REVIEWER: HCI CHAPTER AND 5

THE OVERALL DESIGN PROCESS

HCI design is an integral part of a larger software design (and its architectural development) and is defined as the process of establishing the basic framework for user interaction (UI), which includes the following iterative steps and activities.

HCI design includes all of the preparatory activities required to develop an interactive software product that will provide a high level of usability and a good user experience when it is actually implemented.

REQUIREMENTS ANALYSIS

Any software design starts with a careful analysis of the functional requirements

For interactive software with a focus on the user experience, we take a particular look at **functions that are to be activated directly by the user through interaction** (functional requirements) and **functions that are important in realizing certain aspects of the user experience** (functional-UI requirements), even though these may not be directly activated by the user

Developing a system requires initial research, which must be conducted thoroughly from a user perspective. In order to optimize the interaction between the system and the user, it is necessary to know the user needs and characteristics, usage context, preceding systems and the competitive system aspects

For an interactive software or application that focuses on user experience, it is important to consider and assess the functions that the user will be activated directly through interaction and that are significant in understanding certain aspects of the user experience.

Requirements analysis basically establishes the goal for the development of a system from the viewpoint of the target users.

FUNCTIONAL REQUIREMENTS

- **Define the basic system behavior.**
- Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs.
- Functional requirements usually define if/then behaviors and include calculations, data input, and business processes.

NONFUNCTIONAL REQUIREMENTS (NFRS)

Define system attributes such as security, reliability, performance, maintainability, scalability, and usability.

They serve as constraints or restrictions on the design of the system across the different backlogs.

Operational Requirements

- These specify the operating environment(s) in which the system must perform, as well as how these might change over time.
- These usually refer to operating systems, system software, and information systems with which the system must interact. These might also include the physical environment, such as in what building area the system must be installed.
- For example, the system of a company must work over Web environment with Web browsers, or the system must be able to work with different operating system platforms such as Linux, Windows, and Mac OS.

Performance Requirements

- These deal with issues related to performance such as response time, capacity, and reliability of the system.
- For example, the system's database must be updated in real time, and the system will process and store data on approximately 5,000 customers for a total of about 2 Mb of data.

Security Requirements

- These address issues with security, such as who has access to the system's data and must have the ability to protect data from disruption or data loss.
- For example, only manager levels can change inventory items on the system within their own department; this way, the system's data will be encrypted to provide security and files will be scanned for viruses before saving them on the system to prevent the system from a potential threat.

Cultural and political requirements

- These deal with issues related to the cultural and political factors and legal requirements that affect the system.
- For example, the system should not use any terms or icons that might offend anyone, the system should use the English language only, and personal information of customers are not transferrable outside the country. These also address the issues related to the rules of the company.

USER ANALYSIS

This design process reinforces the original requirements analysis to satisfy potential system use more comprehensively. It also prioritizes user experience and usability standards. The results will always reflect the original requirements, which could identify supplementary user interface (UI) requirements.

The results of the user analysis will be reflected back to the requirements, potentially identifying additional UI requirements, whether functional or non-functional. It is simply a process to reinforce the original requirements' analysis to further accommodate potential users in a more complete way.

For instance, a particular age group might necessitate certain interaction features such as a large font size and high contrast, or there might be a need for a functional UI feature to adjust the scrolling speed.

SCENARIO AND TASK MODELING

This is the crux of interaction modeling: identifying the application task structure and the sequential relationships between the different elements.

With a crude task model, we can also start to draw a more detailed scenario or storyboard to envision how the system would be used and to assess both the appropriateness of the task model and the feasibility of the given requirements.

Again, one can regard this simply as an iterative process to refine the original rough requirements.

INTERFACE SELECTION AND CONSOLIDATION

For each of the subtasks and scenes in the storyboard—particularly software interface components (e.g., widgets), interaction techniques (e.g., voice recognition), and hardware (sensors, actuators, buttons, display, etc.)—choices will be made.

The chosen individual interface components need to be consolidated into a practical package because not all of these interface components may be available on a working platform (e.g., Android™-based smartphone, desktop PC, MP3 player).

