Human Computer Interaction- 1st Course

تفاعل الانسان مع الاله

Fourth Grade

Software Branch

Dr. Matheel Emaduldeen

1. Foundations Contexts for HCI

Contexts for HCI

Human–Computer Interaction (HCI), alternatively Man–Machine Interaction (MMI) or Computer–Human Interaction (CHI) is the study of interaction between people (users) and computers.

"HCI is a branch concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."

GOALS

- A basic goal of HCI is to improve the interactions between users and computers by making computers more usable and receptive to the user's needs.
- A long term goal of HCI is to design systems that minimize the barrier (الحاجز) between the human's cognitive (الادراكي) model of what they want to accomplish and the computer's understanding of the user's task.

WHY IS HCI IMPORTANT

1. User-centered design is getting an important role.
2. It is getting more important today to increase competitiveness (المنافسة) via HCI studies.
3. High-cost e-transformation investments (استثمارات).
4. Users lose time with badly designed products and services.
5. Users even give up (يستسلم) using bad interface.
6. Ineffective allocation of resources.

DEFINING THE USER INTERFACE

• User interface, design is a subset of a field of study called *human-computer interaction* (HCI).

• HCI is the study, planning, and design of how people and computers work together so that a person's needs are satisfied in the most effective way.

• HCI designers must consider a variety of factors:

1. what people want and expect, physical limitations and abilities people possess,
2. how information processing systems work,
3. what people find enjoyable and attractive.
4. Technical characteristics and limitations of the computer hardware and software must also be considered.

• The *user interface* is the part of a computer and its software that people can see, hear, touch, talk to, or otherwise understand or direct.

• The user interface has essentially two components: input and output.

• *Input* is how a person communicates his / her needs to the computer. Some common input components are the keyboard, mouse, trackball, one's finger, and one's voice.

• *Output* is how the computer conveys the results of its computations and requirements to the user.

• Today, the most common computer output mechanism is the display screen, followed by mechanisms that take advantage of a person's auditory (السمعية) capabilities: voice and sound.

• The use of the human senses of smell and touch output in interface design still remain largely unexplored.

• Proper interface design will provide a mix of well-designed input and output mechanisms that satisfy the user's needs, capabilities, and limitations in the most effective way possible.

• The best interface is one that it not noticed, one that permits the user to focus on the information and task at hand, not the mechanisms used to present the information and perform the task.

THE IMPORTANCE OF GOOD DESIGN

With today's technology and tools, and our motivation to create really effective and usable interfaces and screens, why do we continue to produce systems that are inefficient and confusing or, at worst, just plain unusable.

Is it because:

• We don't care?

• We don't possess common sense?

• We don't have the time?

• We still don't know what really makes good design?

• But we never seem to have time to find out what makes good design, nor to properly apply it. After all, many of us have other things to do in addition to design interfaces and screens.

• So we take our best shot given the workload and time constraints imposed upon us. The result, too often, is woefully inadequate.

• Interface and screen design were really a matter of common sense, The developers would have been producing *almost identical* screens for representing the real world.

• Example bad designs

– Closed door with complete wood

– suggestion : glass door

THE BENEFITS OF GOOD DESIGN

- Poor clarity forced screen users to spend one extra second per screen.
- The benefits of a well-designed screen have also been under experimental scrutiny (فحص دقيق) for many years.
- Proper formatting of information on screens does have a significant positive effect on performance.
- Training costs are lowered because training time is reduced.
- support line costs are lowered because fewer assist calls are necessary.
- Identifying and resolving problems during the design and development process also has significant economic benefits.

A BRIEF HISTORY OF THE HCI

• The need for people to communicate with each other has existed since we first walked upon this planet.

• The lowest and most common level of communication modes we share are movements and gestures.

• Movements and gestures are language independent, that is, they permit people who do not speak the same language to deal with one another.

• The next higher level, in terms of universality and complexity, is spoken language.

• Most people can speak one language, some two or more. A spoken language is a very efficient mode of communication if both parties to the communication understand it.

• At the third and highest level of complexity is written language. While most people speak, not all can write.

• But for those who can, writing is still nowhere near as efficient a means of communication as speaking.

• In modem times, we have the typewriter, another step upward in communication complexity.

• Significantly fewer people type than write. (While a practiced typist can find typing faster and more efficient than handwriting, the unskilled may not find this the case.)

• Spoken language, however, is still more efficient than typing, regardless' of typing skill level.

• Through its first few decades, a computer's ability to deal with human communication was inversely related to what was easy for people to do.

• The computer demanded rigid, typed input through a keyboard; people responded slowly using this device and with varying degrees of skill.

• The human-computer dialog reflected the computer's preferences, consisting of one style or a combination of styles using keyboards, commonly referred to as Command Language, Question and

Answer, Menu selection, Function Key Selection, and Form Fill-In.

• Throughout the computer's history, designers have been developing, with varying degrees of success, other human-computer interaction methods that utilize more general, widespread, and easier-to-learn capabilities: voice and handwriting.

• Systems that recognize human speech and handwriting now exist, although they still lack the universality and richness of typed input.

2. User-Centered Development Process

Software Development Process Model

• Process model

– Segmentation of the overall (team) activity of software development into smaller portions of work

» High-level structure: phases

» Low-level structure: steps, activities

• Basic activities covered in all models:

