Module -2
Design Process and Interaction Styles

SYLLABUS
HCI patterns, Design frameworks, Design methods, Prototyping. Understanding interaction styles - Direct Manipulation and Immersive environments, Fluid navigation Navigation by Selection, Small Displays, Content Organization, Expressive Human and Command Languages-Speech Recognition, Traditional Command Languages, Communication and Collaboration-Models of Collaboration, Design considerations.

- Design patterns play an important role when managing design knowledge for later reuse.
- In the Human Computer Interaction (HCI) community, **design patterns** are an often used **tool for sharing design knowledge among user interface (UI) designers as well as non UI experts.**
- An HCI design pattern consists of several different components.
    - The first component is the **structure of a pattern**, which encapsulates the description of the problem, its context, and the solution suggested by the pattern.
    - **Relationships and semantics** are important when design patterns are used in pattern management tools.
- To make sure that the developed patterns satisfy their users, it is important to evaluate and validate the patterns' content**.**

WHAT IS DESIGN?

- A simple definition is: **achieving goals within constraints**
- **Goals:** what is the **purpose of the design** we are intending to produce? **Who is it for**? **Why do they want it**? For example, if we are designing a wireless personal movie player, we may think about young affluent users wanting to watch the latest movies whilst on the move and download free copies, and perhaps wanting to share the experience with a few friends.
- **Constraints:** What materials must we use? What standards must we adopt? How much can it cost? How much time do we have to develop it? Are there health and

safety issues? In the case of the personal movie player: does it have to withstand rain? Must we use existing video standards to download movies? Do we need to build in copyright protection?

- **Trade-off** Choosing which **goals or constraints can be relaxed so that others can be met**. For example, we might find that an eye-mounted video display, a bit like those used in virtual reality, would give the most stable image whilst walking along. However, this would not allow you to show friends, and might be dangerous if you were watching a gripping part of the movie as you crossed the road. The golden rule of design The designs we produce may be different, but often the raw materials are the same.
- This leads us to the golden rule of design: **understand your materials**
    - understand computers
        - limitations, capacities, tools, platforms
    - understand people
        - psychological, social aspects, human error.

HCI PATTERNS

HCI patterns or **design patterns**, are a **key concept in interface and interaction design**. Originating from urban planning and later software engineering, they represent **best-practice solutions to commonly occurring problems within a given context**.

- Design patterns are described as a **way to approach design by learning from examples that have proven successful in the past**.
- They capture design practice and embody knowledge about successful solutions, often stemming from practice rather than psychological theory.
- They are reusable solutions to common problems in user-interface and interaction design.
- For novice designers, patterns are presented as **valuable "experience-in-a-can**," compensating for limited prior experience.
- **They aim to create a vocabulary that designers and users can use to communicate and document design experience**.
- Patterns are akin to guidelines but offer a more **orderly structure**, including the **problem, context, solution, examples, rationale, and cross-referencing**.
- They capture the essential common properties of good design. Instead of prescribing *how* to do something, they articulate *what* needs to be done and *why*.
- A key characteristic of design patterns is that they are **generative**, meaning they can be implemented in various ways.
- The concept of a **pattern language** is seen as generative, potentially assisting in the development of complete designs.

- Using a web design pattern language is suggested as a practical application, such as for designing an e-commerce site. Such languages can help establish a shared language and promote consistency among multiple designers.
- Patterns can **represent design knowledge at varying levels**, from social and organizational issues down to detailed widget design.
- **Limitations:** Design patterns are a distillation of previous common practice, but this practice is not always good. Design approaches representing poor practice are referred to as **anti-patterns**. A common source of anti-patterns involves migrating designs from large screens to smartphones without appropriate adaptation.

In summary, HCI patterns serve as documented, reusable solutions to common design challenges, capturing accumulated knowledge and best practices to inform and structure the design process across various scales and application areas.

DESIGN FRAMEWORKS

- HCI design frameworks **provide structured approaches to designing interactive systems.**
- They offer a set of **guidelines, principles, and methodologies** to guide the design process, ensuring a systematic and comprehensive approach to creating user interfaces.
- Characteristics:
    1. S**tructured Process:** Design frameworks provide a structured process, outlining steps from ideation to implementation, ensuring a systematic approach.
    2. **Flexibility:** Frameworks are adaptable to different project sizes, scopes, and contexts, allowing designers to tailor their application based on specific requirements.
    3. **User-Centered:** Emphasizing a user-centered approach, design frameworks prioritize understanding user needs, behaviors, and preferences.
    4. **Iterative Design:** Many frameworks support an iterative design process, encouraging designers to gather feedback, refine, and iterate on their designs.
    5. **Cross-Functional Collaboration:** Design frameworks often involve collaboration between designers, developers, and stakeholders to ensure a holistic and integrated design process.
    6. **Usability Testing:** Incorporating usability testing and evaluation methods to validate design decisions and identify areas for improvement.
- Examples:

1. **Double Diamond Design Framework:** A framework that involves divergent and convergent thinking stages, focusing on problem exploration and solution refinement.
2. **IDEO's Human-Centered Design Toolkit:** A toolkit emphasizing empathy, brainstorming, and prototyping in the design process.
3. **Lean UX Framework:** A framework that integrates user experience design with Lean and Agile methodologies, emphasizing continuous iteration and collaboration.

DESIGN METHODS

- HCI (Human-Computer Interaction) design methods are **systematic approaches and techniques used in the design and evaluation of interactive systems**.
- These methods guide designers through the various stages of the design process, helping them understand user needs, create prototypes, and assess the usability of the final product.
- Common HCI design method with its purpose

  1. User Research:
  - Method: Interviews, Surveys, Observations
  - Purpose: Understand user behaviors, needs, and preferences.
  - Benefits: Informs the design process by gathering insights from the target audience.

  2. Task Analysis:
  - Method: Task Decomposition
  - Purpose: Break down user activities into discrete tasks.
  - Benefits: Identifies the steps users take to achieve their goals, informing interface design.

  3. Card Sorting:
  - Method: Open and Closed Card Sorting
  - Purpose: Understand how users categorize information.
  - Benefits: Informs information architecture and helps create intuitive navigation.

  4. Prototyping:
  - Method: Paper Prototyping, Wireframes, Interactive Prototypes

- Purpose: Visualize and test design concepts.
- Benefits: Facilitates early feedback, iteration, and validation of design ideas.

5. Usability Testing:

- Method: Think-Aloud Protocol, Remote Testing
- Purpose: Observe users interacting with the system to identify usability issues.
- Benefits: Validates design decisions, uncovers user challenges, and guides improvements.

6. Heuristic Evaluation:

- Method: Expert Review
- Purpose: Evaluate a design against a set of usability heuristics or guidelines.
- Benefits: Identifies potential usability problems based on established principles.

7. Contextual Inquiry:

- Method: Observation, Interview
- Purpose: Study users in their natural environment to provide context for design decisions.
- Benefits: Gains insights into real-world use and uncovers user needs.

8. A/B Testing:

- Method: Comparing two versions of a design (A and B)
- Purpose: Determine which version performs better in terms of user engagement.
- Benefits: Informs design decisions based on quantitative data.

9. Cognitive Walkthrough:

- Method: Simulate the thought process of users as they interact with the system.
- Purpose: Evaluate the system's learnability for new users.
- Benefits: Identifies potential usability issues related to user understanding.

10. Wizard of Oz Testing:

- Method: Simulate system functionality with a human "wizard" behind the scenes.
- Purpose: Test interactions without fully implementing the system.
- Benefits: Gathers early user feedback without the need for a fully functional system.

11. Accessibility Evaluation:

- Method: Evaluate the design for accessibility standards.
- Purpose: Ensure inclusivity and usability for users with diverse abilities.
- Benefits: Addresses ethical considerations and broadens the user base.

- These HCI design methods can be used individually or in combination, depending on the project's goals, scope, and constraints.
- Incorporating a mix of these methods throughout the design process contributes to the creation of user-friendly and effective interactive systems.

PROTOTYPING

Prototyping is a fundamental activity in interaction design and the user interface design process.

What is a Prototype?

- A prototype is a **simple, incomplete model or mock-up of a design**.
- It is a **limited representation of a design** that allows users to interact with it and explore its suitability.
- Prototypes can take many forms, ranging from **paper-based storyboards** or **sketches** to **complex pieces of software**, or even **physical mockups** made of materials like cardboard or wood. They can also be a collection of wires and ready-made components or a digital picture.
- A prototype is limited; it usually **emphasises one set of product characteristics** while de-emphasising others.
- It provides a **concrete manifestation of an idea** that allows designers to communicate their concepts and users to try them out.
- Prototypes can also be called **physical sketches**.
- **A prototype is a preliminary model or version of a product, system, or interface. It is used to validate design concepts, test functionality, and gather feedback from users.**

Purpose and Benefits of Prototyping

Prototyping serves a variety of crucial purposes in the design process:

- **Communication:** Prototypes are a **communication device** among team members and are useful for discussing ideas with stakeholders. They can convey what a product will look like to clients. Prototypes are effective in winning contracts as they give clients a rough idea of the finished system. For mobile projects involving multiple disciplines, paper prototyping can provide a vehicle for everyone to be part of the design process collaboratively.
- **Exploring Ideas:** They allow designers to **test out ideas** for themselves. Quick sketches are important for exploring multiple alternatives in early stages. Prototyping helps in exploring different conceptual models and design ideas.
- **Obtaining User Input and Feedback:** A primary purpose is to **obtain user input** in design and **provide feedback to designers**. Users can evaluate the design by interacting with

prototypes. They help designers get feedback on **emerging designs**. Low-fidelity video prototypes can elicit comparable feedback to high-fidelity ones, but are quicker and cheaper to produce.

- **Answering Questions:** Prototypes are built to **answer specific questions** about the design's feasibility or appropriateness.
- **Choosing Between Alternatives:** They support designers in **choosing between alternative designs**. Usability criteria, technical feasibility, and user feedback on prototypes can all be used to choose among alternatives.
- **Testing Feasibility:** Prototypes can test the **technical feasibility** of an idea and its production. High-fidelity prototyping is useful for testing technical issues.
- **Clarifying Requirements:** They can clarify some **vague requirements**. Hand sketches can be used to define requirements. Programmed facades are not effective for requirements gathering.
- **Reflection in Design:** Building prototypes encourages **reflection in design**.
- **Visualisation:** A prototype enables a design to be better **visualised** and provides insights into how the software will look and work. High-fidelity prototypes are useful for demonstrating the look and feel of the final product.
- **Defining Tasks and Flow:** It aids in defining tasks, their flow, and the interface itself. Limited understanding of navigation and flow is a disadvantage of low-fidelity prototypes. High-fidelity prototypes clearly define the navigational scheme.
- **Inexpensive and Rapid Feedback:** Early prototypes, particularly low-fidelity ones, are inexpensive, rapid, and usually productive for gathering feedback. Low-fidelity prototypes support the exploration of alternatives because they are simple, cheap, and quick to produce and modify.
- **Making Ideas Tangible:** Prototyping helps you get ideas out of your head and into something more tangible that you can feel, experience, work through, play with, and test.
- **Overcoming Misunderstandings:** Prototyping is often used to overcome potential client misunderstandings. Paper prototyping provides a vehicle for reducing misunderstandings within a multidisciplinary team.
- **Supporting Iteration:** Prototypes are central to the iterative design process, allowing for repeated design-evaluation-redesign cycles involving users. Low-fidelity prototypes are quick to revise.
- **Selling Ideas:** High-fidelity prototyping can be useful for selling ideas. They can serve as a marketing and sales tool.
- **Proof-of-Concept:** Prototypes can serve as a proof-of-concept. Creating a proof of concept is a reason for conceptualising ideas initially.