Certain choices will have to be retracted in the interest of employing a particular interaction platform. For instance, for a particular subtask and application context, the designer might have chosen voice recognition to be the most fitting interaction technique. However, if the required platform does not support a voice sensor or network access to the remote recognition server, an alternative will have to be devised.

Such concessions can be made for many reasons besides platform requirements, such as constraints in budget, time, personnel, etc.

HARDWARE PLATFORMS

Different interactions and subtasks may require various individual devices, such as sensors and displays. We take a look at the hardware options in terms of the larger computing platforms, which are composed of the usual devices.

The choice of a design configuration for the hardware interaction platform is largely determined by the characteristics of the task/application that necessitates a certain operating environment. Therefore, the different platforms listed here are suited for and reflect various operating environments.

DESKTOP

Monitor (typical size: 17–42 inches; resolution: 1280×1012 or higher); keyboard, mouse, speakers/headphones (microphone).

SMARTPHONE/HANDHELD

Suited for: Simple and short tasks, special-purpose tasks.

TABLET/PAD

Suited for: Simple, mobile, and short tasks, but those that require a relatively large screen (e.g., a sales pitch).

EMBEDDED

Suited for: Special tasks and situations where interaction and computations are needed on the spot (e.g., printer, rice cooker, MP3 player, personal media player).

TV CONSOLES

Suited for: Public users and installations, limited interaction, short series of selection tasks, monitoring tasks.

VIRTUAL REALITY

Suited for: Spatial training, tele-experience, and tele-presence, immersive entertainment.

SOFTWARE INTERFACE COMPONENTS

Most of these software components are quite well-known and familiar to most readers, so we only highlight important issues to consider in the interface selection.

WINDOWS LAYERS

Modern desktop computer interfaces are designed around windows, which serve as visual output channels and abstractions for individual computational processes.

For a single application, a number of subtasks may be needed concurrently and thus must be interfaced through multiple windows.

For relatively large displays, overlapping windows may be used. However, as the display size decreases (e.g., mobile devices), non-overlapping layers (a full-screen window) may be used in which individual layers are activated in turn by "flipping" through them (e.g., flicking movements on touch screens).

ICONS

Interactable objects can be visually represented as compact and small pictograms, such as icons. Similarly, for the auditory modality, they may be represented as "earcons." Clickable icons are simple and intuitive for users.

As a compact representation designed to facilitate interaction, icons must be designed to be as informative or distinctive as possible despite their small size and compactness.

MENUS

Menus allow activations of commands and tasks through selection (recognition) rather than recall.

DIRECT INTERACTION

The mouse/touch-based interaction is strongly tied to the concept of direct and visual interaction. Before the mouse era, Human-Computer Interaction (HCI) was mostly in the form of keyboard input, involving the typing of text commands. The mouse made it possible for users to apply a direct, metaphoric "touch" upon the target objects, which are visually and metaphorically represented as concrete objects with icons. This was a departure from "commanding" the operating system via keyboard input to indirectly invoke the job.

In addition to this virtual "touch" for simple enactment, the direct and visual interaction has further extended to direct manipulation, such as moving and gesturing with the cursor against the target interaction objects. "Dragging and dropping," "cutting and pasting," and "rubber banding" are typical examples of these extensions.

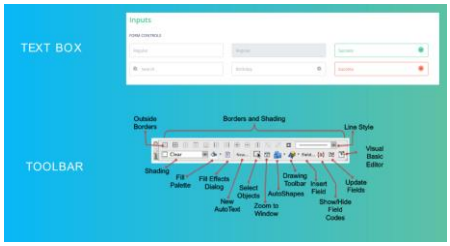
GUI COMPONENTS

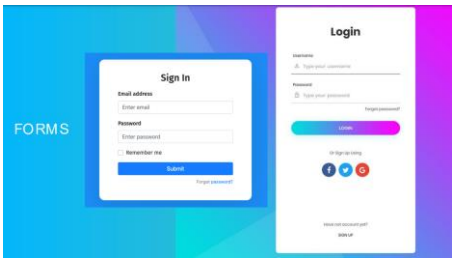
Software interaction objects are mostly visual. We have already discussed windows, icons, menus, and mouse/pointer-based interactions, which are the essential elements for the graphical user interface (GUI), also sometimes referred to as the WIMP (window, icon, mouse, and pointer).

