establishes the tone of a picture or graphic. By giving a business a unique personality through the use of logos and other marketing materials, it aids in branding. Additionally, colour creates visual hierarchy by using contrast to draw the viewer's attention to the key elements.

The arrangement of text, graphics and shapes on a page or screen is referred to as layout. A well-designed layout should be both aesthetically beautiful and practical, allowing users to engage with content without being overloaded or disoriented.

3.2.1 Color Theory, Typography and Imagery

Typography is regarded as one of the most important components in graphic design and visual communication. It describes the skill and method of setting type in a way that makes printed words readable, aesthetically pleasing and legible. Typography is a crucial skill for designers to learn because it greatly influences how people view material.

Font size, line spacing, kerning, tracking, leading, alignment and hierarchy are the primary typographic components. Font size is important because it affects how visible and readable the text is; readers have to work harder to read smaller fonts, while larger fonts are simpler to read. Since line spacing helps keep lines from mixing or overlapping, it also has an impact on readability.

Tracking modifies space consistently throughout entire text blocks, whereas kerneling modifies the space between individual letters in a word. While alignment defines whether the text is centred, left-aligned (flushed left), right-aligned (flushed right), or right-aligned (flushed right), leading measures the vertical space between lines of type in a paragraph.

When typesetting elements like headers with larger sizes than body material are arranged, hierarchy plays a part. Together, these typographic components provide the best readability and presentation possible, successfully grabbing the attention of the user.

Since colour theory deals with using colour to elicit feelings and convey messages, it is essential to graphic design. Designers of visual communication must comprehend the connections between colours, their connotations and how people perceive them. Colour theory is crucial for designers to understand since colour combinations have the power to make or ruin a design.

Knowing the colour wheel is a key component of colour theory in graphic design. All other colours are derived from the main colors—red, blue and yellow—whereas secondary colors—green, purple and orange—are produced by combining two primary colours. Furthermore, combining a primary and a secondary colour creates tertiary colours.

Hue, saturation and brightness are three more key concepts in colour theory that graphic designers need to understand. Hue is the predominant wavelength that gives rise to a particular colour; saturation is the degree of intensity or purity of a given hue; brightness is the degree of lightness or darkness of a shade. When choosing appropriate colour schemes for various projects, designers need to be well-versed in these ideas. This is especially true when choosing colour schemes for events like weddings or parties, when clients may have specific colour schemes in mind.

(c) Amity University Online

Notes

Techniques for Visual Communication in Graphic Design

Visual communication is the foundation of every successful project in graphic design. In graphic design, visual communication techniques include using typography, images and other elements to communicate ideas or messages to a target audience. These techniques are necessary to produce eye-catching designs that can draw in and retain viewers.

Colour theory is a popular visual communication technique in graphic design. This is realising how certain hues combine to evoke particular feelings or moods. Composition, or the arrangement of pieces within a design area, is another technique. Because typography uses font sizes and styles to assist express meaning, it is also a crucial visual communication technique in graphic design.

All things considered, graphic designers may produce more powerful designs that successfully convey their intended message to their target audience by learning about these visual communication techniques.

Typography

Typography is the art of organising text and letters so that the reader can easily read, understand and find the content visually pleasing. Typography is the study of typeface appearance, style and structure with the goal of expressing particular feelings and ideas. To put it succinctly, typography is what gives the writing life.

The invention of moveable type in the eleventh century is when typography first emerged. Typography was a specialised craft used in books, publications and finally public works before the advent of the digital age. The Gutenberg Bible contains the earliest known example of type, which ignited a revolution in printing throughout the West. Fun fact: the typeface that was used in the Gutenberg Bible is now called Textura and most modern desktop apps have it available in the font drop-down menu.

What makes typography significant?

Strong visual hierarchy, graphic balance and the overall tone of the product are all established by well-designed typography. Typography should maximise readability and accessibility, provide a great user experience and direct and enlighten your users. Effective typography not only adds individuality to your website, but it will also cause consumers to subconsciously start associating your business with the typeface you select.

Imagery

A key component of design, imagery uses the visual language to express concepts, arouse feelings and improve user experiences in general. Effective use of images is essential for communicating ideas in print, internet and branding contexts.

in design, imagery is a potent tool that can improve user engagement, elicit strong feelings and facilitate effective communication. To produce powerful visual storytelling, designers need to make use of the criteria of relevance, quality and consistency. Responsible usage of imagery is ensured by ethical concerns, which improve user experiences and brand perception. The use of imagery in design is changing as technology does, opening up new avenues for communication and creative expression.

Types of Imagery:

a. Photographs:Photographic representations of the real world evoke realism and frequently provide a sense of intimacy.

b. Illustrations:Visual communication gains a creative and stylistic touch from artistic drawings or graphical representations.

(c) Amity University Online

Notes

c. Icons:Particularly in user interfaces, simplified and symbolic representations are employed for rapid recognition.

d. Infographics:Information can be made easier to understand and more palatable by using graphic visual representations of facts.

e. Symbols and Logos:Distinct symbols and logos serve as visual identifiers for brands, products, or concepts.

Functions of Imagery:

a. Storytelling: A captivating story can be told with the help of imagery, which takes the audience on a visual tour.

b. Emotion Elicitation:Emotionally charged images can impact viewers' perceptions and interactions with the material.

c. Information Reinforcement:Textual information is strengthened by imagery, which increases its memorability and comprehension.

d. Branding:Brand identification is enhanced by the consistent use of imagery, which develops a unified visual language.

e. User Engagement: In a variety of situations, visually appealing graphics draw viewers in and improve user engagement.

3.2.2 Iconography

The use of symbols or images to express ideas, indicate concepts, or initiate activities is known as iconography. Icons are essential to design because they improve both visual communication and user experience. Let's examine the importance, tenets and uses of iconography in several design contexts:

1. Significance of Iconography:
   a. Universal Language: Icons function as a universal language that is comprehended by a wide range of audiences, overcoming linguistic barriers.
   b. Visual Efficiency: Icons are a great choice for interfaces with limited space because they effectively and rapidly transmit information.
   c. Wayfinding and Navigation: Icons make complicated information structures easier to understand and serve as visual indicators for navigation.
   d. Branding and Identity: Distinctive symbols help build a unified visual identity and aid in brand identification.
2. Principles of Iconography:
   a. Simplicity: To guarantee immediate comprehension, make icons easily recognisable and uncomplicated.
   b. Consistency: To create a cohesive visual language, keep all of your icons using the same design and style.
   c. Clarity: To prevent misunderstandings, make sure that icons accurately convey the ideas intended.
   d. Relevance: Icons have to make sense in the context and match the related ideas or actions.
   e. Scalability: When creating icons, especially for responsive and flexible designs, make sure they are scalable without sacrificing clarity.
3. Types of Icons:

© Amity University Online

Notes

a. Pictorial Icons: Reflect the actual physical form of the idea or object being represented.

b. Symbolic Icons: Often used to symbolise concepts or actions, abstract symbols can be used to convey meanings.

c. Action Icons: Show particular activities, like searching with a magnifying glass or deleting something with a trash can.

d. Notification Icons: Use a bell-like icon to notify users of particular occurrences or information.

3.2.3 Designing for Different Screens

Users access websites and applications on a variety of devices, each with a different screen size and quality, in today's digital world. For designers, creating a smooth and aesthetically pleasing user experience throughout this range offers both a problem and an opportunity.

It is vital to have interface optimisation and adaptability skills for various screen sizes in order to guarantee user delight, accessibility and usability. The methods and best practices for designing for various screen sizes and resolutions will be discussed..

Designing for various screen sizes and resolutions is a complex process that calls for careful testing, technological know-how and a user-centered approach. Key techniques for developing flexible interfaces include adopting responsive design, giving mobile-first principles top priority and optimising media and graphics.

Through the implementation of consistent testing across various devices, the establishment of breakpoints, the maintenance of design consistency and the consideration of touch interactions, designers are able to produce seamless user experiences across the wide range of digital devices. In the end, designing for varied displays is about making sure that users can engage with interfaces and access content in a natural way on any device, which promotes a welcoming and good user experience.

1. Adopt responsive design: Designing for a variety of screens starts with responsive design. It entails designing fluid elements and adaptable layouts that change automatically according to the user's device. Designers may make sure that the interface adjusts to different screen sizes and orientations without any problems by utilising media queries, flexible images and responsive grids.

2. Give Mobile-First Design top priority: It makes sense to design with mobile devices in mind, as they are becoming more and more common. Design for the smallest screens first, then gradually improve the experience for larger ones. This methodology fosters an emphasis on crucial content and functionality, resulting in a design that is both efficient and user-focused.

3. Employ Flowing Fonts: The use of typography is essential for both readability and aesthetic appeal. Use scalable, flowing font that adjusts to the size of the screen. This guarantees that, whatever of the device's size, the text will always be readable and will balance well with other design elements.

4. Make Media and Image Optimisation: Media files and images ought to be adjusted for varying connection speeds and resolutions. Make use of responsive images, which adjust to the screen size of the user's device automatically. To balance performance and image quality, think about utilising contemporary image formats like WebP.

5. Examine Different Devices: It is essential to test your design on a variety of devices on a regular basis. Different screen sizes can be simulated by emulators and testing tools,

(c) Amity University Online

Notes

but actual testing on physical devices yields more realistic results. You can find and fix any layout, functionality, or performance problems using testing.

6. Breaking Points for Adapting Designs: Breakpoints are precise screen widths where the layout of the design changes to fit various devices. A seamless transition and the best possible user experience should be ensured by carefully selecting breakpoints based on typical device sizes and user behaviour.
7. Steady User Experience: A unified user experience depends on consistency. It is important for elements like buttons, navigation menus and branding to hold true on various screen sizes. Maintaining consistency helps users navigate easily and strengthens brand identity.
8. Adaptable Navigation Design: User interaction depends on navigation. Use off-canvas designs, icon-based navigation, or collapsible menus for smaller screens. Give priority to necessary navigation choices and make sure that important content is easily accessible to consumers.
9. Touch and Gesture Interaction: Create interactions for touch-enabled devices based on touch gestures. When creating interactive features, take swipe, pinch and tap gestures into account. Sufficient distance between touch points minimises unintentional touches and improves usability.
10. Compatibility with All Display Sizes: Accessibility considerations are also included while designing for various screen sizes. Make sure the interactive features are easy to navigate, the language is readable and the colours keep their contrast. The accessibility of a responsive design shouldn't be compromised for users with disabilities.

3.2.4 Responsive Web Design

These days, almost all prospective customers request a mobile version of their website. After all, having separate designs for the BlackBerry, iPhone, iPad, netbook and Kindle is practically necessary. All screen resolutions also need to work together. We will probably need to design for several more inventions in the next five years. When will this crazy end? Of course it won't.

We're rapidly reaching a stage in Web design and development where we can no longer keep up with the constant stream of new resolutions and devices. For many websites, it would be impractical, if not impossible, to create a version of the site for every resolution and new device.

The idea behind responsive web design is that, depending on the screen size, platform and orientation of the user, design and development should adapt to their behaviour and environment.

The technique uses a combination of pictures, adaptable grids and layouts and deft CSS media query application. The website ought to adapt immediately to the user's shift from a laptop to an iPad, taking into account differences in resolution, image size and scripting capabilities.

It could also be necessary to take into account the settings on one's devices; for instance, if an iPad is using a VPN for iOS, the website shouldn't prevent the user from accessing the page. Stated differently, the website ought to be equipped with technology that can react to the user's preferences on its own. This would do away with the requirement for a distinct design and development stage for every new device released onto the market.

If we apply this discipline to Web design, we get a whole new but comparable concept. Why should we make a unique Web design for every user group? After all, architects don't

© Amity University Online

Notes

make buildings that are suitable for every kind and size of group that uses them. Similar to responsive architecture, Web design need to adapt on its own. There shouldn't be an endless supply of unique solutions for every new user segment.

Naturally, we are unable to achieve this in the same way that a building would using robotics and motion sensors. Thinking more abstractly is necessary while designing responsive websites. Nonetheless, several concepts are already in use: media queries, fluid layouts and scripts that easily (or automatically) restructure HTML and markup.

However, responsive Web design is about a completely new way of thinking about design, not only about changing screen resolutions and automatically resizing pictures. Let's discuss each of these aspects as well as upcoming concepts.

Adjusting Screen Resolution

As the number of devices increases, so do the screen orientations, definitions and resolutions. Every day, new gadgets with larger screens are created and each of these gadgets might be able to accommodate changes in terms of usefulness, size and even colour. Some are in portrait orientation, some in landscape and some are even square in shape. Numerous modern devices have the ability to automatically flip between portrait and landscape orientations, as evidenced by the growing popularity of the iPhone, iPad and sophisticated smartphones. How can one create for these kinds of circumstances?

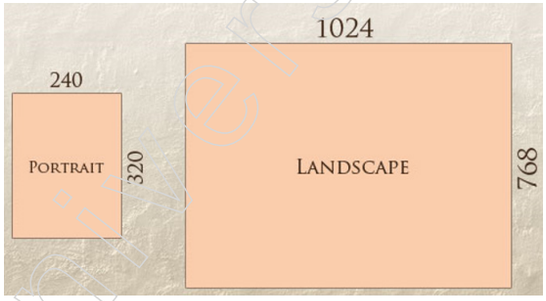


Figure 61: Portrait and Landscape modes

We need to take into account the hundreds of different screen sizes in addition to planning for both landscape and portrait orientations (and possibly enabling those orientations to transition in an instant upon page load). It is feasible to classify them into broad groups, create designs specifically for each and adjust the flexibility of each design as needed. However, that might be too much to handle and who can predict how much will be used in five years? Furthermore, a lot of people don't make the most of their browsers, which leaves way too much area for different screen sizes.

Only the layout columns (structural components) and the text were flexible in a design a few years ago, when flexible layouts were considered a "luxury" for websites. Layouts were easily broken by images and when pushed too far, even the form of a flexible structural element would break. Although they could accommodate a few hundred pixels more or less, flexible designs frequently couldn't transition from a huge computer screen to a netbook. We can now adjust the level of flexibility. We have solutions so that layouts never fail, even though images can be automatically resized (albeit they could get crushed and unreadable in the process). The solution gives us a lot more alternatives even though it's not a perfect fix. It works well with gadgets that can quickly flip from portrait to landscape mode and with consumers moving from a big computer screen to an iPad.

Why Responsive Web Design Is Important

Websites could only be viewed on desktop computers in the past. Most of them

(c) Amity University Online

Notes

had monitors of the same size. Websites were created with the typical user in mind. Expectations have evolved as a result of the wide range of devices that people use to view websites these days, with screens that can be as little as a few inches or as large as 27 inches or more. Customers anticipate that the website they're on will recognise that they are using a tablet instead of a PC.

Instead of the other way around, they anticipate the website adapting to them.There are also variations in the expectations for the usage of different devices. People have been "trained" to anticipate to be able to swipe left, right, up and down, for instance, by the Apple iPhone. When a smartphone user visits a website, they expect to be able to click on a phone number and have the option of having their phone call that number automatically. They also anticipate that an address will have a GPS-powered "click for directions" option on their phone. No matter what device a visitor uses to view the site, a responsive website considers all of this and instantly adjusts to give them the greatest possible user experience.

Summary

- Evaluation should occur at every stage of the design life cycle and is an essential component of the process. Its objectives are to test the design's usability and functionality and to find and fix any issues. Additionally, it may attempt to ascertain the user's mindset and reaction to the system. It may or may not require the user's active engagement and can occur in a specialised laboratory or at their place of employment.
- To reduce the expense of early design mistakes, a design can be assessed prior to the commencement of implementation activities. At this point, the majority of evaluation methods are analytical and entail having a specialist evaluate the design in accordance with usability and cognitive concepts. At this point, prior experimental findings and modelling techniques can also offer insight. Both quantitative and qualitative data can be obtained using experimental and observational techniques once an artefact (a prototype or a complete system) has been built.
- Techniques for asking questions elicit subjective information from the user. Physiological monitoring can record the user's physical reactions to the device if more objective data is needed. What is expected of the evaluation will determine the assessment technique selection in major part. The stages and contexts in which evaluation techniques are often applied differ. Some offer qualitative measurements as opposed to quantitative ones and are more subjective than others.
- While some give information right away, others receive input after the event. But the more direct approaches also have a tendency to impede the interaction the most. Lastly, some take more resources than others in terms of time, tools and knowledge.
- Icons symbolize actions, categories, or concepts in a simplified and visual manner. Consistent design language and recognizable icons improve usability and user understanding. Icons should remain clear and identifiable at various sizes and resolutions.
- Design elements like color, typography, imagery and iconography play vital roles in shaping user perception, aiding usability and creating visually compelling and functional interfaces in software design.
- Designing for different screens, especially in the context of responsive web design, involves creating web experiences that adapt seamlessly to various devices, screen sizes, orientations and resolutions. Responsive web design is fundamental for creating adaptable, user-friendly web experiences across a multitude of devices. Prioritising mobile-first, utilising flexible design principles and continuous testing are essential for successful responsive designs that cater to diverse user needs and devices.

© Amity University Online

Notes

Glossary

- HCI: Human Computer Interaction
- UA: User's Action
- SD: System's Display or Response
- ANOVA: Analysis of Variance
- EVA: Experimental Video Annotator
- MUSiC: Measuring the Usability of Systems in Context/Metrics for Usability Standards in Computing

Check Your Understanding

1. What is the primary objective of evaluation in software engineering?
   a) Assessing user preferences b) Identifying design trends
   c) Ensuring software quality and effectiveness
   d) Enhancing coding speed
2. What does evaluation seek to achieve concerning software development?
   a) Increasing budget allocation b) Continuous process improvement
   c) Reducing team collaboration d) Limiting user feedback
3. What is a significant advantage of using expert evaluation in software assessment?
   a) Gathering subjective opinions b) Providing diverse user feedback
   c) Reducing the need for testing
   d) Offering specialised insights and recommendations
4. What is a primary focus in expert evaluation during software assessment?
   a) Identifying user demographics b) Providing extensive user training
   c) Leveraging specialised knowledge and experience
   d) Simplifying the software interface
5. Which evaluation method is focused on gathering real-time insights into user interactions with software?
   a) Contextual Inquiry b) Expert Review
   c) Surveys and Questionnaires d) Controlled Experiments

Exercise

1. What are Goals of Evaluation? Explain briefly.
2. Explain in detail Choosing an Evaluation Method.
3. What are Color Theory, Typography and Imagery? Give examples.
4. What do you understand by Iconography?
5. What is Designing for Different Screens? Explain.

Learning Activities

1. Complete the cognitive walkthrough example for the video remote control design.
2. What are the benefits and problems of using video in experimentation? If you have access to a video recorder, attempt to transcribe a piece of action and conversation (it does not have to be an experiment – a soap opera will do!). What problems did you encounter?

(c) Amity University Online

Module - IV: Models and Theories

Notes

Learning Objectives

At the end of this module, you will be able to:

- Understand Cognitive Model
- Infer Communication and Collaboration Model
- Know Task Analysis and Decomposition
- Define Knowledge Based Analysis
- Discuss Dialog notation and Semantics
- Understand Natural Language Processing

Introduction

Interaction Design (IxD) is a pivotal component of Human-Computer Interaction (HCI), concentrating on the creation of meaningful user experiences with digital systems. This field emphasizes user-centered design principles, conducting task analysis and employing prototyping and wireframing techniques. Visual design is integral to interaction Design, aiming for aesthetically pleasing interfaces with consistent color, typography and imagery.

4.1 Interaction Design in Human Computer Interaction (HCI)

The design process includes incorporating feedback, usability testing and ensuring accessibility. Responsive design, micro interactions and thoughtful navigation and information architecture contribute to creating intuitive, engaging and efficient digital experiences. Interaction Designers consider user personas and scenarios to align designs with user needs and motivations, ultimately shaping positive interactions between users and technology.

4.1.1 Cognitive Model

A model is a readily manipulable depiction of something. In this case, we look at models of users' cognitive capacities. That is, instead of examining the environment in which users operate, we will take a look at behavioural models. These cognitive models are significant since they offer a means of comprehending individuals' cognitive capacities, which may be utilised to guide system evaluation and design. Let's examine the GOMS and ICS cognitive models.

GOMS

GOMS makes use of an explicit model of cognitive processes that was created through user research. Goals, Operators, Methods and Selection Rules make up the acronym GOMS. The tasks that users complete and how they accomplish them are described by this basic collection of concepts. But GOMS is more than just a task description tool; it offers a family of modelling approaches that, given task descriptions and the user interfaces to be utilised, may be used to anticipate how long a task will take to complete.

These forecasts, which have an absolute accuracy of 10% to 20%, are based on a basic cognitive model known as the Model Human Processor (MHP). The name of the system makes clear the fundamental premise of GOMS: users know what their goals are and GOMS evaluates their activities in relation to those goals. GOMS can provide us with comments regarding the coverage and consistency of the user interface now that this

(c) Amity University Online

Notes

assumption has been made. When we talk about coverage, we're talking about whether the UI has all the features required to fulfil the tasks for which it was intended. When we talk about consistency, we're talking about how similar tasks are carried out with the user interface. Additionally, GOMS can indicate whether frequent targets can be completed rapidly using the provided interface.

Model Human Processor

The central part of GOMS is the Model Human Processor (MHP). This is a simplified view of the psychological processes involved in human cognition which can be used to make approximate predictions of user behaviour. Card et al. based this model on empirical evidence i.e. psychological studies of users. Even though it is based on psychological evidence and theory the motivation for the MHP is that is should be usable by non-psychologists e.g. designers and evaluators. The MHP is consists of:

- set of processors – systems that can process data and make judgements
- set of memories – regions where processors store data in different formats
- set of principles of operation – They direct the processors' operations and are essential to a realistic portrayal of human behaviour.

The perceptual, motor and cognitive subsystems comprise the three interacting subsystems that make up the MHP itself. As shown in the following picture, each of these subsystems operates on distinct types of information and has its own memories and processing capabilities.

1. Perceptual system -The cognitive system processes data from the eyes and hearing in working memory and may store it in long-term memory for later use. This symbolic storage occurs in visual and auditory image stores. A automobile horn, for instance, is coded as a loud warning.
2. Cognitive system - decides how to react by combining information from working and long term memory. For instance, recognising that there is a loud warning sound, remembering that under certain circumstances, evasive action is advised and choosing to run to the pavement.
3. Motor system - carries out reactions to choices the cognitive system has made. like sprinting towards the sidewalk.