Types of Prototypes

Prototypes can be categorised by their fidelity, purpose, and construction method:

- **Low-Fidelity Prototypes:**
    - Do **not look much like the final product** and do not provide the same functionality.
    - Use materials very different from the final version (e.g., paper, cardboard, sticky notes, transparencies).
    - Are **simple, cheap, and quick to produce and modify**, supporting the exploration of alternatives, especially in early stages.

    - Their very vagueness or sketchy look can communicate uncertainty and **invite improvements or rejection**.
    - Examples include **sketches**, **paper mockups**, **storyboards**, **prototypes using index cards**, and **Interactive Paper Prototypes**.
    - They are **not intended to be kept** and integrated into the final product.
    - Often **facilitator-driven** rather than user-driven.
    - Limited error checking, detailed specification, and usefulness for usability tests.
    - Useful in the early stages of design.
    - Users have been shown to respond more openly to lower-fidelity designs.
    - Paper-based techniques like PICTIVE and CARD are also low-fidelity.
- **Medium-Fidelity Prototypes:**
    - Are often called **wireframes**.
    - Provide some standardized elements and have some basic navigation functionality.
    - Medium-fidelity prototypes offer a more detailed representation than low-fidelity ones.
    - They incorporate digital elements and visuals to evaluate design aesthetics and refine user interface components.
    - These prototypes strike a balance between simplicity and detail, allowing designers to assess visual design elements effectively.
    - Tools such as **Figma, Sketch,** or **Adobe XD** are commonly used for creating medium-fidelity prototypes.
- **High-Fidelity Prototypes:**
    - Use materials expected in the final product and **look much more like the final thing**.
    - Typically provide **more functionality** than low-fidelity prototypes.
    - Often **software-based**.
    - Can also be physical models made with final-like materials.
    - More **expensive and time-consuming to develop** than low-fidelity ones.
    - Can be **fully interactive** and user-driven.
    - Clearly define navigational schemes.
    - Useful for exploration, testing, and demonstrating the look and feel of the final product.
    - Can serve as a "living" or evolving specification.
    - May incorporate actual code or be built using programming languages.
    - Provide a more realistic environment for expert reviews and usability testing compared to low-fidelity.
    - **Disadvantages** include taking too long to build, reviewers commenting on superficial aspects, developers being reluctant to change, setting expectations too high, and bugs halting testing. They can solidify the design too soon and create false expectations. They are inefficient for proof-of-concept designs and not effective for requirements gathering.
- **Wireframes:**
    - Documents showing the **structure, content, and controls** of an interface.
    - Can be constructed at varying levels of abstraction.
    - Often considered medium-fidelity.
    - Popular design method.

    - Supported by dedicated tools.
    - Initial conceptual models may be captured in wireframes.
- **Interactive Prototypes:**
    - Allow users to select from menus, click buttons, use scrolling lists, or drag icons.
    - Users can navigate through screens.
    - High-fidelity prototypes are often fully interactive.
    - Interactive paper prototypes simulate program dynamics by manually manipulating components.
    - Tools supporting the creation of interactive prototypes from static graphical elements are available.
- **Paper Prototypes:**
    - Paper prototypes involve hand-drawn representations on physical paper, offering a tangible and easily modifiable format.
    - They are particularly useful for quick ideation, testing basic interactions, and facilitating collaborative design sessions.
    - Designers commonly use simple tools like paper, markers, and sticky notes for creating paper prototypes.
- **Functional Prototypes:**
    - Prototypes can provide **limited functionality** to simulate only a small aspect for evaluation.
    - **Limited functionality simulations** require some programming support.
    - **Full functionality** can be provided at the expense of other performance characteristics like speed or error tolerance.
    - High-fidelity prototypes typically provide more functionality and can have almost complete functionality.
    - Programmed facades illustrate some program functioning but are shallow.
- **Digital/Software-based Prototypes:**
    - Digital prototypes encompass a broad category of representations, including wireframes, medium-fidelity, and high-fidelity designs created using digital tools such as Figma, Sketch, Adobe XD, InVision, or Marvel.
    - They allow for iterative testing of design elements and seamless integration with user testing platforms, providing a flexible and comprehensive approach to prototyping.
- **Storyboards:**
    - A **series of sketches** depicting the system's appearance or a user's progression through a task at different points in the interaction.
    - Can be paper-based or computer-based.
    - Often used in conjunction with scenarios.

    - Provide snapshots of the interface.
    - Allow stakeholders to role-play by stepping through a scenario.
    - Can be generated from scenarios.
    - Considered low-fidelity prototyping.
    - Dynamic visualizations, like computer-based storyboards, capture initial ideas about interface and functionality.
- **Virtual Reality (VR):**
    - Mentioned as a potential technology to consider for prototypes. Used for manipulating proteins or visualizing data. VR interfaces are listed as an interface type.
    - Virtual Reality (VR) prototypes are designed for immersive experiences, often including 3D models and spatial interactions.
    - These prototypes are crucial for testing interactions in virtual environments and simulating user experiences within a VR context.
    - Development platforms like Unity, Unreal Engine, and specific VR prototyping tools are employed for creating VR prototypes.
- **Horizontal Prototyping:** Provides a **wide range of functions but with little detail**. Useful for presenting concepts and overall organisation. A horizontal, high-fidelity prototype might be used for feedback based on first impressions.

- **Vertical Prototyping:** Provides a **lot of detail for only a few functions**. Useful for focusing on individual design aspects or tasks.

- **Throwaway Prototyping:** The prototype is **built and tested, but then discarded**. The design knowledge gained is used to build the final product from scratch. Less rigorous testing of the prototype itself is needed. The decision of whether prototype software is throwaway or the actual system software should be made upfront when using programming languages.

- **Evolutionary Prototyping:** The prototype is **not discarded** and serves as the basis for the next iteration of design. The system evolves from an initial limited version to its final release. This approach fits well with modifications needed during operation and maintenance. It requires rigorous testing of the prototype along the way.

- **Physical Models:** Creating physical representations, sometimes with interactive components, from simple wood blocks to 3D printed objects or assembled hardware components. A lump of wood prototype for the PalmPilot simulated use scenarios. Foam models were used to decide screen size and simulate display formats.

Techniques and Tools for Prototyping

Various techniques and tools are used:

- **Hand Sketches and Scenarios:** Rough hand-drawn sketches or mock-ups of screens, often used with scenarios. Can be created by hand or using a drawing package. Focusing on the design, not interface mechanics. Inexpensive and easy to modify. Can be used very early and are suited for the entire design team. Storyboards are sequences of sketches.

- **Paper Mockups / Interactive Paper Prototypes:** Using paper, sticky notes, transparencies, scissors, etc., to create interface components that can be manually manipulated to simulate dynamics. Inexpensive and quick. More illustrative of program dynamics than simple sketches. PICTIVE and CARD are specific paper-based techniques.
- **Prototyping with Index Cards:** Using cards (approx. 3x5 inches) to represent interface elements like screens, icons, or menus, allowing users to step through them. Used for developing websites and smartphone apps. Useful for exploring dialogue exchanges and user end-to-end experience. A card-based prototype can be generated from a use case.
- **Wizard of Oz:** The user interacts with a seemingly functional system, but a human operator secretly simulates the software's responses.
- **Wireframes:** Documents showing the structure, content, and controls of an interface. Supported by dedicated tools.
- **Software Tools:**
    - General-purpose drawing tools (Visio, Microsoft Paint).
    - Word processors.
    - Slide-show presentation software (Microsoft PowerPoint, Apple Keynote).
    - Graphics packages (Adobe InDesign, Photoshop, Illustrator).
    - Multimedia construction tools (Macromedia Director, Flash MX, Dreamweaver).
    - Programming languages and environments (Visual Basic, C++, Java, Smalltalk, Powerbuilder, HyperCard).
    - Dedicated prototyping environments (Balsamiq Mockups, Axure RP, Sketch, overflow.io, MockingBird, SmartDraw, Gliffy).
    - Software tools can allow designers to create more detailed high-fidelity prototypes with minimal effort.
    - Many tools use actual buttons, dropdowns, etc., but dedicated tools like Balsamiq use a sketchy, hand-drawn look to convey that it's not final.
    - Computer programs can be used to prototype specific interfaces or apps.
    - Digital tools support various design activities including sketching, simulation, and creating prototypes of various levels of fidelity.
    - Online demo tools exist (e.g., moqups).
- **Physical Models:** Creating physical representations. 3D printing is increasingly common.
- **Physical Computing Toolkits:** Kits like Arduino provide hardware and software to create physical prototypes with interactive components.
- **Software Development Kits (SDKs):** Provide a basis for construction and facilitate the transition from design to construction.

Prototyping Process:

- Clearly articulate the goals and objectives of the prototype. Identify the specific aspects of the design or user interactions that need validation or improvement. Establishing clear objectives helps guide the prototyping process effectively.

1. **Select Prototype Type:**
    - Choose the appropriate level of fidelity based on the project stage and goals.
    - Decide whether a low-fidelity sketch, wireframe, medium-fidelity prototype, or high-fidelity interactive prototype is most suitable.

- Consider the complexity of interactions and the depth of user feedback needed.

2. **Create Prototype:**
    - Use prototyping tools to build the representation of the design concept. Leverage wireframing or prototyping software such as Sketch, Figma, or Adobe XD to translate design ideas into a tangible form.
    - Pay attention to the chosen prototype type and ensure it aligns with the defined objectives.
3. **User Testing:**
    - Gather feedback through usability testing and observations. Conduct user testing sessions with the created prototype to evaluate user interactions, identify pain points, and assess overall usability.
    - Observe users as they navigate through the prototype, and document their feedback and suggestions.
4. **Iterate and Refine:**
    - Incorporate feedback and make necessary improvements. Based on the insights gained from user testing, iterate on the prototype to address identified issues, refine design elements, and enhance user experience.
    - The iterative process allows for continuous improvement.
5. **Present to Stakeholders:**
    - Share the prototype with stakeholders for review and validation.
    - Present the refined prototype to project stakeholders, including clients, product managers, or other relevant team members.
    - Seek their input, validate design decisions, and ensure alignment with project goals.
6. **Handoff to Development:**
    - Provide developers with the necessary information to implement the final product.
    - Deliver a comprehensive handoff package that includes design specifications, assets, and any relevant documentation.
    - Collaborate closely with the development team to ensure a smooth transition from prototype to the actual product.

- The prototyping process is iterative, allowing for ongoing refinement and enhancement.

- By involving users early in the design process, addressing their feedback, and collaborating with stakeholders and developers, the final product can be developed with a greater understanding of user needs and preferences.