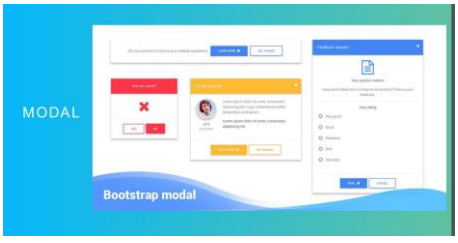
The term WIMP is deliberately chosen for its negative connotation to emphasize its contrast with a newer upcoming generation of user interfaces (such as voice/language and gesture-based). However, WIMP interfaces have greatly contributed to the mass proliferation of computer technologies.

GUI interface components:

- **Text box**: Used for making short/medium alphanumeric input
- **Toolbar**: A small group of frequently used icons/functions organized horizontally or vertically for quick direct access
- **Forms**: Mixture of menus, buttons, and text boxes for long thematic input
- **Dialog/combo boxes**: Mixture of menus, buttons, and text boxes for short mixed-mode input.







3D INTERFACE (IN 2D INTERACTION INPUT SPACE)

3D Interface: It is described as "an image that provides the perception of depth." Making a 3D image interactive and involving the user in the scene provides the experience of virtual reality.

3D interfaces are commonly presented and operated in a 2D space, which is controlled by a mouse or a touch screen. Though the 2D control for a 3D application is possible, it is often inadequate. The mismatch in the degrees of operation brings about exhaustion and inconvenience to the users.

3D interfaces are now used in developing video games and very large displays. Also, smartphones and tablets today are embedded with advanced sensors for 3D spatial input, which allows users to interact with the device.

WIREFRAMING

The interaction modeling and interface options can be concretely brought together using the so-called wireframing process. Wireframing originated from making rough specifications for website page design and resembles scenarios or storyboards. Usually, wireframes look like page schematics or screen blueprints, which serve as a visual guide that represents the skeletal framework of a website or interface.

Through wireframing, the developer can specify and flesh out the kinds of information displayed, the range of functions available, and their priorities, alternatives, and interaction flow.

CHAPTER 6: PROTOTYPE

Prototype - a prototype is an expression of design intent.

- Prototyping allows designers to present their designs and see them in action.
- In the context of digital products, a prototype is a simulation of the final interaction between the user and the interface.

Why we need Prototypes?

The primary goal of building a prototype is to test designs (and product ideas) before creating real products.

Two Cases That Requires A Prototype

ENSURE THE DESIGN CONCEPT WORKS AS INTENDED

- In most cases, it' s relatively easy to test a concept with real users. Once an interactive version of a product idea is in the hands of real users, a product team will be able to see how a target audience wants to use the product.

DETERMINE IF PEOPLE ARE ABLE USE A PRODUCT

- Prototyping is essential for finding and resolving usability issues before launch. Testing reveals areas that need improvement.

FIDELITY

Prototypes don' t necessarily looks like final products — they can have different fidelity.

- The fidelity of a prototype refers to how it conveys the look- and- feel of the final product (basically, its level of detail and realism).

Fidelity can vary in the areas of:

- Visual design
- Content
- Interactivity

There are many types of prototypes, ranging anywhere between these two extremes:

- Low- Fidelity
- High- Fidelity

LOW FIDELITY PROTOTYPING

Low-fidelity (lo- fi) prototyping is a quick and easy way to translate high- level design concepts into tangible and testable artifacts.

Basic Characteristics of low fidelity prototyping:

Visual Design: Only some of the visual attributes of the final product are presented (such as shapes of elements, basic visual hierarchy, etc.).

Content: Only key elements of the content are included.

Interactivity: The prototype can be simulated by a real human. During a testing session, a particular person who is familiar with design acts as a computer and manually changes the design' s state in real- time. Interactivity can also be created from wireframes, also known as "connected wireframes."

Pros:

Inexpensive. The clear advantage of low- fidelity prototyping is its extremely low cost.

Fast. It' s possible to create a low- fi paper prototype in just five to ten minutes. This allows product teams to explore different ideas without too much effort.

Collaborative. This type of prototyping stimulates group work. Since lo- fi prototyping doesn't' t require special skills, more people can be involved in the design process. Even non- designers can play an active part in the idea- formulation process.