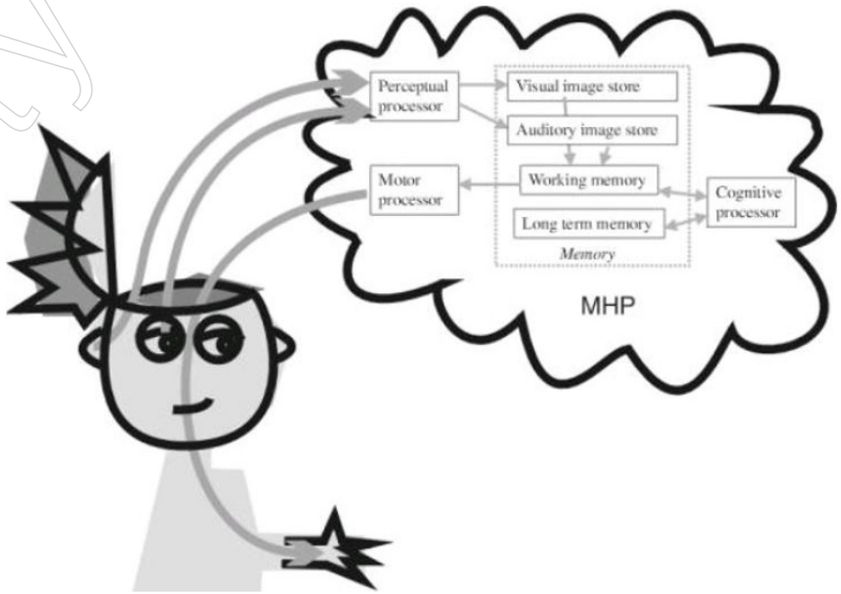


Figure 62: MPH

Notes

The parameters of each part of these systems are used to generate behavioural predictions. The store capacity (the number of bits of information that a memory system can hold at once) and the decay time (the amount of time that passes before an item is forgotten) are the two characteristics that define each memory system. In a similar vein, processor cycle times, or the amount of time needed to process a single piece of data, are used to characterise each processor. Based on psychological studies, Card et al. determined approximations for these factors, which are listed below. It should be noted that these values only forecast the time required for responses to stimuli, not for self-initiated actions and they assume expert (errorless) performance.

Table 13: Memory parameters

Memory type	Storage capacity	Decay time
Auditory	4 letters	1500 msec
Visual	17 letters	200 msec
Working memory	7 chunks of information	7 sec
Long term memory	Unlimited	Never

Table 14: Processor parameters

Processor type	Processing cycle time
Processor type	100 msec
(eye movement)	230 msec
Cognitive	70 msec
Motor	70 msec

Now that the MHP has been explained, let's go back to the basics of GOMS, including its objectives and operators, to see how these characteristics are used to prediction.

Goals

A goal is anything the user wishes to do, such as making a directory, going to the airport, or deleting a file. They are made up of several smaller goals that must be accomplished in order to reach the main objective, giving them a hierarchical structure.

Operators

Operators are fundamental perceptual, motor, or cognitive activities that are required to alter the user's surroundings or mental state and cannot be broken down into smaller operations. They represent the lowest level of a GOMS analysis as a result. A user's behaviour can be captured using GOMS as a series of operators as operators cannot occur simultaneously. For instance, the following actions could be taken by the user to relocate a file to a new folder:

- Move cursor to item
- Hold mouse button down
- Locate destination icon
- Let go of mouse button

Methods

We construct methods, or sets of actions that achieve an objective, from operators. Similar to goals, these techniques can have additional (sub)goals. A basic tenet of GOMS

(c) Amity University Online

Notes

is that knowledge is stored in a single fashion by users and that techniques are routine and taught, meaning they do not require problem solving.

Selection Rules

The Selection rules specify the approach to take if there are multiple ways to achieve a goal. They presume error-free performance, just like methods do (so the user does not accidentally select the incorrect method). They are formatted as the following IF... THEN statements:

IF <condition> THEN accomplish <GOAL>

For example:

IF <restaurant accepts credit cards> THEN <pay by credit card>

ELSE

IF <restaurant accepts cheques> THEN <pay by cheque>

ELSE

<pay by cash>

Keystroke Level Model

The lowest level of GOMS analysis is called the Keystroke Level Model (KLM). This produces quantitative predictions of the time it would take a skilled operator to complete a task. Again, it assumes error-free performance by the operator.

Execution of a task is described in terms of

- 5 physical-motor operators:
  1. Tk: (k)eying – the duration of time required to press a key, including when utilising modifiers such the shift key
  2. Tp: the duration of time required to press a key, including when utilising modifiers such the shift key
  3. Th: (h)oming – the duration of time required to switch between input devices, such as a mouse and keyboard
  4. Td: (d)rawing – how long it takes to use a mouse or other input device to draw a line
  5. Tb: click (b)utton – How much time does it take to click the mouse?

As a result, the total number of operators utilised to complete a task is utilised to characterise its execution time. Let's say, for instance, that we had wrote, "The quick fox jumps over the lazy dog." Using a word processor, we now want to put brown right after quick. Assuming that the current point is at the conclusion of the phrase, we need to carry out the following actions:

1. move hand to mouse
2. position mouse just after quick
3. move hand to keyboard
4. formulate word to insert - brown
5. type brown
6. reposition insertion point at end of sentence

(c) Amity University Online

Notes

Interacting Cognitive Subsystems

The complex framework known as ICS makes the assumption that human perception, thought and behaviour can be broken down into separate, connected information processing units. As we will see in this section, ICS is a considerably richer means of describing human cognition than GOMS.

Subsystems

Each ICS subsystem functions independently and in a distinct processing area that consists of three primary components:

- sensory – visual and auditory stimulus
- representational – representations of information
- effector – body movement

Each subsystem has a common structure, as shown in the accompanying figure, even if it operates on different types of information and in distinct domains. Every subsystem has a memory record, a set of transformations, an input to its left and one or more outputs to its right. Every piece of data that enters the subsystem is kept in the memory record (and may eventually be utilised as an input into the subsystem).

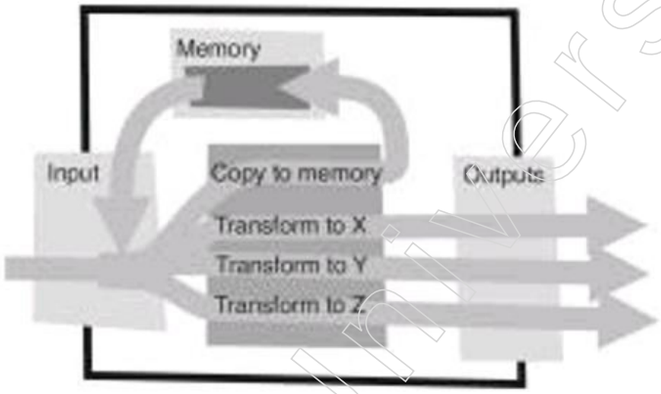


Figure 63: ICS subsystem

The transformations that change the form of information determine the outputs. Three transformations are used in the example to change the input data into the three distinct codes, X, Y and Z.

ICS uses a combination of these interconnected subsystems to model human cognition. Later on, we will examine the entire network; for now, let us examine the functioning of a single subsystem to gain an understanding of the ICS architecture. The visual subsystem, which is a component of the sensory domain and is depicted below, is a useful place to start when trying to comprehend ICS. This converts input from the eyes—which are viewed as edges, angles, shades, contrasts, hues and so on—into object representations—which are viewed as depth, position, orientation, shapes and so on—by utilising sensory representations.

In this case, the object representation describes the image as a grey pentagon. It should be noted that the memory record keeps a copy of the visual representation for future usage. Also take note of the fact that the effectiveness of the transformation into an object representation depends on how clear the visual representation is. Though at distinct information levels, it's crucial to keep in mind that these two mental representations—visual and object—are both representations.

© Amity University Online

Notes

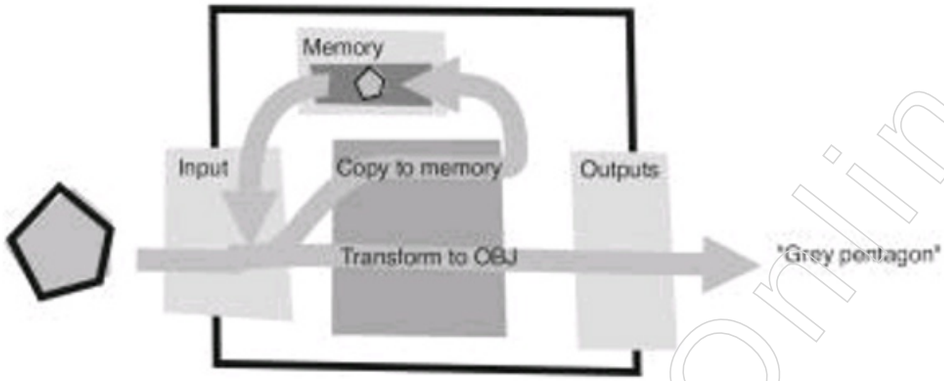


Figure 64: Visual subsystem

The interconnection of the subsystems is essential to use ICS. An illustration of the entire ICS framework is shown below; the Visual Subsystem (VIS), which was previously discussed, is located in the sensory domain, or input from the senses, at the bottom left of the picture. To derive meaning and make decisions, we must process information further. For example, we might have to interpret the grey pentagon as a fifty-pence piece before using it effectively.

ICS can be used to explain what goes through a user's head when they are working on a task. The relative quantities of cognitive resources—that is, the number of transformations—necessary to finish tasks with various interfaces can then be evaluated using such descriptions. In the example, a word processor user is editing some text. They choose to reword a particularly challenging sentence they come across while reading the material. One way to begin an ICS description of the cognitive processes needed to complete this activity would be to think about the sensory inputs. The user is using their visual sensory input as they read the words off the screen. The Visual Subsystem, which converts the visual representation into an object representation (VIS ® OBJ), is the first subsystem to be involved. The information about the letters being viewed is produced by this transformation. The output is processed by the Object Subsystem, which carries out the OBJ ® MPL transformation, because it is in OBJ form.

By encoding the surface structure of the sentences rather than the letters that make up the words, this morphonolexical representation encodes the meaning of the words in a speech-based code that ignores the surface structure of the words. The Morphonolexical Subsystem then processes the MPL representation, transforming it into a representation that encodes the meanings of each word in the sentence as well as their relationships with one another. This process is known as MPL ® PROP transformation. The Propositional Subsystem converts this representation into an implication representation (PROP ® IMPLIC), which identifies the meaning of the statement.

If many interpretations of the sentence are found, PROP ® IMPLIC (Propositional Subsystem) and IMPLIC ® PROP (Implicational Subsystem) may be used multiple times until the meaning is understood satisfactorily. To make the repair at this point, the representation must be changed into a format that works with the Limb Subsystem. The IMPLIC® PROP (Implication Subsystem), PROP ® OBJ (Propositional Subsystem), OBJ ® LIM (Object Subsystem) and LIM ® MOT(hand) (Limb Subsystem – converts LIM into actual hand actions) would all need to be transformed in order to accomplish this.

(c) Amity University Online

Notes

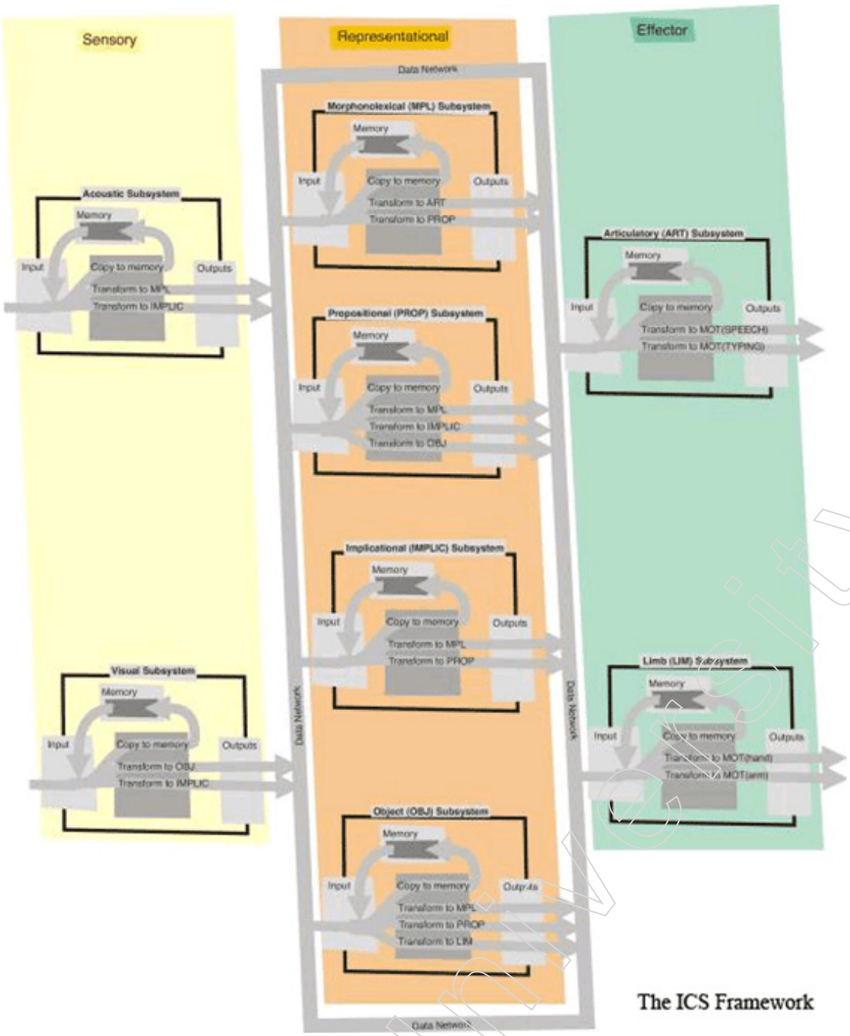


Figure 65: ICS Framework

This description provides an understanding of the processes involved; of course, there is much more processing involved in finding the text on the screen, putting the cursor where it belongs, reading the menu bar, etc. The following is a summary of the sequence described: \ stands for external input or output; ® denotes a transformation; » denotes data transported over the data network; and {} encloses a collection of transformations that may occur many times.

Þ VIS ® OBJ »

» OBJ ® MPL »

» MPL ® PROP »

{» PROP ® IMPLIC »

» IMPLIC ® PROP »}

» PROP ® OBJ »

» OBJ ® LIM »

» LIM ® MOT(hand) Þ

The goal of ICS is to give a thorough explanation of the mental resources required to operate and get familiar with a system. Because of this, it is significantly more powerful than

Notes

GOMS, but it is also more difficult to use and understand the output it generates. It would be difficult for even experienced psychologists to pinpoint which Subsystems are active while users are trying to finish activities. Creating a software toolkit to assist evaluators in using the framework is one way to address this issue. This could help with decisions about which representations are appropriate and which Subsystems are employed.

4.1.2 Communication and Collaboration Model

It is obvious that multiple people are involved in groupware systems, such email or conference calls. All systems, however, both impact and are influenced by the social groupings and circumstances in which they are situated to some degree. Computer-supported cooperative work (CSCW) is a field that includes the effects of computers on cooperative working in general as well as groupware systems in particular.

Face-to-face Communication

When it comes to technology, face-to-face interaction is the most archaic method of communication. Conversely, when we take into account the communication style, the way various channels interact and productivity, we discover that face-to-face communication is the most advanced method now in use. First of all, in addition to speech and hearing, face-to-face communication also includes the nuanced use of body language and eyegaze. Let's talk about a few of these occurrences and how they affect how we utilise computer-mediated communication.

Transfer Effects and Personal Space:

All of our social conventions and expectations from face-to-face communication transfer over to computer-mediated communication. Individuals are highly flexible and can pick up new customs to fit in with emerging media (e.g., turning over a walkie-talkie by saying "over"). Nonetheless, the ability of the participants to apply their own norms is frequently a prerequisite for success with new media. Moreover, we are not always aware of the real issue when face-to-face communication norms are breached since we are not mindful of them. We can simply feel uncomfortable, or we might think our coworker has been impolite. One of these issues is with respect to personal space. We usually stand with our heads at a pretty constant distance apart when we are speaking. People will quickly approach each other until they are a few feet apart if they begin talking at opposing ends of the room. The precise distance varies somewhat depending on the situation; loud noises may compel people to approach in order to be heard. Nonetheless, even in packed spaces, conversants will incline their heads in the direction of one another when conversing before straightening up to maintain their personal space again. It's also crucial to have direction. Those who are behind or at our sides are easier for us to accept than those who are in front of us. As a result, when speaking, people on tube trains who are compelled to sit close to one another will tilt their faces inward.

Eye Contact and Gaze:

Long-term eye contact is typically saved for romantic relationships. On the other hand, casual conversation makes substantial, if not intense, use of eye contact. Our eyes can indicate interest, bewilderment, or boredom, thus we can detect if our colleague is paying attention or not. Making sporadic direct eye contact, or staring into each other's eyes, is crucial for creating a sense of social presence and involvement. When you stare at someone, they could look shifty and like they're hiding something. Furthermore, there is a strong correlation between authority and power and the relative frequency of eye contact as well as who "gives way" from direct eye contact.

(c) Amity University Online

Notes

Motions and Body Language:

We signal objects of attention with our hands in a manner that is comparable to but more direct. This could be as simple as a small hand wave or body alignment, or it could be intentional and purposeful as we point to the object. Once more, it's possible that our colleagues won't be able to read our actions with just a video connection. This could be a big issue because we frequently use gestures (or eye contact) to signify "this" and "there" in our speech, such as "let's move this one there."

Then, much as in person, the participants can point at the appropriate item on the screen. It goes without saying that the group pointer is utilised in both synchronous co-located groupware—like meeting rooms—and distant groupware. In other words, despite being able to communicate in person, the participants still require deictic aids. One explanation for this is that, despite being conceptually shared (allowing them to all see the same image), their electronic screens are not physically shared (they are located in separate locations). Hence, Jemima's coworkers are unaware of what she is pointing to when she gestures to her own screen. The use of electronic equipment can affect the body language that is typical of face-to-face conversation even when the participants are in the same room.

Eye contact chances may be diminished when attention is fixed on the keyboard and screen. Large monitors may also obstruct participants' views of one another's bodies, making it harder for them to decipher body language and gestures. To lessen these issues, the majority of computer-supported conference rooms integrate monitors into the desks.

Conversation

- Although transcripts are still devoid of back channel information, they can be utilised as a highly annotated discussion structure.
- An additional structure involves taking turns, which can be understood as adjacent pairs, such as A-x, B-x, A-y, B-y.
- The discourse determines the context.
- Adjacency pairings make it challenging to maintain context awareness since the context's attention can change.
- Conversational breakdowns are frequently seen and can be identified through transcript analysis.
- It is crucial to find a common ground or grounding in order to comprehend the shared context.
- The propositional meaning of the sentences forms the foundation of speech act theory.
- Taking into account these actions as illocutionary points in the diagram, a state diagram of the aforementioned can be created. We refer to this as "Conversation for Action."

Text Based Conversation

Text-based communication is the primary means of direct communication for asynchronous groupware (and even some synchronous systems). There are several exceptions to this rule, such as answerphones and voice messaging systems. In addition to text, other material can be utilised, such as images, audio annotations and even short videos. However, text continues to be the most often used media. Most people are accustomed to text-based communication since they have written and received letters. But writing a letter and communicating with someone in person have quite distinct styles. Groupware systems use text-based communication to replace speech, which causes certain adaptation issues between the two media.

© Amity University Online

Notes

Currently available groupware offers four different textual communication formats:

- discrete – targeted communication, similar to an email. Other than when a message's text makes reference to a prior one, there isn't a clear relationship between them.
- linear – The messages from participants are appended to the end of a single transcript in a (typically chronological) order.
- non-linear – when text messages are connected to each other using hypertext.
- spatial – where a two-dimensional surface is used to organise communications.

Furthermore, there's a chance that the communication is linked to other shared computer artefacts.

Structured messaging systems like Coordinator, where each message's function and order are predetermined, are a special case of a linear transcript. The transcript can also be displayed as a single stream without any unique fields other than the contributor's name, which is the other extreme.

4.1.3 Task Analysis and Decomposition

The process of examining how employees carry out their work, including what they do, what they act upon and what knowledge they require, is known as task analysis. If we were thinking about the housekeeping position, for instance, we would want to mention things like:

in order to clean the house

get the vacuum cleaner out

fix the appropriate attachment

clean the rooms

when the dust bag gets full, empty it

put the vacuum cleaner and tools away

In order to complete this assignment, we must be knowledgeable about vacuum cleaners, their attachments, dust bags (if applicable), cabinets (where the vacuum cleaner is stored), cleaned rooms and other related topics.

Let's examine three distinct task analysis methodologies that share certain similarities but place slightly different emphasis on them. These are listed in the following order:

- Task decomposition examines how a task is divided into smaller tasks and what order those tasks are completed.
- approaches based on knowledge that examine the organisation of knowledge and what users need to know about the tools and procedures used in a task.
- The focus of entity-relation-based analysis, an object-based method, is on identifying the objects and actors as well as their relationships and the activities they do.

Task analysis uses various forms of observation as its primary instrument. It is concerned with current systems and procedures.

Task Decomposition

The hoover cleaning example provided above demonstrated how a single task, such as "clean the house," might be divided into multiple smaller ones, such as "get the hoover out." To express this kind of behaviour, task decomposition is a common component of task analysis methodologies. An example of such a technique is hierarchical task analysis (HTA). The results of HTA include a hierarchy of tasks and subtasks as well as plans outlining the sequence and circumstances in which subtasks are completed.

(c) Amity University Online

Notes

0. in order to clean the house
    1. get the vacuum cleaner out
    2. fix the appropriate attachment
    3. clean the rooms
        3.1. clean the hall
        3.2. clean the living rooms
        3.3. clean the bedrooms
    4. empty the dust bag
    5. put the vacuum cleaner and attachments away

Plan 0: do 1 – 2 – 3 – 5 in that order.
when the dust bag gets full do 4

Plan 3: do any of 3.1, 3.2 or 3.3 in any order
depending on which rooms need cleaning

Figure 66: How to clean a house

For instance, we may further break down the subtask "clean rooms" in the house-cleaning example as shown in the above Figure. The tasks are numbered to highlight this structure and the levels in the task hierarchy are indicated by indentation. The tasks that the plans relate to are marked on the labels. Plan 0 therefore describes how we will carry out Task 0's subtasks 1 through 5. Plan 3 also describes how we carry out steps 3.1–3.3. Subtasks 1, 2, 4 and 5 have not been broken down, hence there are no plans for them.