Compromises in Prototyping

- By their nature, prototypes involve **compromises**, as they are produced quickly to test specific aspects.
- Common trade-offs include **breadth vs. depth** of functionality (horizontal vs. vertical prototyping) and **robustness vs. changeability**.
- A danger of high-fidelity prototypes is that users or developers might mistake them for the final product or become reluctant to make changes, potentially compromising engineering principles if integrated into the final product.
- Potential managerial challenges include underestimating the **time and cost** of prototyping.

UNDERSTANDING INTERACTION STYLES

Interaction styles are the methods by which a user and a computer system communicate with one another. The choice of interface style can have a profound effect on the nature of this communication, seen as a dialog between the computer and the user. Designers can choose from various primary interaction styles when designing interfaces. The choice of interaction style may be limited by the type of system being developed and the characteristics of the input and output devices used. Blending several interaction styles may be appropriate when the required tasks and users are diverse.

Different interaction styles include:

- **Command line interfaces**
    - This is the **oldest and original user interaction style**.
    - It requires the user to press a function key or type a command into a designated entry area on a screen. Commands can be single characters, abbreviations, words, or multiple words and codes.
    - It provides a means of expressing instructions to the computer directly.
    - While it was the first interactive dialog style commonly used, it is still widely used, often supplementary to menu-based interfaces, providing accelerated access for experienced users.
    - **Advantages**: Powerful, offering direct access to system functionality, flexible (commands can have options or parameters), and can be applied to many objects at once, making it useful for repetitive tasks. It conserves screen space and appeals to expert users.
    - **Disadvantages**: Commands must be remembered, as no cue is provided on the screen to indicate which command is needed, testing one's power of recall. They can be cryptic and obscure with complex syntax. Commands are very prone to, and intolerant of, typing errors that can lead to user frustration. Commands also often vary across systems, causing confusion and increasing learning overhead. Command line interfaces are therefore better for expert users than for novices. Using consistent and

meaningful commands and abbreviations can alleviate some of these difficulties. Commands used should be terms within the vocabulary of the user.
    - Grammars for command languages are effective. Command syntax, semantics, and sequences are aspects of interface design.
    - An example includes a Second Life command-based interface for visually impaired users.
- **Menus-based interfaces**
    - This is listed as a **primary interaction style**.
    - They present the user with a list of alternatives or choices from which the user selects an item.
    - Menus are a major form of navigation through a system and, if properly designed, assist the user in developing a mental model of the system.
    - They rely on recognition memory, which is easier than recall.
    - **Advantages**: Shortens learning, reduces keystrokes, structures decision making, permits use of dialog-management tools, and allows easy support of error handling. They reduce interaction complexity, eliminate memorizing commands, syntax errors, and reduce typing. The user knows what is available.
    - **Disadvantages**: Menus must be structured logically and have meaningful descriptions. They may slow knowledgeable users and consume screen space. Experienced users may find continually stepping through a series of menus to achieve an objective time-consuming and frustrating. Design must consider the conflicting needs of inexperienced and experienced users.
    - Graphical and Web systems are heavily menu-oriented and vary in form. Common graphical menu types include menu bars, pull-down, pop-up, cascading, tear-off, and iconic menus. Web menus include textual links to other pages, command buttons, and graphical and textual toolbars.
    - Menu selection is done via pointing devices in graphical systems. WIMP interfaces include menus. Precise behavior of menu elements can differ between environments (e.g., pull-down vs fall-down menus, MacOS vs Windows menu behavior).
    - Menu design considerations include task-related organization, single menus, combinations of multiple menus, content organization, fast movement through menus, and data entry with menus, form filling, and dialog boxes.
- **Question-answer and Query Dialog boxes interfaces**
    - **Question and answer dialog** is a simple mechanism for providing input to an application in a specific domain. The user is asked a series of questions (mainly with yes/no responses, multiple choice, or codes) and so is led through the interaction step by step.
    - They are easy to learn and use but are limited in functionality and power. As such, they are appropriate for restricted domains (particularly information systems) and for novice or casual users.
    - An example of this would be web questionnaires.
- **Forms Fills and spreadsheet interfaces**
    - **Form fill-in** is listed as a primary interaction style.
    - It requires the user to type information into labeled fields. This style is good for data entry, such as completing an online order.
    - It simplifies data entry.

    - Requires screen form design that helps the user understand its purpose and allow fast and easy entry of information. Conversely, a poorly designed screen form can be inefficient and aggravating to complete.
    - Dialog boxes commonly include forms. A downside of dialog boxes is the tendency to cram too much information or data entry fields, making the interface confusing.
    - **Spreadsheets** are listed as an interaction style alongside form-fills. VisiCalc spreadsheets are mentioned as an example in the context of the evolution of direct manipulation.
- **Point and click**
    - This interaction technique is a **key component of graphical user interfaces and direct manipulation.**
    - In direct manipulation, typed commands are replaced by a pointing action on the object of interest. Users point at visual representations of objects and actions to carry out tasks rapidly.
    - Users typically select screen objects and actions by using pointing mechanisms, such as the mouse or joystick, instead of the traditional keyboard. Operations on objects include accessing and modifying them by pointing, selecting, and manipulating.
    - "Pick-and-Click" interaction is a characteristic of graphical interfaces. It involves identifying an object, picking it, and then performing an action such as clicking, double-clicking, or dragging. This encourages system learning through exploration.
    - Pointing devices are key elements of WIMP interfaces. The mouse controls the cursor as a point of entry to interface elements.
    - A variety of pointing devices exist, including the mouse, stylus, eye-gaze, gestural, and haptic devices, presenting both opportunities and challenges. Design issues include speed, accuracy, fatigue, error correction, and subjective satisfaction.
    - Novel pointing devices are used in games. More direct pointing interfaces emerged as computing advanced.
- **3D interfaces**
    - These are listed as an interaction style, specifically referred to as "three-dimensional interfaces".
    - Game designers lead the way in creating visually compelling 3D scenes with characters controlled by novel pointing devices.
    - While 3D interfaces are distinct, WIMP elements can be given a 3D appearance using shading to highlight active areas.
    - Virtual reality involves applications of direct manipulation.
    - Virtual reality is listed as an interface type, with research and design considerations discussed.
- **WIMP Interfaces**
    - The WIMP interface is described as the **most common and complex** interface style.
    - It is the **default interface style for the majority of interactive computer systems in use today**, particularly in the PC and desktop workstation arena.
    - WIMP stands for **windows, icons, menus, and pointers** (sometimes windows, icons, mouse, and pull-down menus).
    - Examples include Microsoft Windows, MacOS, and various X Windows-based systems for UNIX.
    - The Xerox Star interface led to the birth of the graphical user interface (GUI), which was originally called a WIMP.

- Elements of the WIMP interface include windows, icons, pointers, and menus, as well as buttons, toolbars, palettes, and dialog boxes.
- While WIMP systems appear to have virtually the same elements, the precise behavior of these elements differs both within a single environment and between environments.
- Interactivity, focusing on how elements react to user actions, is crucial in determining the 'feel' of a WIMP environment. Dialog design is centered on the sequences of actions and corresponding changes in the interface state.
- Windowing systems are the foundation of modern WIMP interfaces.
- The WIMP interface, as a paradigm, focuses on addressing the single user and allows for flexible switching between tasks.
- Early WIMP interfaces had a boxy design, and interaction primarily occurred through windows, scroll bars, checkboxes, panels, palettes, and dialog boxes. Developers were often constrained by the available widgets, with the dialog box being prominent.
- GUIs have evolved and been adapted for mobile devices and touchscreens, using swipe and touch input.

DIRECT MANIPULATION AND IMMERSIVE ENVIRONMENTS

Direct Manipulation:

- Direct manipulation is a user interface interaction paradigm that **allows users to interact with digital objects on the screen in a physical and intuitive manner.**
- Unlike traditional command-line interfaces or indirect methods, direct manipulation provides a more **hands-on experience by enabling users to directly engage with graphical representations of objects.**
- This approach **enhances user control, responsiveness, and overall user experience.**
- Definition: Direct manipulation (DM) is an **interaction style in which users act on displayed objects of interest using physical, incremental, reversible actions whose effects are immediately visible on the screen.**

Characteristics:

1. **Graphical Interaction:** Users interact with visual elements, such as icons or objects, rather than relying on abstract commands or textual inputs.
2. **Immediate Feedback:** Actions produce instant visual feedback, providing users with a direct and tangible connection between their input and the system's response.
3. **Physical Metaphors**: Design often incorporates real-world metaphors, making the interface more intuitive. For example, dragging an icon mimics physically moving an object.

4. **User Engagement:** Promotes a more engaging and immersive user experience by allowing users to directly manipulate on-screen elements.

Examples:

1. **Dragging and Dropping Files:** Users can move files or folders by clicking and dragging them to a new location, providing a tangible sense of relocation.
2. **Resizing Windows:** Users can adjust the size of application windows by dragging the edges or corners, allowing for customization based on preferences or content.
3. **Rotating and Zooming in Images:** Touch gestures on touch-enabled devices, such as pinching to zoom or rotating with two fingers, provide a natural way to manipulate images.
4. **Painting and Drawing Applications:** Users can directly draw or paint on a digital canvas using a stylus or touch input, simulating the experience of traditional artistic tools.
5. **Interactive Maps:** Users can zoom in and out, pan, and rotate maps directly on touchscreens or with pointing devices to explore geographic information.

Advantages:

- Intuitiveness: **Mimics real-world interactions,** making it easier for users to understand and learn.
- Efficiency: **Reduces the cognitive load** by eliminating the need for users to memorize commands or navigate complex menus.
- User Satisfaction: Provides a **satisfying and responsive user experience,** contributing to increased user satisfaction.

Disadvantages:

- Device Dependency: Some direct manipulation interactions may be **device-specific,** requiring touch screens or specific input devices.
- **Complexity**: While intuitive for simple interactions, designing effective direct manipulation interfaces for complex tasks can be challenging.
- **Consistency**: Ensuring consistency in direct manipulation metaphors across different applications or platforms can be crucial for user understanding.

2D and 3D Interfaces

2D Interfaces:

- A 2D interface involves **two-dimensional representations, typically displayed on a flat surface** like a computer screen or a piece of paper.

- In this environment, **information is presented in the form of images, text, and graphics that have width and height but lack depth.**
- Common examples of 2D interfaces include traditional computer desktops, websites, and mobile applications.
- Interactions in 2D interfaces often rely on input devices like a mouse, keyboard, or touch screen, where users navigate and interact with objects on a flat plane.

3D Interfaces:

- A 3D interface, on the other hand, extends into the third dimension, **adding depth to the visual representation.**
- In a 3D environment, objects have height, width, and depth, creating a more **realistic and spatial representation.**
- This type of interface is often used in applications where a sense of depth and perspective is essential, such as computer-aided design (CAD), virtual reality (VR), and simulations.
- In 3D interfaces, **users can interact with and manipulate objects as if they exist in a three-dimensional space.**
- This can enhance the user's understanding of complex structures, models, or virtual environments.
- Input devices for 3D interfaces may include tools like 3D mouse, motion controllers, or VR headsets that enable users to navigate and manipulate objects in a spatial context

Augmented Reality (AR):

- Augmented Reality (AR) is a technology that **enriches the real-world environment by overlaying digital information, such as graphics, text, or 3D models, onto the user's physical surroundings.**
- Unlike Virtual Reality (VR), **AR does not replace the real world but enhances it by providing additional context or interactive elements.**
- AR experiences can be accessed through devices like smartphones, tablets, smart glasses, and heads-up displays.
- Users interact with AR content in real-time, often using gestures, touch input, or voice commands.