Clarifying. Both team members and stakeholders will have a much clearer expectation about an upcoming project.

Cons:

Uncertainty during testing. With a lo-fi prototype, it might be unclear to test participants what is supposed to work and what isn' t. A low- fidelity prototype requires a lot of imagination from the user, limiting the outcome of user testing.

Limited interactivity. It' s impossible to convey complex animations or transitions using this type of prototype.

POPULAR TECHNIQUES

PAPER PROTOTYPING

- Paper prototyping allows you to prototype a digital product interface without using digital software.
- The technique is based on creating hand drawings of different screens that represent user interfaces of a product.

BENEFITS:

Leverage common design skills. Means that everyone can Eeveryone can sketch (even those who say they can' t) and this build paper prototypes.

Allow early testing. Testing prototypes early lets product teams find big- picture problems — such as unclear information architecture — before they become too difficult to handle.

Support rapid experimentation. Different user interface elements can be drawn, cut out, copied to make extras, and then assembled on a new piece of paper. With paper prototypes, it' s also possible to mimic complex interactions, such as scrolling.

Serve as documentation. Unlike digital prototypes, paper prototypes can be used as a reference for future iterations. Notes and revisions can be written either directly on the prototype or on sticky notes attached to the pages.

Facilitate adjustments. Using paper prototypes, it' s possible to make changes during the testing session. If designers need to add a change to the prototype, they can quickly sketch a response or erase part of the design.

PAPER PROTOTYPING LIMITATIONS

An additional person is required to conduct the test session. You'll need at least two people to conduct the test. One person will be the

facilitator ('computer') that' s helping the test participant walk through the design and the other person will be actually testing the app.

It' s hard to convey complicated operations. Paper prototypes are less suitable for visually- complex or highly- interactive interfaces.

CLICKABLE WIREFRAMES

- A wireframe is a visual representation of a product page that the designer can use to arrange page elements.
- Wireframes can be used as a foundation for lo-fi prototypes.

WIREFRAMES BENEFITS

- **Existing design deliverables can be reused.** During a particular phase of the design process, you' ll have wireframes or sketches that represent your product' s UI design. In most cases, it' s possible to use them to create a clickable flow.
- **Layouts can be easily changed.** Designers can easily adapt wireframes based on user feedback and repeat the testing process. With the right tool, it' s easy to create or modify click- through prototypes without spending a lot of extra time.

HIGH FIDELITY PROTOTYPING

- Hi-fi prototypes appear and function as similar as possible to the actual product that will ship.
- Teams usually create high- fidelity prototypes when they have a solid understanding of what they are going to build and they need to either test it with real users or get final- design approval from stakeholders.
- The basic characteristics of high- fidelity prototyping include:
- **Visual design:** Realistic and detailed design — all interface elements, spacing, and graphics look just like a real app or website.
- **Content:** Designers use real or similar- to- real content. The prototype includes most or all of the content that will appear in the final design.
- **Interactivity:** Prototypes are highly realistic in their interactions.

POPULAR TECHNIQUES

DIGITAL PROTOTYPES

The most common form of hi- fi prototyping. Nowadays, the variety of specialized

software allows designers to create visually rich, powerful prototypes full of interactive effects and complex animations.

Benefits:

Optimization for devices. Specialized software

allows designers to preview a prototype in a web browser or on any desktop or mobile device. This helps UX and UI designers achieve optimal layouts on different types of devices.

Less clarification during usability testing. High-fidelity interactivity frees the designer from having to clarify concepts during testing,

allowing the designer to focus on observation instead.

CODED PROTOTYPES

A hi- fi, coded prototype is a solution that is pretty close to the ready- to- release version of a product. An example of such prototype

would be a rich interactive sandbox that allows test participants to explore a product' s different features. This type of prototyping is recommended for designers who are confident in their coding skills.

Benefits:

Familiarity with the platform' s constraints.

Coding allows designers to understand the true capabilities and constraints of the platform they' re designing for.

Efficiency.

A coded prototype can be a good foundation for a fully- functioning app. Assuming you' re not building a one- time throwaway prototype, what you build will provide the groundwork for the final product.