Human Machine Interface

TELLI AbdelmoutiA
Class A Associate Professor
Computer Science Department
Biskra University

2023-2024

Domaine: Mathematics and Computer Science
Title of the ACADEMIC LICENSE: Computer Science
Speciality: Informatic Systems
Semestre: S5
Fundamental teaching unit: UEF2
Title of the material: Human Machine Inteface.
Credits: 5
Coefficients: 3
Objectives of teaching: Allow students to acquire skills to create visual graphical interfaces while respecting ergonomic criteria and design standards of interactive and user-friendly interfaces.
Knowledge of ergonomic rules.
Knowledge of HMI development methods.
Coupling with the object-based development method.
Implementing these methods in a project.
Recommended prior knowledge: Algorithms and data structure, software engineering.
Content of the material:

1. **Chapter I:** Notions of interaction.
2. **Chapter II:** HMI construction methodology.
3. **Chapter III:** Models and architectures.
4. **Chapter IV:** Ergonomic rules in HMIs.
5. **Chapter V:** Multi-users interface design.
6. **Chapitre VI** : Interfaces adaptatives
7. **Chapter VII:** Multimodal interfaces and future interfaces.

Evaluation method: Exam (60%) + Continuous assessment (40%).

The author has prepared this work by collecting a wide range of different scientific sources for educational purposes. The use of these courses is authorized as part of university training with mention of the author.

Rferences

- **Ménadier Jean-Paul**, l'interface utilisateurs: Pour une informatique conviviale, DUNOD, Informatique et Stratégie, 1991.
- **Coutaz Joelle**, Interface homme ordinateur: conception et réalisation Dunod Informatique 1990.
- **Kolski, C, Ezzedine, H et Abed, M**, Développement du logiciel: des cycles classiques aux cycles enrichis sous l'angle des IHM », ouvrage collectif, Analyse et conception de l'IHM, Interaction homme machine pour les systèmes d'information Vol 1, Hermès, 2001.
- **Drouin, A, Valentin, A et Vanderdonckt, J**, Les apports de l'ergonomie à l'analyse et à la conception des syteme d'information, in Christophe KOLSKI, Analyse et conception de l'IHM, Interaction homme machine pour les systèmes d'information Vol 1, Hermes, 2001.
- **David Benyon**, Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design,Pearson; 3 edition, 2013.
- **Yvonne Rogers**, Helen Sharp et Jenny Preece, Interaction Design: beyond human computerinteraction (3rd edition), Wiley, 2011
- **Norman DA**, The Design of Everyday Things, Basic Books, 2002. Serengul Smith Atakan The Fast Track to Human Computer Interaction, (Paperback) Thomson Learning, 2006.

Contents

Chapter 1

Notions of interaction

1. Definitions: Interaction, Interactivity, ...
2. Causes for rejection of certain applications.
3. Challenges.
4. Difficulties.
5. Definition of an HMI.
6. History of HMIs.