- Common applications include navigation assistance, educational tools, and engaging gaming experiences where digital elements interact with the real world.
- Examples of AR include navigation apps that display directions on a live camera feed and social media filters that augment users' faces with virtual effects.

Virtual Reality (VR):

- Virtual Reality (VR) creates a **fully immersive, computer-generated environment that transports users into a digital world, isolating them from their physical surroundings.**
- VR typically **requires specialized hardware** like VR headsets or goggles, equipped with sensors for tracking head and body movements.
- Users interact with the virtual environment using devices such as motion controllers, providing a more immersive and natural experience.
- VR finds extensive **applications in gaming, simulations for training purposes, and virtual tourism.**
- In VR, users experience a sense of presence, feeling as if they are physically present in the simulated environment.
- Notable examples include VR gaming platforms like Oculus Rift and HTC Vive, as well as VR applications in healthcare for realistic training simulations and therapy.

Immersive Environments:

Immersive environments offer **users a deeply engaging and interactive experience, often within virtual or augmented reality settings.**

- These environments strive to simulate a **sense of presence**, allowing **users to feel as if they are physically present in a computer-generated or enhanced environment**.
- Immersive technologies leverage devices such as **headsets, motion sensors, and other sensory inputs to create a convincing and interactive user experience.**

Examples:

1. **Virtual Reality (VR) Simulations**: Users are placed in computer-generated 3D environments, providing a fully immersive experience.
2. **Augmented Reality** (AR) Applications: Digital information is overlaid onto the real world, enhancing the user's perception of the physical environment.
3. **3D Interactive Environments**: Used for design, training, or educational purposes, allowing users to interact with 3D objects in a virtual space.

Advantages:

- **Engagement**: Provides a high level of **user engagement and immersion, enhancing the overall experience.**
- **Training and Simulation**: Valuable for training scenarios, simulations, and educational applications.
- **Innovative Experiences**: Enables new and innovative ways of interacting with digital content and information.

Disadvantages:

- Hardware Requirements: Often **requires specialized hardware** such as VR headsets, which can be a barrier to widespread adoption.
- Motion Sickness: Some users may experience **motion sickness or discomfort** in immersive environments.
- **Content Creation**: Developing high-quality content for immersive environments can be **resource-intensive and complex**

FLUID NAVIGATION

Fluid navigation is a design goal that **aims to enable users to move through interactive systems effectively and comfortably.** The goal for designers is to enable fluid navigation that allows users to **gracefully and confidently get to where they want to go, explore novel possible routes, and backtrack when necessary**.

Navigation is distinct from search, which typically involves users typing keywords into a search box. Instead, navigation harnesses users' ability to rapidly skim choices, **recognise what is relevant, and select what they need**. It depends on the recognition of landmarks that users utilise to guide their choices.

Techniques such as **small or large menus, embedded links, or tool palettes are described as the "workhorses" of navigation**. **Users interact with these techniques using a touch, tap, swipe of the finger, or a pointing device, receiving immediate feedback on their actions.**

Navigation by selection is an interaction style that is particularly effective for users who are novice or first-time users, are knowledgeable intermittent users, or need help structuring their decision-making processes. With careful design, including complex menus and rapid interaction, menu selection can also be appealing to expert frequent users. Careful design, keyboard shortcuts, and gestures can allow expert users to navigate quickly through large information structures.

Interactive visualization of information can also help analysts navigate large amounts of data in a fluid visual manner. Related concepts like pan and zoom operations allow users to control the size and position of the viewport on a visualization, which can be useful for focusing on relevant items.

These operations are common in conventional user applications like Google Maps, Adobe Photoshop, and Microsoft Word.

- Navigation harnesses users' ability to rapidly skim choices, recognize what is relevant, and select what they need to realize their intentions.
- The goal for designers is to enable fluid navigation that allows users to gracefully and confidently get to where they want to go, explore novel possible routes, and back-track when necessary.
- Navigation depends on recognition of landmarks that travelers use to guide their choices, which differs greatly from search, which requires users to describe what they want by typing keywords in a blank search box.
- However, with careful design of complex menus and rapid interaction, menu selection can be appealing even to expert frequent users.
- These strategies can be used in combination with command languages, allowing users to transition smoothly from novice to expert because menus offer cues to elicit recognition rather than forcing users to recall the syntax of a command from memory.
- Careful design, keyboard shortcuts, and gestures allow expert users to navigate quickly through large information structures.
- Is key to:
    - successfully operating interactive applications, such as installing a mobile app, completing an online survey, or purchasing a train ticket (**task navigation**)
    - finding information on a website or browsing social media (**web navigation**)
    - finding the action needed in a desktop application (**command menu navigation**)
- Note that even if a designer uses slick graphical menus, elegant form fill-in, or well known gestures there is no guarantee that the interface will be appealing and easy to use
- Effective interfaces emerge only after careful consideration of and testing for numerous design issues, such as:
    - task-related organization
    - phrasing of items
    - sequence of items
    - graphic layout and design
    - responsive design to adapt to various sizes of devices

- shortcuts for knowledgeable frequent users
- online help and error correction

NAVIGATION BY SELECTION

- Graphical techniques are a particularly attractive way to present choices while providing context to help users specify what they want.
- Interactive visualization of information can also help analysts navigate large amounts of data in a fluid visual manner.
- The simplest case of explicit menus is a binary menu for yes/no, true/false choices.
- When users need to make a series of choices (e.g., in a survey or to select parameters of an application), there are well-established methods of presenting choices.
- Radio buttons support single-item selection from a multiple-item menu, while checkboxes allow the selection of one or more items in a menu.
- A multiple-selection menu is a convenient method for handling multiple binary choices, since the user is able to scan the full list of items while deciding.

Navigation by selection is a key interaction style contributing to **fluid navigation**. Fluid navigation aims to enable users to move through interactive systems effectively and comfortably, allowing them to gracefully and confidently get to where they want to go, explore novel routes, and backtrack when necessary.

In navigation by selection, users are presented with a set of options or choices, review them, and select the one most appropriate to their task. After making a selection, users observe the effect. The user typically indicates their choice using a pointing device such as a mouse, trackball, light pen, or finger, or by using a keystroke or key. Visual feedback is provided to indicate the option selected, often by highlighting the choice. Separate actions for selection and execution are often provided.

This interaction style is effective because it harnesses users' ability to rapidly skim choices, recognize what is relevant, and select what they need, rather than requiring them to recall information from memory. This recognition-based approach is less demanding of the user than recall-based methods like command languages. It also provides a clear structure for decision making by presenting all possible choices at once.

Navigation by selection is particularly appropriate for users who are novice or first-time users, knowledgeable intermittent users, or those who need help structuring their decision-making processes. If the terminology and meaning of the items are understandable and distinct, users can accomplish tasks with little learning or memorization. With careful design, including complex menus and rapid interaction, menu selection can also be appealing and rapid for expert frequent users. Techniques like careful design, keyboard shortcuts, and gestures allow expert users to navigate quickly through large information structures.

Techniques and Menu Types used for Navigation by Selection:

- **Menus (general):** Can be small or large. Range from trivial choices between two items to complex systems with thousands of displays. Can be single, in linear sequences, tree structures (most common), acyclic, or cyclic networks. Navigation using menus assists the user in developing a mental model of the system.
- **Pull-downs:** Modern menus often appear as pull-downs, which drop down from a menu bar.
- **Checkboxes and Radio buttons:** Used in dialog boxes or menus; checkboxes allow multiple selections, while radio buttons support single-item selection.
- **Embedded links (Hotlinks):** Links selectable within text or graphics. They allow items to be viewed in context, eliminating the need for separate item enumeration. Popularised by the Hyperties system and inspired World Wide Web hotlinks.
- **Tool palettes:** Used in addition to menus, and can be moved.
- **Dialog boxes:** Contribute to allowing users to express choices.
- **Pop-up menus:** Appear on the display in response to a click or tap.
- **Context menus:** Pop-up menus whose content depends on cursor position.
- **Circular menus (Pie menus, Marking menus):** Options arranged in a circle. Advantageous as the pointer is equidistant from selections, potentially making selection easier and quicker. With practice, they can be used without visual attention by memorizing directions.
- **Scrolling menus:** Display a portion of a long list with navigation options (like arrows or typing) to access other items. Can be less effective for long lists than hierarchical menus.
- **Combo boxes:** Combine a scrolling menu with a text-entry field, allowing users to type characters to quickly scroll through the list.
- **Fisheye menus:** Display all items but show items near the cursor at full size and others smaller. Can improve speed over scrolling menus for certain list sizes.
- **Sliders and Alphasliders:** Used for selecting continuous numerical values (sliders) or items from a large categorical list (alpha sliders). Often used in interactive visualization systems.
- **Two-dimensional menus (Multiple-column menus, Mega menus):** Present choices in columns or a grid, giving an overview and reducing scrolling. Useful in web page design.
- **Cascading menus:** Selecting an item opens another menu adjacent to it, allowing refinement. Requires precise control and can lead to errors.
- **Collapsible menus:** Allow further options to be made visible by selecting a header, providing an overview and reducing scrolling.
- **Simultaneous menus:** Multiple menus are displayed together. Can be effective for experienced users doing complex tasks and are a powerful application in faceted search interfaces to browse large databases.
- **Split menus and Folded menus:** Techniques to provide faster access to frequently used items by positioning them at the top or showing them first.
- **Graphical menus:** Use visual representations like maps or calendars for selection options while providing context. Visual search interfaces provide context and help users refine needs.
- **Audio menus:** Used in interactive voice response (IVR) systems or when hands and eyes are busy; important for public access and visually impaired users.
- **Novel designs combining selection and direct manipulation:** Examples include modified pie menus for specifying parameters, FlowMenu which chains selections and direct manipulations without lifting the pointing device, and Toolglass which uses two-handed operation.
- **Gestures:** Can serve as a shortcut for rapid selection, especially on touchscreens.