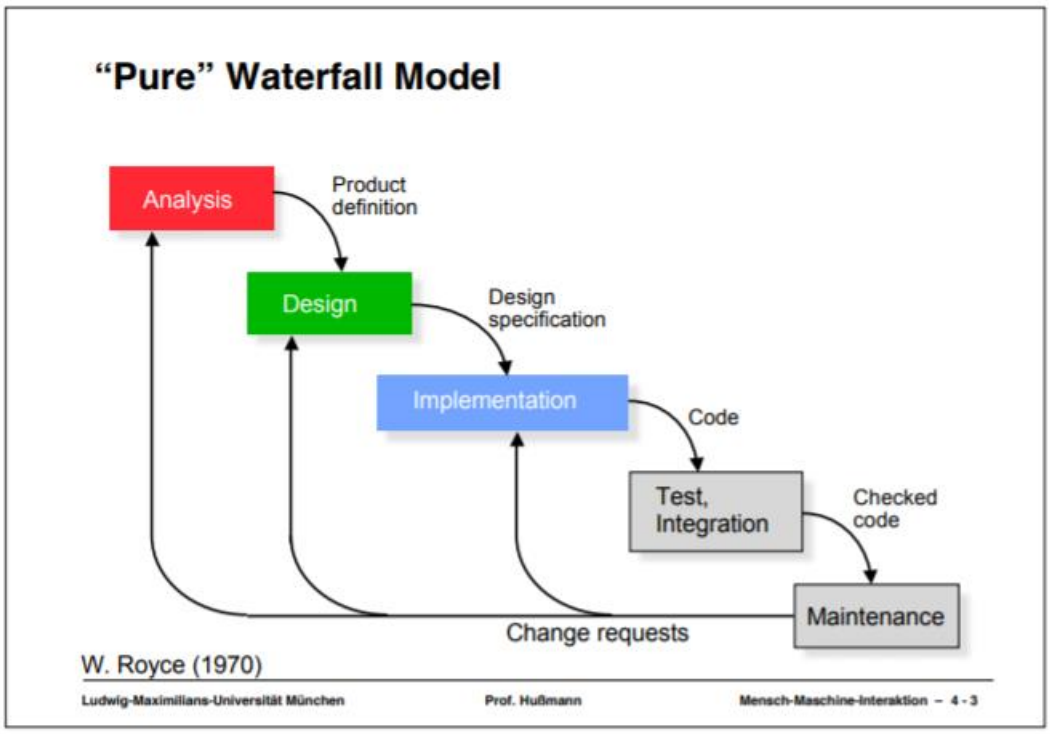
– Analysis

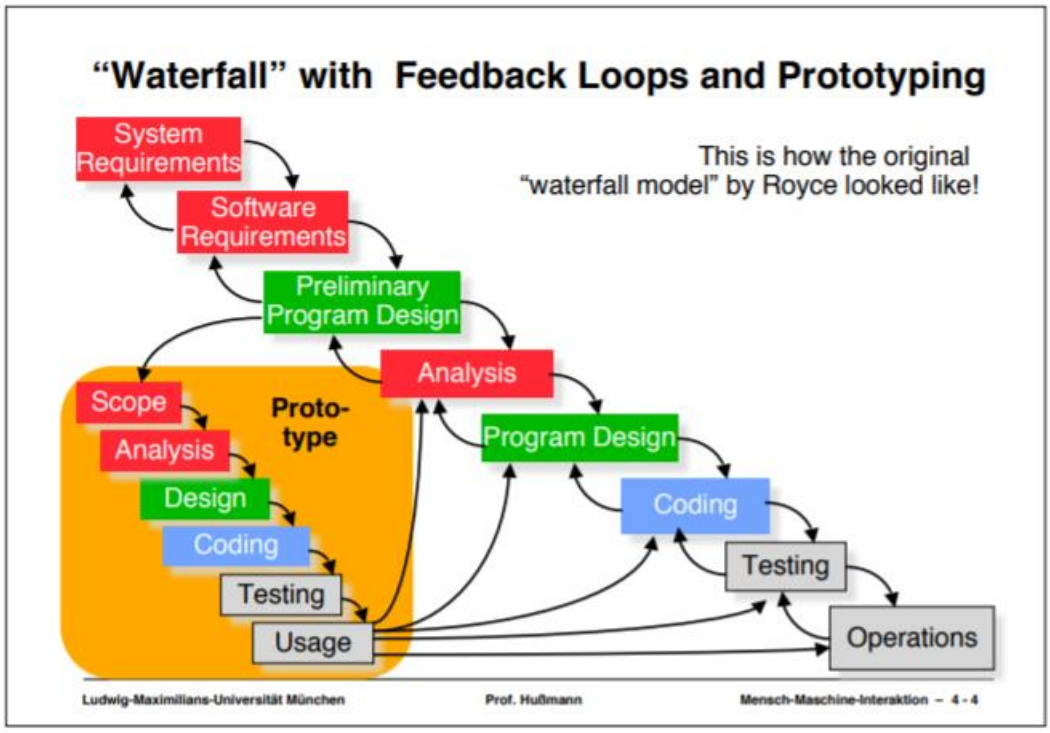
– Design

– Implementation

– Validation (in particular Test, Integration)

– Evolution (in particular Maintenance)


"Pure" Waterfall Model
Analysis
Product definition
Design
Design specification
Implementation
Code
Test, Integration
Checked code
Maintenance
Change requests
W. Royce (1970)
Ludwig-Maximilians-Universität München
Prof. Hußmann
Mensch-Maschine-Interaktion – 4 - 3


"Waterfall" with Feedback Loops and Prototyping
System Requirements
Software Requirements
Preliminary Program Design
Analysis
Program Design
Coding
Testing
Operations
Scope
Analysis
Design
Coding
Testing
Usage
Proto-type
This is how the original "waterfall model" by Royce looked like!
Ludwig-Maximilians-Universität München
Prof. Hußmann
Mensch-Maschine-Interaktion – 4 - 4

It gives mixed information about success in practice:

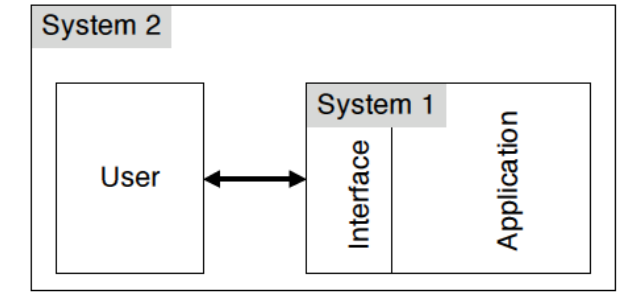
1. Good experiences in small and innovative مبتكر projects
2. Large-scale projects tend to stay "conservative"متحفظ , mainly due to transparency for project management

Usability Aspects are Mostly Ignored by Software Engineers

System perspectives وجهات نظر النظام

– SW Engineers take the "System 1" perspective

– Usability Engineers take the "System 2" perspective.



Separation between Interaction Design and Technical Design

For interactive applications a separation into a two stage process is often advisable

• 1st – Interaction design (iterative)

– concept

– Interaction analysis

– Prototypes

– Evaluation

– Stable and tested design

• 2nd – technical realization

– Technical analysis

– Technical specification (e.g. architecture, platform)

– Implementation

– Evaluation and Quality management

LUCID Development Process

Logical User Centered Interactive development Methodology (LUCID)

• Stage 1: Envision التخيل

– Develop UI Roadmap which defines the product concept, rationale الاساس المنطقي, constraints and design objectives.

• Stage 2: Analyze

– Analyze the user needs and develop requirements.

• Stage 3: Design

– Create a design concept and implement a key screen prototype.

• Stage 4: Refine

– Test the prototype for design problems and iteratively refine and expand the design.

• Stage 5: Implement

– Support implementation of the product making late stage design changes where required. Develop user support components.

• Stage 6: Support

– Provide roll-out support as the product is deployed and gather data for next version.

Scenarios and Claims المطالبات

• Scenario

– Scenarios describe an existing or envisioned system from the perspective of one or more real or realistic users.

» Example: "A person turned on a computer; the screen displayed a button labeled Start; the person used the mouse to select the button."

• Claim

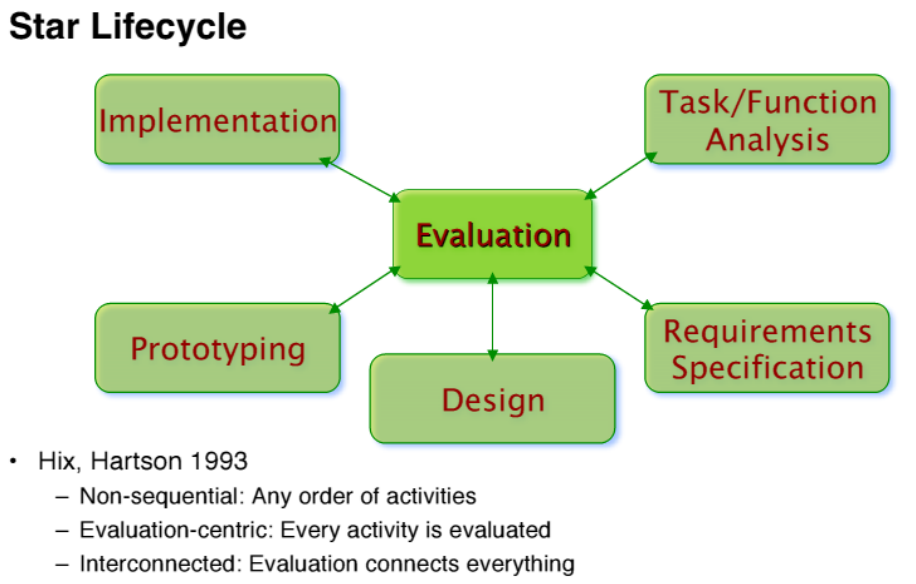
– Claims are psychologically motivated design rationales that express the advantages and disadvantages of a design as a usability issue

– Relating properties of the artifact with specific psychological consequences, under the scope of a basic task usage situation.