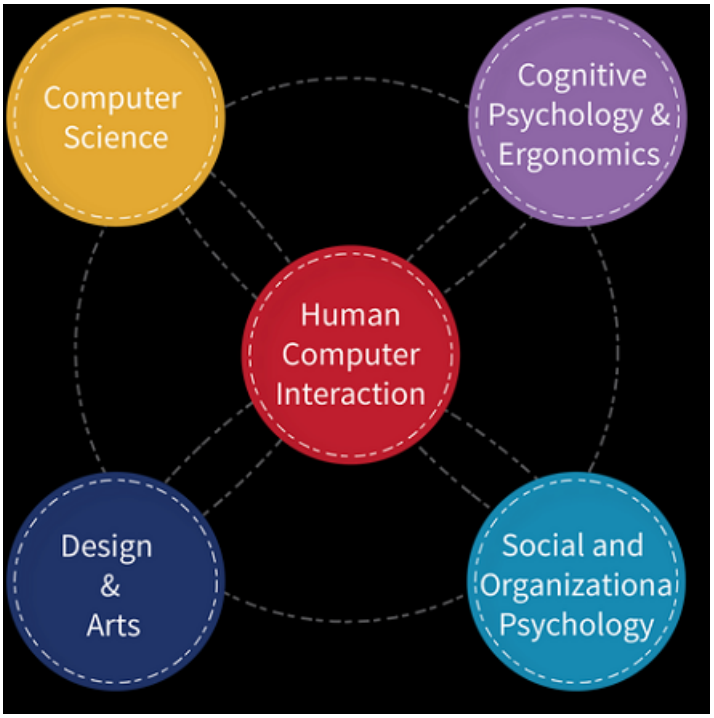
Introduction

Human Machine Interface (HMI) is a technology that serves as a bridge between humans and machines, allowing users to interact with and control machines or systems. It is a critical component in various industries, particularly in automation and control systems. HMI technology plays a crucial role in simplifying human interaction with complex machines and systems across various industries, enhancing efficiency and safety in industrial processes.

1.1 Definitions

Notions of interaction on Human-Machine Interfaces (HMI) refer to the ways in which humans and machines communicate and collaborate in various contexts:

Human-Computer Interaction (HCI): HCI is a multidisciplinary field that explores how humans interact with computers and HMI systems. It encompasses the study of user interfaces, user experience design and usability.



User-Centered Design: Interaction on HMI is often guided by user-centered design principles, where the focus is on creating interfaces that meet the needs and preferences of users.

Interactivity: HMI systems can vary in terms of interactivity, ranging from simple push-button controls to advanced touchscreens, voice recogni-

tion, and gesture-based interfaces.

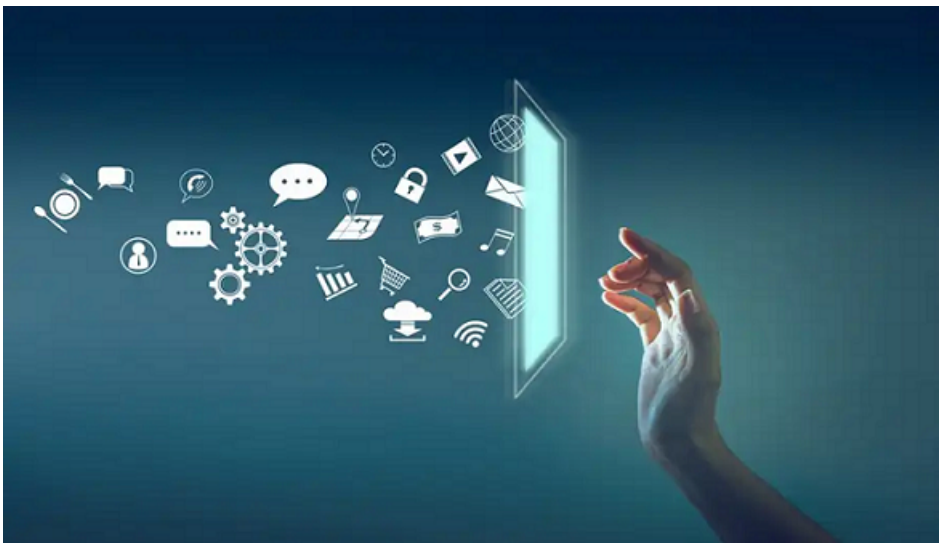
Feedback and Response: Effective interaction involves providing users with feedback on their actions and the system's response. This can include visual cues, sounds, or haptic feedback.

Adaptive Interfaces: Some HMIs incorporate adaptive features that learn from user behavior and adjust the interface accordingly, aiming to enhance user satisfaction and efficiency.

Cross-Platform Interaction: With the increasing use of mobile devices, cross-platform interaction is essential. HMIs may need to adapt to different screen sizes and input methods.

Safety and Ergonomics: Interaction design in critical systems, such as industrial control panels, focuses on safety and ergonomics to prevent accidents and minimize user fatigue.