Upon reviewing the plans, it is evident that not every subtask must be completed, nor must they be done so in the specified order. Initially, plan 0 must be followed. Subtask 4, "empty the dust bag," need only be carried out if it is discovered that the dust bag is full. Since this is included in plan 0, we take it for granted that we may empty the dust bag at any time, even when we initially take the hoover out and when we store it. We could decide to include this subtask under "clean the rooms" in step three if we are aware that we only ever notice the bag is full when we are actively using the machine. This kind of reorganisation—identifying the proper and significant hierarchy—is a step in the HTA process.

Upon examining plan 3, which outlines how to clean the rooms, we can observe that we are free to clean the rooms in any sequence. Had the assignment involved polishing the floors instead of washing them, the hallway would likely be completed before the remaining rooms! In addition, we only hoover the rooms that require cleaning. We won't need to clean the bedrooms as frequently because they won't get dirty as quickly as the hallway. We could create a more detailed plan if we wanted to be more exact about when the rooms are cleaned:

Plan 3: do 3.1 every day

3.2 once a week

when visitors are due 3.3

This process may continue indefinitely, so a stopping criterion of some kind is used to determine when the jobs are sufficiently simple. Naturally, the goal of the task analysis will determine the extent to which we carry out this. Let's take an example where we were examining a chemical facility and have created a basic breakdown of what to do in case of emergency:

0. in an emergency

1. read the alarms

2. work out appropriate corrective action

3. perform corrective action

We would be interested in extending tasks 1 and 3 if installing computer monitoring

Notes

of the plant is our final goal. Nevertheless, job 2 would need to be expanded if the goal is to provide online operating guides. Actually, the analyst would definitely enlarge each subtask at this high level of task description since she should adopt a somewhat broader perspective. Nonetheless, it stands to reason that one would work harder on the subtasks that are directly related to the main goal.

The P × C rule is one such rule that is very applicable for creating training materials. According to this, expansion should cease if the task's probability of error (P) multiplied by its error cost (C) is less than a certain value. In other words, unless they are absolutely necessary, basic jobs don't need to be increased (because nobody needs training).

When a task calls for complicated motor responses (such as mouse movement) or requires internal decision-making, this is another clear indication to halt. In the first instance, decomposition would be counterproductive; it is unlikely to be truthful or helpful to explain how such actions are carried out. In the second scenario, we would go further if the decision-making involved actions that were external, like consulting a document or using a tool, rather than when the activity was solely cognitive. This might not apply if we were going to develop a decision support system, in which case we might need to comprehend how someone approached a problem in order to develop helpful tools. Nevertheless, whether HTA is the right tactic in this situation is up for debate.

The task hierarchy can be expressed both textually and diagrammatically. A task hierarchy for preparing a cup of tea is shown in the figure below. There are six subtasks that make up the primary task, "make a cup of tea." Only the first one, "boil water," is elaborated upon. The subtasks of 1.1–1.4 and the remaining tasks 2–6 are highlighted, indicating that the study was purposefully ended at that time. This clearly indicates the same information as the written form, but it might be easier to quickly scan.

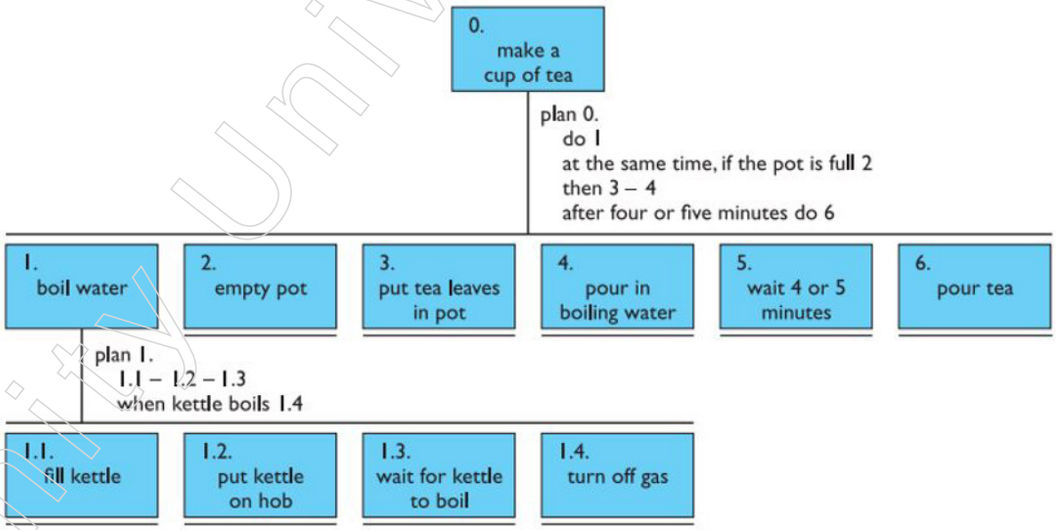


Figure 67: Hierarchical task analysis: making a cup of tea

Once a task hierarchy was created, it would be reviewed for mistakes or omissions. Speaking with a domain expert about the steps in the task hierarchy is one method to go about this. This would demonstrate very quickly that there is a serious mistake in the tea-making plan: the pot is not warmed. Between jobs two and three, this must be added.

Due to the nature of expert knowledge, items that should be clear are often overlooked for task descriptions. Examining the subtasks' format is one way the analyst might look for these omissions. For instance, 1.4 instructs you to "turn off the gas," but it makes no mention of turning it on! This was perhaps implied in the phrase "put kettle on hob," but it

(c) Amity University Online

Notes

needs to be added in between tasks 1.2 and 1.3. We may now observe that there is some imbalance in the task hierarchy. This may be accurate, but it's possible that we added too many intricate duties at the top level. We decide to add a new top-level node called "make pot," which would include the additional "warm pot" task in addition to tasks 3 and 4.

Now, the most important responsibilities would be

0. make a cup of tea
1. boil water
2. empty pot
3. make pot
4. wait 4 or 5 minutes
5. pour tea

Plan 0. do 1

at the same time, if the pot is full 2

then 3 – 4

after four or five minutes 5

This is almost there: it's obvious that the commands "empty pot" and "wait 4 or 5 minutes" don't need to be expanded. They are also not required to be a part of any of the other tasks. While we can empty the pot while the kettle is boiling, we have to wait for the kettle to boil before we can conduct any of the other tea-making chores. This is why we would assume that "empty pot" should go in with "make pot." Similarly, since the pouring of the tea depends on it, the 'wait' node ought to be at the top. There is a tiny oddity with the "pour tea" node. Is this really that much easier than, like, putting together a pot? Maybe we ought to extend this node as well. It could be divided into three sections:

5. pour tea

5.1. put milk in cup

5.2. fill cup with tea

5.3. add sugar to taste

Plan 5. 5.1 – 5.2 if

desired 5.3

However, the mention of cups makes us wonder: do we really only want to describe the making of a single cup of tea? Perhaps we ought to allow several cups of tea to be made. To do this we modify the plan to allow repetitions of steps 1–3 for each cup. We could describe this plan in words, or use a simple diagram as in Figure below.

Plan 5.

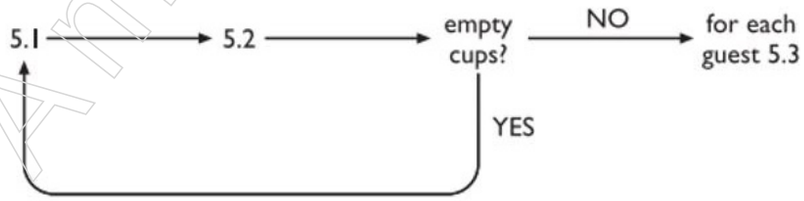


Figure 68: Plan for pouring tea

Following all of this investigation, the modified HTA is shown in the figure below. Asking the visitors if sugar is necessary has also been included to the process of adding the sugar. Furthermore take note of the change in the primary objective from "make a cup of tea" to "make cups of tea."

Notes

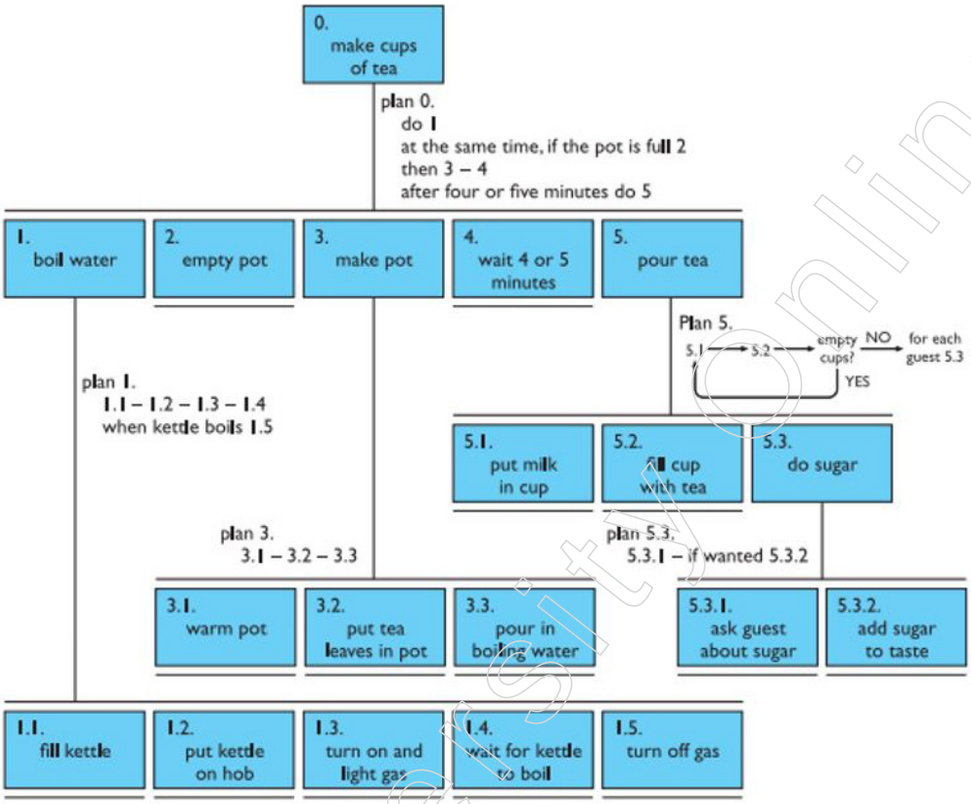


Figure 69: Modified task hierarchy for making lots of cups of tea

4.1.4 Knowledge Based Analysis

Building taxonomies of all the items and behaviours involved in the task is the first step in knowledge-based task analysis. Taxonomies resemble the kind of hierarchical statements we frequently encounter in biology: fish, birds, reptiles, amphibians, mammals and other vertebrates are examples of animals, while invertebrates are vertebrates. The goal is to comprehend the information required to complete a task, which will aid in the creation of instructional materials and the evaluation of the degree of knowledge shared across tasks.

Let's examine it using the example below: composing a letter and getting it ready to be sent. First, we have a look at Linden's activity description:

Description	Object	Action
First, we observe Linden, who has no computer support, performing this task in her office. She finds a sheet of paper, a mathching envelope and a pen, and settles down at her desk to write.	Paper Envelope Pen	Find
She starts by putting her own address in the top irght-hand corner, then picks up her address book (which is lying on the desk beside her) and flicks to the page containing the addressee's address information. She copeis the addressee's address to ther paper (left-hand side), them writes today's date and starts on the body of the text: " Dear...." When she has finished, she signs the letter	Address (Sender) Address (addressee) Address book Page Date Body text Signature	File up open & flick write sign
& then reaches for the envelope, writes the name on it and copies the address from the address book into the envelope, &	Envelope Address (addressee)	Pick up write

(c) Amity University Online

Notes

& folds the letter, puts it into the envelope and seals the envelope. Task completed.	Letter Envelope	fold Insert Seal

Figure 70: Linden's activity

Upon perusing this inventory, it is evident right away that there are two categories of objects: the tangible items that Linden is dealing with and the letter's constituent pieces (the text that transforms a blank page of paper into a letter). There are arguably more distinctions as well, such as the difference between an address as a type of thing-in-the-world (used, for instance, by the postal service) and an address as a brief passage of text that provides details about the related thing-in-the-world. On the other hand, it's possible that the actions "find" and "pick up" are practically interchangeable for this purpose as they both depend on the user having the related object in their possession.

This is how a basic taxonomy might appear. To express whether an item can be or have all of the sub-components at the same time, we use the terms AND, OR and XOR. In this case and denotes "all of," OR denotes "any number of," and XOR denotes "exactly one or exclusive."

- Letter-writing object XOR
  - Physical object XOR
    - Paper XOR
      - Blank sheet
      - Letter
    - Pen
    - Envelope
    - Address book
      - Page (lots!)
  - Portion of text XOR
    - Address AND
      - Owner XOR
        - Sender
        - Addressee
      - Location XOR
        - Letter
        - Envelope
    - Date
    - Body text
    - Signature

The majority of the taxonomy is XOR (for example, the address owner is either the sender or the addressee and we do not take into account the possibility that these might be the same), but we also state that an address has both an owner AND a location (where it is written). In this specific case, there are no "OR" objects. Some peculiarities exist, such as the fact that a sheet of paper can be either blank or written on; it would be good to have a third choice that is half written. We'll stop here and provide the action-corresponding taxonomy instead of expanding on this one:

© Amity University Online

Notes

Letter-writing action XOR

Gain possessionof

Find (e.g. locate address in address book)

Write XOR

Write header information

Write general text

Sign

Fold

Insert

Seal

Neither the action taxonomy nor its revelations are particularly intriguing at this level of description. On the other hand, more specific details may make it extremely valuable. Take into consideration, for instance, the pull-down menus seen in word processors like Microsoft Word. There is a solid taxonomic basis for the organisation if you examine how the menus are arranged under the top-level headers ('file,'edit,'insert,' etc.), how the items are grouped inside each menu and the options that are made available. Check out the Figure below, for instance. You can even go a step further and pinpoint locations where the taxonomic organisation may be vulnerable to dispute (for example, where ought to "insert table" be placed? And why does this confusion occur?).

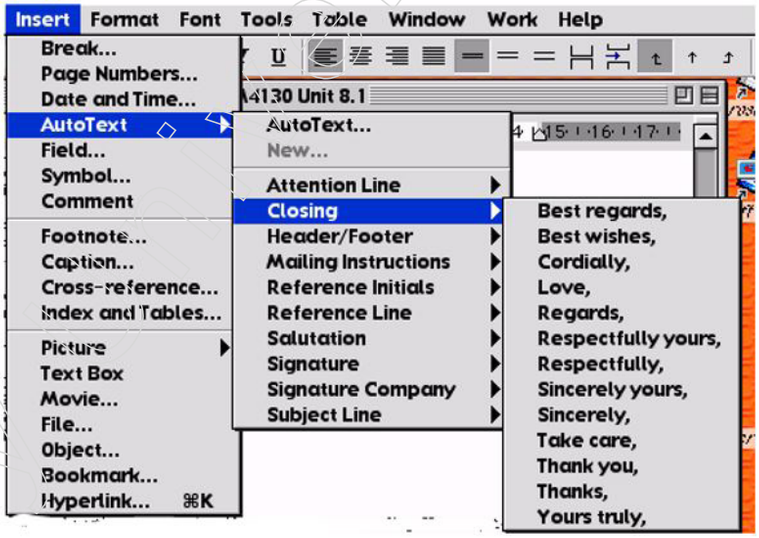


Figure 71: Typical letter closings

Depending on the goal, different levels of detail are needed. A fine level of detail (a list of all the most typical letter closings) is shown in the above picture, which is helpful for this stage of implementation.

4.2 Human Computer Dialogues

A dialogue is a discourse between two or more people, as opposed to a monologue. It now also seems to indicate a degree of cooperation or at the very least a desire to settle disputes. The term "dialogue" refers to a more precise meaning in user interface design, which is the format of the user-computer dialogue.

Computer language can be examined on three different levels:

(c) Amity University Online

Notes

Lexical: The form of the icons on the screen and the actual keys hit represent the lowest level. The spelling and sound of words in human language.

Syntactic: The arrangement and hierarchy of the inputs and outputs. The sentence formation grammar of human language.

Semantic: The significance of the discussion about its impact on the internal data structures of the computer and/or the outside environment. The meaning that each participant assigns to the communication in human language.

The phrase "dialogue" in user interfaces is frequently understood to mean "the syntactic level." The lexical/syntactic divide is not often clear-cut, though and lexical aspects are frequently included in actual dialogue description usage.

4.2.1 Dialog Notation

Let's now examine a few of the notations that have been applied to human-computer dialogues. Some of these may be known to the computer scientist because the user interface developer "appropriated" them from other areas of computing.

So what's the point of using a unique notation? Why not use the programming languages we already have? Let us examine a basic mortgage repayment calculator provided by a financial advising program. (This is not meant to serve as a model for effective dialogue design.)

```
rate = 10
term = 25
print "Our current interest rate is 10%"
print "What is your annual salary?"
input salary
max_loan = 3 * salary
print "How much do you want to borrow?"
input amount
if amount > max_loan
then print "That is too much money"
print "Please consult our financial advisor"
goto finish
end if
repeat forever
print "Our standard term is 25 years."
Print "Do you want this (yes/no)?"
input answer
if answer == "yes" goto calc
if answer == "no" goto rd_trm
print "You must answer yes or no"
end repeat
rd_trm: print "What term do you require (years)?"
input term
```

Notes

```
calc: r = ( 100 + rate ) / 100
payment = r^term * ( r – 1 )
* amount / ( r^(term-1) – 1 )
print "Monthly repayment is ", payment
finish: stop
```

Take note of how the user's dialogue and the program's other elements are mingled together; input-output and computations are interwoven. While some of the if then statements indicate the system's selections (amount > max_loan), others indicate the user's selections (response == "yes"). If you are a programmer, you have probably also observed the usage of goto and the shoddy program structure. It's not that the writers are poor programmers. The program could be rewritten exclusively with structured programming constructs. But the resulting program would be just as cryptic, because programs that require parsing typically contain a lot of ugly structures. The most challenging aspect of this program is making sure the response is "yes" or "no." Interactive programs are frequently dominated by error checking and correction.

Suppose you are now required to perform some sort of analysis on the dialogue: perhaps you are to provide a list of all potential combinations of user inputs and system answers, or perhaps you are to instruct the user on how to obtain the repayments for a fifteen-year loan. This is unexpectedly challenging due to the combination of system and user options and the complex program structure.

Alternately, you could be required to adjust the program's interface style or set it up with a mouse-and window-based interface. Is this challenging? Recall that this is a brief program that consists primarily of user interaction and should be rather simple. Consider a 10,000 or 100,000 line program. Many commercial programs that were first developed for text-based terminals are now accessible on Windows or Macintosh platforms. Such programs' provenance is frequently all too evident, which is hardly shocking.

This provides us with two justifications for employing distinct dialogue description notation: simplicity of analysis and division of the program's interface components from the computations themselves (semantics). Both of these justifications assume that the program already exists; a third justification is to record the dialogue prior to writing the program. This gives the designer the opportunity to evaluate the suggested structure or even run the dialogue using a prototype tool. Members of a design team can also communicate about the design using a dialogue notation, which is finally used by the designer to transfer the intended dialogue to the application's programmer. For this reason, dialogue notations are frequently a crucial component of prototype techniques and equipment.

4.2.2 Dialog Semantics

It might be sufficient to annotate the formal dialogue with the intended meaning of the actions, or to leave it up to the reader to deduce the semantics, if the goal of a dialogue description is just to facilitate communication among designers, or as a "tool for thought" early in the design process. However, there needs to be a method for properly describing the dialog's semantics if the dialogue description is to be used as a formal specification, possibly as part of a contract, or for use as a prototype. The structure of the dialogue is more or less described by the dialogue notations that we have seen. We have to move on to the meaning now.

Dialogue semantics has two dimensions: internal to the application and external to the display. The semantic portion of the dialogue should act as a connector between the two,

(c) Amity University Online

Notes

taking on as little of an active role as feasible. Consider how one determines, for instance, that the lines created by the drawing tool are at the locations indicated by the user's mouse pointer. Let's talk about three methods for tying semantics and dialogue together:

notation-specific semantics: certain semantic forms intended to be a component of a dialogue notation;

links to programming languages – affixing lines of code in the programming language to the dialogue;

links to specification notations – comparable, but with the usage of formal specification notation.

Notation-specific Semantics

One type of state transition network is an augmented transition network (ATN). It is expected in the ATN that the system contains a set of registers, or storage places, that can be set and tested by the transition network. Remember that the event causing the transition and the system response can be labelled on the arcs in a STN. This is extended in an ATN. The arcs have an event and a condition; the event may have to do with registers in the system and the arc is only followed if both the condition and the event are met. The setting of registers is added to the system reaction, in addition to feedback and display. Simply put, these registers can be used to explain more intricate dialogues, like a cash dispenser that holds onto your card after three incorrect numbers. They can also be used to hold mouse values and communicate with the application.

There is a great deal of variation in production norms and the relationship to semantics is no different. Variables are frequently stored in the system's memory and can be used to store input values, like the mouse location, which the rules' conditions and actions can then check. The underlying application's functions may also be accessed using these variables, or they may be called directly by unique types of action. As an example, the production rule that follows places a dot at the mouse location and calls the application procedure "another_point" when the user clicks within a target region is as follows:

click_at(x,y) → dot_at(x,y), call another_point(x,y)

Links to programming languages

Dialogue notations are frequently "attached" to programming languages that are more traditional. For instance, the dialogue semantics are expressed in C by input tools, a regular expression-based notation. The description of the input tool is made up of definitions for the "tool," which include regular expressions mixed in with standard C code. For instance, the following is how the Plasmeijer number reading tool starts:

```
tool number
{ char buf[80];
int index;
int positive;
input { ( digit* + sign; digit; digit* ) ; return }
tool digit
{ input { key key_c>='0' && key_c<='9' | }
if ( index< 79 ) /* append character to string */
{ buf[index] = key_c;
index = index + 1;
```

Notes

```
echo(key_c);
}
}
tool sigl..
tool return
{ input { key:| key_c == '\n' | }
...
}
...
}
```

Input tools have additional operators and their own regular expression syntax. Semi-c"l"n ";" is used to indicate sequencing instead than just basic juxtaposition and the "i"n "+" is used for choice (much "i"e "|" in BNF). A postfix guard is an expression "ike "key:|condit"on|," which matches only if the condition is true.

The expressions for the particular input tools are as follows. The regular expression that the keyword tool indicates is enclosed in the input statement. This new tool is akin to a non-terminal in a BNF grammar. The digit, sign and return tools are private to the number tool due to the scoped hierarchy of the tools. To put it simply, the call to echo returns the character to the user. The last fundamental tool is called key and it matches a single character that is read from the keyboard. The actual character that is read is kept in the global variable key_c.