» Example: Including open-ended exercises in an instruction manual supports learning-by-exploration

» Example: “Returning the user to the Create or Revise menu is adequate feedback that an option change attempt is successful (but may not be enough feedback for users who are unsure or confused)”

– Encourages designer to reason about trade-offs rather than accepting a single guideline or principle



Problems of User Centered Design

• Users may be wrong

• Users may be resistant to change

• Users may expect disadvantages (e.g. being replaced by software)

• Be aware – you are expected to create an optimal system with regard to the goals specified

– this is unfortunately NOT necessarily the system users would like to have (e.g. trade-off between employers and employees)

Testing prototypes to choose among alternatives

3. Principles of GUI

INTRODUCTION OF THE GRAPHICAL USER INTERFACE (GUI)

The Xerox systems, Altus and STAR, introduced the mouse and pointing and selecting as the primary human-computer communication method. The user simply pointed at the screen, using the mouse as an intermediary. These systems also introduced the graphical user interface as we know it a new concept was born, revolutionizing the human-computer interface.

THE POPULARITY OF GRAPHICS

A graphical screen carry scant resemblance (تشابها ضئيلا) to its earlier text-based colleagues. Older text-based screen possessed a one dimensional. Graphic screens assumed a three-dimensional look. Controls appeared to rise above the screen and move when activated. Information could appear, and disappear, as needed.

Text could be replaced by graphical images called icons. • These icons could represent objects or actions. Selection fields such as radio buttons, check boxes, list boxes, and palettes coexisted with the reliable old text entry field. More sophisticated text entry fields with attached or dropdown menus. Objects and actions were selected through use of pointing mechanisms. User's actions to be reacted to quickly, dynamically, and meaningfully.

(a) (b)

Figure A scrolling menu of country names
Source: https://www.jollyflorist.com

WIMP interface: windows, icons, menus, and pointers. Graphic presentation is much more effective than other presentation methods. Properly used, it reduces the requirement for perceptual and mental information recoding and reorganization, and also reduces the memory loads.

It permits faster information transfer between computers and people by permitting more visual comparisons of amounts, trends, or relationships; more compact representation of information; Graphics also can add appeal or charm (سحر) to the interface and permit greater customization to create a unique corporate or organization style.

GRAPHICAL SYSTEMS ADVANTAGES AND DISADVANTAGES

ADVANTAGES

1. Reduce the memory requirements.
2. More effective use of one's information.
3. Dramatically reduce system learning requirements.
4. Experience indicates that for many people they have done all these things.
5. Symbols recognized faster than text
6. Faster learning
7. Faster use and problem solving
8. Easier remembering
9. More natural
10. Fewer errors
11. Increased feeling of control
12. Immediate feedback
13. Predictable system responses

14. More attractive
15. May consume less space
16. Replaces national languages
17. Easily augmented with text displays
18. Smooth transition from command language system

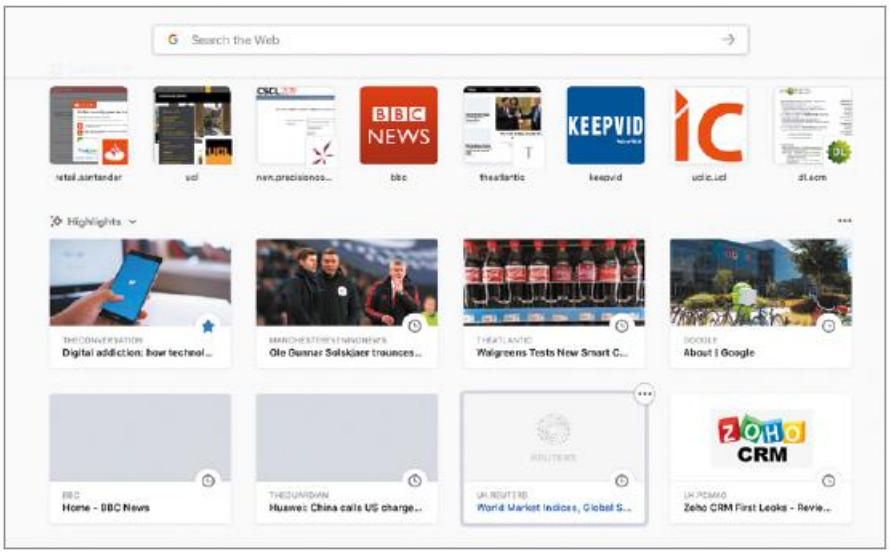


Figure Part of the home page for the Firefox browser showing thumbnails of top sites visited and suggested highlight pages (bottom rows)

DISADVANTAGES

1. Greater design complexity.
2. Learning still necessary.
3. Lack of experimentally-derived design guidelines.
4. use a pointing device may also have to be learned.
5. Working domain is the present.
6. Human comprehension limitations.
7. Window manipulation requirements.

8. Production limitations.
9. Few tested icons exist.
10. Inefficient for touch typists.
11. Inefficient for expert users
12. Not always the preferred style of interaction
13. Not always fastest style of interaction
14. Increased chances of clutter and confusion
15. May consume more screen space
16. Hardware limitations

THE CONCEPT OF DIRECT MANIPULATION

The system is portrayed as an extension of the real world: It is assumed person is already familiar with the objects and actions in his or her environment of interest. The system simply replicates them and portrays them on a different medium, the screen. A person has the power to access and modify these objects, among which are windows. A person is allowed to work in a familiar environment and in a familiar way, focusing on the data, not the application and tools.

The physical organization of the system, which most often is unfamiliar, is hidden from view and is not a distraction. Continuous visibility of objects and actions: Like one's desktop, objects are continuously visible. Reminders of actions to be performed are also obvious, labeled buttons replacing complex syntax and command names. Cursor action and motion occurs in

physically obvious and natural ways. One problem in direct manipulation, however, is that there is no direct analogy on the desk for all necessary windowing operations.

A piece of paper on one's desk maintains a constant size, never shrinking or growing. Windows can do both. Solving this problem required embedding a control panel, a familiar concept to most people, in a window's border. This control panel is manipulated, not the window itself. Actions are rapid and incremental with visible display of results, the results of actions are immediately displayed visually on the screen in their new and current form. Auditory feedback may also be provided. The impact of a previous action is quickly seen, and the evolution of tasks is continuous and effortless. Incremental actions are easily reversible.

INDIRECT MANIPULATION

In practice, direct manipulation of all screen objects and actions may not be feasible (ممكن) because of the following:

1. The operation may be difficult to conceptualize (التصور) in the graphical system.
2. The graphics capability of the system may be limited.
3. The amount of space available for placing manipulation controls in the window border may be limited.
4. It may be difficult for people to learn and remember all the necessary operations and actions.

When this occurs, indirect manipulation is provided. Indirect manipulation substitutes words and text, such as pull-down or pop-up menus, for symbols, and substitutes typing for pointing. Most window systems are a combination of both direct and indirect manipulation. A menu may be accessed by pointing at a menu icon and then selecting it (direct manipulation). The menu itself, however, is a textual list of operations (indirect manipulation). When an operation is selected from the list, by pointing or typing, the system executes it as a command.

CHARACTERISTICS OF THE GUI

A graphical system possesses a set of defining concepts. Included are sophisticated visual presentation, pick-and click interaction, a restricted set of interface options, visualization, object orientation, extensive use of a person's recognition memory.

1. Sophisticated Visual Presentation:

Visual presentation is the visual aspect of the interface. It is what people see on the screen. The sophistication (تكلف) of a graphical system permits displaying lines, including drawings and icons. It also permits the displaying of a variety of character fonts, including different sizes and styles.

The display of 16 million or more colors is possible on some screens. Graphics also permit animation and the presentation of photograph and motion video.