Future Trends: The future of HMI is likely to involve more advanced technologies like augmented reality, virtua reality, and natural language processing, enabling more intuitive interactions.



1.2 Causes for rejection of applications

There are several reasons why certain applications on Human-Machine Interfaces (HMI) may be rejected or encounter issues:

Access Restrictions: errors can occur when the user does not have the necessary permissions to run or modify an application on the HMI.

High CPU Usage: Some applications may lead to excessive CPU usage, causing performance problems or even system crashes.

Connection Failures: Applications relying on network connections, like Vijeo Designer Air App, may fail due to network issues or misconfiguration.

Errors in Application Design: Poorly designed applications may trigger errors, indicating issues within the application.

User Administration: In complex systems like WinCC (TIA Portal), incorrect user administration settings can lead to access problems.

Compatibility: Compatibility issues with specific HMI hardware or software versions may cause rejection or malfunction of applications.

Communication Problems: Applications using protocols like OPC UA may encounter communication problems if not properly configured or integrated.

File Copying Errors: Copying HMI applications can lead to issues like "Problems opening InTouch after copying an application" if not done correctly.

Known Software Issues: Some software versions may have known issues, as documented in the list of known issues for iX Developer.

Documentation: Users may face challenges when applications are not well-documented, making it difficult to troubleshoot or operate.

1.3 Challenges

Human-Machine Interface (HMI) design and implementation present various challenges in different domains. Addressing these challenges requires expertise in interface design, human factors, technology integration, and continuous adaptation to evolving user needs and expectations.

User Interface Complexity: Complex user interfaces can confuse users and hinder efficient interaction.

HMI Failures: Failures such as unresponsive touchscreens or system crashes can disrupt operations.

Brain-Computer Interface (BCI) Limitations: BCI integration with HMI faces limitations in accuracy and practicality.

Next-Generation HMI Development: Developing advanced HMI solutions requires overcoming technical and design challenges.

Technology Integration: Integrating new technologies into HMIs, like augmented reality or voice commands, can be challenging.

Automotive HMI Design: The automotive sector grapples with designing intuitive and safe HMIs for vehicles.

Transportation Interface: Ensuring user-friendly interfaces in transportation systems is vital for safety.

Audio-Based HMIs: Challenges in designing audio-based HMIs include noise interference and user adaptability.

LCD HMI Challenges: Overcoming challenges associated with LCD-based HMIs, such as visibility and durability.

Software Development: Developing automotive software for HMIs involves addressing specific challenges.

1.4 Difficulties

Designing Human-Machine Interfaces (HMI) presents several difficulties, including:

Diverse User Base: HMIs must cater to a wide range of users with varying levels of technical expertise, making user-friendly design crucial.

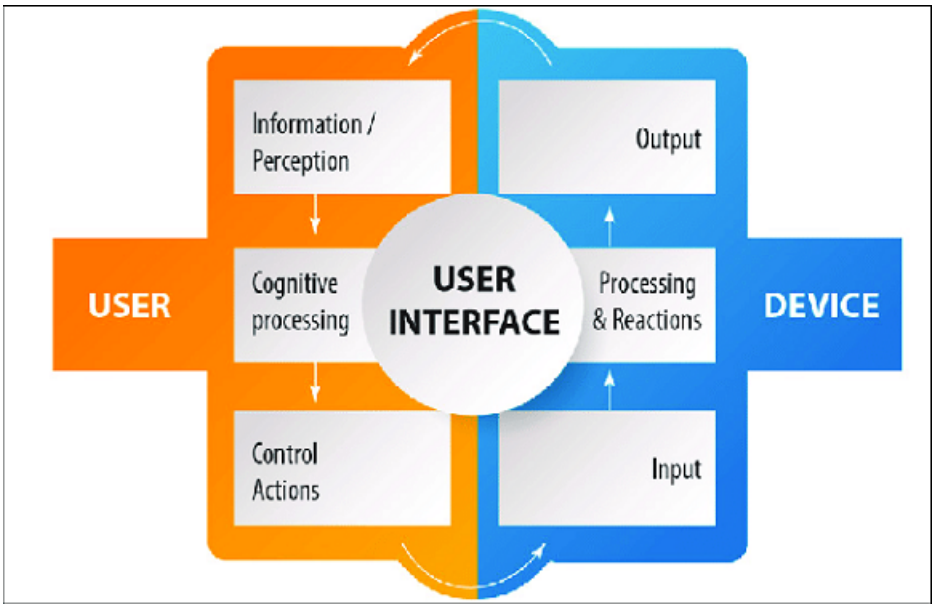
Complex Manufacturing Environments: In manufacturing, HMIs are used in complex settings with machinery and processes. Ensuring these interfaces are intuitive and efficient is challenging.