Take note of how the shared variable buf is used by the subtool digit to relay its findings back to the main tool. This is a messy process, as is the way it takes values from the key tool. This particular notation is not the only one messy; other regular expression-based notations are considerably messier!

Links to Formal Specification

The two components of Alex'nder's executable specification/prototyping language, SPI (Specifying and Prototyping Interaction), are eventISL (which provides the dialogue semantics) and eventCSP (a dialogue notation heavily based on CSP).

EventISL is semi-standardised and semi-dependent on the sele'ted 'host' language. Me-too: a formal specification notation based on VDM, was the first host language; a C counterpart is also available. A clause providing the global varia ormalizlised and changed by the event, a precondition indicating when the event can occur and input and output sections make up the component that is independent of the host language. The changes and exact outputs are only described in the host language portion.

Examine the eventCSP description of a login sequence that follows:

Login = login-mess → get-name → Passwd

Passwd = passwd-mess → (invalid → Login

[ ] valid → Session)

Session = (logout → Login

[ ] command → execute → Session)

A typical unsuccessful login sequence might be

login: fred

passwd: b9fGk (invisible)

Sorry bad user-id/password

Notes

The two events login-mess and get-name handle the first line of the above dialog:

event: login-mess =

prompt: true

out: “login:”

event: get-name =

uses: input

set: user-id = input

The first event states that user input is requested and prints the pr“mpt “l”gin:” (the out: clause). This input from the user is kept in a unique variable n“med “i”put.” The second event simply sets the vari“ble “us”r-id” to the input, which will be set to the name the user inputs. Keep in mind that the event sets the valu“ of “us”r-id” to a new value; any preceding value is ormalizlised.

The steps to retrieve the password are similar, with the exception of two choices based on whether the user entered a valid password or not:

event: passwd-mess =

prompt: invis

out: “passwd:”

event: valid =

uses: input, user-id, passwd-db

when: passwd-id = passwd-db(user-id)

event: invalid =

uses: input, user-id, passwd-db

when: passwd-id ≠ passwd-db(user-id)

out: “Sorry bad user-id/password”

With the exception of not requiring echoing, the password prompt is the same as the login prompt. Still, the latter two incidents highlight two more characteristics. They also make use of the varia“les “pass”d-db”“and “use”-id,” which were set by the get-name event, in addition to the user input variable. Since it is presumed that this is a password database, the’user’s correct passwor‘ is ‘passwd-db(use’-id)’. There is al“o a ”when” clause for the two events. This is a precondition, meaning it outlines what needs to be true in order for the event to happen. Accordingly, a password entered correctly can only result “n a “”alid” event and an erroneous password can only result “n a “in”alid” event.

Observe that eventISL mostly relies on the usage of global variables to transfer data between events, just as input tools. Several dialogue nota’ions’ dependency on global variables goes against every accepted principle of software engineering. Tracing the consequences of events is still not easy, but at least in SPI, the globals that are used and altered are made clear. It is unfortunate that the effic ormalizedising methods found in dialogue notations—hierarchical networks, non-terminals in grammars and processes in CSP—do not translate into semantic effects.

Distributed Ormalized Lised Dialog Description

It is simple to examine and assess the dialogue syntax separately if it is described by a pure grammar devoid of any semantic elements. Conversely, in an average interactive program, the dialogue will be dispersed across the code, making it challenging to follow the flow of a typical interaction.

By keeping the syntactic and semantic components separate, a notation that aims to represent both the semantics and syntax of a dialogue might attempt to preserve the

(c) Amity University Online

Notes

benefits of a straightforward syntactic description, enabling the dialogue designer to focus solely on the dialogue syntax. This describ ormalized lised dialogue. This method is best illustrated by Alex'nder's SPI, which contains distinct sublanguages for the two sections. This illustrates yet another benefit of this strategy.

Often, the syntactic dialogue description in the same form can be used for both automatic coding (or run-time interpretation) and high-level analysis. In contrast, the form of the semantic description generated for specification or prototyping will probably differ from that of the semantic description generated for integration into a production system. By separating the two, the dialogue description can be reused with distinct semantic components, representing various applications of the specification.

As an alternative, the notation may decide to group together related syntactic and semantic elements, as in input-tools. The benefit of this is that certain aspects of the interaction may be studied in-depth, making it possible to assess the syntax and semantics simultaneously. Another benefit of abstraction is that related syntax and semantics can be bundled together. Its drawback is that it has a dispersed dialogue description just like any other program. Gaining insight into the large-scale flow of the interaction requires looking at several parts of the specification.

It is not true that the two methods are inherently incompatible. elements of the dialogue syntax could be easily separated and presented with the relevant elements of the semantic description given a notation of the former type. In a similar vein, it is feasible to isolate the sections that define the dialogue syntax and examine them together using certain distributed notations. For example, all input clauses using the regular expression subtool syntax may be extracted using input-tools.

4.2.3 Standard Formalisms

Standard formalisms are well acknowledged and extensively used notations or languages that are used to express computing concepts, specifications and designs. Formalisms are crucial for generating accurate and unequivocal depictions of systems, algorithms, or processes. Below are some instances of standard formalisms in the field of computing:

1. Unified Modeling Language (UML): UML ormalized dised modelling language widely used in software engineering to visually represent, describe, build and document software systems. The tool offers a collection of diagrams and symbols that depict many elements of software design, such as classes, objects, relationships and behaviour.
2. Z Notation: Z is a formal specification lang ormalizlised to articulate the behaviour and structure of computer systems. The sy ormalizlises mathematical language to express specificati ormalized ising accurate descriptions of data kinds, actions and limitations.
3. Specification and Description Language (SDL): SDL ormalized lised lang ormalizlised to precisely define the actions and operations of systems, particularly in the field of telecommunications. The system specifications are presented in a formal and graphical manner, which includes state machines and message forwarding.
4. Temporal Logic of Actions (TLA+): TLA+ is a formal specification language that enables the articulation of system behaviour and attributes. It encompasses a temporal logic that facilitates the analysis of the behaviour of concurrent and distributed systems.
5. B Method: The B Method is a rigorous approach that employs mathematical notations to precisely define, design and validate software systems. The B-Method language, included in this, is founded on set theory and first-order logic.

(c) Amity University Online

Notes

6. Process Calculi (e.g., π-calculus): Process calculi are a set of formal languages employed for the purpose of modelling and analysing concurrent systems. The π-calculus is a process calculus that enables the representation of processes, communication channels and process mobility.

These conventional formalisms serve as a basis for accurately articulating concepts in computing, assisting in the process of designing, specifying and verifying systems. They enhance the development of superior and dependable software by enabling meticulous investigation and validation of system propert.

4.2.4 Interaction Models

Interactive systems should adhere to the principl‘ of ‘what you see is what yo’ get’ (WYSIWYG), maintain consistency and provide a universal undo feature, among other requirements. However, if a supplier claims that its word proce‘sor ‘Sludge’Word’ is WYSIWYG (What You See Is What You Get), how can we verify this assertion? Do the screen fonts have the same appearance as the fonts on the page? The technique may seem WYSIWYG (What You See Is What You Get) for straightforward tasks, but it becomes less effective when dealing with more intricate ones. What is the true definition of WYSIWYG?

The methods outlIned in this section was created to tackle these challelges. In contrast to formal notations that describe specific systems, the objective in this section is to establish interaction models that are general and formal representations of interactive systems. By employing a universal model, it is possible to establish principles in a precise manner that may be subsequently applied to other systems. By considering Sludge-Word as an exemplification of the overarching framework, we can substantiate (or disprove) certain assertions made by its provider.

The specific architecture we will discuss, known as the PIE model, was created to challenge WYSIWYG-like characteristics. Upon completing the reading of this part, you will possess a precise understanding of the meaning of WYSIWYG. Regrettably, the phrase is too extensive and diverse ormalized ardised.

The conc‘pt of ‘us’bility’ cannot be fully defined; it is not possible to claim that a system will be usable simply by adhering to a set of formal standards. However, certain formal rules are essential for usability. Any system that violates these principles is likely to encounter issues. The formal principles serve as a safeguard to prevent major errors in an interactive system, but they do not guarantee a well-designed outcome. The outcome is contingent upon the expertise of a skilled designer.

The ‘safety net’ provided by interaction models, while valuable, is not the primary advantage of utilising them. The primary significance of these ideas is derived from the examination of interaction features, free from the extraneous details of actual or hypothetical systems. These insights assimilate into your existing knowledge, influencing your approach to new subjects, regardless of whether a formal method is explicitly adopted. Moreover, several of these observations ormalized ralised into informal principles, which, although obtained from formal analyses, can, once expressed, be supported independently. It is an intriguing paradox that certain informal concepts, derived through formal analysis, cannot be comp ormalizedmalised.

The PIE Model

A black-box model is the PIE model. It just discusses a computer’system’s user inputs and outputs; it makes no attempt to depict the internal architecture and development of the

(c) Amity University Online

Notes

system. Keyboard and mouse inputs and the c'mputer's display screen and final printed output would be the usual outputs for a basic single-user system (Figure below).

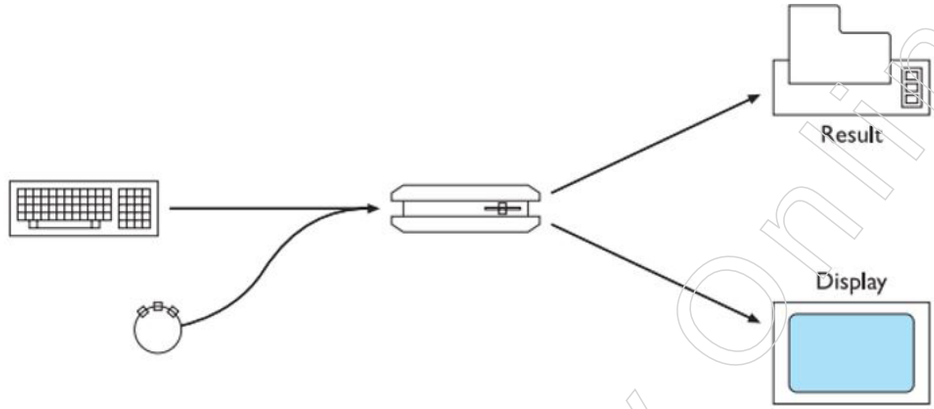


Figure 72: Inputs and outputs of a single-user system

The core of the PIE model is the distinction between a system's transient appearance and its lasting outcome. The set of possible outcomes will be called R and the set of possible displays will be called D. We should discuss the relationship between display and consequence in order to articulate the concepts of observability. In other words, is it possible to infer the outcome (what you will receive) from the display (what you see)?

Discussing the system's internal state is helpful, but not necessary, for a formal declaration of predictability. This does not refute our black-box model assertion. First, the state we define will be ill-defined; we will just assume it exists without considering its structure. Second, the state we'll be talking about is an idealised version of the system, not the real one. It will be the very minimum needed to explain the exterior behaviour in the future. We'll refer to this as the effect (E). From this minimal state, functions display and result in the current outputs:

display : E → D

result : E → R

What is shown right now will be the actual display. The outcome that is being displayed is actually not what is available, but rather what would occur if the interaction were to conclude. In a word processor, for instance, these are the pages that would be produced if the document were printed in its current condition.

We will refer to a single-user action as a command (from a set C). An interpretation function can be used to determine the present impact from the program (P = seq C), which is the history of all user commands.

I : P → E

The PIE model would no longer have its acronym if the input history had been given the more appropriate designation, H! When all the pieces are assembled, we get a diagram of sets and functions (see Figure below), which resembles the original drawing quite a bit.

All desired features can, in theory, be expressed in terms of the interpretation function, I. But typically, this entails expressing characteristics as a quantifiable set over all potential historical periods. We will also employ a state transition function doit to simplify the expression of some of the properties:

doit : E × P → E

(c) Amity University Online

Notes

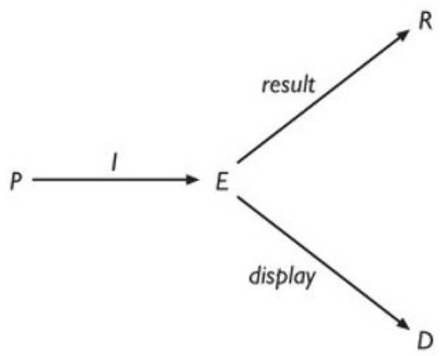


Figure 73: The PIE model

The function doit takes the present state e and some user commands p and gives the new state after the user has entered the commands doit(e, p). It is related to the interpretation function I by the following axioms:

doit(I(p), q) = I(p ^ q)

doit(doit(e, p), q) = doit(e, p ^ q)

Several layers of abstraction can be used to interpret the PIE diagram. Figure (Inputs and outputs of a single-user system) provides a straightforward analogy. The physical display is the display set D, the keystrokes and mouse clicks are the commands set C and the written output is the result R.

C = {‘a’, ‘b’, ... , ‘0’, ‘1’, ... , ‘*’, ‘&’, ... }

D = Pixel_coord → RGB_value

R = ink on paper

At the lexical/physical level of interpretation, this is. For any system, a similar mapping of the raw physical inputs and outputs can be generated. Utilising the model at the logical level is frequently more beneficial. In this case, the user commands are higher-level operations like “select bold font,” which can be accessed with a combination of mouse clicks and keystrokes. Likewise, we can rationally characterise the screen in terms of windows, buttons, fields and so forth. Additionally, for some purposes, we may consider the document on disc to be the result rather than dealing with the ultimate physical product.

The PIE model’s versatility allows for application at various levels of abstraction. While many features ought to hold true across all levels of system description, some might only be true at one. The PIE model can also be applied solely to the user, in that the display represents the user’s perceived response and the commands represent the intended actions of the user. It is feasible to map between the levels when using the PIE model at various levels. Level conformance features follow from this, such as the requirement that changes observed at the interface level match those observed at the application object level.

Predictability and Observability

WYSIWYG is obviously connected to what is discernible from the display. There are two ways to view WYSIWYG, as Harold Thimbleby has pointed out.

One is that at the printer, you will receive exactly what you see. This is correlated with your ability to deduce the outcome from the display. According to the second interpretation, what you see is what the system contains. We’ll inquire what the display can tell us about the effect in response to this. Both of these might be regarded as observability principles.

Notes

Predictability is a similar problem. Consider yourself using a drawing program when, in the middle, you decide you're thirsty and go grab a cup of tea. When you get back, you see the screen. What should you do now? Mouse clicks may be interpreted by the graphics package as affecting the "top" shape if there are two shapes stacked on top of each other. There might not be a visible cue as to which is highest, though.

You must keep in mind how you get there and your command history because the screen image does not indicate the outcome of your activities. The phrase "gone away for a cup of tea problem" describes this situation. Actually, the state of the system dictates how any further commands will work, thus if we have a system that is predictable in that its state is determined by the display, then it is also observable. Observability has a specific instance known as predictability.

We are going to try to codify these attributes. It is said that there is a function transparentR that connects displays to results and that we can infer the result from the display:

$$\exists\ transparent_R : D \rightarrow R \bullet$$
$$\forall\ e \in E \bullet transparent_R(display(e)) = result(e)$$

The second part of the preceding statement states that the function provides us with the identical result that we would receive from the system, hence it is useless to have any old function from the display to the result. This characteristic result is known as transparency.

Similar actions can be taken for the impact, or the system state:

$$\exists\ transparent_E : D \rightarrow E \bullet$$
$$\forall\ e \in E \bullet transparent_E(display(e)) = (e)$$

This quality is known as transparency.

In which of these senses would a system be considered transparent? If the system were transparent, after returning from our cup of tea, we could examine the display and use transparentR to mentally create the precise image of the printed artwork. It's another question entirely whether we could achieve this in our thoughts. The role of the majority of drawing programs would be to just "photocopy" the screen and disregard the options.

Even more powerful is plain transparency. It would claim that everything about the system's current status can be deduced from the display. If any modes exist, they must be visually indicated; if the displayed shapes behave differently from one another, there must be a visual difference to match. Forgetting about the formal concepts entirely, this is a powerful and practical design heuristic.

Regretfully, both of these ideas are a bit too rigid. If we envision a word processor instead of a painting program, the information on the display will only represent a portion of the document. It is evident that the display does not allow us to deduce anything about the remainder of the document or the printed output. Similarly, it might not be practical to provide a visual representation of, example, object grouping within a sophisticated drawing program (and this can cause issues for the user).

A word processor user only needs to scroll the display up and down to see what is contained in a document. Although the current display does not provide all the information about the system, you can still learn more. A strategy is the method by which the user investigates the system's present condition. Even when cognitive limitations are disregarded, the formalisation of a method is highly intricate. Users will employ different tactics, but system documentation should explain to users how to access relevant data.

(c) Amity University Online

Notes

How to identify the groups of objects in the sketching tools, for instance. This will lay out a number of practical tactics that the user can employ.

A passive strategy would be ideal for monitoring the system since it wouldn't change the application objects' current state. A method of seeing a document that required erasing it, for instance, would not be particularly helpful. Though it may seem apparent, think about grouping in drawing tools once again. Often, disassembling a collected thing piece by piece is the only way to learn how it is put together. After that, you'll need to recall how to reassemble it. It becomes clear why a passive approach is preferable.

When one employs this tactic, the system becomes more visible than it does on the display. We refer to this as the observable impact (O). This would be the entire view of the document that might be accessed by scrolling in a word processor, together with any mode indications that are active and a brief glance at the cut/paste buffer's condition. The observable effect is positioned ahead of the display in a functional diagram because it is strictly more informative than the display (Figure below).

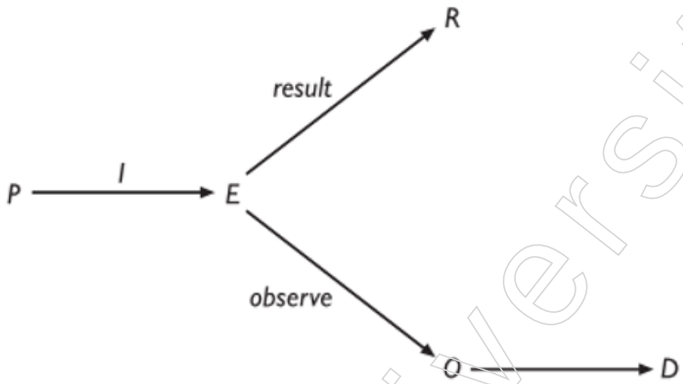


Figure 74: The observable effect

Observable effect reformulation of principles is now possible. First of all, if the system's outcome can be ascertained from its observable effect, it is result observable:

$$\exists\, predict_R : O \rightarrow R \bullet$$
$$\forall\, e \in E \bullet predict_R(observe(e)) = result(e)$$

This indicates that there is an equal amount of information in the observable effect and the result. It will, however, also include more details regarding the system's interactive status. For instance, you will be able to see the current pointer location, but the printed page will not be affected by this.

Now you are aware of what will occur if you press the print button. After having a cup of tea, you get back to work. You hit a function key that, unbeknownst to you, is linked to a macro meant for a completely other program. To your dismay, the screen scrolls, the disc spins and your document along with the contents of the entire disc are destroyed. You get up from the computer and grab another drink, maybe not even tea.

The requirement that the system be completely predictable is a stronger one:

$$\exists\, transparent_E : O \rightarrow E \bullet$$
$$\forall\, e \in E \bullet predict(observe(e)) = (e)$$

This indicates that you are able to view the system's entire status. At that point, everything the system does can be predicted (in theory). In theory, you could have prevented your catastrophe if the system had been completely predictable, as you would have known the function keys' bindings.

Notes

This concludes the formal story's portion in this book. Observability and predictability, however, have more complex rules that account for user attention and problems like keyboard buffers.

Predictability principles are not sufficient on their own; you could have accidentally struck the function key or forgotten what was connected to it even if you were aware of it. There are other protective rules that must be followed, such as proportional effort. Additionally, one likes a system that functions more like the transparency principles than one that necessitates intricate searching in order to get information, even though this is difficult to fully formalise. This is an attempt at observation that is somewhat comparable.

4.2.5 Hypertext and Multimedia

Advances in desktop computing power have made it possible for the multimedia business to expand rapidly. Stereo audio, CD-ROMs and frequently video input/output are now included in the specifications of typical personal computers. Furthermore, the world wide web, which connects computers, information and ultimately people worldwide, has made the dreams of Ted Nelson and Vannevar Bush a reality.

The internet is only one, albeit a significant, example of a hypertext system. First, we'll go over some typical characteristics of hypermedia, such as its various content kinds and application domains. Text can be made more engaging by diagrams and images, but the most amazing results come from using a computer's processing power to enable animation, video and interactive elements.

Hypertext definition – text, hypertext and multimedia

Even though sculpture and images predate writing, text, or the written word, is often recognised as the turning point in human civilisation; preliterate and oral cultures are sometimes cast off as prehistoric solely because they lack surviving narratives. The Book of Kells, the Caxton press, hieroglyphic writing and Babylonian mud tablets are some of the early examples of our modern information-driven culture.

These classic texts are all linear in some way. According to Aristotle's Poetics, a story needs a beginning, middle and finish. In fact, even non-linear postmodern narrative is written in a linear form, even though the events may not be causally related.

The media utilised, such as the paper book, painted frieze, or papyrus scroll, contributes to this linearity in part, but maybe more importantly, we are creatures of linear time. We hear stories naturally and tell them naturally too. That's why scenarios have such great power. The author's expertise is creating a reading experience for the reader that incorporates new ideas and events while making them make sense in relation to earlier occurrences.

There are kinds of activities, though, where the reader must choose how to navigate a text on their own. For instance, the user wants to know what is best for him right now, not a detailed explanation of the system, while using online documentation or assistance services (although tutorials are usually linear). In certain types of exploratory learning, students may choose to go at their own pace through the content, going into detail in separate sections. Subject matter experts may also wish to refresh their memory on a specific problem or piece of information.

A standard text's linear format is inconvenient for these goals. When utilising manuals, for instance, the user might not be familiar with all of the terminology used and will have to repeatedly refer back to a different set of pages in order to look up the meanings, then go back to the original pages and attempt to continue the conversation from there.

(c) Amity University Online

Notes

Encyclopaedias, dictionaries and table of contents sections in paper books are intended for non-linear reading. The reader is still forced to physically flip back and forth, though.

Hypertext structures text as a mesh instead of a line in an effort to overcome these constraints. This makes it possible to access other sites from the one that is currently open and if the hypertext is done correctly, it should be simpler for the user to follow his own specific notion across the mesh as opposed to being led in a certain direction. Hypertext systems typically combine text with other material, such as images, diagrams and photos. Although the phrases are frequently used interchangeably, these systems are commonly referred to as multimedia or hypermedia systems.