The meaningful interface elements visually presented to the user in a graphical system include windows (primary, secondary, or dialog boxes), menus (menu bar, pull down, popup, cascading), icons to represent objects such as programs or files, assorted screen-based controls (text boxes, list boxes, combination boxes, settings, scroll bar and buttons), and a mouse pointer and cursor. The **objective** is to reflect visually on screen the real world of the user as realistically, meaningfully, simply, and clearly possible.

2. **Pick-and-Click Interaction:**

Elements of a graphical screen upon which some action is to be performed must first identified. The motor activity required of a person to identify this element for a proposed action is commonly referred to as pick, the signal to perform an action as cue. The primary mechanism for performing this pick-and-click is most often the mouse and its buttons. The user moves the mouse pointer to the relevant element (pick) and the action is signaled (click).

Pointing allows rapid selection and feedback. The hand and mind seem to work smoothly and efficiently together. The secondary mechanism for performing these selection actions is the keyboard most systems permit pick-and-click to be performed using the keyboard as well.

3. **Restricted Set of Interface Options:**

The array of alternatives available to the user is what is presented on the screen or may be retrieved through what is presented on the screen, nothing less, nothing more. This concept fostered the acronym WYSIWYG.

4. **Visualization:**

Visualization is a cognitive process that allows people to understand. Information that is difficult to perceive, because it is either too voluminous or too abstract presenting specialized graphic portrayals facilitates visualization. The best visualization method for an activity depends on what people are trying to learn from the data.

The goal is not necessarily to reproduce a really graphical image, but to produce one that conveys the most relevant information. Effective visualizations can facilitate mental insights, increase productivity, and for faster and more accurate use of data.

5. **Object Orientation:**

A graphical system consists of objects and actions. Objects are what people see on screen. They are manipulated as a single unit. Objects can be composed of sub objects. For example, an object may be a document. The document's sub objects may be a paragraph, sentence, word, and letter. A collection is the simplest relationship-the objects sharing a common aspect. A collection might be the result of a query or a multiple selection of objects. Operations can be applied to a collection of objects.

A constraint is a stronger object relationship. Changing an object in a set affects some other object in the set. A document being organized into pages is an example of a constraint. A composite exists when the relationship between objects becomes so significant that the aggregation itself can be identified as

an object. Examples include a range of cells organized into a spreadsheet, or a collection of words organized into a paragraph. A container is an object in which other objects exist. Examples include text in a document or documents in a folder.

A container often influences the behavior of its content. It may add or suppress certain properties or operations of objects placed within it, control access to its content, or control access to kinds of objects it will accept. These relationships help define an object's type. Similar traits and behaviors exist in objects of the same object type.

Another important object characteristic is persistence. Persistence is the maintenance of a state once it is established. An object's state (for example, window size, cursor location, scroll position, and so on) should always be automatically preserved when the user changes it.

6. **Use of Recognition Memory:**

Continuous visibility of objects and actions encourages use of a person's more powerful recognition memory. The "out of sight, out of mind" problem is eliminated

Application Examples:

QR code appearing on a magazine page

4. Principles of GUA II

CONCURRENT PERFORMANCE OF FUNCTIONS

Graphic systems may do two or more things at one time. Multiple programs may run simultaneously. When a system is not busy on a primary task, it may process background tasks (cooperative multitasking).When applications are running as truly separate tasks, the system may divide the processing power into time slices and allocate portions to each application.

Data may also be transferred between programs. It may be temporarily stored on a "clipboard" for later transfer or be automatically swapped between programs.

THE GRAPHICAL USER INTERFACE

A user interface is a collection of techniques and mechanisms to interact with something. In a graphical interface the primary interaction mechanism is a pointing device of some kind. This device is the electronic equivalent to the human hand. What the user interacts with is a collection of elements referred to as objects. They can be seen, heard, touched, or otherwise perceived. Objects are always visible to the user and are used to perform tasks. They are interacted with as entities independent of all other objects.

People perform operations, called actions, on objects. The operations include accessing and modifying objects by pointing, selecting, and manipulating. All objects have standard resulting behaviors.

THE WEB USER INTERFACE

The expansion of the World Wide Web since the early 1990s has been truly amazing. Once simply a communication medium for scientists and researchers, its many and spread deeply into businesses, organizations, and homes around the world.

Unlike earlier text-based and GUI systems that were developed and nurtured(رعايتها) in an organization's Data Processing and Information Systems groups, the Web's roots were sown in a market-driven society thirsting for convenience and information(زرعت في مجتمع يحركه السوق متعطشا للراحة والمعلومات).

Web interface design is essentially the design of navigation and the presentation of information. It is about content, not data. Proper interface design is largely a matter of properly balancing the structure and relationships of menus, content, and other linked documents or graphics. The design goal is to build a hierarchy of menus and pages that feels natural, is well structured, is easy to use, and is truthful.

The Web is a navigation environment where people move between pages of information, not an application environment. It is also a graphically rich environment.

Web interface design is difficult for a number of reasons. **First**, its underlying (اساسية) design language, HTML, was never intended (مقصودة) for creating screens to be used by the general population.

Its scope of users was expected to be technical. HTML was limited in objects and interaction styles and did not provide a means for presenting information

in the most effective way for people. **Next**, browser navigation retreated to the pre-GUI era (حقبة). This era was characterized by a "command" field whose contents had to be learned, and a navigational organization and structure that lay hidden beneath a mostly dark and blank screen.

GUIs eliminated the absolute necessity for a command field, providing menus related to the task and the current contextual situation. Browser navigation is mostly confined to a "Back" and "Forward" concept, but "back-to where" and "forward-to where" is often unremembered or unknown. Web interface design is also more difficult because the main issues (مسائل) concern information architecture and task flow, neither of which is easy to standardize.

It is more difficult because of the availability of the various types of multimedia, and the desire of many designers to use something simply because it is available. It is more difficult because users are ill defined, and the user's tools so variable in nature. The ultimate goal of a Web that feels natural, is well structured, and is easy to use will reach fruition.

GUI VERSUS WEB PAGE DESIGN

GUI and web interface design do have similarities. Both are software designs, they are used by people, they are interactive, they are heavily visual experiences presented through screens, and they are composed of many similar components.

CONCEPT GUI WEB

• User hardware variations limited

• User hardware characteristics well defined.

• Screens appear exactly as specified.

• User hardware variations enormous (اختلافات هائلة).

• Screen appearance influenced by hardware being used (يتائر مظهر الشاشة بالاجهزه المستخدمة).

PRINCIPLES OF USER INTERFACE DESIGN

• An interface must really be just an extension of a person. This means that the system and its software must reflect a person's capabilities and respond to his or her specific needs.

• It should be useful, accomplishing some business objectives faster and more efficiently than the previously used method or tool did.

• It must also be easy to learn, for people want to do, not learn to do.

• Finally, the system must be easy and fun to use.

• The interface itself should serve as both a connector and a separator

• a connector in that it ties the user to the power of the computer, and a separator in that it minimizes the possibility of the participants damaging one another (اضرار المشاركين بعضهم البعض).

GENERAL PRINCIPLES

The design goals in creating a user interface are described below. They are fundamental to the design and implementation of all effective interfaces, including GUI and Web ones. These principles are general characteristics of the interface, and they apply to all aspects. The compilation is presented alphabetically, and the ordering is not intended to imply degree of importance.

1. Aesthetically Pleasing (تعطي جمالية)

• Provide meaningful contrast between screen elements.
• Create groupings.
• Align screen elements and groups.
• Provide three-dimensional representation.
• Use color and graphics effectively and simply.