Communication Gap: Bridging the gap between designers and developers is essential. Often, designers create HMI concepts that developers need to implement accurately.

To address these difficulties, HMI designers focus on user-centered design principles, emphasizing user needs and usability testing. Collaborative efforts between designers and developers, as well as continuous feedback loops, help refine HMI designs. Additionally, staying updated with innovative HMI technologies can simplify complex manufacturing environments.

1.5 Definition of an HMI

A Human-Machine Interface (HMI) is a technology or interface that allows communication between a human operator and a machine, system, or device. It serves as a bridge for users to interact with and control machines or processes efficiently. HMIs typically include components like touchscreens, graphical displays, buttons, and software applications.



Common elements of an HMI include:

Graphical User Interface (GUI): This visual component presents information in a user-friendly manner, often using icons, graphics, and menus.

Input Devices: HMIs offer various input methods such as touchscreens, keyboards, or mice to allow users to input commands or data.

Control Functions: Users can control machines or processes through the HMI, issuing commands, setting parameters, and monitoring performance.

Feedback: HMIs provide feedback to users, showing the current status, alarms, and relevant data.

Integration: They integrate with the underlying systems or machinery, facilitating data exchange and control.

HMI technology is widely used in industries such as manufacturing, automation, healthcare, and more, enhancing operational efficiency and enabling users to interact with complex systems effectively.

1.6 History of HMI

The history of HMI is a fascinating journey that has evolved significantly over time. Here's a concise overview:

1. Early Precursors: The predecessors of HMI include early mechanical interfaces used for controlling machinery. Examples include levers, buttons, and dials, which were manual and lacked the sophistication of modern HMIs.

2. Digitization: With the advent of digital technology, the evolution of HMIs accelerated. The transition from mechanical interfaces to digital screens allowed for more dynamic and intuitive interactions with machines.

3. Graphical User Interfaces (GUIs): The development of GUIs revolutionized HMI design. GUIs made use of icons, menus, and visual elements to simplify complex interactions, making machines more user-friendly.

4. Touchscreens: The introduction of touchscreens further enhanced user experiences. These interfaces eliminated the need for physical buttons, offering a more adaptable and versatile way to control and interact with machines.

5. Advanced Functionality: Modern HMIs are capable of sophisticated tasks, including real-time monitoring, data visualization, and remote control. They have become integral in various industries, including manufacturing, automation, and process control.

6. Integration with AI and IoT: The latest trend in HMI development involves integrating Artificial Intelligence (AI) and the Internet of Things (IoT) to create smarter and more responsive interfaces.

To explore the rich history of HMI in more detail, the provided sources offer comprehensive insights into its evolution, including articles on the roles of women in shaping HMI interfaces, the transition toward easier design.