A hypertext system is made up of several pages and a collection of links that link the pages together. Each page can have more than one link and links can connect any two pages. As a result, a hypertext document takes many different paths, some of which end and others of which link back into other sections of the document (see Figure below, which illustrates the difference between linear text and hypertext). A hypertext document does not simply start a linear progression and follow it to an end.

A hypertext page can be read in a variety of ways, just as there are numerous ways to navigate a network. The goal is to provide the user the flexibility to read the content however best suits him. Links can be embedded within the content itself, or they can be found at the end of each page, with the user selecting which to follow.

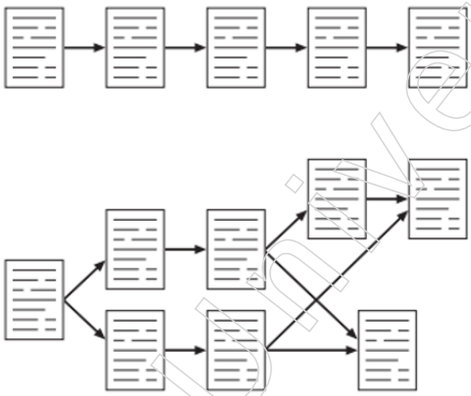


Figure 75: Typical structures of linear text and hypertext

For instance, all of the technical terms in an online handbook might have direct links to the glossary meanings. A single click on an unfamiliar term directs the user to the appropriate entry in the glossary. The meaning of every further unfamiliar word that is encountered there can also be found, making it simple for the user to go back to where they were in the manual. Because these links react to mouse clicks, their locations are referred to as hot-spots. Hotspots can also be incorporated into maps, images and diagrams to help the user concentrate on the areas that catch his interest.

Rich Content

In addition to text and static images, diagrams and photographs, hypertext systems can also contain more dynamic content including audio and video clips, animation and even entire computer programs.

Animation

The process of giving images motion so they can move, change and evolve over time is called animation. A clock is a basic illustration of animation in an interface. Some digital clocks can mimic Salvador Dali's style of bending and warping one numeral into the next,

(c) Amity University Online

Notes

while others can only flicker by the seconds. The hour and minute hands on analogue clocks move and a second hand that circles the clock face is optional. Many interface configurations include this kind of desktop item and the extra processing time needed to generate these effects is no longer a significant consideration.

Cursor animation is another prominent usage for animation in modern windowing systems. Many interfaces now employ the conventional 16 × 16 bitmap that makes up the cursor to indicate more complex information, rather than just keeping a simple pointer on the screen at all times. While there are many various static cursors available, animation goes one step further by incorporating motion into the visuals. Usually, this is done to let the user know that a process is underway and to reassure them that something is truly happening. Animating the cursor is a clean and simple technique to convey the needed information because it eliminates the need to print messages to a window. On the Macintosh, work-in-progress is denoted by a spinning disc or a watch symbol with rotating hands. In one method, strenuous effort is represented by a stick figure who appears to be lifting weights, while in another, an hourglass is slowly lowering.

Non-cursor process indicators frequently resemble a stick figure going back and forth or a pop-up box with a sliding slider. These are necessary to avoid inaction from being mistaken for a system malfunction and to provide the user with a sense of progress. Moreover, animated icons can be used to display system status; Netscape Navigator, for instance, uses shooting stars to convey that it is downloading data. Moreover, animated icons can be utilised to clarify an icon's meaning. This is particularly helpful for icons that stand in for actions that static graphics can't adequately convey. To prevent distraction, though, this kind of motion needs to be utilised sparingly!

Animation is a powerful tool for displaying changes in data sets. It can be used to demonstrate dramatic changes, such as those caused by abrupt discontinuities in otherwise regular motion, or steady variations, such as those represented by rippling three-dimensional coloured surfaces.

Another application of animation is cartoon-like, in which the user can employ animated objects to carry out certain tasks. An animated assistance system, for instance, might have a character that interacts with the user and leads him through several help stages before being dismissed. Such a guide can be given a distinct personality, which makes the user experience less impersonal and more engaging. As a result, the user becomes more engaged with the system and is eager to learn more about it.

Video and Audio

There are compelling reasons, in a world dominated by media, to incorporate audio and video content into hypertext systems for reference, amusement, or instruction. Although creating audio and video content is costly and time-consuming, home PC systems are increasingly coming equipped with video and audio editing as basic features.

For instance, the iMac comes with a suite for burning DVDs and editing video. When paired with digital video cameras, these enable the creation of audiovisual content for a wider audience. Moreover, multimedia content can be easily distributed by being embedded in web pages thanks to standard formats like QuickTime. Of course, producing a high-quality film involves a lot of experience, but so does writing!

The memory needs for music and video remain maybe the largest issue. As memory capacities rise, this is getting more tolerable, but it will likely remain a significant issue for some time to come, particularly for web-based content. Naturally, longer video sequences are more linear than text. This could be reasonable if the hypertext serves as a useful index

(c) Amity University Online

Notes

for the video content. For instance, a website may provide access to a collection of silent films that are available online. But the video must also be interactive if it is to be considered a fully realised component of the hypermedia experience. Hotspots can be included in video images in the same way that they can be in graphics; however, if a gymnast is in the middle of an activity on the asymmetric bars, it might be challenging to click on her to view details about her most recent performance. This is accomplished with interactive television by offering fewer alternatives that are context-sensitive or by allowing pictures to be frozen to facilitate investigation. The ideal way to handle this interplay between video and other content might be seen on gaming stations, where game play is smoothly woven with live action or pre-recorded high-quality generated visuals.

Similar to "clips" in a paper, audio content can also be used in an interactive tour to Beethoven's compositions. Since the majority of hypertext content is visual, audio can also be utilised as "voice-over" for static pictures or animations, background music, or sound effects to set the mood.

Sound samples, or digital recordings of instruments, voices, or other sounds, can be used to store audio content. It can be more difficult to compress sound than video because sounds are by nature continually changing and we are frequently more sensitive to poor sound than poor video images. To accomplish such great compression, MP3, which is widely used for downloaded music from the internet, uses an understanding of the auditory system.

When two sounds occur simultaneously, for instance and one of them is significantly louder than the other, the quieter sound might be played with reduced fidelity because the louder sound obscures its nuances. MIDI files are essentially a digital version of sheet music that are used for pure instrumental music. They simply specify which instrument needs to be played along with the duration, pitch and loudness of each note.

Naturally, any sound in the interface has the potential to be bothersome, particularly in open-plan work spaces. For this reason, audio should be used sparingly and should be simple to muffle. The current author's wife is having trouble turning off a bothersome musical accompaniment to a webpage she is visiting while these words are being typed!

4.2.6 World Wide Web

Users can read and write on the World Wide Web, or "Web," using computers that are linked to the Internet. Although the Web and e-mail are both services that run over the Internet, the phrase is frequently used incorrectly to refer to the Internet itself. The Internet's history predates the World Wide Web by a considerable margin. All that is the internet is a network of computers connected by some kind of data connection, be it a high-bandwidth optical link or a sluggish phone line and modem.

Common data transmission protocols (TCP/IP) and addressing systems (IP addresses and domain names) are used by all internet-connected machines to communicate with one another. In theory, this allows anyone to read anything from anywhere as long as it complies with the protocol. The web expands on this by utilising a global naming scheme (uniform resource locators, or URLs), a standard markup syntax (like HTML) for information page layout and its own layer of network protocol (http).

Text, colour photos, videos, sound and—most importantly—hypertext links to other web sites can all be found on web pages. Therefore, anyone with access to a computer linked to the internet can publish hypermedia documents.

Web Servers and Web Clients

The web is dispersed, as contrast to a traditional PC program that runs and is viewed

(c) Amity University Online

Notes

on a single machine. Its various components operate on several computers, frequently located in other nations. Of course, the internet, a massive worldwide computer network, connects them.

The web servers that house the pages might be found in specialised data centres or on the premises of the business. A site's webmaster can upload pages to the server from anywhere in the world because they are networked. For instance, the data centre where the webpages for www.hcibook.com are kept is located thousands of kilometres away from any of the writers' residences!

Because it requests the documents from the servers, the computer operating your web browser is referred to as a client. Your web browser determines the complete URL of the page it needs when you click on a link; for example, "http://www.hcibook.com/e3/authors.html." This is divided into sections. The first section is the protocol "http," which specifies how it communicates with the server (ftp is an additional option). The host name, or the name of the web server hosting the requested page, is the second portion, "www.hcibook.com." The specific file on the website is provided by the final portion, "/e3/authors.html." Next, the browser connects to the necessary web server (in this case, "www.hcibook.com") and uses the HTTP protocol to format a message to the web server. The web server locates the requested HTML file (or image, or other file type) and sends it back to the browser, which displays it to you.

The identical procedure is carried out for each image on the page if it has any and for each frame on the page if it is framed.

Static Web Content

The term "static web content" describes web pages or pieces that don't change and always display the same information to users. Static content is unchanging unless it is changed manually, in contrast to dynamic content, which is subject to change in response to user interactions or other factors. The following are salient features and illustrations of static web content:

Characteristics of Static Web Content:

- ixed Information: Static web content is made up of fixed data that doesn't change for every user or user session.
- No User Personalisation: Content is not customised or personalised according to the interests or interactions of any particular user.
- Predefined Layout: Static online content has a preset layout and structure that doesn't change over time.
- Fast Loading: Because the server only provides the specified files without processing or creating content on the fly, static content typically loads rapidly.
- Images, CSS stylesheets, HTML pages and other items that don't need server-side processing are examples of common static content types.

Examples of Static Web Content:

- Brochure Websites: Simple websites are usually static and are used for informational reasons or to highlight a business's goods and services. Unless it is manually updated by the owner of the website, the content stays unchanged.
- Personal Blogs: A lot of personal blogs are made up of static content, meaning that the pages and blog posts are prewritten and don't alter in response to user interactions.

(c) Amity University Online

Notes

- Online Portfolios: To display a person's artwork, photography, or writing, static websites are frequently utilised as online portfolios.
- Documentation Websites: Static content is frequently used on documentation sites to facilitate quick access to information and easy navigation. These sites may include information about programming languages, products, or services.
- Landing Pages: Marketing-related landing pages are frequently static, particularly those that have a predetermined campaign theme or promotional message.
- News stories (Without User Comments): While some news websites contain interactive parts, certain news stories may be static, particularly if they don't include user comments or other interactive elements.
- Educational Resources: Syllabi, general information and course descriptions are examples of the types of educational resources that frequently use static online material.

Static web content is simple and quick to load, whereas dynamic content offers more interactive and customised user experiences. In order to balance content and user engagement, websites frequently incorporate both static and dynamic features.

Dynamic Web Content

Web pages or features that can alter dynamically in response to user input, user preferences, or real-time data are referred to as dynamic web content. A more dynamic and customised user experience is offered by dynamic content, as opposed to static material, which stays the same for every user. The following are essential traits and illustrations of dynamic online content:

Characteristics of Adaptive Online Content:

1. Updates in real time: Real-time adjustments to dynamic content can be made in response to user input, new data, or external circumstances.
2. User Communication: Changes in dynamic content can be brought about by user interaction, such as form submissions, clicks, or other user activities.
3. Customisation: Content on dynamic web sites can be customised based on user profiles, preferences, or previous exchanges.
4. Integration of Databases: In order to retrieve and provide the most recent data, dynamic content frequently requires interaction with databases or other APIs.
5. Adaptable to User Input: Based on user behaviour, preferences, or the circumstances of the user's session, content may alter dynamically.
6. Adaptive Structures: Dynamic content includes forms that, in response to user input, update or validate data in real-time.
7. Instant Messaging: Notifications, live updates and chat apps are examples of dynamic content that facilitates real-time conversation.

Instances of Interactive Web Content:

1. Feeds on Social Media: Users' feeds on social media platforms show dynamic content that is continuously updated in response to new posts, comments and user interactions.
2. Online shopping Suggestions: Dynamic content is frequently used by e-commerce websites to make product recommendations based on a user's browsing history, preferences, or purchasing patterns.

© Amity University Online

Notes

3. Maps that are Interactive: Dynamic maps let users engage with the map by panning, zooming and presenting real-time information (like traffic conditions).
4. Dashboards in real time: Dynamic dashboards are those that show real-time analytics or statistics about things like stock prices, website traffic, or system performance.
5. Online Resources for Collaboration: Dynamic content is a component of real-time collaboration tools like shared whiteboards and document editing.
6. Platforms for user-generated content: Dynamic content can be seen on websites such as wikis and forums where people can dynamically add, edit and alter content.
7. Real-time streaming: Websites that stream video content live, such webinars or live events, give users dynamic experiences in real time.
8. Tailored News Streams: Dynamic features are used in news feeds by websites that personalise material according to user interests, preferences, or location.

In comparison to static material, dynamic online content improves user engagement, personalisation and responsiveness while offering a more customised and engaging experience. It frequently combines data storage and retrieval techniques, client-side scripting (JavaScript) and server-side scripting.

4.3 Advanced HCI Technologies

Advanced Human-Computer Interaction (HCI) technologies encompass a range of innovative approaches to enhance the interaction between humans and digital systems. These technologies aim to create more intuitive, immersive and efficient interactions, with ongoing advancements shaping the future of HCI.

4.3.1 Augmented and Virtual Reality

Virtual Reality

Virtual reality (VR) is the term used to describe the computer-generated simulation of a world, or a portion of it, in which the user is fully submerged. While focusing mostly on the visual senses, it embodies the state of the art in multimedia systems. Virtual reality (VR) enables users to experience scenarios that are too risky or costly to enter “in the flesh.” Users can investigate the real world with hidden characteristics revealed and at a different scale. On the other hand, the generated virtual environments could be wholly artificial; authentic in their own right, but only an expression of electronic frameworks.

The phrase “virtual reality” usually conjures up an image of a person clutching irrationally into empty space while wearing a helmet or goggles. The user navigates a replicated countryside while alone in his virtual world, picking up artefacts along the way. This VR experience is totally immersive.

A headset is typically necessary in a VR setup since the user must “see” a new environment. In order to provide a 3D view, the headset is often a huge, relatively bulky piece of head-mounted equipment with separate screens for each eye. Nonetheless, more compact and lightweight VR goggles are already on the market and could soon be replaced by somewhat larger glasses.

Large quantities of processing power are needed to create and render realistic visuals in real time and the resources needed to achieve complete realism are rarely available or might not even exist yet. This indicates that the user’s environment is often “blocky,” with little variation in texture and dull lighting. This greatly simplifies and raises the feasible level of the computations. A dataglove that records gestures is commonly utilised as an input

(c) Amity University Online

Notes

method for virtual reality applications. It is possible to integrate feedback into the glove so that it feels resistant when a user grasps a virtual object. Additionally, speech recognition technologies can be included and audio feedback is generally used in one way or another. On the opposite end of the spectrum, there is a full-body dataglove that can be purchased in addition to basic stereo headphones for virtual reality!

Immersive VR

Environments that resemble our real world can be created with virtual reality. Architects have always shown their clients what a building will look like using models and sketches. They may now utilise virtual reality (VR) to give clients a virtual tour of the building, allowing them to fly over it from above, view it from the streets outside, enter through the doors and navigate the hallways. Gardens and kitchens are planned using similar methods.

But there are also a lot of things that are too small or too big for humans to perceive, or that are invisible to the unaided sight (heat, magnetism, gravity). VR technology is used by scientific and data visualisation tools to reveal and investigate these traits.

In the realm of protein chemistry, for instance, individual molecules are naturally too tiny to be seen without the use of an electron microscope.

Made up of folded, coiled chains of smaller molecules called amino acids, proteins are intricate molecules. The specific behaviour and function of the protein are determined by the specific acids involved, their position in the chain and the type of folding that occurs. Scientists have been using virtual reality (VR) techniques to attempt and comprehend the behaviour of certain proteins in order to learn more about these essential components of life.

Hubbard's work has resulted in a VR system that uses datagloves and a headset. After donning the headset, the user is thrust into a three-dimensional environment where the protein appears as a ball of hairy yarn and atomic dimensions translate into tangible distances. To better understand the molecule's intricate structure, the user can reach out into space with the dataglove and grasp hold of it, twisting it this way and that. As the intricate chemistry is formed, the chain of amino acids may be traced from one end to the other, wrapping around and nearly back on itself. Scientists now have a better understanding of the structure and function of some extremely complicated proteins.

For example, a certain protein interacts with certain enzymes, but the reactive site is buried deep within the coiled helix. When the enzyme gets close, the protein opens up what is essentially a molecular trapdoor, allowing the enzyme to dock with the reactive site, according to data gathered during reactions and utilised in the system. When observed in animated form, the reaction's nature becomes evident, but the issue becomes far less clear in the absence of this visual aid. Primitive colour and shading are employed in the VR setup to give the visuals depth and dimension, but efforts are ongoing to create effective algorithms on specialised processors to enable more detailed imaging.

Augmented Reality

Electronic pictures are projected over the physical world in augmented reality systems, fusing virtual and real worlds. One example of this would be the head-up displays used in many cars and aircraft, however the blend between virtuality and reality is somewhat faint because the data in these displays is usually unconnected to the objects seen through them.

You can get a stronger sensation of closeness using semi-transparent goggles. While moving about in the actual world and seeing genuine items, computer graphics are superimposed over the real ones by reflecting off the glass's inside. Once more, technology can be used to display irrelevant content. For instance, certain wearable computers let

(c) Amity University Online

Notes

users read their email while on the go. But when the projected image somehow connects to or alludes to the thing it covers, that's when two worlds truly collide. One experimental system, for instance, allows users to pick up and toss virtual balls. The simulated ball bounces off the actual wall when it "hits" it. There is an actual staircase that the balls can bounce down.

Ensuring that the virtual and physical worlds are properly synchronised is a challenging task for these kinds of systems, a challenge known as registration. The virtual ball would appear to go through the wall and then bounce back from the other side if it was not correctly registered, or it would bounce just short of the actual wall! The majority of trials have taken place in highly regulated settings with known object placements and devices that track the user's position and gaze direction. In practical applications, image processing techniques are frequently required. A digital camera records what the user sees, identifies each object using image recognition and then creates the relevant electronic overlay image based on this information.

This issue is resolved by one type of augmented reality technology, which projects graphics onto the actual environment. Video pictures might be projected, for instance, onto a paper desktop. The system may be calibrated and registration ensured as long as the desktop and camera remain fixed.

4.3.2 Natural Language Processing

At first glance, using natural language to communicate with machines may seem like the most appealing option. People who are lost in a maze of menus or unable to recall commands may yearn for a computer that can comprehend instructions stated in simple language! Research and interest in natural language understanding, both for spoken and written input, are high. Unfortunately, natural language is particularly difficult for a machine to understand because of its ambiguity. There are several degrees of ambiguity in language. First, a phrase's syntax, or structure, might not be obvious. If the sentence is presented to us.

The boy hit the dog with the stick

It's unclear to us if the dog is grasping the stick when the youngster hits it or if the boy is hitting the dog with it.

There may be ambiguity in a sentence's meaning even when its structure is evident from the words chosen. A sports pitch, a throw, a waterproof material, or even, informally, a region can all be referred to as "pitch." To filter through these ambiguities, we frequently rely on the context and our general knowledge. It is challenging to give the machine this information. The addition of ambiguity through the use of pronouns and relative words further complicates issues.

It appears doubtful that a general natural language interface will be accessible for some time given these issues. Still, it is possible to construct systems that comprehend limited parts of a language. The system can be given enough information in a known and restricted region to distinguish between terms. When utilising restricted natural language in interfaces, it's critical that users understand the system's limitations and don't anticipate too much comprehension.

While there is some success in using natural language in certain sectors, it is questionable if this truly qualifies as natural language. The user may experience frustration if they have high expectations for the computer and yet need to discover which language it can understand. On the other hand, the use of a broad natural language interface is likewise unclear. Due to language's inherent vagueness and imprecision, it is flexible

(c) Amity University Online

and encourages innovation in expression. On the other hand, computers need exact instructions. If we were allowed to go wild, could we articulate our needs well enough to ensure a certain response? And even if we were successful, would the language we employed end up being a limited subset of natural language?

4.3.3 Biometric Interfaces

In Human-Computer Interaction (HCI), biometric interfaces represent a paradigm shift in the way people engage with digital systems by using distinct behavioural and physiological characteristics for authentication and identification. These interfaces are made to improve user interactions in a variety of applications by making them more secure, convenient and organic. One prominent aspect is fingerprint recognition, which uses each individual's unique fingerprint patterns to provide accurate and secure identification. Another well-known biometric interface is facial recognition, which uses complex algorithms to evaluate and authenticate people based on their facial traits. Iris and retinal scanning accuracy and uniqueness as well as speech recognition, which uses a person's unique voice features to identify them, provide an additional degree of protection.

Applications for these biometric interfaces can be found in a variety of industries, including mobile devices, financial transactions, access control and authentication and even healthcare, where biometric data is used for user identification and monitoring. The broad popularity of biometric interfaces can be attributed to their ease of use and convenience, especially when it comes to complementing or replacing conventional authentication methods like passwords or PINs. Emerging trends like continuous authentication, which guarantees users stay authenticated during interactions and the investigation of multimodal biometrics, which combines multiple identification techniques for increased accuracy, are indicative of the interfaces' ongoing progress.

Although biometric interfaces have great potential, their use is fraught with difficulties and moral dilemmas. Because biometric data is sensitive, privacy issues are critical, necessitating strong security measures and transparent user communication surrounding data utilisation. Biometric systems must be accurate and reliable since mistakes might result in false positives or negatives, which undermine user confidence and system efficacy. Furthermore, the possibility of security flaws like spoofing efforts emphasises the necessity for continuous improvements in biometric technology.

In the future, a growing trend that will enable ongoing authentication and surveillance is the incorporation of biometric sensors into wearable technology. More attention is being paid to ethical issues, which highlight the value of justice and deal with biases in biometric algorithms. The ethical criteria that protect user privacy and preserve the integrity of these cutting-edge interaction techniques are established together with a more efficient and safe user experience through the responsible development and implementation of biometric interfaces in HCI. With continued development, biometric technology has the potential to significantly alter the HCI landscape and provide users with a safe and seamless digital experience in a variety of contexts.

Summary

- In this module, we have discussed a wide selection of cognitive models of the users of interactive systems. Cognitive models attempt to represent the users as they interact with a system, modeling aspects of their understanding, knowledge, intentions or processing. We divided cognitive models into three categories.
- The first described the hierarchical structuring of the user's task and goal structures. The GOMS model and CCT were examples of cognitive models in this category.