2. Clarity

The interface should be visually, conceptually, and linguistically clear, including

• Visual elements
• Functions
• Metaphors (استعارات)
• Words and Text

3. Compatibility (التوافق)

Provide compatibility with the following:

- The user
- The task and job
- The Product
- Adopt the User's Perspective (تبني وجهة نظر المستخدم)

4. Configurability (التكوين)

- Permit easy personalization, configuration, and reconfiguration of settings.
- Enhances a sense of control
- Encourages an active role in understanding

5. Comprehensibility (الشمولية)

A system should be easily learned and understood. The flow of actions, responses, visual presentations, and information should be in a sensible order that is easy to recollect and place in context.

6. Consistency (الاتساق)

A system should look, act, and operate the same throughout. Similar components should:

- Have a similar look.
- Have similar uses.
- Operate similarly.

• The same action should always yield the same result.

• The function of elements should not change.

• The position of standard elements should not change.

7. Control

The user must control the interaction.

- Actions should result from explicit user requests.

- Actions should be performed quickly.

- Actions should be capable of interruption or termination.

- The user should never be interrupted for errors

• The means to achieve goals should be flexible and compatible with the user's skills, experiences.

8. Directness

Provide direct ways to accomplish tasks.

- Available alternatives should be visible.

- The effect of actions on objects should be visible.

9. Flexibility

A system must be sensitive to the differing needs of its users, enabling a level and type of performance based upon:

- Each user's knowledge and skills.

- Each user's experience.

- Each user's personal preference.
- Each user's habits.
- The conditions at that moment.

10. Efficiency

Minimize eye and hand movements, and other control actions.

- Transitions between various system controls should flow easily and freely.
- Navigation paths should be as short as possible.
- Eye movement through a screen should be obvious and sequential.

11. Familiarity

Employ familiar concepts and use a language that is familiar to the user. Keep the interface natural, mimicking (تقليد) the user's behavior patterns.

12. Forgiveness (المسامحة)

Tolerate and forgive common and unavoidable human errors. When an error does occur, provide constructive messages.

13. Recovery

A system should permit commands or actions to be abolished or reversed. Immediate return to a certain point if difficulties arise.

14. Responsiveness

The system must rapidly respond to the user's requests. Provide immediate acknowledgment for all user actions: Visual, Textual and Auditory.

15. Transparency

Permit the user to focus on the task or job, without concern for the mechanics of the interface. Workings and reminders of workings inside the computer should be invisible to the user.

16. Simplicity

Provide as simple an interface as possible. Five ways to provide simplicity:

- Use progressive disclosure (استخدم الكشف التدريجي), hiding things until they are needed.
- Present common and necessary functions first.
- Hide more sophisticated and less frequently used functions.
- Minimize screen alignment points (تقليل نقاط محاذاة الشاشة).

5. Task Analysis

OBSTACLES AND PITFALLS IN DEVELOPMENT PATH

•Nobody ever gets it right for the first time.

• Good design requires living in a sea of changes.

• Designers need good tools.

• Performance design goals.

• People may make mistakes while using a good system also.

COMMON PITFALLS

• No early analysis and understanding the users needs and expectations.

• A focus on using design features or components .

• No usability testing.

• No common design team vision.

• Poor communication

COMMON USABILITY PROBLEMS

• Ambiguous menus and icons.

• Languages that permit only single direction movement through a system.

• Input and direct manipulation limits.

• Complex linkage.

• Inadequate feedback.

• Lack of system anticipation (وجود ترقب للنظام).

• Inadequate (غير كافية) error messages.

DESIGN TEAM

Development, Human factors, Visual Design, Usability assesment, Documentation and Training.

HUMAN INTERACTION WITH COMPUTERS

Understanding how people interact with computers characteristics of computer systems, past and present, that have caused, and are causing, people problems. We will then look at the effect these problems have –

• Why people have trouble with computers

• Responses to poor design

Why People Have Trouble with Computers

• Extensive technical knowledge but little behavioral training.

• With its extensive graphical capabilities.

• Poorly designed interfaces.

What makes a system difficult to use in the eyes of its user?

• Use of jargon (المصطلحات).

• Non-obvious design.

• Fine distinctions (الفروق الدقيقة).

• Disparity (التباين) in problem-solving strategies.

• an "error-preventing" strategy.

• Design inconsistency (تضارب).

HUMAN CONSIDERATIONS IN DESIGN

The knowledge possessed by a person, and the experiences undergone, shape the design of the interface in many ways. The following kinds of knowledge and experiences should be identified.

• Computer Literacy- (محو الامية الحاسوبية) Highly technical or experienced, moderate computer experience, or none.

• System Experience (تجربة النظام)- High, moderate, or low knowledge of a particular system and its methods of interaction.

• Application Experience - High, moderate, or low knowledge of similar systems.

UNDERSTAND THE BUSINESS FUNCTION

•Business definition and requirements analysis

--Direct methods

--Indirect methods

--Requirements collection guidelines

• Determining basic business functions

--Developing conceptual modes

--Understanding mental models

--Users new mental model

• **Design standards or style guides**

--Value of standards and guidelines

--Document design

--Design support and implementation

• **System training and documentation**

-- Training

--Documentation

DIRECT METHODS

• Individual Face-to-Face Interview

• Telephone Interview or Survey

• Traditional Focus Group

• Facilitated Team Workshop

• Observational Field Study

• User-Interface Prototyping

• Usability Laboratory Testing

• Card Sorting for Web Sites

• A technique to establish groupings of information for web sites

INDIRECT METHODS

• Paper Surveyor Questionnaire (الاستبيان)

• Electronic Surveyor Questionnaire (استبيان الكتروني)

- Marketing and Sales
- E-Mail or Bulletin Board
- Other Media Analysis
- System Testing

UNDERSTANDING THE USER'S MENTAL MODEL (فهم نموذج المستخدم العقلي)

A goal of task analysis, and a goal of understanding the user, is to gain a picture of the user's mental model. A mental model is an internal representation of a person's current conceptualization and understanding of something. Mental models are gradually developed in order to understand, explain, and do something. Mental models enable a person to predict the actions necessary to do things if the actions have been forgotten or have not yet been encountered.

PERFORMING A TASK ANALYSIS

• Task analysis involves breaking down the user's activities to the individual task level. Knowing why establishes the major work goals; Complete description of all user tasks and interactions. Work activities are studied using the techniques just reviewed; Direct observation, interviews, questionnaires, or obtaining measurements of actual current system usage. Listing of the user's current tasks. Another result is a list of objects the users see as important to what they do.

DEVELOPING CONCEPTUAL MODELS

The output of the task analysis is the creation, by the designer, of a conceptual model for the user interface. A conceptual model is the general conceptual framework through which the system's functions are presented.
Such a model describes how the interface will present objects, the relationships between objects, the properties of objects, and the actions that will be performed. A conceptual model is based on the user's mental model. Since the term mental model refers to a person's current level of knowledge about something, people will always have them.

Guidelines for Designing Conceptual Models

• Reflect the user's mental model, not the designer's.
• Draw physical analogies or present metaphors.
• Comply with expectancies, habits, routines, and stereotypes.
• Provide action-response compatibility.
• Make invisible parts and process of a system visible.
• Provide proper and correct feedback.
• Avoid anything unnecessary or irrelevant.
• Provide design consistency.
• Provide documentation and a help system that will reinforce the conceptual model.
• Promote the development of both novice and expert mental models.

Design goals

• Reduce visual work

• Reduce intellectual work

• Reduce memory work

• Eliminate burdens or instructions.

VISUALLY PLEASING COMPOSITION

• Provide visually pleasing composition with the following qualities –

Balance, Symmetry, Regularity, Predictability, Sequentiality, Economy, Unity Proportion, Simplicity, Groupings.