Design Considerations for Navigation by Selection:

Effective design of navigation by selection requires several considerations:

- **Content Organisation:** Meaningful **grouping and sequencing of choices** based on task semantics or other logical schemes. Alphabetic, categorical, or frequency-of-use ordering can be used; categorical is generally preferred, but frequency-of-use can speed access to top items in small lists. Dynamically reordering menus based on use can be disruptive.
- **Phrasing and Labels:** Clear and consistent terminology, phrasing, and layout are crucial. Menu titles should accurately represent the contents and ideally mirror the selection made on the previous menu. Link text should match destination page headings.
- **Structure:** Choose appropriate structures (**single, linear, tree, network**) based on the task. Breadth-shallow structures are often preferred over narrow-deep ones, but excessively broad or deep structures can challenge navigation. Effective **sub-grouping** reduces perceived breadth. For hierarchical menus, a concave structure (broad initial, narrow intermediate, broader terminal) may be optimal for browsing.
- **Navigation Aids and Feedback:** Provide explicit cues to elicit recognition. **Show current position or path** (navigation history). Enable landmarks to show themselves. Use intent indicators to show if more information is needed or if an item leads to a submenu. Provide maps or overviews of the menu hierarchy or site structure. Provide clickability cues. Change link color after visiting a destination. Maintain a sense of place (e.g., through a home base, recurring elements).
- **Efficiency and Shortcuts:** Allow rapid display and selection mechanisms for frequent users. Offer type ahead, jump ahead, keyboard shortcuts, or gestures. Position initial cursor at the most likely choice. Minimize the distance traveled for common tasks.
- **Error Prevention and Recovery:** Errors are more preventable in menu selection systems compared to command languages. Design cascading menus to minimize errors. Allow users to backtrack or return to previous menus. Permit erroneous selections to be unselected before execution.
- **Handling Large Lists:** For long lists, consider **scrolling menus, combo boxes, fisheye menus, sliders, alpha sliders, or two-dimensional layouts**. However, requiring scrolling should ideally be avoided on navigation-only pages.
- **User Flexibility:** Allow different **selection techniques** (pointing, keyboard) to provide flexibility. Provide both simple and complex menu sets for novice and expert users. Support dynamic queries and filtering to narrow choices.
- **Accessibility:** Navigation by selection should consider accessibility issues, such as for visually impaired users. Audio menus are important for blind or vision-impaired users.

Navigation is distinct from search, which typically involves typing keywords. While search initiates the process in vast information spaces, navigation techniques like menus are the workhorses for moving through systems. However, some interfaces, like faceted search, tightly integrate category browsing (navigation) with keyword searching. Browsing, a form of navigation by selection, often works better than searching for less specific items and is preferred by users due to less effort, better context, and faster access to content. Navigation links and a search function should ideally be included on all key web pages.

SMALL DISPLAYS

S**mall displays** are used on a wide range of devices, including mobile phones, digital cameras, smartwatches, and wearables.

The **primary challenge** is their **limited screen space**, which makes most desktop designs impractical and often leads to failure if simply "dumbed down". This limitation results in a more **temporal interface**, showing only a portion of the information at any one time.

Designing for small displays requires a **radical rethinking** of functionality and often leads to **novel interface designs**. Key design considerations include:

- **Simplification:** The rule "Less is More" is important, focusing on essential functions.
- **Content:** Concise writing is crucial, as "Every word counts". Abbreviations may be necessary, but consistency is needed. **Reduce short-term memory load** by avoiding designs where users must remember information across displays.
- **Navigation:** This often relies on dedicated keys or soft keys with dynamically changing labels. Menu styles are adapted; flat lists are good for few options, while sequential menus may be needed for hierarchical content. Simultaneous menus requiring multiple filters are harder to fit. **Scrolling is a constant challenge.**
- **Data Entry: Avoid data entry as much as possible** as it is very difficult. Using contextual information (like location) or making selectable items (like phone numbers or dates) "tap-away" can help. Complex tasks might be handed off to larger devices.
- **Responsiveness:** Designing for mobile first can be an effective strategy. Touch interfaces require sufficiently large touch targets.

Small displays are **portable** and provide a **private interaction experience**. Display characteristics like **resolution, color, and glare** are important. New technologies are expanding possibilities, such as electronic ink for paperlike resolution or pico projectors to share content externally.

Small displays play a significant role in Human-Computer Interaction (HCI) across various devices, influencing how users engage with technology in their everyday lives. HCI refers to the design and use of computer technologies, with a focus on the interaction between humans and computers. Here are some key contexts where small displays are commonly encountered in HCI:

- Mobile Devices:
    - ***Smartphones****:* Small displays on smartphones are fundamental to HCI, providing users with a compact yet powerful interface for various applications, communication, and information retrieval. Touchscreens on these devices enable direct manipulation, enhancing the user experience.
    - ***Smartwatches****:* Wearable devices, like smartwatches, incorporate small displays that allow users to receive notifications, track fitness, and interact with apps directly from their wrists. The design of these displays prioritizes quick and convenient access to information.

- Wearable Technology:
    - ***Fitness Trackers****:* Devices such as fitness trackers often feature small displays to provide real-time feedback on health metrics, activity levels, and notifications, fostering user engagement with their personal wellness.
    - ***Augmented Reality Glasses****:* Some AR glasses use small displays to overlay digital information onto the user's field of view, facilitating hands-free interaction with the environment and digital content.
- Embedded Systems:
    - ***Embedded Interfaces****:* Small displays are employed in various embedded systems, such as digital cameras, MP3 players, and handheld gaming devices, allowing users to interact with these devices through a compact visual interface.
- Human-Robot Interaction:
    - ***Robotic Interfaces****:* In the realm of robotics, small displays are often integrated into human-facing robots, enabling communication and conveying information to users in a visually accessible manner.
- Interactive Gadgets:
    - ***Consumer Electronics****:* Small displays are utilized in interactive gadgets, including remote controls, e-readers, and digital cameras, to provide users with control options, status information, and user-friendly interfaces.

In HCI, the design of small displays focuses on optimizing usability, responsiveness, and visual clarity within the constraints of size. It involves considerations such as touch gestures, navigation menus, and the presentation of information to ensure a seamless and effective user experience.

CONTENT ORGANIZATION

C**ontent organization** is a fundamental principle and practice in designing user interfaces. It refers to **how information, options, elements, or data are arranged and structured within a display, menu, system, or collection to make it comprehensible, memorable, and easy for users to interact with and navigate.**

The **primary goal** of content organization is to create a **sensible, comprehensible, memorable, and convenient structure that is relevant to the user's tasks**. Effective display designs must provide all the necessary data in the proper sequence to carry out a task. Well-organized content facilitates task performance, speeds up learning, increases navigation speed, and lowers the burden on a user's short-term memory. It helps users recognize essential elements and ignore secondary information. Well-organized information hierarchies can even be as effective as search engines for finding information.

Content can be organized in various ways:

1. **Grouping:**
    - Items or information are **grouped into categories of similar items**. This is a common way to create a **tree structure** when dealing with a large collection of items.
    - Groups should be **logical, distinctive, meaningful, and mutually exclusive, maximising the similarity of items within a category and minimising similarity across categories.**
    - **Examples** of grouping include: separating produce, meat, and dairy in an online grocery store; grouping word processing commands by function (e.g., customise, compose, edit, print); or grouping countries, states, and cities hierarchically.
    - On screens, related fields can be grouped by contiguity or similar patterns, or surrounded by boxes. Unrelated fields should be kept separate by blank space.
    - Providing **meaningful titles** for groupings aids search and is preferred by users.
    - Perceptual principles like **proximity** (elements close together are perceived as a group) are used to establish visual groupings. Adequate separation using white space or borders enhances this.
    - For navigation, links of a similar purpose and function should be grouped.
2. **Ordering:**
    - Items or groups of items are **sequenced in a logical or meaningful way.** This helps users anticipate where items will be located.
    - Ordering schemes can include:
        - **Natural sequence:** such as chapters in a book, days of the week, or months of the year.
        - **Task-related:** reflecting the sequence of steps to accomplish a user's goals.
        - **Frequency of use:** placing the most frequently used items first.
        - **Importance:** placing the most important items first or in a prominent position.
        - **General to specific:** common for hierarchical data, where broader categories precede more detailed ones.
        - **Alphabetical:** useful for long lists (eight or more items) or short lists with no obvious pattern.
        - **Semantic similarity:** ordering items along a dimension like impact or emphasis.
        - **Conventional:** following established patterns.
    - Ordering normally reflects a combination of these techniques.
    - For lists or menu choices, a columnar orientation is preferred for faster scanning. Left-justification is common for text and alphanumeric data.
    - Consistent ordering should be maintained across related menus.
3. **Structuring Hierarchies:**
    - Large collections of items are often organised into hierarchical **tree structures** by forming categories at different levels.
    - While hierarchical decompositions can be natural and comprehensible, deep hierarchies can be difficult to navigate, and users may get lost if the groupings are unfamiliar.
    - It is often better to have broad, shallow top-level categories or present several levels of menu on one screen.
    - A well-balanced hierarchical tree is a recommended organisation scheme for Websites. A document structure (table of contents, chapters, sections) is familiar and effective.

    - **Taxonomies** are formal hierarchical structures that classify phenomena into understandable categories. They are used to impose order and can guide designers. Examples include taxonomies for input devices, tasks, user experience levels, and information visualizations.
    - Headings and subheadings are used to indicate hierarchical structure and classify information on screens and Web pages. Establishing a hierarchy of font styles and sizes for headings helps communicate structure and importance.
4. **Layout and Formatting:**
    - The physical arrangement of elements on a screen or page is crucial. **A clear and clean layout, using space and lines, supports comprehension and makes screens visually pleasing.**
    - Related fields can be shown by contiguity or similar structural patterns. Unrelated fields can be separated by blank space.
    - Using adequate **white space** is essential for readability and organization, separating groupings and making pages inviting. It should be considered a design component.
    - **Alignment** of elements contributes to clarity and guides the user's eye.
    - Using similar sizes, shapes, or colours for related information promotes **unity**.
    - **Consistency** in design, ordering, and positioning contributes to clarity and predictability.
    - For multi-part displays (like sections or pages), they should appear to go deeper from left to right and top to bottom.
    - Reducing graphic complexity and textual density towards the bottom of a page can help users focus on content.
5. **Relevance and Filtering:**
    - Only **display information necessary to perform actions, make decisions, or answer questions**. Irrelevant information is noise.
    - Ensure information that must be compared is visible simultaneously, avoiding the need for users to remember things across different views or scrolled content.
    - For search results or large collections, providing overviews of the number of items in categories (**faceted search**) is a useful strategy when metadata is available.

Content Organization in Specific Contexts:

- **Menus:** Task-related menu organization is a primary concern. Content organization is a heavily researched area for menus. Rules for forming menu trees and guidelines for content phrasing and ordering are provided.
- **Displays/Screens:** Organizing the display involves grouping, ordering, layout, and ensuring only relevant information is presented.
- **Web Pages/Sites:** Web content organization deals with structuring the entire site (depth vs. breadth, number/length of pages) and individual pages (layout, components, links). A clear and meaningful site structure is crucial because the Web is a nonlinear hypertext system. Content should be organized naturally for the audience, reflecting either the information structure or the users' tasks. Headings and subheadings play a key role in organising and making Web page content scannable. Using white space and formatting techniques aids readability and scanning. Navigation elements should be clearly differentiated and consistently located.
- **Help and Documentation:** Documentation should be organised into topics and subtopics. Content can be managed as single-source topics for reuse in different formats. Well-labeled

headings and text formatting (lists, tables, short paragraphs) make content scannable and understandable.
- **Information Retrieval and Data:** Data can be organised in structured databases, textual documents, or multimedia libraries. Supplemental finding aids like tables of contents and indexes help users understand the content structure. Visualisation techniques and taxonomies are used to organise and explore data. Categorising items based on content analysis is used in tasks like automatically analysing news stories. Organizing personal information (like documents, images) for later retrieval is a design challenge, often involving folders, albums, or tags.