© Amity University Online

Notes

The second category was concerned with linguistic and grammatical models, which emphasized the user's understanding of the user–system dialog. BNF and TAG were described as examples in this category.

- Most of these cognitive models have focussed on the execution activity of the user, neglecting his perceptive ability and how that might affect less planned and natural interaction strategies. The third category of cognitive models was based on the more solid understanding of the human motor system, applicable in situations where the user does no planning of behaviour and executes actions automatically.
- KLM was used to provide rough measures of user performance in terms of execution times for basic sequences of actions. Buxton's three-state model for pointing devices allowed for a finer distinction between execution times than with KLM. We concludethat with a discussion of cognitive architectures, the assumptions of which form the foundation for any cognitive models.
- In addition to the basic architectural distinction between long- and shortterm memory, we discussed two other cognitive architectures – the problem space model and ICS – which apply different assumptions to the analysis of interactive systems.
- We saw, by example, how the dialog of a user interface can become unmanageable and impossible to analyse, if we do not have a separate description. We have looked at two main classes of description: diagrammatic and textual. Diagrammatic notations included various forms of state transition network, Petri nets, Harel state charts, flow charts and JSD diagrams. Under textual notations, we considered grammars (regular expressions and BNF), production rules (event based and state based) and CSP.
- We found that some very different notations were essentially equivalent: for example, JSD diagrams and regular expressions can describe exactly the same dialogs. However, we also saw that there were major differences in expressive power between different formalisms. For example, BNF grammars are able to deal with recursively nested dialogs and are thus more powerful than regular expressions or state transition networks.
- One important difference in expressive power is whether or not the formalism can handle concurrent dialogs. We found that some notations favour sequential dialogs (for example, STNs) and some concurrent (for example, production rules), but few could handle both (the exceptions being Harel state charts and CSP). However, increased expressive power is not always desirable as it may mean a more complex and less easily understood description. The important point is to match the notation to the form of the desired dialog. Another important distinction was between state-oriented and event-oriented descriptions.
- We found that presentation and lexical issues are not (as is commonly supposed) a separable issue from dialog design. We need to consider the visibility of modes and the style of interaction. Even the layout of keys and menu items can influence the likelihood of making serious errors.

Glossary

- HCI: Human Computer Interaction
- MHP: Model Human Processor
- IxD: Interaction Design
- GOMS: Goals, Operators, Methods and Selection
- ICS: Interacting Cognitive Subsystems
- VIS: Visual Subsystem
- UML:Unified Modeling Language

(c) Amity University Online

Notes

- SDL: Specification and Description Language
- TLA: Temporal Logic of Actions
- VR: Virtual Reality

Check Your Understanding

1. In the context of Interacting Cognitive Subsystems, what does the term "perceptual processor" primarily focus on?
   a. Decision-making b. Sensory input and interpretation
   c. Memory retrieval d. Goal setting
2. Which cognitive subsystem is responsible for higher-level cognitive functions such as planning and problem-solving?
   a. Perceptual processor b. Motor processor
   c. Cognitive processor d. Emotional processor
3. According to the Interacting Cognitive Subsystems model, what is the primary role of the "motor processor"?
   a. Processing sensory information b. Initiating physical actions
   c. Making decisions d. Storing long-term memories
4. What is the key characteristic of the "emotional processor" in the Interacting Cognitive Subsystems model?
   a. Logical reasoning b. Handling emotional responses
   c. Motor coordination d. Language processing
5. How do Interacting Cognitive Subsystems contribute to understanding human cognition in interactive systems?
   a. By isolating cognitive processes for analysis
   b. By emphasising only emotional responses
   c. By excluding motor coordination
   d. By focusing solely on perceptual processing

Exercise

1. Define Cognitive Model.
2. What do you mean by Communication and Collaboration Model
3. Explain Task Analysis and Decomposition
4. Explain Dialog Notation and Semantics.
5. What is Hypertext and Multimedia?

Learning Activities

1. Choose a website of your preference and analyse its URL. Identify and describe the different components of the URL, such as the protocol, domain, path and any parameters. Explain how these components contribute to the overall functionality of the website.
2. Open your favourite web browser and access its developer tools. Explore the various features available, such as the Elements tab, Network tab and Console. Perform a simple inspection of a web page, modify an HTML element andanalyse the network activity. Share your observations and insights gained from using the browser's developer tools.

© Amity University Online

Notes

Module - V: Research Frameworks

Learning Objectives

At the end of this module, you will be able to:

- Define Speech Interfaces
- Understand Information Visualisation
- Define Ubiquitous Computing
- Analyse Ethical Issues in HCI
- Understand Emerging Trends

Introduction

Strong research is built on a framework that connects disparate themes. It supports the exploration of subtopics, idea generation and new research production.

A research framework makes it easier to recognise and follow the steps necessary to accomplish the desired aim and purpose. The robust literature review and several case studies form the foundation of the research framework.

5.1 Overview of Research Framework

The program that enables communication between voice-activated devices and people is called a speech interface. Understanding 3D displays is particularly crucial because they are utilised in data visualisation. Computing used to be fixed in terms of location, device and format, but thanks to ubiquitous computing, it is now feasible to compute in any format, at any time and from any location.

5.1.1 Speech Interfaces

Numerous academics and dreamers have been drawn to the idea of interacting with computers and having them talk for themselves. The 1968 science fiction novel and film 2001: A Space Odyssey by Arthur C. Clarke, which features the HAL 9000 computer, set a benchmark for computer performance for science fiction and certain sophisticated developers. Compared to dreams, life can occasionally be more startling and intricate. Speech generation, recognition and processing hardware designers have come a long way, yet current achievements pale in comparison to science-fictional visions. Even science-fiction authors have changed their plots, as evidenced by Minority Report and Star Trek's Voyager, which use more visual displays than verbal interaction.

These days, the idea of machines conversing with people informally sounds more like an ignorant dream than a desirable reality. Rather, successful voice interaction programs are those that meet users' expectations for quick tasks that require little cognitive burden and few mistakes. While recognition algorithms and technological issues are being resolved, designers are grudgingly realising that voice directing requires more working memory from users than hand-eye coordination and may thus cause more disruptions to users during job completion. While hand-eye coordination is processed in a different area of the brain, allowing for a higher level of parallel processing, speech demands the use of restricted resources. While hand-eye coordination can progress in tandem with problem-solving and planning, these tasks are more challenging to do when speaking.

Unfortunately, the problem of voice recognition remains unabated due to variances in user speech performance and background noise. In contrast, due to the widespread

(c) Amity University Online

Notes

usage of telephones and the small size of speech chips, speech store and forward as well as speech generation are remarkably predictable, inexpensive and widely accessible. But there are three challenges that each designer has to face when it comes to voice output: speech output moves more slowly than visual displays, speech is ephemeral and it is more difficult to scan or search. Speech store and forward works well because voice communications, museum tours and educational settings benefit from the emotional depth and natural prosody of human speech.

While the advantages for those with certain physical limitations are evident, most users of personal or business computers are not hurrying to add speech input and output devices. The bicycle of user interface design is speech: It plays a crucial role and is a lot of fun to use, but it can only support a small weight. Advocacy for sobriety is aware that replacing the car's graphical user interfaces will be difficult.

Speech fans can rejoice in the triumph of digital circuitry in telephony, which has enhanced voice quality and network capacity. In wealthy nations, cellular phones have been a great success and they frequently quickly provide phone service to developing nations. VoiceoverIP (Internet Protocol), sometimes known as internet telephony, is becoming more and more popular. It provides many consumers with low-cost long-distance service at a reduced sound quality. One of the most appealing aspects of human-to-human communication is the immediacy and emotional effect of a phone conversation.

Opportunities

- When users have vision impairments
- When the speaker's hands are busy
- When mobility is required
- When the speaker's eyes are occupied
- When harsh or cramped conditions preclude use of a keyboard

Technologies

- Speech store and forward
- Discrete-word recognition
- Continuous-speech recognition
- Voice information systems
- Speech generation

Obstacles to speech recognition

- Increased cognitive load compared to pointing
- Interference from noisy environments
- Unstable recognition across changing users, environments, and time

Obstacles to speech output

- Slow pace of speech output when compared to visual displays
- Ephemeral nature of speech
- Difficulty in scanning/searching

Figure 76: Speech Systems

There are at least five types of speech and audio technologies available to designers of systems for human-computer interaction: voice information systems, speech production, continuous speech recognition, discrete word recognition and nonspeech auditory interfaces. These elements can be combined in a variety of inventive ways, ranging from straightforward systems that only record or play back messages to intricate interactions

© Amity University Online

Notes

that can produce speech feedback, take voice instructions, sonify scientific data and enable annotation and editing of recorded speech.

For designers working in this industry, a fuller grasp of how the brain processes sounds might be beneficial. Why does listening to radio news reporting halt creative thought processes while listening to Mozart symphonies fosters them? Is listening to a radio news program requiring linguistic processing that is disturbing, whereas listening to Mozart in the background is somehow energising? Of all, it would take all of a musicologist's mental energy to listen to Mozart with any seriousness. Are there any speech or sound-related techniques or attention-getting strategies that could be less disruptive or even helpful for visual design, symbolic processing, or analytical reasoning? Is sound a better feature for drawing programs than word processors?

Discrete-word Recognition

Discrete word recognition systems can identify individual words spoken by a particular speaker with a reliability of 90 to 98% for vocabularies of 100 to 1000 words or more. Many systems include speaker-dependent training, where users repeat the entire vocabulary once or twice. Compared to speaker-independent systems, such training produces higher accuracy; yet, the removal of training broadens the range of possible commercial uses. Both situations benefit from quiet surroundings, head-mounted microphones and deliberate vocabulary selection in terms of recognition rates.

People who are ill, bedridden, or paralysed have been able to expand their horizons thanks to applications designed for individuals with disabilities. They are capable of operating wheelchairs, machinery and personal computers for a range of purposes. Applications for the elderly or those with cognitive or emotional disabilities may also help people become more independent, regain confidence in their talents and retrieve lost skills. Unfortunately, those advantages will be severely limited by speech problems, which are frequently linked to other disabilities.

In recent years, there has been a boom in telephone-based information services offering weather, sports, stock market and movie information. Even on cell phones, phone companies provide voice-dialing services that let customers just say "Call Mom" to get connected. Even in autos, where hands-free operation is a huge advantage, acceptance appears to be sluggish due to issues with training for numerous users in a home, user reluctance to utilise speech commands and unreliable recognition.

Successful and widely used phone-based solutions include number recognition, yes/no response processing and voice menu selection. But orders that are complete sentences, such "Book two seats on the first flight from New York to Washington tomorrow," are only now making the transition from research challenge to practical application. There have been a number of suggestions for organised speech, but interaction remains challenging even if users pick up a basic language, like <action><object><qualifier> ("Reserve seats two" "Depart New York tomorrow").

Speech recognition has been tried in a variety of advanced development settings, including offices, training labs, medical operating rooms and military aircraft. The findings show that even speaker-dependent training systems have issues with identification rates when background noise changes, users are sick or under stress and vocabulary phrases that are similar (e.g., dime/time or Houston! Austin).

When at least one of these circumstances is met, other applications have had success:

- Speaker's hands are busy.
- Mobility is required.

(c) Amity University Online

Notes

- Speaker's eyes are occupied.
- Harsh or cramped conditions preclude use of a keyboard (for example, in underwater or rescue operations).

Systems for aviation engine inspectors, who walk around the engine opening cover plates or adjusting components while wearing wireless microphones, are one example of such application. With a 35-word vocabulary, they can read serial numbers, issue orders and access past maintenance information. Developers of such difficult apps must to think about educating speakers dependently within the task setting.

Speech input is not very beneficial when a display is utilised for basic computing tasks. Research using user-controlled speech cursor movement confirms reduced cursor movement performance for tasks like button clicking and web browsing.

However, a study of users of sketching programs revealed that performance times were improved by an average of 21% when users were given the option to choose one of 19 commands by voice rather than from a palette. The benefit appears to have come from not having to make the tedious and annoying effort of continuously shifting the cursor from the diagram to the tool panel and back. Using word processing tasks that included eighteen spoken instructions, including boldface, down, italic, paste andundo, a replication of this finding was conducted. While voice instructions were generally faster than mouse pointing (mostly because of mouse acquisition time), speech users made more mistakes when performing tasks requiring a large short-term memory load. Psychologists addressed this surprising outcome by pointing out that short-term memory is also sometimes referred to as "acoustic memory." Speaking commands requires more working memory than executing the hand-eye coordination required for mouse pointing, which is processed concurrently by different brain regions.

The slower adoption of speech interfaces relative to graphical user interfaces may be explained by this phenomena; saying commands or listening interferes more with problem solving and planning than does choosing an action from a menu with a mouse. Product assessors for an IBM dictation package made note of this, noting that "for many people, thought is very closely linked to language." When typing, people can keep refining their words even as their fingers produce a less-than-perfect rendition. Users may encounter more interruptions when dictating when trying to expand on an original concept."

Playthings such as dolls and tiny robots that can converse with their owners and react to voice commands are a significant example of speech recognition technology's success. Compact microphones and speakers, along with inexpensive speech chips, have made it possible for designers to incorporate fun systems into high-volume products. In games and with similar toys, mistakes frequently add to the challenge and enjoyment. Although there have been proposals, voice-activated consumer appliances have not yet been implemented.

The goals of current research efforts are to increase the vocabularies handled to 100,000 words or more, remove the need for speaker-dependent training and improve recognition rates under challenging settings.

Discrete word speech recognition is useful for specialised applications, but it is not a general-purpose medium for interaction. Direct manipulation pointing devices, function keys and keyboards are frequently faster and the commands or actions can be easily edited by making them visible. Moreover, error correction and relevant feedback are slow and challenging to use with voice input. However, as Pausch and Leatherby's work suggests, voice and direct manipulation together may be helpful.

(c) Amity University Online

Notes

Continuous-speech recognition

While it was an attractive illusion to think that HAL could read the astronauts' lips and comprehend their spoken sentences, the truth is more grim. Continuous-speech recognition has been the subject of numerous research initiatives and the dot-com boom gave rise to broad expectations for solutions that would be profitable. Although the highly advertised products were purchased by consumers, many were disappointed by the overstated promises. Though they function, spoken dictation devices have significant issues with error rates and error remediation. Furthermore, dictation's cognitive demands disrupt sentence structure and planning, which frequently lowers the quality of materials produced in comparison to typewritten production.

The distinction between spoken words can be challenging for software designers to make since typical speech patterns tend to obfuscate these boundaries. Variable speaking rates, varying dialects, distracting background noise and shifting emotional intonation are further issues. Trade press product reviews have amusing segments featuring the slips generated by speech recognition software. The most challenging issue is, of course, connecting human-level contextual comprehension and semantic interpretation to word prediction and disambiguation. An IBM technical report with the amusing title "How to wreck a nice beach" (a parody of "How to recognise speech") effectively brought attention to this issue.

IBM's speech-dictation system, ViaVoice, is "trained" by users by having them read typical text passages (such as Treasure Island snippets) for fifteen to thirty minutes. This helps the system address some of these issues. Companies like Philips are manufacturing versions of these specialised devices in at least 22 languages. They are commercial successes designed for healthcare staff, solicitors and specific business experts. Paradoxically, because jargon-heavy technical subjects have unique language, they make suitable candidates.

Continuous-speech recognition systems allow users to write reports and dictate letters to be automatically transcribed. Keyboards and displays are typically used for review, correction and revision. When creating typical reports, users seem to do better using speech input and require practice with dictation. Entire utilisation of the limited cognitive resources of human working memory is necessary for creative writing and analytical articles. Although keyboard typing is frequently the most efficient way to complete these chores, some users still find that pen and paper is more comfortable. With practice, writers can become more proficient dictators and developers can advance voice editing techniques, error-correction techniques and system accuracy.

Automatic scanning and retrieval of specific words or subjects from radio or television programs, court proceedings, lectures, or phone calls is another benefit of continuous speech recognition systems. Even with flaws, these applications can be quite effective and helpful. Although faults in closed captioning might be annoying, most television viewers find them tolerable, therefore producing closed caption text for programs is also financially lucrative. Continuous-speech recognition can also help with the indexing of audio and video archives, as it does not require real-time performance.

Another practical element for security systems is speech recognition for identity purposes. When users are requested to utter a new phrase, the system determines who of the users who have registered is speaking. Nevertheless, there are still issues with maintaining stable performance, managing users with colds and handling noisy surroundings. Voice graphs are acceptable in legal settings and could find application in security systems.

Despite the fact that numerous businesses and research teams have made progress, the assessment below is still accurate: "Comfortable and natural communication in a

(c) Amity University Online

general setting (no constraints on what you can say and how you say it) is beyond us for now, posing a problem too difficult to solve".

Voice Information Systems

The human voice has a powerful allure as a communication tool and information source. Stored speech is frequently utilised for after-hours communications from organisations as well as telephone-based information about tourist attractions and government services. If appropriate development techniques and metrics are applied, these voice information systems, also known as Interactive Voice Response (IVR), can offer quality customer support at a reasonable cost. Users are guided by voice prompts to press keys to obtain medicine prescription refills, reserve theatre tickets or check airline aircraft departure and arrival schedules.

Speech recognition can be used to shortcut through menu trees when consumers are familiar with the names of the items they are looking for, such as a city, person, or stock name. On the other hand, consumers are irritated when lengthy speech information segments contain useless information or when menu structures get deep and complicated. Voice information systems are widely used because they enable services that would otherwise be too expensive; hiring well-trained customer-service representatives available 24 hours a day is not practical for many organisations. However, the slow pace of voice output, the ephemeral nature of speech and the difficulty of scanning/searching remain great challenges.

Popular personal voicemail systems also use voice information technologies. With the help of keypads, users can input commands to store and forward spoken messages using these telephone-based speech systems. Notifications can be received, replayed, replied to, forwarded to other users, archived, deleted and forwarded to other users. Listening times can be reduced by half by automatically eliminating silences, speeding up replays and altering frequencies while maintaining original frequency ranges.

Voicemail technology is generally well-liked by consumers, operates with reliability and is reasonably priced. The discomfort of using the 12-key phone pad for commands, the requirement to ring in to see if messages have been left and the possibility of receiving an excessive number of "junk" phone messages due to the simplicity of broadcasting a message to a large number of recipients are the main causes of issues. Some email developers think that customers could be able to view their emails from a phone by using speech recognition for user requests. Designers are still having difficulty striking the ideal mix between user control and guided instruction, though. Even if they manage to solve the three issues with voice output, there are still concerns about the application's usefulness given the three challenges with command input.

Digital methods are becoming more prevalent in audio recording and compact, portable voice note-takers are successfully capturing consumer demand. An hour's worth of voice quality notes can be randomly accessed and stored on credit card-sized devices that cost less than fifty dollars. More advanced portable devices, like the iPod from Apple, allow users to manage enormous audio databases and retrieve specific music or lecture chunks that have been recorded.

The reason audiobooks and museum audio tours are so popular is because they let the listener set their own speed while expressing the passion of the author or the curator. According to the theories of educational psychologists, learning can be made easier if many senses—sight, touch and hearing—are used. Enhancing an online help or education system with a spoken voice could help students learn more effectively. Voice annotation could make it simpler for company leaders to write thorough instructions or comments on

(c) Amity University Online

Notes

student papers, or for teachers to provide feedback on student work. However, once more, speakers' cognitive burdens, the challenge of editing voice annotations and other issues might limit adoption.

5.1.2 Information Visualisation

Scientific data and other complex information can be visualised using virtual reality and 3D displays. Whether or not 3D representations are used, animation techniques can foster a sense of engagement with the data and promote pattern creation and discovery, particularly when they are controlled by the user interactively.

Scientific and Technical Data

The amount of virtual world dimensions that match actual spatial dimensions as opposed to those that relate to more abstract factors can be used to categorise three-dimensional scientific and technological data representations.

Images where all three dimensions have some degree of physical realism are possibly the most captivating. The virtual wind tunnel is one illustration of this. A precise model of an aeroplane is built in a physical wind tunnel and exposed to winds that, when scaled suitably, match to real-world conditions. The goal is to look at air movement and pressure patterns, such as identifying the locations where turbulence arises. Since air is invisible, it is possible to reveal the concealed airflows by using polarised light, tiny bubbles released into the chamber, or ribbons connected to the aircraft's surface.

The fluid dynamics equations are used to determine air movements in the virtual wind tunnel. The engineer can then navigate around a (virtual) baton that shoots (virtual) bubbles and view the simulated aircraft with virtual reality goggles (Figure below). Areas of interest can be studied by shifting the baton to different portions of the aircraft.

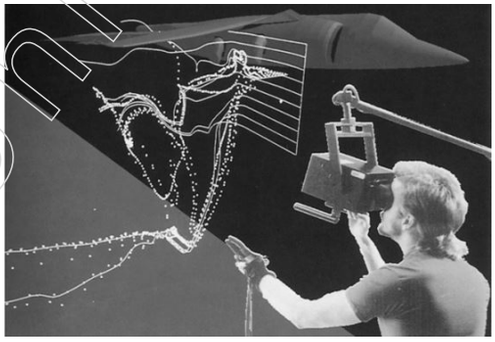


Figure 77: Virtual wind tunnel

The virtual wind tunnel can reduce costs and speed up the process of developing and assessing models, which can be highly time and money consuming. There are, nevertheless, certain instances in which the virtual wind tunnel performs better than the actual one. Engineers can study specific details by getting as near to the simulated aircraft as they desire in the virtual wind tunnel. This would require entering the actual tunnel, which would disrupt the airflow and render any measurements useless.

When a physical plane is represented by two dimensions and certain data is represented by a third dimension for each point, 3D reality advances to the next level. For instance, instead of representing the height of the terrain, a relief map can be created where the height at any given position indicates the average annual rainfall. Selecting a vantage point is a challenging task in any depiction, as it is probable that towering buildings in the forefront will obscure significant elements in the backdrop. This likely explains why static paper articles rarely feature these kinds of representations.

(c) Amity University Online

Notes

The last scenario is one in which a spatial dimension is represented by either one or none of the dimensions. For instance, the majority of spreadsheet programs include a variety of 3D graphing choices, such as the ability to use two columns for the x and y coordinates and a third column for the height. In this instance, there are no 2D or 3D patterns in the picture that match features found in the actual world. Still, we may make use of our 3D perception skills to better comprehend and value this more abstract data.

Structured Information

Since scientific data are usually numerical, mapping them onto a virtual space dimension is simple. On the other hand, information systems data sets usually consist of several distinct features and structures, such as networks, hierarchies and—above all—free text. Organisation charts and file trees are two instances of hierarchies. Hypertext structures and program flow charts are two instances of networks.