The eye trends to be attracted to :

• A brighter element before one less bright

• Isolated elements before elements in a group

• Graphics before text

• Color before black and white

• Highly saturated colors before those less saturated.

• Dark areas before light areas

• A big element before a small one

• An unusual shape before a usual one

• Big objects before little objects

6. User-Centered Design and Testing

Different kinds of requirements

In software engineering, two different kinds of requirements have traditionally been identified: functional requirements, which say what the system should do, and non-functional requirements, which say what constraints there are on the system and its development. For example, a functional requirement for a word processor may be that it should support a variety of formatting styles. This requirement might then be decomposed into more specific requirements detailing the kind of formatting required such as formatting by paragraph, by character, and by document, down to a very specific level such as that character formatting must include 20 typefaces, each with bold, italic, and standard options. A non-functional requirement for a word processor might be that it must be able to run on a variety of platforms such as PCs, Macs and Unix machines. Another might be that it must be able to function on a computer with 64 MB RAM. This represents a constraint on the development activity itself rather than on the product being developed.

Environmental requirements or context of use refer to the circumstances in which the interactive product will be expected to operate.

Data gathering

The purpose of data gathering is to collect sufficient, relevant, and appropriate

data so that a set of stable requirements can be produced. Even if a set of initial requirements exists, data gathering will be required to expand, clarify, and confirm those initial requirements.

There is essentially a small number of basic techniques for data gathering, but they are flexible and can be combined and extended in many ways; this makes the possibilities for data gathering very varied, to give full leverage (تاثير ايجابي) on understanding the variety of requirements we seek. These techniques are questionnaires, interviews, focus groups and workshops, naturalistic observation (الملاحظة), and studying documentation.

Some of them, such as the interview, require active participation (مشاركة) from stakeholders (اصحاب المصلحة), while others, such as studying documentation, require no involvement at all. In addition, various props (دعائم) can be used in data-gathering sessions, such as descriptions of common tasks and prototypes (نماذج) of possible new functionality.

Studying documentation are procedures and rules are often written down in manuals and these are a good source of data about the steps involved in an activity. Such documentation should not be used as the only source, however, as everyday practices may augment them and may have been devised (ابتكر) by those concerned to make the procedures work in a practical setting. Taking a user-centered view of development means that we are interested in the everyday practices rather than an idealized account.

Other documentation that might be studied includes diaries (يوميات) or job logs that are written by the stakeholders during the course of their work. In the

requirements activity, studying documentation is good for getting some background information on the work. It also doesn't involve stakeholder time, which is a limiting factor on the other techniques.

Technique	Good for	Kind of data	Advantages	Disadvantages
Questionnaires	Answering specific questions	Quantitative and qualitative data	Can reach many people with low resource	The design is crucial. Response rate may be low. Responses may not be what you want
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee if necessary. Encourages contact between developers and users	Time consuming. Artificial environment may intimidate interviewee
Focus groups and workshops	Collecting multiple viewpoints	Some quantitative but mostly qualitative data	Highlights areas of consensus and conflict. Encourages contact between developers and users	Possibility of dominant characters
Naturalistic observation	Understanding context of user activity	Qualitative	Observing actual work gives insights that other techniques can't give	Very time consuming. Huge amounts of data
Studying documentation	Learning about procedures, regulations and standards	Quantitative	No time commitment from users required	Day-to-day working will differ from documented procedures

Task analysis

Task analysis is used mainly to investigate an existing situation, not to envision new systems or devices. It is used to analyze the underlying rationale and purpose of what people are doing: what are they trying to achieve, why are they trying to achieve it, and how are they going about it? The information gleaned (المستخلصة) from task analysis establishes a foundation of existing practices on which to build new requirements or to design new tasks.

Task analysis is an umbrella term that covers techniques for investigating cognitive processes and physical actions, at a high level of abstraction and in minute detail. In practice, task analysis techniques have had a mixed reception. The most widely used version is Hierarchical Task Analysis (HTA). Another well-known task analysis technique called GOMS (goals, operations, methods, and selection rules).

Hierarchical task analysis

Hierarchical Task Analysis (HTA) was originally designed to identify training needs. It involves breaking a task down into subtasks and then into subtasks and so on. These are then grouped together as plans that specify how the tasks might be performed in an actual situation. HTA focuses on the physical and observable actions that are performed, and includes looking at actions that are not related to software or an interaction device at all. The starting point is a user goal. This is then examined and the main tasks associated with achieving that goal are identified. Where appropriate, these tasks are subdivided into subtasks.

Prototyping and construction

It is often said that users can't tell you what they want, but when they see something and get to use it, they soon know what they don't want. Having collected information about work practices and views about what a system should and shouldn't do, we then need to try out our ideas by building

prototypes and iterating through several versions. And the more iterations, the better the final product will be.

What is a prototype?

When you hear the term prototype, you may imagine something like a scale model of a building or a bridge, or maybe a piece of software that crashes every few minutes. But a prototype can also be a paper-based outline of a screen or set of screens, an electronic "picture," a video simulation of a task, a three-dimensional paper and cardboard mockup of a whole workstation, or a simple stack of hyper-linked screen shots, among other things.

In fact, a prototype can be anything from a paper-based storyboard through to a complex piece of software, and from a cardboard mockup to a molded or pressed piece of metal. A prototype allows stakeholders to interact with an envisioned product, to gain some experience of using it in a realistic setting, and to explore imagined uses.

So a prototype is a limited representation of a design that allows users to interact with it and to explore its suitability.

Why prototype?

Prototypes are a useful aid when discussing ideas with stakeholders; they are a communication device among team members, and are an effective way to test out ideas for yourself.

Prototypes answer questions and support designers in choosing between alternatives. Hence, they serve a variety of purposes: for example, to test out the technical feasibility of an idea, to clarify some vague (مشاكل) requirements, to do some user testing and evaluation, or to check that a certain design direction is compatible with the rest of the system development. Which of these is your purpose will influence the kind of prototype you build. So, for example, if you are trying to clarify how users might perform a set of tasks and whether your proposed device would support them in this, you might produce a paper-based mockup (مجسم).

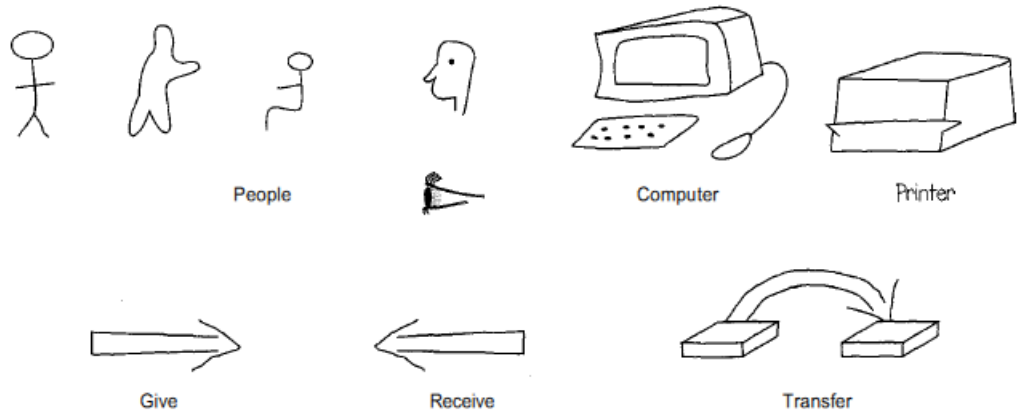
Low-fidelity protoiyping

A low-fidelity prototype is one that does not look very much like the final product. For example, it uses materials that are very different from the intended final version, such as paper and cardboard rather than electronic screens and metal. Such as is the cardboard-box in laser printer.