Techniques like **card sorting** are used to establish hierarchical groupings and the information architecture for Web sites by asking users to group content topics placed on cards. **Content analysis** and **grounded theory** are methods for analysing data (including text, video, etc.) by classifying it into themes or categories. Determining meaningful and non-overlapping categories is a challenging aspect of these techniques.

In summary, effective content organization ensures information is presented clearly, logically, and in a manner that supports user tasks, facilitating efficient navigation, reducing cognitive load, and improving overall usability and satisfaction.

EXPRESSIVE HUMAN AND COMMAND LANGUAGES

"Expressive human and command languages" refers to **how users communicate with computer systems using language-based methods**, including both formal command languages and attempts at natural language interaction.

- **Command languages** require users to **recall and type specific syntax and commands to initiate actions**. They are powerful for frequent, expert users, allowing rapid specification of complex tasks and the creation of macros or scripts for automation. However, they are **difficult to learn and retain** for novices, can be **error-prone**, and may lack visual context or cues. Design benefits from meaningful names, consistent abbreviations, and structure.
- **Natural language interfaces** (like spoken commands or typed queries) aim to let users interact using everyday language. While intuitively appealing, they are challenging due to the **subtlety and complexity** of human language. They often require **clarification dialog**, can be **unpredictable**, may not show context, and the "habitability" (knowing what the system understands) is a key difficulty.
- Speech recognition is improving, especially for specific tasks or familiar phrases, enabling hands-free use, but faces obstacles like noise, accents, and error correction.
- Well-designed messages and text in any interface should be **brief, simple, and directly usable**. This involves using short sentences (under 20 words) and paragraphs (under 6 sentences, one idea), clear, familiar, and complete words, and an affirmative, active voice with a proper tone.
- Some example speech technologies are (Box 9.1):
    - Store and replay (museum guides)

• Dictation (document preparation, web search)

• Closed captioning, transcription

• Transactions over the phone

• Personal “assistant” (common tasks on mobile devices)

• Hands-free interaction with a device

• Adaptive technology for users with disabilities

• Translation

• Alerts

• Speaker identification

SPEECH RECOGNITION

The concept of speaking to computers has long been a dream for researchers and visionaries. While true natural conversation with computers remains challenging, significant progress has been made in speech interfaces and human language technologies.

Speech recognition is the technology that enables computers to **convert spoken input into text or commands**. Its goal is primarily to **produce text based on spoken input.** Although it has been a promising area for years, it is still used mainly in limited situations. While reported recognition rates can exceed 97%, this still represents a significant error rate (e.g., one spelling mistake every six or so words), particularly for unrestricted vocabulary. Early systems were often limited to discrete-word recognition, sometimes requiring extensive user training. Recent decades have seen major breakthroughs in continuous-speech recognition algorithms, aided by large online voice data repositories for training. Speaker-independent systems, which require little or no user training, have expanded the scope of commercial applications. Recognition rates can be improved by using quiet environments, high-quality microphones, and carefully chosen, limited vocabularies.

However, significant challenges remain:

- **Background noise** and variable speech performance make recognition difficult.
- Recognising word boundaries in continuous speech is challenging due to **normal speech patterns blurring** them.
- **Diverse accents, variable speaking rates, and changing emotional intonation** cause problems.
- Systems can be unstable across different users, environments, and time.
- Errors occur when words in the vocabulary are similar (e.g., "dime/time" or "Houston/Austin").
- **Matching semantic interpretation and contextual understanding** remains the most difficult problem.
- Speech input can impose a higher **cognitive load** compared to pointing devices, potentially disrupting tasks, as it relies more on working memory. Hand/eye coordination, processed elsewhere in the brain, allows for higher parallel processing.
- Correcting errors can be very taxing, especially without a keyboard or pointing device.

Applications of Speech Recognition

Speech recognition is used successfully in a growing number of applications, particularly when alternative input methods are difficult or impossible. Applications include:

- **Interactive Voice Response (IVR) systems** for telephone-based transactions and information, such as checking flight times, ordering prescriptions, or accessing banking services. Simple IVRs can be seen as audio menus.
- **Dictation** for document preparation, email composition, and web searches. Specialized systems have found commercial success in fields with specific jargon, such as medical or legal professions.
- Enabling **hands-free interaction** when users' hands are busy (e.g., while driving, performing maintenance checks, or in military applications).
- Providing **adaptive technology for users with disabilities**, such as those with physical or visual impairments. Screen readers for the blind rely on textual descriptions of visual elements.
- **Personal "assistants"** on mobile devices (e.g., Siri, Google Now, Cortana, Hound) for common tasks like finding locations, setting reminders, and communicating.
- Automatic scanning and retrieval of specific words or topics from recorded speech (e.g., radio, TV, court proceedings, telephone calls).
- Generation of **closed-caption text** for television.
- **Speaker verification** for security systems.
- **Translation** applications.
- Use in **toys**.

For common computing applications with displays, speech input often offers marginal benefits. Studies show it can be slower for tasks like cursor movement and web browsing but can improve performance in tasks where frequently moving the cursor (e.g., between a diagram and a tool palette) is time-consuming. Combining voice with direct manipulation may be useful.

Designing Spoken Interaction

When designing speech interaction, it is crucial to determine if using voice is appropriate for the task and user context.

- Often, **integrating voice with visual channels and displays is preferable**, as reading on a screen is faster than listening, and visual displays allow for rapid selection and provide context. Access to a keyboard for error correction is also helpful.
- Designers must define what users can say and what the system can reliably recognise. **Learnability** is a key issue.
- In IVR systems, prompts guide users, and the possible transactions are kept simple. Speech recognition can allow users to shortcut menus if they know the name of what they seek.
- For personal assistants, users face the burden of learning effective commands; providing command examples helps, but habitability remains a key problem.
- **Error correction** design is critical; having correction commands distinct from dictation, allowing easy deletion of the last spoken text, and offering alternative text or spelling options are important.

- The success of many systems relies on being **limited to narrow application domains** where the world of actions and vocabulary is constrained. Careful selection of easily differentiated terms dramatically improves recognition.
- A careful **task analysis** is the first step to determine necessary functions, actions, and objects, which informs the command design. Observing users speaking "naturally" can reveal potential commands.
- Meaningful, specific names for commands aid learning and retention.
- Feedback, often showing the recognised text, is important. Spoken feedback can be generated by systems integrated tightly with speech recognition.

Speech Production (Synthesis)

Speech production or synthesis is the technology that **generates spoken output**. It is successful for simple, short messages, especially when visual channels are overloaded, mobility is required, or conditions are unsuitable for displays. There are three general methods:

- **Formant synthesis** produces artificial, machine-generated speech using algorithms.
- **Concatenated synthesis** combines recorded human speech segments (phonemes, words, phrases) to create more natural-sounding sentences.
- **Canned speech** uses a fixed set of pre-recorded, digitised segments assembled for longer messages; while the number of possible sentences is limited, it can sound natural, though seams between segments may be awkward. IVR systems often mix canned segments and synthesis.

Obstacles to speech output include:

- The **slow pace** compared to visual displays.
- The **ephemeral nature** of speech, making review or browsing difficult.
- **Difficulty in scanning/searching** spoken messages.
- **Acceptability and privacy issues** in public spaces.
- Intolerance of imperfections; synthesised speech can sound monotonic or robot-like, although a computer-like sound may sometimes be preferred. Achieving natural intonation requires understanding the domain.

Successful applications of speech generation include:

- Providing **alerts**.
- Giving instructions, such as in **automobile navigation systems**.
- In **IVR systems** where hiring live agents is impractical.
- Providing **access for the blind and visually impaired** via screen readers and book readers that convert text to speech.
- Serving as a **communication tool for people with physical disabilities** affecting their speech.
- Adding **voice annotation** to documents.
- Used in **toys and consumer products**.
- Synchronising with **"talking heads"** for lip-reading or low-bandwidth video calls.
- Supplementing displays when visual attention is focused elsewhere.

Non-speech auditory interfaces (tones, sounds, music) can also provide feedback or information, sometimes assimilated faster than speech and not language-dependent.

Human Language Technology (HLT)

Human language technology (HLT) encompasses **computational techniques for dealing with human language, aiming towards systems that can process, understand, or generate natural language**. While people have long dreamed of computers that truly understand natural language, achieving full comprehension and generation of open-ended language remains an inaccessible goal due to the subtlety, complexity, context dependency, and emotional aspects of human communication.

Current successes in HLT do not rely on true "understanding" or parsing of intent but rather on **statistical methods based on analysing vast datasets** of text or spoken language. Applications include:

- **Question-answering systems** that match queries to previously asked questions or use usage logs to predict user needs.
- **Grammatical error detection and proofreading** in instructional systems.
- Automated scoring of essays.
- Providing **feedback in natural language** in tutoring systems.
- Machine **translation** between human languages, increasingly using statistical methods based on large databases of human translations.
- Serving as an important part of **search technology**.

HLT faces challenges like the complexity of language structure (syntax, semantics, prosody), ambiguity, and the need for clarification dialogue. Despite research investment, widespread use is sometimes limited because alternative interfaces are more appealing.

TRADITIONAL COMMAND LANGUAGES

A **command language** is an **interaction style where users type specific commands, often followed by arguments or qualifiers, into a designated area on a screen or terminal to instruct a computer system.** These commands are typically **brief**. In the early days of computing, the **command-line interface** was the first interactive dialogue style commonly used. While graphical user interfaces (GUIs) led to a recession in their widespread use, command languages are still widely used. Natural language interaction can even be considered a complex form of command language.

Characteristics and Use Cases

- Command languages **originated with operating system commands**. Early systems often used command lines, sometimes requiring interaction via virtual character-based terminals within windows for remote access. Examples include DOS and Unix. Specialized languages for mathematics, music, and chemistry emerged because they facilitated communication and problem solving.
- **Mechanism:** Users issue a command and observe the result. This is contrasted with menu-selection, where users view or hear options and respond. Command-language users must recall notation and initiate action. Commands can be single characters, abbreviations, words, or multiple words and codes. Key combinations (like Shift+Alt+Ctrl) and fixed

function keys (delete, enter, undo) are also ways of issuing commands, and some function keys can be programmed as specific commands.

- **Purpose:** Commands have immediacy and impact on devices or information. They are used to **express instructions directly to the computer**. A typical form is a verb followed by a noun object with qualifiers or arguments, such as PRINT MYFILE 3 COPIES.
- **Expert Users:** Command languages are often the domain of **expert frequent users**. They are attractive when frequent system use is anticipated, users are knowledgeable about the task and interface concepts, screen space is limited, response time is slow, and numerous functions can be combined in a compact expression. Experts gain satisfaction from mastering a complex set of semantics and syntax.
- **Specialised Applications:** Command languages are still used by expert users of specialized applications, such as computer programmers and engineers/scientists using tools like MATLAB, which combine a command language and graphical environment. Database-query languages like SQL are also a form of command language, attractive for users who want additional control over searching databases.
- **Assistive Technology:** Command-line interfaces do not require a pointing device, which can make them useful for users with visual impairments, for whom mouse and touchscreens may be impractical. Command-based interfaces have been developed for visually impaired people to interact in virtual worlds.
- **Modern Contexts:** Twitter tags (#hcil, $TWTR, @benbendc) and text-message abbreviations (LOL) can be considered examples of new command languages that need to be learned and remembered. The spread of speech interfaces is re-invigorating the development of command languages, as designers choose which word combinations will be recognised as spoken commands.