Converting the discrete structure into a similarity metric is a popular method. In a hypertext network, this may be the total number of links that must be crossed to get from one node to another; in a free text document, the similarity between two pages could be measured by the percentage of words that they share. The data points can then be mapped into two or three dimensions using a variety of methods, keeping as much of the similarity measurements as feasible (similar points are closer). Statistical multi-dimensional scaling, certain types of self-organising neural networks and simulated gravity are examples of these methods. Users can see clusters and other structures within the data set thanks to the visual mapping, even though the dimensions that result from these procedures are arbitrary.

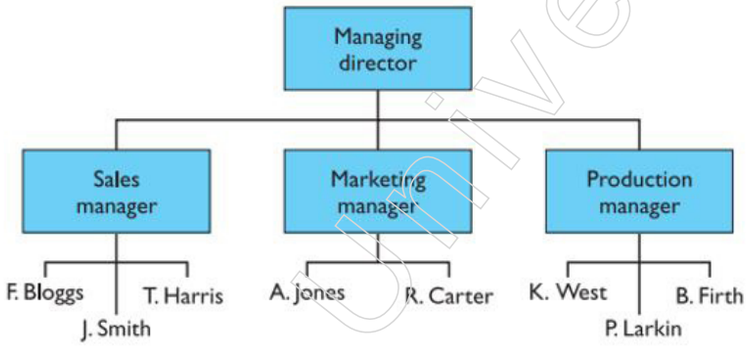


Figure 78: Two-dimensional organisation chart

Two-dimensional hierarchies and networks can be represented in usual ways. A example organisation chart is shown in the above figure. You'll see that, despite the chart's small size, the names have been somewhat shifted to free up more room. Large wall displays are actually typically required, even by relatively tiny businesses.

Flow charts and network diagrams also have space issues; these diagrams are often divided over multiple pages. Moreover, only specific types of networks—referred to as planar graphs—can be arranged in two dimensions without any lines crossing (Figure below).

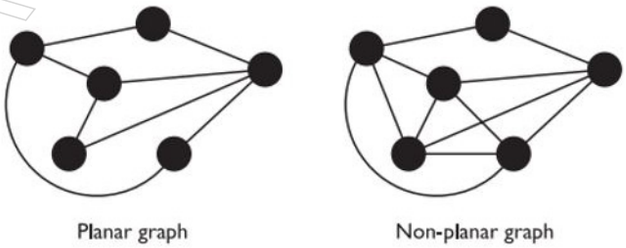


Figure 79: Two-dimensional network layout

Notes

Network and hierarchical designs can both benefit from the third dimension. Nodes in a network can be arranged in three dimensions, which reduces clutter and means that lines just pass by one another rather than crossing. Naturally, this has the drawback that nodes and lines may blend together, but these hidden nodes are still visible as long as the user can fly over or rotate the network. For hierarchies, comparable methods can be applied.

The Camtree, a feature of the Xerox PARC Information Visualizer, is depicted in the figure below. From left to right, the Camtree hierarchy is shown, with the parent node at the top and the subordinate nodes grouped in a circle around the base of the cone. Even though the tree develops in a left-to-right direction, it does so far less quickly than the matching two-dimensional representation. The mechanism rotates the cones in order to bring the selected node and its parents as close to you as possible. As a result, even though some nodes are hidden at any given time, the complete tree can be seen.

Observe also how the shadows add to the image of three dimensions and provide more details about the tree's hidden areas. Similar to the Camtree, the Conetree is another hierarchy visualisation that shows the hierarchy as it appears in the organisation chart, but in a top-down manner. Despite the striking similarities between these two representations, it is much simpler to display names for each node in the Camtree. This is due to the fact that most hierarchies have a greater breadth than depth. Using the Camtree and Conetree, hierarchies with up to 10,000 nodes have been visualised.

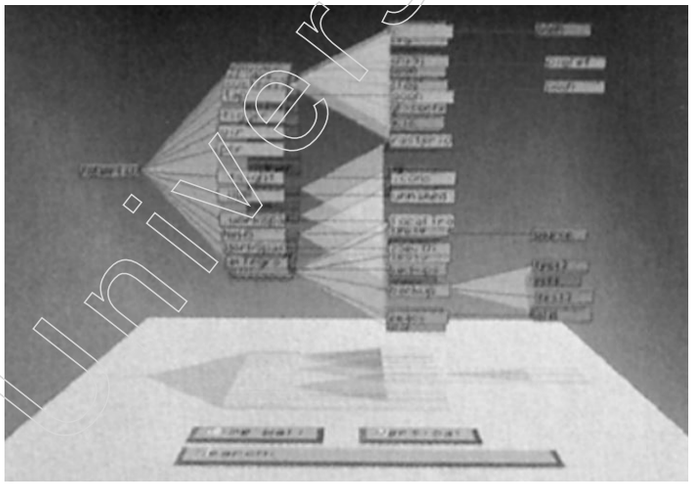


Figure 80: Camtree – visualising hierarchies

Time and Interactivity

Time and visualisation can be viewed from two perspectives. On the one hand, we would like to visualise temporal values (dates, periods, etc.) in numerous data sets. However, different kinds of data can also be visualised by using the flow of time itself. Time is frequently plotted onto a single spatial dimension in 2D graphs, such as when a company's monthly sales numbers are displayed. More than one snapshot can be used when the time-varying data are 2D images themselves. Technical manuals and comic books both employ sequential images to depict movement and changes. These images are frequently enhanced with arrows, streamlines, or blurring to provide the feeling of speed and direction.

Events that happen at irregular intervals are another kind of temporal data. This kind of data is frequently represented using timelines, in which the type of activity is represented by the second axis and time is represented by the first. On the time axis, events and periods are shown by icons or bars along the line that corresponds to their nature. A popular

(c) Amity University Online

Notes

illustration of this is the Gantt chart, which shows activity on project tasks. This kind of method can also be applied to computer visualisation; Lifelines, for instance, visualises judicial and medical information using a model akin to a Gantt chart (Figure below).

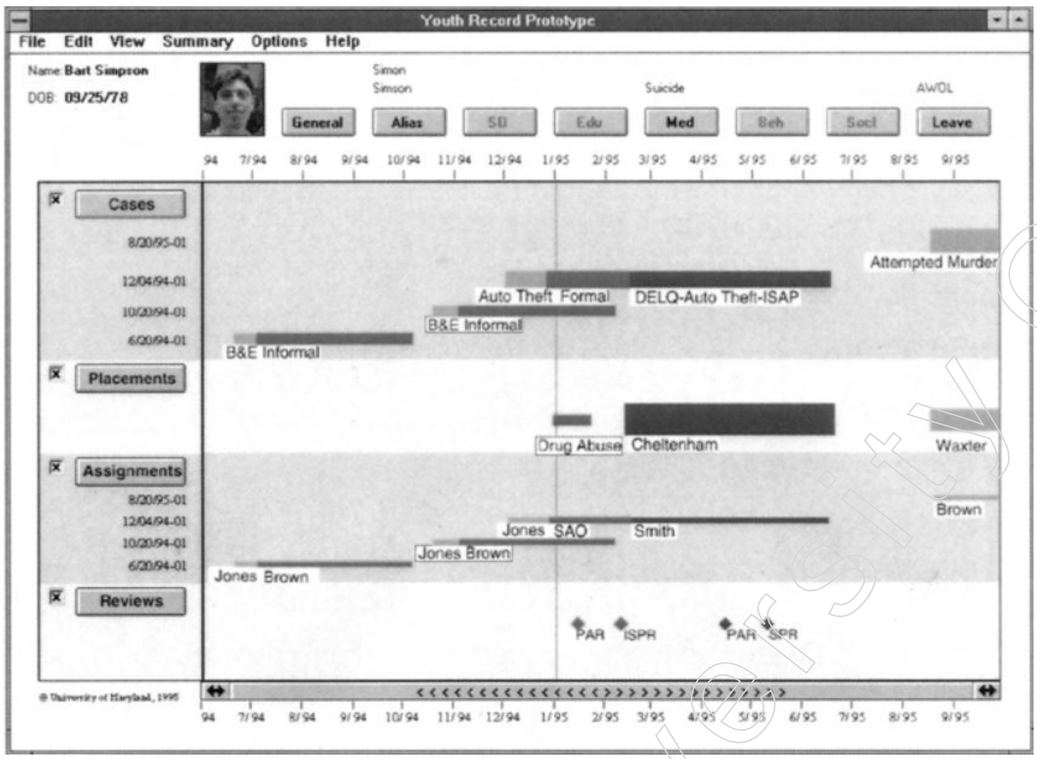


Figure 81: Lifelines – visualising personal histories

Another way to help with visualisation is through the passage of time. When time at the interface is directly mapped onto time in the data, this is the most straightforward scenario. In other words, the statistics that vary over time are only repeated. repeat rates do not have to be in real time; for example, one might repeat weekly sales data for a year at a pace of less than one minute each week. An interface for such a visualisation can be created using standard video controls.

As an alternative, the passage of time at the interface may be translated onto a spatial dimension (or other parameter). Think of a solid, immobile three-dimensional object. One can get a decent idea of the object's exterior by glancing at a few 3D produced photos. That being said, it is extremely difficult to look inside. It is possible to display multiple slices of the item at once, albeit there are clearly restrictions on the quantity and size of each. One way to address this is to map the distance across the object onto time by progressively changing the "cutting" plane and displaying subsequent cross-sections as an animation.

5.1.3 Ubiquitous Computing

According to the notion of ubiquitous computing, information processing is connected to every task or thing that is encountered. It entails integrating microprocessors with electronic devices in order to communicate information. Ubiquitous computing-powered devices are always available and fully connected.

By reducing the complexity of computing and boosting productivity while utilising computers for various daily tasks, ubiquitous computing puts an emphasis on learning.

Pervasive computing, every ware and ambient intelligence are other names for ubiquitous computing.

Notes

In software engineering and computer science, ubiquitous computing refers to the idea of making computing accessible to everyone, anywhere and at any time. Ubiquitous computing, as opposed to desktop computing, is possible with any device, anywhere and in any format.

Examples of Ubiquitous Applications

Home Automation

We've all heard of "home automation" in this day of computing. During this procedure, we link every device that can be connected to the internet or a local area network (LAN). To instruct those gadgets, we use a parent device (such as Google Home, Apple Hub, or Amazon Alexa). We provide processing power to every linked gadget in order to suggest this idea. We make use of ubiquitous computing here.

Figure 82: Ubiquitous applications

IP Camera

Consumer electronics are being networked these days. Some high-end cameras come equipped with built-in wireless networking, which they use to upload images and movies to the internet or broadcast them to a nearby display. "A digital video camera that transmits and receives image data and control data over the Internet is known as an Internet Protocol camera, or IP camera."

RFID Devices

Radio Frequency Identification is referred to as RFID. RFID tags can interact passively with RFID scanners, much like stamps. Our credit and debit cards contain RFID chips, which we use extensively to hold encrypted financial data.

Smartphones

They are the most evident and direct example because, in most cases, all it takes is downloading an app to manage other commonplace applications, such as home automation systems.

Wearables

Wearables such as smart glasses and watches are instances of ubiquitous computing. Both Fitbits and Apple Watch fall under the category of ubiquitous computing.

Self-driving Vehicles

Autonomous vehicles Excellent instances of ubiquitous computing as well. An even better example of ubiquitous computing would be an autonomous car that can recognise its authorised passenger by the proximity of a smartphone, dock and charge itself when necessary and manage payments for fast food, tolls and emergency services by interacting with infrastructure.

(c) Amity University Online

Notes

Holistic Framework for UBICOM: SMART DEI

The smart DEI framework, which forms a comprehensive foundation for ubiquitous computing, is built on three ways to study and create UbiCom systems:

1. Create architectures with UbiCom systems in mind: There are three primary design approaches for UbiCom systems: smart interaction, smart environment and smart device.
2. Five basic qualities form the basis of an internal model of the UbiCom system properties: distributed, iHCI, context-awareness, autonomy and artificial intelligence. The degree to which these five attributes are supported and interconnected determines the various subtypes of ubiquitous system design that can be created.
3. An interaction model of the UbiCom system with its surroundings. Two different forms of interaction are highlighted in addition to a traditional distributed ICT system device interaction within a virtual8 environment (C2C):
   a. between computer systems and humans as systems (HCI);
   b. (b) between computers and the physical world (CPI).

The Smart Devices, Environments and Interactions model is referred to as Smart DEI. Pronouncing it as “Smart Day” alludes to the model’s emphasis on using systems support for day-to-day tasks.

A comprehensive framework for developing various UbiCom systems based on smart devices, smart surroundings and smart interaction is represented by the Smart DEI model. These three design styles can also be integrated to enable other kinds of smart environments. For example, smart mobile devices can optimise the routing of physical assets or the computer environment from a different location by combining a knowledge of their changing physical environment location. As a fundamental feature, every smart device is networked and has access to information services and the ability to share data.

Ubiquitous System Environment Interaction

1. For every specific UbiCom system, there are three different kinds of system environments that we can identify at a high level of abstraction: other UbiCom systems that make up the ICT infrastructure, supporting services and middleware for that specific ICT system applications (virtual worlds).
2. human beings as individuals and as organisations.
3. biological and ecological systems are examples of the physical world’s systems. For UbiCom systems, the real world and the virtual (computer) environment together constitute the external environment. Take note of how distinct the design concepts and procedures seem to be in each of these three primary environments. Well-known principles of physical forces like gravity and electromagnetic regulate physical world events. Ecological models of their ecosystem govern living things in the real world. Economic and social models frequently rule living things, including humans.

Conventionally, a UbiCom system is arranged as a layered information system stack, with user information abstractions to view and interact with the information at the top, middle processing layer and bottom layer information resources. Their distribution across many ICT systems is made possible via a shared communications conduit.

For UbiCom systems, there are three primary types of environments:

a) the infrastructure of other ICT systems.

© Amity University Online

Notes

b) the physical world environment

c) the human environment.

Interaction between humans and ICT systems, HCI between ICT systems and the physical world, CPI between ICT systems and C2C or CCI are some of the fundamental forms of system environment interaction that take place. Furthermore, interactions—also known as social interactions—can take place between non-ICT systems, such as between various physical elements and between people (H2H or HHI). All of these kinds of interactions are present.

Architectural Design for UbiCom Systems: Smart DEI Model, Smart Devices and Services

For ubiquitous ICT systems, three fundamental architectural design patterns are suggested: smart devices, smart environments and smart interactions. In this context, "smart" merely refers to the entity's ability to function as an active, digital, networked, partially autonomous, reconfigurable and locally controlled resource (e.g., energy, data storage, etc.). Therefore, it is possible for these three primary system design types to have sub-systems and lower-level components that are also capable of being categorised as smart; for example, a smart environment device could have smart sensors and a smart controller, among other things.

For ubiquitous ICT systems, three fundamental architectural design patterns are suggested: smart devices, smart environments and smart interactions. In this context, "smart" merely refers to the entity's ability to function as an active, digital, networked, partially autonomous, reconfigurable and locally controlled resource (e.g., energy, data storage, etc.). Therefore, it is possible for these three primary system design types to have sub-systems and lower-level components that are also capable of being categorised as smart; for example, a smart environment device could have smart sensors and a smart controller, among other things.

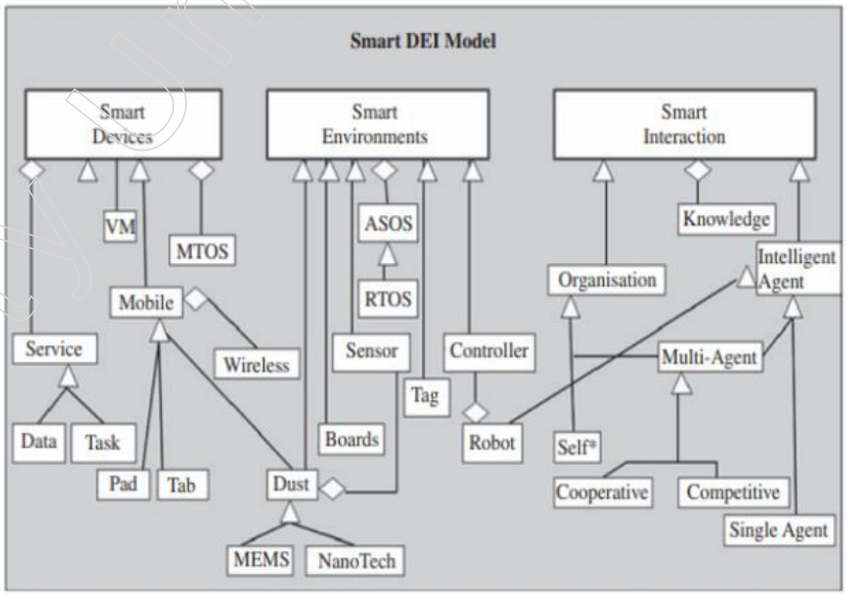


Figure 83: Smart DEI Model

Smart Devices

Smart devices, such as mobile phones and personal computers, are typically multifunctional ICT devices that serve as a single portal to a variety of widely used multiple

Notes

application services that may be hosted locally on the device or remotely on servers. Smart gadgets come in various shapes and sizes. Smart gadgets are typically individualised, belonging to a certain user or owner. The user interface and locus of control are housed in the smart device in the model.

Mobility, dynamic service discovery and intermittent resource access (concurrency, updating, etc.) are the three primary features of smart devices. Because multifunctional devices make multifunctions easier to access and facilitate their run-time interoperability, multifunctional devices are frequently developed with this in mind. Reduced system openness to support more dynamic, flexible run-time interoperability and maintain (upgrade) hardware components is the trade-off, though.

Weiser's ICT Device Forms: Tabs, Pads and Boards

ICT Device Forms from Weiser: Tabs, Pads and Boards When we think of computers, we usually think of multi-application personal or server computers, which are devices with a keyboard and some kind of pointing device for data input and a screen display for data output. Humans regularly interact with a far greater number of items, such as home appliances, that have a single embedded computer in them as well as complicated machinery that have several embedded computers. Weiser observed that there was a shift from having many users per computer to having one user per computer and then back to having many users per computer.

Computer-based gadgets are often become lighter, smaller and less expensive to make. As a result, gadgets may become more common, more portable and less noticeable. In his early work, Weiser took into consideration a variety of device sizes, including wearable centimeter-sized devices (tabs), hand-held decimeter-sized devices (pads) and meters-sized displays (boards). ICT tabs for tracking items and ICT pads for accessing mobile services are widely used. Wall displays are helpful for group viewing, cooperative work and the viewing of big, intricate structures like maps. In addition to being used vertically, board devices can also be utilised horizontally as surface computers.

Extended Forms for ICT Devices: Dust, Skin and Clay

Weiser (1991) proposed three distinct forms for devices, namely tabs, pads and boards. These forms are distinguished by their macro-sized design, planar shape and integration of visual output displays. We may broaden this spectrum of ubiquitous computing devices into a far more useful and diverse range if we relax each of these three features. First, ICT devices, such as Micro ElectroMechanical Systems (MEMS), which range in size from nanometers to millimetres, can be reduced in size without having visual output displays. We refer to this kind as Smart Dust. Some of them have the ability to integrate several microscopic mechanical and electronic components, allowing an expanding range of functionalities to be integrated into ICT devices, the real world and people.

These days, MEMS—such accelerometers—are included into a growing number of gadgets, including laptops, to detect falls and park moving parts, like disc arms. These systems are progressively integrated into commonly used systems. Additionally, a lot of devices use them to facilitate gesture-based interaction. One of the main factors enabling the concept of ubiquitous computing is miniaturisation combined with inexpensive manufacturing. Second, more flexible non-planar display surfaces and items, such clothing and curtains, can be created using fabrics based on light-emitting and conductive polymers, organic computer devices.

In order to transform a range of real-world structures into networked MEMS surfaces, MEMS devices can also be painted onto different surfaces. We refer to this kind as Smart

Notes

Skins. Third, MEMS ensembles can be shaped into any desired three-dimensional shape to create artefacts that mimic a wide variety of real-world objects. We refer to this kind as Smart Clay.

Mobility

Generally speaking, mobile gadgets are communicators, gaming consoles and business processing equipment that can be carried by their human users. Examples of these devices are cell phones and gaming consoles. There are numerous varieties of mobile phones, including the following:

- Accompanied: These are not implanted or worn devices. They might be carried in clothing or fashion accessories, or they can be handheld and portable.
- Portable: like laptop computers, which are designed to be operated while seated with two hands. These are typically the gadgets with the most resources.
- Hand-held: Devices that combine several functions including communication, audio-video recording and playback and mobile office are typically used one handed and occasionally hands-free. These gadgets have few resources.
- Wearable: Accessories and jewellery, like watches that manage personal information, earpieces that function as audio transceivers, glasses that function as visual transceivers and contact lenses, are typically controlled hands-free and independently. These gadgets have few resources.
- Implanted or embedded: They are frequently employed in medicine to improve human functions, such as in heart pacemakers. They could potentially be applied to improve the cognitive and physical capacities of people. Implants can be carbon-based, such as in nanotechnology, or silicon-based, such as in macro- or micro-sized integrated circuits.

Service Architecture Models

Distribution and Partitioning of Service Elements.

A variety of designs exist for segmenting and allocating services based on:

1) the application
2) the type of communication service
3) the type of access device used.

A chess program runs remotely on low resource devices since there is insufficient CPU power to run the program locally. Devices with relatively high resource access can function independently, functioning as standalone or monolithic appliances in an offline state.

Access devices with limited resources, like lightweight mobile devices, are frequently made with the utility model in mind, which primarily executes services across networks. Nonetheless, the need for some degree of self-reliance and local processing support features of an appliance model is supported by the necessity to deal with erratic and low-performance networks as well as the requirement to tailor power consumption to the power reserves available in mobile apps. The amount of local processing and information storage must be weighed against the amount of remote processing and communication bandwidth needed. Depending on the kind of applications and ICT infrastructure that are available, the balance could need to change dynamically.

Multi-tier Client Service Models

When an application service runs in a single tier, monolithic system, it all lives locally. The system could be networked so that, in some situations, when its operation is disrupted

(c) Amity University Online

Notes

or due to local malfunctions, it can go online and request assistance. In a two-tier thin client server, service processes run remotely, information related to these services is stored remotely and the access device (also known as the client device or terminal) facilitates data access or display. The access device in a two-tier, fat client-server model can invoke distant services in addition to supporting certain local processing and service usage.