Low-fidelity prototypes are useful because they tend to be simple, cheap, and quick to produce. This also means that they are simple, cheap, and quick to modify so they support the exploration of alternative designs and ideas. This is particularly important in early stages of development, during conceptual design for example, because prototypes that are used for exploring ideas should be flexible and encourage (مشجعة) rather than discourage exploration and modification. Low-fidelity prototypes are never intended to be kept and integrated into the final product. They are for exploration only.

Story boarding

Story boarding is one example of low-fidelity prototyping that is often used in conjunction with scenarios. A storyboard consists of a series of sketches showing how a user might progress through a task using the device being developed. It can be a series of sketched screens for a GUI-based software system, or a series of scene sketches showing how a user can perform a task using the device. When used in conjunction with a scenario, the storyboard brings more detail to the written scenario and offers stakeholders a chance to role-play with the prototype, interacting with it by stepping through the scenario. The example storyboard shown in Figure below depicts a person using a new system for digitizing images. This example doesn't show detailed drawings of the screens involved, but it describes the steps a user might go through in order to use the system.



Sketching Low-fidelity prototyping often relies on sketching, and many people find it difficult to engage in this activity because they are inhibited about the quality of their drawing.

High-fidelity prototyping

High-fidelity prototyping uses materials that you would expect to be in the final product and produces a prototype that looks much more like the final thing. For example, a prototype of a software system developed in Visual Basic is higher fidelity than a paper-based mockup; a molded piece of plastic with a dummy key-board is a higher-fidelity.

If you are to build a prototype in software, then clearly you need a software tool to support this. Common prototyping tools include Macromedia Director, Visual Basic, and Smalltalk. These are also full-fledged development environments, so they are powerful tools, but building prototypes using them can also be very straightforward.

Type	Advantages	Disadvantages
Low-fidelity prototype	• Lower development cost. • Evaluate multiple design concepts. • Useful communication device. • Address screen layout issues. • Useful for identifying market requirements. • Proof-of-concept.	• Limited error checking. • Poor detailed specification to code to. • Facilitator-driven. • Limited utility after requirements established. • Limited usefulness for usability tests. • Navigational and flow limitations.
High-fidelity prototype	• Complete functionality. • Fully interactive. • User-driven. • Clearly defines navigational scheme. • Use for exploration and test. • Look and feel of final product. • Serves as a living specification. • Marketing and sales tool.	• More expensive to develop. • Time-consuming to create. • Inefficient for proof-of-concept designs. • Not effective for requirements gathering.

7. Introducing evaluation

What to evaluate

There is a huge variety of interactive products with a vast array of features that need to be evaluated. Some features, such as the sequence of links to be followed to find an item on a website, are often best evaluated in a laboratory, since such a setting allows the evaluators to control what they want to investigate (تحقيقه). Other aspects, such as whether a collaborative toy is robust and whether children enjoy interacting with it, are better evaluated in natural settings, so that evaluators can see what children do when left to their own devices.

There has also been a growing trend towards observing how people interact with the system in their work, home, and other settings, the goal being to obtain a better understanding of how the product is (or will be) used in its intended setting.

For example, at work people are frequently being interrupted by phone calls, others knocking at their door, email arriving, and so on-to the extent that many tasks are interrupt-driven. Only rarely does someone carry a task out from beginning to end without stopping to do something else. Hence the way people carry out an activity (e.g., preparing a report) in the real world is very different from how it may be observed in a laboratory. Furthermore, this observation has implications for the way products should be designed.

Why you need to evaluate

Just as designers shouldn't assume that everyone is like them, they also shouldn't presume that following design guidelines guarantees good usability, Evaluation is needed to check that users can use the product and like it. Furthermore, nowadays users look for much more than just a usable system.

Now that there is a diversity of interactive products, it is not surprising that the range of features to be evaluated is very broad. For example, developers of a new web browser may want to know if users find items faster with their product. Government authorities may ask if a computerized system for controlling traffic lights results in fewer accidents. Makers of a toy may ask if six-year-olds can manipulate the controls and whether they are engaged by its furry case and pixie face. A company that develops the casing for cell phones may ask if the shape, size, and color of the case is appealing to teenagers. A new dotcom company may want to assess market reaction to its new home page design.

This diversity of interactive products, coupled with new user expectations, poses interesting challenges for evaluators, who, armed with many well tried and tested techniques, must now adapt them and develop new ones. As well as usability, user experience goals can be extremely important for a product's success.

Evaluation paradigms and techniques

" Any kind of evaluation, whether it is a user study or not, is guided either explicitly or implicitly by a set of beliefs that may also be underpinned by theory. These beliefs and the practices (i.e., the methods or techniques) associated with them are known as an evaluation paradigm, which you should not confuse with the "interaction paradigms". Often evaluation paradigms are related to a particular discipline in that they strongly influence how people from the discipline think about evaluation. Each paradigm has particular methods and techniques associated with it. So that you are not confused, we want to state explicitly that we will not be distinguishing between methods and techniques. The techniques associated with usability testing are: user testing in a controlled environment; observation of user activity in the controlled environment and the field; and questionnaires and interviews.

We identify four core evaluation paradigms: (1) "quick and dirty" evaluations; (2) usability testing; (3) field studies; and (4) predictive evaluation. Other texts may use slightly different terms to refer to similar paradigms.

1. "Quick and dirty" evaluation

A "quick and dirty" evaluation is a common practice in which designers informally get feedback from users or consultants to confirm that their ideas are in line with users' needs and are liked. "Quick and dirty" evaluations can be done at any stage and the emphasis (التشديد) is on fast input rather than carefully documented findings. For example, early in design developers may

meet informally with users to get feedback on ideas for a new product. At later stages similar meetings may occur to try out an idea for an icon, check whether a graphic is liked, or confirm that information has been appropriately categorized on a webpage. This approach is often called "quick and dirty" because it is meant to be done in a short space of time. Getting this kind of feedback is an essential ingredient of successful design.

2. Usability testing

Usability testing involves measuring typical users' performance on carefully prepared tasks that are typical of those for which the system was designed. Users' performance is generally measured in terms of number of errors and time to complete the task. As the users perform these tasks, they are watched and recorded on video and by logging their interactions with software. This observational data is used to calculate performance times, identify errors, and help explain why the users did what they did.

3. Field studies

The distinguishing feature of field studies (دراسة ميدانية) is that they are done in natural settings with the aim of increasing understanding about what users do naturally and how technology impacts them. In product design, field studies can be used to (1) help identify opportunities for new technology; (2) determine requirements for design; (3) facilitate the introduction of technology; and (4) evaluate technology.

We introduced qualitative techniques such as interviews, observation, participant observation, and ethnography that are used in field studies. The exact choice of techniques is often influenced by the theory used to analyze the data. The data takes the form of events and conversations that are recorded as notes, or by audio or video recording, and later analyzed using a variety of analysis techniques such as content, discourse, and conversational analysis.

4. Predictive evaluation

In predictive evaluations experts apply their knowledge of typical users, often guided by heuristics, to predict usability problems. Another approach involves theoretically-based models. The key feature of predictive evaluation is that users need not be present, which makes the process quick, relatively inexpensive, and thus attractive to companies; but it has limitations.

Techniques

There are many evaluation techniques and they can be categorized in various ways, but in this text we will examine techniques for: observing users, asking users, asking experts, user testing and modeling users' task performance. Be aware that some techniques are used in different ways in different evaluation paradigms.

1. **Observing users (مراقبة المستخدمين)**

Observation techniques help to identify needs leading to new types of products and help to evaluate prototypes. Notes, audio, video, and interaction logs are well-known ways of recording observations and each has benefits and drawbacks. Obvious challenges for evaluators are how to observe without disturbing the people being observed and how to analyze the data, particularly when large quantities of video data are collected or when several different types must be integrated to tell the story.