Advantages

- **Power and Flexibility:** Command languages are powerful, offering direct access to system functionality. They allow rapid specification of complex tasks. Commands are flexible and can incorporate options or parameters to vary their behaviour, and can be applied to many objects at once, making them useful for repetitive tasks.
- **Feeling of Control:** For frequent users, command languages can provide a strong feeling of being in control.
- **Efficiency for Experts:** They allow users to express complex possibilities rapidly without having to read distracting prompts. System administrators, programmers, and power users often find them much more efficient and quicker. Experts can rapidly specify actions involving multiple options.
- **Scripting and Macros:** Powerful advantages include easy history keeping and simple macro creation. Command histories are sometimes kept, and macros can be created. A complex sequence of commands can be easily specified and stored for future use as a macro or script. Macro facilities can be full programming languages, allowing extensions and tailoring to personal work styles.

Disadvantages

- **Learning Curve:** Command languages require users to learn and retain specific syntax. They can be difficult for novices to learn and retain. Commands must be remembered, as there is no

cue provided on the screen. This requires memorization and typing. The burden of learning effective commands falls on the user.

- **Error Proneness:** Error rates are typically high. They are very prone to, and intolerant of, typing errors. Correcting errors can be very taxing.
- **Lack of Context/Assistance:** Command-language interaction usually provides little context for issuing the next command. Error messages and online assistance can be hard to provide due to the diversity of possibilities and complexity of mapping tasks to interface concepts and syntax. Command prompts can help with simple errors like incorrect syntax, but they assume knowledge of the command.

Design Considerations

- **Task Analysis:** Designers should begin with a careful task analysis to determine what functions should be provided. This helps determine the necessary functions, actions, and objects.
- **Syntax and Structure:** Commands may have simple or complex syntax. A typical form is a verb followed by a noun object with qualifiers or arguments. Imposing a **meaningful structure** on the command language can be highly beneficial. Sources of structure that have proved advantageous include positional consistency, grammatical consistency, congruent pairing, and hierarchical form. Hierarchical structure (e.g., verb-object-qualifier) and congruence (meaningful pairs like ADVANCE/RETREAT) have been shown to be advantageous in experiments, aiding memory and problem-solving and lowering error rates. Consistency is a relevant design principle.
- **Naming:** Command names are the most visible part of a system. Meaningful, specific names aid learning and retention. Using terminology familiar to the user community and keeping a list promotes consistent use. Commands should use terms within the vocabulary of the user rather than the technician. Specific commands may be more descriptive and memorable than general ones, and studies have shown better recall and recognition for specific command names. Names should be distinct from one another. Command names should reflect the actions that will occur.
- **Abbreviations:** Permitting abbreviations may be useful. Brevity is a worthy goal, as it can speed entry and reduce error rates. For expert users, abbreviations become attractive and even necessary. A consistent strategy for abbreviation is important. Potential strategies include simple truncation, vowel drop with truncation, first and final letter, first letter of each word, standard abbreviations, and phonics. A simple primary rule for most items, with a secondary rule for conflicts, is recommended. Designers might first instruct new users to type full commands and later introduce abbreviations.
- **Harmony with Entry Mechanism:** Commands should be in harmony with the keyboard, using brief and kinesthetically easy codes. Commands requiring SHIFT or CTRL keys, special characters, or difficult-to-type sequences are likely to cause higher error rates.
- **Feedback and Errors:** Feedback may be generated for acceptable commands, and error messages may result from unacceptable forms or typos. Error messages should be clear and constructively guide the user. Auto-completion is critical to help prevent errors.
- **Command Menus and Shortcuts:** To relieve the burden of memorization, some designers offer users brief prompts of available commands in a format called a command menu. Experienced users may not need to read the prompt, while intermittent users use it to jog memory. Keyboard shortcuts also remain heavily used by users who take the time to learn them. These shortcuts can provide accelerated access for experienced users and offer more

rapid performance than mouse selection for experts. Command menus on high-speed displays can be effective.

While grammar notations like BNF are useful to specify textual commands and are good for describing command-based interfaces, they are not as good for graphical interfaces and may not adequately model the directionality of events or information flow. Task-action grammar (TAG) is an approach that attempts to incorporate concepts like consistency and user world knowledge, which standard BNF lacks, potentially providing a better predictor of learning and performance for command languages. However, grammar-based techniques largely ignore system output and assume users execute commands blindly.

COMMUNICATION AND COLLABORATION

Communication and collaboration are central aspects of human activity, deeply intertwined with the design and use of technology. Human beings are inherently social, living, working, learning, playing, interacting, and talking together. Interactive systems are developed to support and extend these different kinds of sociality.

Face-to-Face Communication and Interaction: Face-to-face interaction is described as being in the same place at the same time. Despite its technological simplicity, it is considered the most sophisticated communication mechanism available due to the complex interplay of channels. It involves not just speech and hearing, but also the subtle use of **body language**, **eye gaze**, tone of voice, and facial expressions. These non-verbal cues are crucial for enabling smooth conversation and providing **back channels** (like "aha's" and "umms") that signify agreement or manage turn-taking.

When people transition to computer-mediated communication, they carry forward their expectations and social norms from face-to-face interaction. Breaking these unconscious rules can lead to a feeling of unease or rudeness.

In face-to-face settings, especially when working together, people use **deictic reference** – pointing or gesturing to items. The presence of electronic equipment or screens can interfere with this, potentially requiring technological aids like **group pointers** on shared displays.

The **physical layout** of a room profoundly affects face-to-face working relationships. For example, the positioning of monitors can block participants' views of one another, reducing opportunities for eye contact and hindering the interpretation of gestures and body position. Meeting rooms are sometimes designed with recessed monitors to mitigate these problems. Different layouts can encourage or limit social interaction and cooperative work in classrooms or offices.

Group Dynamics and Working: Group working is considerably more complex than single-user activities. It is influenced by the physical environment and dynamic social relationships among participants. Evaluating systems designed for groups presents additional challenges compared to single-user systems. Even minor factors, such as the arrangement of chairs in a room, can significantly impact group behaviour.

Introducing technology into an organisation can interfere with existing authority and social relationships within work-groups and the organisation as a whole. Potential problems include a mismatch between who performs the work and who benefits, as well as the issue of "free riders"

benefiting from the group's effort with little personal contribution. Understanding normal human-human communication is seen as a fundamental step in grasping the social dimension of technology.

For designing systems for groups, it may be useful to focus on analyzing the **external representations** used by participants rather than attempting to understand individual cognitive processing. Effective group working is also supported by mechanisms like **distributed coordination**, where the division of labour, clarity of responsibilities, and individuals' appreciation and orientation towards the work, tasks, and roles of others are important.

Communication and Collaboration Technologies:

The field dedicated to studying technology used by two or more people emerged in the 1980s as **Computer-Supported Cooperative Work (CSCW)**. It encompasses cooperative, collaborative, and competitive work, as well as non-work activities like play, family activities, education, social exchanges, learning, games, and entertainment. **Groupware** is a term often used to describe team-oriented commercial products in this area. **Computer-mediated communication (CMC)** refers specifically to systems that support direct communication between participants. **Direct collaboration** is also suggested as a potential motivating force for computer use.

People collaborate for various reasons, whether for purely emotionally rewarding purposes or specific task-related goals. Collaborative technologies support a wide range of activities, including communication, conversation, creative production, meeting coordination, and education.

The traditional way to categorize collaborative interfaces is using a **time/space matrix**:

- **Same Place, Same Time (Synchronous Local):** These support face-to-face collaboration using shared technology in the same physical space. Examples include control rooms, meeting rooms, and spaces with wall displays. Technologies include electronic meeting rooms, systems with large interactive screens and digital whiteboards, shareable interfaces like multi-touch tables, and systems for co-located shared screens. These environments can support structured processes like brainstorming, voting, and ranking, sometimes allowing anonymous contributions to encourage participation and reduce ego conflicts. However, technology can sometimes interfere with communication, reducing eye contact or turning discussions into monologues.
- **Different Place, Same Time (Synchronous Distributed):** These allow people in different locations to collaborate simultaneously. Technologies include chat, instant messaging (IM), texting, audio conferencing, and video conferencing. Video conferencing can provide richer experiences by allowing participants to see facial expressions and body language. Studies indicate a clear voice channel is important for coordination when users are looking at shared objects. **Telepresence** systems aim to give remote participants an experience close to being physically co-present, sometimes using immersive environments or robotics.
- **Different Place, Different Time (Asynchronous Distributed):** These systems support collaboration where participants are in different places and interact at different times. Examples include electronic mail, newsgroups, listservers, discussion boards, blogs, wikis, and online communities. **Electronic mail** is widely appreciated for its simplicity and speed, good for conveying facts, but can be loosely structured, overwhelming, or difficult for late joiners. **Online communities** can be lively spaces providing useful information and support,

common for distance education. Discussion groups are often moderated to remove offensive or useless items. Social network sites are a popular form of online communication. **Text-based communication** is dominant in asynchronous groupware.

- **Same Place, Different Time (Asynchronous Local):** These support collaboration where participants are in the same place but interact at different times. Examples include equipment logs or group calendars. Public displays can be used for this purpose. The Notification Collage allows posting information to a large public display or private office displays to share information and promote awareness.

While the time/space matrix is a useful descriptive model and common language in the CSCW community, its binary nature is an oversimplification, as modern tools can blur the lines between synchronous and asynchronous interaction.

Technology and Social Interaction Computing has transformed into a social process, moving away from the introversion and isolation of early computer users. Networks and mobile devices have expanded possibilities for communication. Technology acts as a vector of social change, and its impact should be assessed before introduction. The success of collaborative systems often depends heavily on accommodating existing social and organizational structures, community values, and creating acceptable social norms. Training for online interaction needs to include **etiquette** (netiquette) and cautions about problematic behaviour like 'flame wars'.

Successful design requires understanding user needs for learning from colleagues, consulting partners, annotating documents, and presenting results. Designers must also consider issues like accommodating interruptions, dealing with privacy, and establishing responsibility. Key design considerations include facilitating common ground, providing social cues, managing activity awareness and interruptions, protecting privacy and identity, supporting reputation systems, understanding diverse user motivations, enabling leadership roles, and dealing with deviance and the need for moderation. Designing interfaces to protect users from hostile or malicious behaviour is a significant challenge. As people and technology co-evolve, the design of communication tools must be an iterative process.

Meeting and Decision Support Systems are specifically aimed at helping groups generate ideas and reach decisions. These can be face-to-face (often enhanced by technology) or remote. They can support structured processes and facilitate contributions through shared workspaces or displays.

Shared Applications and Artifacts allow groups to work together on digital objects like documents, images, or code. Communication can occur directly between participants or indirectly **through the artifact** itself (feedthrough), where one person's manipulation of shared objects is observed by others. The **granularity of sharing**, in terms of object size and frequency of updates, varies across different systems. Maintaining **WYSIWIS** (what you see is what I see) across displays is often important for establishing context.