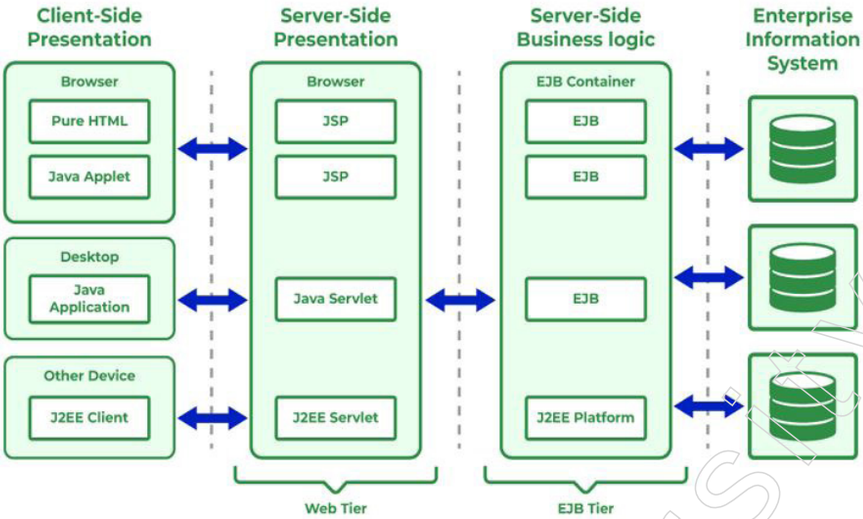


Figure 84: Two-tier, fat client-server model

Distributed Data Storage

Certain elements, including access, processing and information resources, can be further divided to accommodate various application types. Systems that divide and distribute information resources include distributed databases, where queries are mapped to multiple heterogeneous databases, each of which is mapped into a common form using a database wrapper; transaction monitors, where data transactions are created by distributed data sources, such as point of sale terminals where data warehouses; and centralised analysis of centrally stored data sub sets that are periodically extracted from distributed data resources.

Distributed Processing

Even though the most common processing paradigm for distributed systems is a client-server architecture with a single CPU, there are situations when a temporary increase in processing capacity is required. One method to accomplish this is to split the processing, distribute it among several distant processors, have each processor carry out a portion of the processing in parallel and then combine the output from the separate processors to create the whole. In order for this to be beneficial, the time saved by speeding up processing must be far greater than the time needed to divide and assign the tasks, gather the separate outcomes and put them back together. To accomplish parallel processing, a variety of computer architectures are available.

Client–Server Design

The client-server paradigm offers the benefit of more centralised distribution control, but it also has the drawback of fixed server setup and distribution. In terms of resources and interaction orientation, the client-server model of distributed computing is asymmetric. Servers typically have a lot of resources, such as larger storage capacities, more potent CPUs for demanding processing jobs, high bandwidth, a constant network connection to handle a lot of service requests and the ability to serve as a shared data repository.

© Amity University Online

Notes

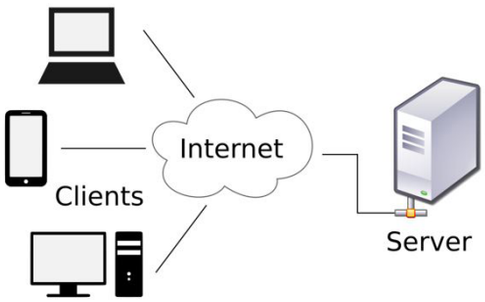


Figure 85: Network connection

Clients have comparatively less resources than servers do. Another example of an asymmetric client-server interaction is when client processes on access devices start the conversation by requesting things from application service processes on servers that are waiting for requests from clients. The asymmetry makes it easier to synchronise requests from clients and servers that begin waiting for requests from clients.

Proxy-based Service Access

Some apps utilise a client proxy to conceal the communication complexity that prevents them from being supported in the client access procedure. Access to services via a proxy can:

- offload network processing and presentation processing from the low-resource client access device to the third proxy node. For example, the client proxy could communicate with a mobile device using a simpler message protocol while speaking XML to a Web server, saving the access terminal from having to parse complex XML data structures. It can also hide the heterogeneity of various network types and terminal types from the access applications. Finally, it can simplify and compose access to multiple service providers. In order to minimise processing and communication with wireless end devices, a proxy may ask for a default connection. It can aggregate content from several service providers to enable a mobile portal approach.
- simplify the communication methods used by access devices. For example, frequently intricate hierarchical data structures must be encoded and decoded into more effective serialised data structures for transmission. This can be done by compressing or transcoding data at the server side, which will use less air wireless bandwidth.
- Permit gadgets to function sporadically when they are disconnected. Devices can turn off, leave the wireless network's coverage area, or just decide to function in disconnected mode, protecting network-based applications from the mobility of the access devices.

Service Provision Life Cycle

In order to provide application services for smart devices, distributed services must be managed at every stage of their life cycle, not just during designated stages like service discovery. For instance, intelligent devices operating in dynamic environments—like intelligent mobile devices—cannot presume that services will remain unchanged while they are being used.

Intermittent service access and handovers between various service instances—such as wireless communication handovers—must be accommodated for in the design of the service offering. This has two distinct components: managing this life cycle and developing a general life cycle model for service delivery. Only two of the five service model components—processing services, or service provision and service access, or clients—are considered

(c) Amity University Online

Notes

active in a basic service provision lifecycle model. The remaining three components, communication, stored information and information sources, are considered passive.

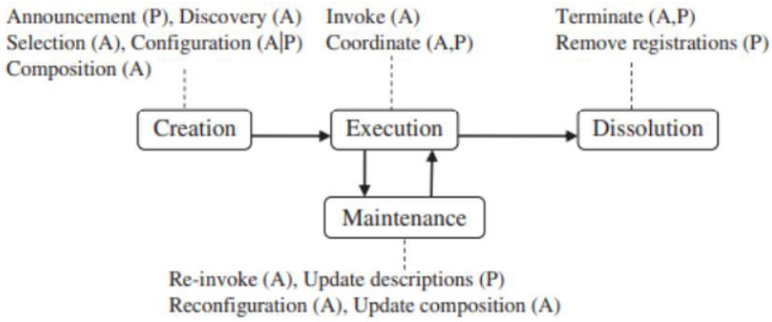


Figure 86: Service Provision Life Cycle

Service processes register with service directories during the service creation phase. In access nodes, service requesters look for services (repositories and information processes). A number of services must be chosen, set up and built; for example, a number of services must be composed in order to capture, annotate, transmit, edit, store, configure and print photographs.

Services are triggered during the operational or service execution phase and it can be necessary to coordinate several interconnected services. Updates to service compositions, access configurations and processes can be made during the service maintenance phase. During the service dissolution phase, requesters or the processes themselves may briefly suspend or stop providing services. It is also possible to withdraw and permanently end services.

The needs of the application, such as the required level of mobility, determine how the service lifecycle is designed. For instance, dynamic service execution and initiation can be supported by a static device like a set-top audio video receiver. This makes it possible for the device to be sent with its default factory settings preconfigured, ready for operation in various countries where it will need to identify and adjust to the various regional sources of RF broadcast signals. If the user has authorisation to access it—possibly through another service contract—this can also allow a static smart device to automatically switch to a different service provider in the event of a malfunction.

5.2 Ethical Considerations and Future Trends

Ethical considerations in Human-Computer Interaction (HCI) emphasize the importance of prioritising user privacy, obtaining informed consent for data usage, ensuring accessibility and inclusivity, addressing biases in algorithms and empowering users with control over their digital experiences.

Anticipated trends in Human-Computer Interaction include the increasing integration of Augmented Reality (AR) and Virtual Reality (VR), the rise of voice and natural language interactions, the adoption of gesture and motion control, the focus on ethical AI and explainability and the exploration of Brain-Computer Interfaces (BCIs).

5.2.1 Ethical Issues in HCI

Human-computer interaction (HCI) ethics cover a wide variety of topics pertaining to the appropriate development, application and use of interactive technologies. As technology becomes more and more integrated into our daily lives, HCI ethics become more and more important.

(c) Amity University Online

Notes

Privacy Issues:

Data Gathering and Surveillance: Concerns regarding user privacy are raised when interactive technologies gather personal data. Without express authorisation, tracking user behaviour, preferences and activities may violate their privacy.

Informed Consent: Users ought to be aware of how interactive systems acquire data and have the power to decide what data is collected and how it is utilised.

Accessibility and Inclusivity:

Digital Divide: Differences in access to technology may be caused by the way interactive systems are designed. Ensuring that systems are inclusive and accessible to users with varying abilities, backgrounds and socioeconomic levels is a crucial ethical factor.

The application of universal design principles entails designing user interfaces that do not require specialised design or adaption for the greatest number of people.

Fairness and Bias:

Algorithmic Bias: Interactive systems that rely on algorithms run the risk of sustaining training data biases, which can provide unfair and biassed results. Finding and eliminating biases is a key component of ethical design in order to guarantee equity for all users.

Representation: Biassed viewpoints may arise from a design team's lack of diverse representation. Promoting diversity in design teams is one ethical way to combat prejudices in the development process.

Openness and Definability:

Algorithms that are Hard to grasp: Complex algorithms may make decisions that are difficult for users to grasp. The creation of transparent systems and giving users justifications for algorithmic choices are key components of ethical HCI.

User Empowerment: It is morally required to provide users with the information and resources they need to comprehend and manage how they engage with systems.

User Input and Self-Governance:

Coercive Design: Ethical questions arise when systems are created to control user behaviour without the users' express consent. Respecting user autonomy and getting informed consent for system functionalities are essential components of ethical HCI practice.

Dark Patterns: One ethical tip is to stay away from using dark patterns, which are design cues that lead people to make decisions that are not in their best interests.

Trust and Security: Data Security: It is morally required to protect user data. Robust security measures must be implemented by interactive systems to guard against unauthorised access and breaches of user information.

Trustworthiness: Establishing trust between people and systems is a key component of ethical design. Being trustworthy is influenced by open communication regarding security precautions and the handling of private information.

User Well-Being: Addiction and Overuse: Taking into account the possibility of addiction or overuse is a necessary part of designing interactive systems that put the needs of users first. Including elements that encourage responsible usage and give users options to control their screen time is part of ethical HCI.

Impact on Mental Health: It is morally right to recognise and take steps to reduce any potential harm that interactive systems may cause to people's mental health. User well-being should come first in design decisions rather than engagement numbers.

(c) Amity University Online

Notes

Sustainability: Ethical HCI takes into account how interactive technologies may affect the environment. Ethical practices are enhanced by sustainable design decisions, such as energy-efficient interfaces and conscientious production methods.

Responsibility & Accountability:

Create Responsibilities: It is the duty of designers and developers to consider any moral dilemmas and put consumers' needs first. This entails maintaining a dedication to moral design principles and keeping up with newly arising moral dilemmas.

Legal and Ethical Compliance: A crucial component of ethical HCI is making sure that interactive systems abide by all applicable laws and regulations. But moral issues frequently transcend legal obligations.

Cultural Appropriation and Cultural Sensitivity: A key component of ethical HCI is identifying and preventing cultural appropriation in design. Respectful and inclusive interactive systems must take diversity, cultural norms and values into account.

End-of-Life Considerations:

E-waste: Disposing of electronic gadgets raises ethical and environmental questions. Designing systems with environmentally conscious end-of-life practices, like recycling and sustainable materials, is part of ethical HCI.

Constant Observation and Assessment:

Ethical Review Procedures: In HCI, ethics are taken into account when continuously observing and assessing interactive systems. Throughout a system's lifecycle, establishing ethical review procedures can assist in identifying and resolving developing ethical difficulties.

Iterative Design and User Feedback: User-Centered Design: It is morally required to include users in the design process. Participatory design, iterative system refinement based on user needs and preferences and collecting and implementing user feedback are examples of ethical HCI techniques.

5.2.2 Emerging Trends

Developing methods that improve user interface and increase computer intuitiveness is the primary goal of research on human-computer interaction. Because there is a significant barrier separating the user and the computer when using physical devices like keyboards and mice for human-computer interaction, the interface is less natural and intuitive. User-computer interface is now more than only keyboard and mouse interaction thanks to the emergence of ubiquitous computing. An appealing alternative to conventional text-based interfaces through graphical-based user interfaces is the direct use of hands as an input device to enable natural human-computer interaction.

Building a reliable hand gesture detection system is still a difficult task for conventional vision-based techniques, despite the enormous market for hand gesture-based interface design. Therefore, a system for recognising hand gestures that is effective at tracking both static and dynamic hand movements would provide users with a natural and straightforward way to interact with their computers.

With relatively little technology, our system transforms the detected gesture into activities like accessing webpages, starting apps and many more. Using gaze movements with a head-mounted display (HMD), a portable interactive display device that tracks eye movement for interaction, is another way to improve the intuitiveness of this interaction.

© Amity University Online

Notes

Given that people have complete control over their eye movements, this technique is both extremely powerful and quite simple to utilise. Therefore, eye-tracking technology can be applied to HCI. Currently, gesture-based HCIs are well-established for voice input; however, they are not appropriate in situations where speech is not possible or when both hands are busy.

Thus, it is imperative to approach HCI with HMDs in a more straightforward and efficient manner. Additionally, an HMD webcam is used in this system to accomplish HMD-based gaze interaction, which analyses the user's intent based on gaze and tracks the direction of human gaze in real time at a close range. Gaze gesture-based HCI research has been developing and growing quickly in the last few years.

Methodology for Hand Gesture Recognition For Human-Computer Interaction

With the use of its camera module, this technology allows the system to instantaneously capture an image of a hand motion made by the user. Next, using a variety of gray-scaling methods, the image is transformed into a grayscale image. After that, the grayscale image is treated to reduce noise and make it smoother.

Otsu binarisation automatically determines a threshold value for a bimodal image—an image with two peaks in its histogram—from the image histogram. Using a threshold value that is determined by binarisation, all pixels in the processed grayscale image are converted to either black or white based on whether their value is less than or greater than that threshold value. This process produces a binary image with improved accuracy. Object detection is accomplished by contour extraction. Convexity flaws and the convex hull are found together. Gestures are then identified based on these flaws.

When there are no convexity flaws in a gesture, such as the palm and first, the Haar cascade classifier is used to identify the gesture. This classifier uses a set of positive images—at least 10 original images—taken at various lighting conditions and angles. Actions are then assigned to each of these motions. At last, the application associated with a specific gesture is started.

Advantages:

- It is a more user-friendly method of engagement that is intuitive and natural.
- Identifies hand movements that are both dynamic and static.
- As an identification system, it is quick and reasonably dependable.
- Simple to integrate into real-time systems.
- You can assign any job to each of these customisable gestures.
- The hardware requirements are minimal.
- Affordable.

Disadvantages:

- Recognition accuracy will decrease in the presence of a complicated background.
- The system of recognition may be misled by hands on irrelevant items.
- It's possible that the system needs the hand to be vertical with the fingers pointing straight towards the camera.
- As the distance between the user and the camera rises, the system's performance decreases.
- Colour detection is impacted by ambient light, which lowers system performance.
- It is still unable to create an interface that can take the place of tangible controls.

(c) Amity University Online

Notes

Methodology For Gaze Gesture And Their Applications In Human-Computer Interaction With a Head-Mounted Display

Eye-tracking, a technique for determining a person's gaze location and degree of movement in relation to head posture, is used in this method. This system achieves a gaze interaction style based on head-mounted displays (HMDs) to track and identify human gaze direction at close range in real-time.

In order to determine the gaze direction, the procedure begins with taking an image of the eye using the near-eye camera built into the headgear. To give head pose information directly, the system will adjust the head pose distribution's range. It gathers information based on range and the information matches the picture that our head-mounted display system took. The image is amplified by a certain amount when the pupil serves as the centre. After generating the pupil coordinates from the centre coordinates and reducing the pixel count at random, the image is subjected to Gaussian filtering. Next, two deep convolutional neural network model modules are employed to categorise the gaze trajectories of the image. Data from nearly ten thousand gaze trajectories gathered from different people are contained in this network.

Learning from synthetic images may not function as predicted because to the difference in feature distributions between the images and real images. This network employs a model that has been pre-trained on the Internet to learn from vast amounts of data and then trains the resulting model to bridge the gap between a synthetic picture distribution and a real image distribution. The data dispersion issue is resolved and system recognition is improved while labelling information is retained when real data is used. These various gaze motions are ultimately assigned to a particular function that the user selects.

Advantages:

- This technology is incredibly reliable and easy to use.
- Extremely useful when speaking is not an option due to hands being occupied.
- The movement of the eye look is much faster than any other gesture.
- For gesture recognition, this method gathers a lot of precise and accurate data.
- Increased recognition accuracy through neural network support.
- Maps several features using two neural networks operating in tandem to guarantee consistency in the outcomes.
- Capable of adjusting to different indoor and outdoor illumination situations.
- Is able to serve as an interface for those with disabilities.

Disadvantages:

- Novice users may frequently create erroneous patterns.
- Takes some time for the user to adjust to the interface after adoption.
- Everybody's eyes and camera are positioned differently.
- Glasses and contact lenses can affect a camera's capacity to track eye movements.
- The cost of eye tracking and neural network training can be high.

Use of These Techniques In Real Time Scenarios:

1. Hand Gesture: There is a great deal of opportunity for the creation of novel interface designs that improve upon the ways in which we now interact with computers. The creation of these kinds of interfaces is the subject of extensive research. Interfaces

© Amity University Online

Notes

based on hand gestures have a vast market. With this type of interface, the user can act conveniently without having to reach for a physical controlling device in a wide range of useful applications.

For instance, when a corporation launches a product, they utilise these hand gestures to advance to the next slide rather than using a real clicker in their hands. Small smart mobile devices and other smart wearables, such as smartwatches, where the interface is constrained and frequently has very small screens forcing users to utilise other types of interface instead of a touch screen, can benefit greatly from this type of interface.

Thus, with the use of infrared sensors, this type of hand gesture movement can be incorporated into such devices. These gestures can be used for simple yet common operations like opening specific apps, adjusting the volume, skipping songs, phoning a preset contact and so on. These gestures can be readily created. This makes it easier for the user to engage with the gadget, which has a restricted physical area, as a visual interface will be difficult for the user to utilise.

2. Gaze Gesture: Other interface paradigms, such as augmented reality and virtual reality (AR and VR), where eye movement tracking is utilised for a variety of purposes, have a high need for this technology. In certain situations where hands are engaged and words may not be an option, traditional ways of interaction may not be appropriate.

   Currently, gaze gestures are supported because these programs do not include any physical interaction devices other than a head mount, which is utilised solely to track the user's eye movements. This technology can be widely applied in these industries because these gadgets simply require the user to engage in a convenient and free way, as gazes gesture interface provides. These kinds of technology are even in use right now; Microsoft HoloLens is one example.

   Gaze motions can be utilised as an interface for navigation and other user-specified functions in smart devices such as smart lenses and wearables that show information. Additionally, this technique can be utilised as a drawing tool in a virtual world because to its great eye-tracking accuracy. These can also be used to gauge a user's level of drowsiness and, if desired, alert them. The identification of the product that has captured the interest of the user viewing the advertisement is another situation in which tracking the user's gaze is crucial in the field of commerce.

Summary

- Moving our focus of interaction away from the traditional two-dimensional graphical user interface on the desktop presents many exciting and new challenges to the field of HCI. Weiser's vision of ubiquitous computing was human-centered and many years later, it still presents a grand challenge for those who wish to address this new interaction paradigm. These challenges cover three main areas. First, the physical interface between humans and a computationally enhanced environment requires a shift in emphasis for both input and output. Secondly, while a 'killer app' for ubicomp has arrived in the form of person–person communication, the more interesting challenge is to understand what general features of ubicomp applications matter.
- The past decade has seen two emergent features – context awareness and automated capture. The next decade should see an increased focus on continuous services that support everyday activities. Thirdly, different theories of human cognition inform the design of ubicomp applications, as 'knowledge in the world' becomes even more important.
- The design process must better incorporate an understanding of the invisible meaning of everyday activities. Evaluation of the utility of ubicomp applications is harder to do

(c) Amity University Online

Notes

in controlled usability laboratories, so we must build a new form of laboratory that facilitates observation and measurement as well as everyday activity.

- The technologies of virtual and augmented reality are no longer reserved for laboratories with multi-million-dollar equipment, but can be experienced on ordinary machines and over the internet. Early applications of virtual and augmented reality were often for military training and for ubicomp it was the office place. Now, much of this technology is showing its promise in the entertainment and leisure domains.
- Many advances of VR and other forms of 3D graphics are an important resource in the visualisation of scientific data and other complex information. In particular, by interacting with the visualisation, the user may obtain a deeper understanding of the data and discover features that would be impossible with static representations.

Glossary

- HCI: Human Computer Interaction
- MHP: Model Human Processor
- IxD: Interaction Design
- IP: Internet Protocol
- LAN: Local Area Network
- RFID: Radio Frequency Identification
- ICT: Information Communication Technology
- CPU: Central Processing Unit
- DEI: Device Environment Integration

Check Your Understanding

1. What is a key ethical consideration in user data collection for HCI applications?
   a. Collect as much data as possible for comprehensive analysis
   b. Prioritise user consent, privacy and secure data handling
   c. Share user data freely to enhance collaboration
   d. Use collected data for targeted advertising without user knowledge
2. Why is addressing biases in algorithms and interfaces important in HCI ethics?
   a. Biases improve system efficiency
   b. Biases align with user preferences
   c. Biases can lead to discrimination and unfair treatment
   d. Biases have no impact on user experiences
3. What does informed consent mean in the context of HCI?
   a. Forcing users to agree to terms and conditions
   b. Obtaining user permission after data collection
   c. Not informing users about data usage
   d. Ignoring user preferences
4. How does the principle of accessibility contribute to ethical HCI practices?
   a. It doesn't have any ethical implications
   b. It limits user options
   c. It ensures that technologies are usable by diverse users

(c) Amity University Online

Notes

d. It prioritises certain user groups over others

5. What is the role of transparency in ethical HCI?
   a. Hiding information to protect the organisation
   b. Sharing only positive outcomes
   c. Providing clear and open communication about system operations and intentions
   d. Keeping users in the dark about data practices

Exercise

1. Define various speech interfaces.
2. Explain information visualisation.
3. What do you mean by ubiquitous computing?
4. Define various ethical issues in HCI.
5. What are the emerging trends in HCI?

Learning Activities

1. Many researchers are beginning to explore the potential of ubiquitous computing technologies and applications in the home environment. Discuss how the application themes of context awareness, automated capture and the continuous interaction of everyday computing are relevant to domestic life. Focus your answer on the challenges of family life or life for an aging population.
2. Virtual reality has found a number of applications in the games market. Is this a suitable use of such technology? Discuss the possible benefits and disadvantages of exploiting leading edge technology in a leisure market.

(c) Amity University Online