2. **Asking users**

Asking users what they think of a product-whether it does what they want; whether they like it; whether the aesthetic design appeals; whether they had problems using it; whether they want to use it again-is an obvious way of getting feedback. Interviews and questionnaires are the main techniques for doing this. The questions asked can be unstructured or tightly structured. They can be asked of a few people or of hundreds. Interview and questionnaire techniques are also being developed for use with email and the web.

3. **Asking experts**

Software inspections and reviews are long established techniques for evaluating software code and structure. Guided by heuristics, experts step through tasks role-playing typical users and identify problems. Developers like this approach because it is usually relatively inexpensive and quick to

perform compared with laboratory and field evaluations that involve users. In addition, experts frequently suggest solutions to problems.

4. User testing

Measuring user performance to compare two or more designs has been the bedrock of usability testing. As we said earlier when discussing usability testing, these tests are usually conducted in controlled settings and involve typical users performing typical, well-defined tasks. Data is collected so that performance can be analyzed. Generally the time taken to complete a task, the number of errors made, and the navigation path through the product are recorded. Descriptive statistical measures such as means and standard deviations are commonly used to report the results.

5. Modeling users' task performance

There have been various attempts to model human-computer interaction so as to predict the efficiency and problems associated with different designs at an early stage without building elaborate prototypes. These techniques are successful for systems with limited functionality such as telephone systems. GOMS and the key-stroke model are the best known techniques.

8. New Interactive Technologies

Multimedia developers are incorporating augmentation, simulation, and real-time streaming in addition to static audio/video content to meet the constructivist, active-learning requirements of an evolving global learning population. To accomplish this, wireless networks need to be moved out of the cellular access network to directly interact with Internet content; enabling the development of multimedia applications for the mobile device. This is a complex issue for the industry requiring legitimate 4G telephony and extension of broadband capacity to mobile network operators.

Many of these organizations are tapping into this segment to educate them and exploit their abundant smartphone usage regardless of socio-economic status. Location, time and differences are less important now than ever before. Today‘s web authoring options such as Flash, HTML5, JavaScript™ and media players provide multimedia developers with the tools to integrate with learning management systems, augmentation and simulation on a 24/7 basis allowing customer access to information both wireless and broadband.

Providing the avenues for students and customers to access multimedia does not guarantee they will learn or buy a product. Multimedia must be combined with solid methodologies and strategies to meet the needs of the users.

Learning Management Systems (LMS)

LMS are software applications that handle all aspects of the learning process (Wikipedia). Learners sign in to the LMS and are given privileges to numerous tools and innovations to enhance the knowledge transfer. LMS have built in conferencing application capabilities, discussion forums, journals and document sharing, and grade books. Functionality includes chat with audio, text, & video. LMS can present documents, display computer screens and robust multimedia products. LMS have interoperable functionality with other multimedia software such as intranets and centralized repositories such as SharePoint. Multimedia in LMS is in support of virtual classrooms, performance-based training and just-in-time tools to create problem-based and collaborative learning.

Augmented Reality

Designers of multimedia solutions for instruction and training are responsible for finding newer and more innovative tools to connect with the real-world and real-time needs of the learners. Augmented reality is multimedia in real-time and in semantic context with environmental elements where the information about the surrounding real world of the user becomes interactive and digitally manipulable. Augmented reality fulfills the basic components of learning through layering instructions that teach and assist the users with information needed at the moment. Educators are looking for ways to actively engage students in the learning process through incorporating some type of

guided discovery that applies to the environment in which the learning exists. Augmented reality is coupled easily with constructivism in teaching and training. The learner becomes the active creator of their own knowledge through having the experience simultaneously with the media.

Move-matching, also called structure-and-motion estimation in multimedia, offer significant possibilities for multimedia innovation in terms of precision but require a lot of manual alignment and they are slow.

Marked augmentation also allows for robust interactivity between the learner and the media through a two-dimensional QR code. This allows connectivity to multimedia websites, such as the International Space Station at which tracks your location and then provides a visual of the real-time location of the Space Shuttle and your live sky chart on both desktop and mobile technologies. This type of interactivity supports a multitude of science, technology, engineering, and mathematics (STEM) instructional environments with minimal coding on the part of the course designers and developers.

Augmented Reality and QR Codes

Virtual reality (VR)

The term virtual reality originally applied only to full immersion VR, in which simulated world is projected onto all walls of a room, or via a head-mounted display (HMD) which uses motion-tracking to change the view as you turn your head. However, actual systems tended to use the glove only for gesture recognition, with all the problems of training, inference and accuracy that this implies. Marketing creep has meant that any interactive 3D environment have often been called VR, even if presented on a standard monitor, and controlled by a mouse. As games players know very well, control of view and camera angle, unless constrained by a script, can make arbitrary action in 3D scenes complex.

Tangible user interfaces

Tangible user interfaces (TUIs) use physical objects to control the computer, most often a collection of objects arranged on a tabletop to act as 'physical icons'. An immediate problem is that physical objects don't change their visible state very easily. You can include motors and displays in each object (expensive), or project overlaid AR information onto them, or just use them as multiple specialized mice/pucks that control elements of the display on a separate screen. In this case, it is necessary to track their positions, perhaps by using a large tablet device. If they are just being used as tokens to select a particular function or piece of data, an embedded RFID chip can be used to sense when they are placed within a certain distance of a reader.

Machine vision

Machine vision is a key technology for both AR and TUIs, as a way of recognizing real world objects such as buildings (in the case of outdoor AR) or objects on a desk (used for TUIs). Many current AR prototypes recognize distinctive objects from a large number of low-level visual features, as in the SIFT algorithm. Key problems are to maintain a sufficiently large database of object features, track them fast enough to give user feedback that responds to camera, gesture or object motion in real time, and do both of these in varying lighting conditions. An alternative is fiducial markers – simple visual markers such as barcodes, that can be used to more reliably identify and track objects from camera input. They are more robust to changes in camera angle and lighting than object recognition algorithms.

Video games

For many people, the most exciting, well-engineered, and commercially successful application of the direct-manipulation concepts lies in the world of video games. The early but simple and popular game PONG required the user to rotate a knob that moved a white rectangle on the screen. A white spot acted as a ping-pong ball that ricocheted off the wall and had to be hit back by the movable white rectangle. Users developed speed and accuracy in placing the "paddle" to keep the increasingly speedy ball from getting past, while the computer speaker emitted a ponging sound when the ball bounced.

Typical games provide a field of action that is visual and compelling. The physical actions-such as button presses, joystick motions, or knob rotations produce rapid responses on the screen. There is no syntax to remember, and therefore there are no syntax-error messages. If users move their spaceships too far to the left, they merely use the natural inverse action of moving back to the right. Error messages are rare, because the results of actions are obvious and can be reversed easily. These principles can be applied to office automation, personal computing, or other interactive environments.

Computer-aided design

Most computer-aided design (CAD) systems for automobiles, electronic circuitry, aircraft, or mechanical engineering use principles of direct manipulation. Building and home architects now have powerful tools such as AutoCAD. With programs like this, the operator may see a circuit schematic on the screen and, with mouse clicks, be able to move components into or out of the proposed circuit. When the design is complete, the computer can provide information about current, voltage drops, and fabrication costs, and warnings about inconsistencies or manufacturing problems. Similarly, newspaper-layout artists or automobile body designers can easily try multiple designs in minutes and can record promising approaches until they find even better ones. The pleasures in using these systems stem from the capacity to manipulate the object of interest directly and to generate multiple alternatives rapidly.