Understanding the social dimension of technology involves studying how people communicate and collaborate through ethnographic studies and using conceptual frameworks like distributed cognition, which analyses the interactions among people, artifacts, and processes in a system. Socio-technical models also emphasize considering technical, social, organizational, and human aspects side-by-side in design.

Ultimately, the **determinants of success** for collaborative interfaces are still not entirely clear. Challenges include overcoming threats to existing power structures, achieving a sufficient **critical mass** of users, avoiding violations of social taboos, and overcoming resistance to change. Measures of success can include system utilization, user satisfaction, sense of belonging, or impact on productivity and goal achievement, but evaluating these can be complex.

MODELS OF COLLABORATION

Understanding communication and collaboration is a key aspect of designing interactive systems. Because working environments often have social aspects, extrapolating current trends suggests that most computer-based tasks will become collaborative. Models, taxonomies, and frameworks are abstractions of reality that, while incomplete, can help to make sense of the design space for collaborative activities and solve specific practical problems.

Here are some models and frameworks related to collaboration:

1. **The Time/Space Matrix (or Framework)**
    - This is described as the **traditional way to decompose or classify collaborative interfaces**.
    - It focuses on two critical dimensions: **time** (same time or different time) and **space** (same place or different places).
    - This creates four quadrants:
        - **Same Place, Same Time (Synchronous Local):** Supports face-to-face collaboration using technology in the same physical space, such as in control rooms, meeting rooms, or with shared table/wall displays.
        - **Different Place, Same Time (Synchronous Distributed):** Allows people in different locations to collaborate simultaneously, using technologies like chat, instant messaging (IM), texting, audio conferencing, or video conferencing.
        - **Different Place, Different Time (Asynchronous Distributed):** Supports collaboration where participants are in different places and interact at different times, such as via electronic mail, newsgroups, listservers, discussion boards, blogs, wikis, and online communities.
        - **Same Place, Different Time (Asynchronous Local):** Supports collaboration where participants are in the same place but interact at different times, like using equipment logs or group calendars.
    - Terms like synchronous (same time), asynchronous (different time), co-located (same place), and remote (different place) are often used.
2. **Model of Coordinated Action (MoCA)**
    - This is presented as a **more contemporary framework** compared to the time/space matrix.
    - It operates at the mezzo and macro level and expands upon the traditional model.
    - It shifts towards a deeper understanding of "**coordinated action**," defined as the "interdependence of two or more actors who, in their individual activities, are working towards a particular goal through one or more overlapping fields of action".

- This definition accounts for situations of **diffuse or even indirect collaboration**, such as crowdsourcing or collaborative recommendation engines.
- The framework has **seven dimensions**:
  - **Synchronicity** (asynchronous to synchronous)
  - **Physical Distribution** (same location to different locations)
  - **Scale** (Number of Participants, from 2 to N)
  - **Number of Communities of Practice** (0 to N, reflecting diversity of disciplines, ways of working, language, culture, etc.)
  - **Nascence** (routine to developing, referring to the novelty of the coordinated action)
  - **Planned Permanence** (short-term to long-term)
  - **Turnover** (low to high, referring to the stability of participants)
- These dimensions influence the design of communication tools but also the nature, qualities, and intents of the people involved.
- The MoCA model can help designers understand and think through diverse contexts for collaboration, such as conversations, markets, meetings, creative work, entertainment, crowdsourcing, and education. Universal designs aim to accommodate users across the spectrum of these dimensions.

3. **Distributed Cognition and DiCoT**

   - **Distributed Cognition** is an approach or theory used to analyze how people carry out work, focusing on the interactions among people, artifacts, and external representations.
   - It involves examining aspects like distributed problem-solving, verbal and non-verbal behaviour, coordination mechanisms, communicative pathways, and knowledge sharing.
   - An analysis can identify problems and predict the consequences of different arrangements of technologies and artifacts.
   - **Distributed Cognition of Teamwork (DiCoT)** is a framework developed to support the application of distributed cognition in practice.
   - DiCoT provides a framework of models constructed from collected data (like ethnographic studies or interviews).
   - Models suggested within DiCoT include:
     - **Information flow model**: shows how information flows and is transformed.
     - **Physical model**: captures how physical structures support communication and access to artifacts.
     - **Artifact model**: captures how artifacts support cognition.
     - **Social structure model**: examines how cognition is socially distributed.
     - **System evolution model**: depicts how the system has evolved over time.

4. **The Language/Action Framework**

   - This framework is based on the premise that **people act through language**.
   - It was developed to inform the design of systems aimed at improving work effectiveness by enhancing communication.
   - It draws on speech act theory and models interactions as structured "conversations for action" (CfA), involving specific types of speech acts like requests, offers, and promises.

5. **Socio-technical Models**

    - These models emphasize the importance of considering both the **human (social) and technical aspects** of a system in parallel during design to achieve a compatible solution.
    - They are relevant because organizational issues and work-groups influence and are influenced by computer systems.
6. **Contextual Design Models**

    - Within the Contextual Design methodology for understanding users' work, several models are created during interpretation sessions to capture different aspects of work, including collaboration and communication.
    - Relevant models include:
        - **Physical model**: shows the physical work environment and constraints.
        - **Flow model**: shows lines of coordination and communication between participants.
        - **Artifact model**: shows how people organize and structure work and manipulate objects.
        - **Sequence model**: shows steps in a task.
        - **Affinity diagram**: groups notes and findings along theme lines.
    - These models help the design team establish a shared understanding and serve as "communication devices" for dialogue.

These different models and frameworks provide various lenses through which to understand, analyze, and design for the complexities of human communication and collaboration, particularly in the context of computer-supported cooperative work (CSCW) and groupware.

DESIGN CONSIDERATIONS

Designing interactive systems involves considering a multitude of factors to ensure the final product is usable, effective, and provides a positive user experience. **Design is described as a deliberate, iterative process of creating specifications for synthetic artefacts like products and services.** It moves from establishing requirements through conceptual and detailed physical design, building, implementation, and evaluation. Evaluation and design are integrated and should be carried out together.

Here are some key design considerations :

1. **Understanding the User(s)**

    - This is paramount in a user-centered approach, where users' concerns direct development over technical ones.
    - Consider user characteristics like physical, cognitive, perceptual, personality, and cultural differences. This includes aspects such as physical dimensions and dynamic actions (reach, finger speed, lifting strength), perceptual abilities (vision, colour deficiencies, flicker, contrast, depth perception), memory and information processing limitations, knowledge, experience, skill level, and individual differences.

    - Design should account for special populations such as children or older adults, and address the needs of users with disabilities, including visual impairments or blindness. Accessibility and inclusiveness aim to make products accessible and empower users.
    - Consider the number of users and potential turnover in collaborative settings.
    - Involving real users throughout the process helps designers gain a better understanding of their needs and goals.
2. **Understanding Tasks and Activities**

    - Designers should begin with a careful task analysis to determine what functions should be provided.
    - Consider the tasks users need to accomplish, how frequently they perform them, and the required precision.
    - Understanding the activities and their context helps in identifying information requirements and how data is needed and transformed.
    - Hierarchical strategies and congruent structures based on tasks can facilitate learning and retention.
3. **Considering the Environment and Context**

    - The environment where work is performed is important for establishing requirements. This includes the physical environment (lighting, temperature, noise, workspace layout), safety, social aspects, organizational structure, and user support.
    - The context of use, including whether users are co-located or distributed, or working synchronously or asynchronously, significantly influences design.
    - Design decisions should also consider how they fit within the planned permanence of the system (short-term or long-term).
4. **Technical and System Constraints**

    - Interface design is affected and constrained by the capabilities of the hardware and software being used. This includes system power, screen size, resolution, display features, and the system platform.
    - Choices about technology (multimedia, VR, web-based) and input/output devices (pen-based, touch screen, speech, keyboard) depend on system constraints derived from requirements.
    - Response time is a critical technical consideration, involving balancing feasibility, costs, task complexity, and user expectations. An unacceptably slow response makes a system unusable.
    - Internal characteristics of the system, while hidden, are important if they affect external behaviour or features visible to the user.
5. **Conceptual and Physical Design**

    - Design involves developing a **conceptual model** and then the **physical design**.
    - **Conceptual design** is concerned with the high-level idea of the product – what users can do and the concepts needed to understand interaction. It involves mapping out high-level concepts like users, controls, displays, navigation, and workflow, and developing the mental model users should have. Decisions about the conceptual model should precede physical design.

- **Physical design** gets concrete, specifying details such as screen and menu structures, icons, graphics, layout, choice of interaction devices, use of colour, sound, and images. It considers how information is presented and interacted with.
- The two are intertwined; concrete issues need consideration for prototyping, and prototyping evolves the concept.

6. **Design Principles, Guidelines, Rules, and Frameworks**

    - These provide guidance for making design decisions and restricting the space of design options. They are derived from theory, experience, and common sense, offering prescriptive "do's and don'ts".
    - Examples include heuristics (like Nielsen's) and usability principles used for evaluation, and Shneiderman's eight golden rules.
    - Guidelines documents (style guides) provide standard practices and organizational identity.
    - Frameworks like the cognitive dimensions framework (viscosity, consistency, visibility, etc.) help analyse design issues. Cognitive frameworks (distributed cognition, etc.) explain and predict user behaviour, informing design.
    - Models within methodologies like Contextual Design (physical, flow, artifact, social structure) help capture aspects of users' work, including collaboration and communication.
7. **Evaluation and Prototyping**

    - Extensive testing and evaluation are necessities in design. Evaluation should occur throughout the design process, from early stages to active use.
    - Evaluation is driven by questions about how well the design satisfies user needs and is shaped by practical constraints like schedules, budgets, and user access.
    - Prototyping involves producing limited versions of the product to answer specific questions about feasibility or appropriateness. Prototypes help overcome misunderstandings and give a better impression of the user experience than descriptions alone. They are built and evaluated by users, facilitating iterative refinement.
8. **Trade-offs and Balancing Requirements**

    - Design is fundamentally about making choices and decisions among numerous alternatives, often involving balancing conflicting requirements.
    - Designers must balance environmental, user, data, usability, and user experience requirements with functional requirements. These requirements can sometimes be in conflict, for example, functionality versus constraints imposed by device type or learnability versus challenge in a game.
    - Trade-offs require experience and the development and evaluation of alternative solutions. They are often based on experimental data, good judgment, and prioritizing user needs. **People's requirements should always take precedence over technical requirements**.
9. **Management, Legal, and Social Issues**

    - Managers promote attention to design through personnel selection, schedules, guidelines, and testing. Managerial issues like time constraints, economic forces, marketability, and training are also relevant.

    - Legal concerns include equal access for disabled users, attention to international laws, and intellectual property. Legal advice may be needed.
    - Considering the social impact of the design early on can help identify problems and produce interfaces with high overall societal benefits.

In summary, design considerations encompass a broad spectrum from deeply understanding the human user and their context to navigating technical possibilities, applying established principles, rigorously evaluating alternatives, and balancing competing demands while addressing project, legal, and social implications.