Chapter 2

HMI construction methodology

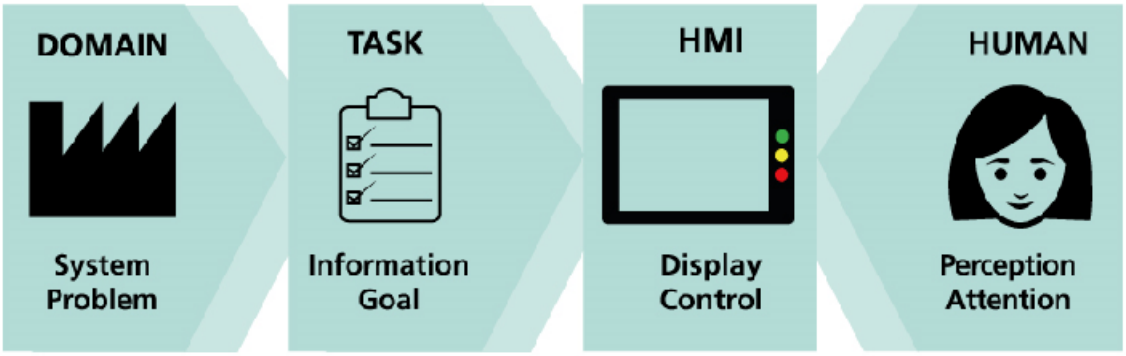
1. Classic Methodology.
2. Identification stage.
3. Task analysis stage.
4. Modeling stage.
5. Specification stage.

Introduction

Human Machine Interface (HMI) construction methodology refers to the process of designing and implementing the interface between humans and machines, particularly in the context of construction equipment and machinery. This involves creating user-friendly interfaces that enable operators to interact with and control complex machinery efficiently and safely.

2.1 Classic methodology

The classic methodology for constructing a HMI involves a systematic approach to designing and implementing interfaces for human interaction with machines or systems.



Here is a general outline of the classic HMI construction methodology:

User Needs Assessment: Begin by identifying the needs and requirements of the users who will interact with the HMI. This includes understanding their tasks, preferences, and expectations.

HMI Design: Create a user-centered design for the HMI based on the gathered user requirements. This design phase includes layout planning, selecting appropriate controls (buttons, touchscreens, etc.), and defining data visualization methods.

Hardware and Software Integration: Implement the hardware components (sensors, displays, input devices) and the software systems (control algorithms, user interface software) required for the HMI.

Prototyping: Develop a prototype or mockup of the HMI to test its usability and functionality. This allows for early user feedback and design improvements.

Usability Testing: Conduct usability testing with representative users to evaluate the HMI's effectiveness, efficiency, and user satisfaction. Make necessary adjustments based on user feedback.

Safety Considerations: If applicable, integrate safety features and emergency shutdown controls into the HMI to ensure user and system safety.

Documentation: Create comprehensive documentation, including user manuals and training materials, to assist users in understanding and using the HMI effectively.

Training: Provide training to users and operators to ensure they are proficient in using the HMI to control the associated machinery or system.

Maintenance and Updates: Develop a plan for ongoing maintenance and updates of the HMI to address issues, improve performance, and adapt to changing user needs or technology advancements.

Validation and Compliance: Ensure that the HMI complies with relevant industry standards and regulations, particularly in safety-critical applications.

Deployment: Finally, deploy the HMI in its intended environment, whether it's in manufacturing, automation, transportation, or another field, and monitor its performance in real-world conditions.

This classic methodology prioritizes user-centric design and iterative development to create an HMI that is intuitive, efficient, and safe for operators. It is adaptable to various industries and technologies, making it a foundational approach in HMI construction.

2.1.1 Identification stage

The identification stage in HMI construction methodology focuses on gathering requirements, identifying assets and access points, understanding user mental models, and fostering innovative solutions for effective HMI design:

1. **Analysis:** During this phase, the design team identifies the goals and requirements of the HMI system. This includes understanding the specific needs of users and the tasks the system will perform. The analysis helps in defining the scope and purpose of the HMI.
2. **User Needs Assessment:** Part of the identification stage involves assessing the needs and preferences of the end-users. This step ensures that the HMI is tailored to meet the users' expectations and requirements.
3. **System Requirements:** Identifying the technical and functional requirements of the HMI system is essential. This includes determining the hardware and software components, data inputs, and communication protocols necessary for seamless interaction.
4. **Goal Definition:** Clearly defining the objectives and goals of the HMI helps in creating a design that aligns with the intended outcomes.

2.1.2 Task analysis stage

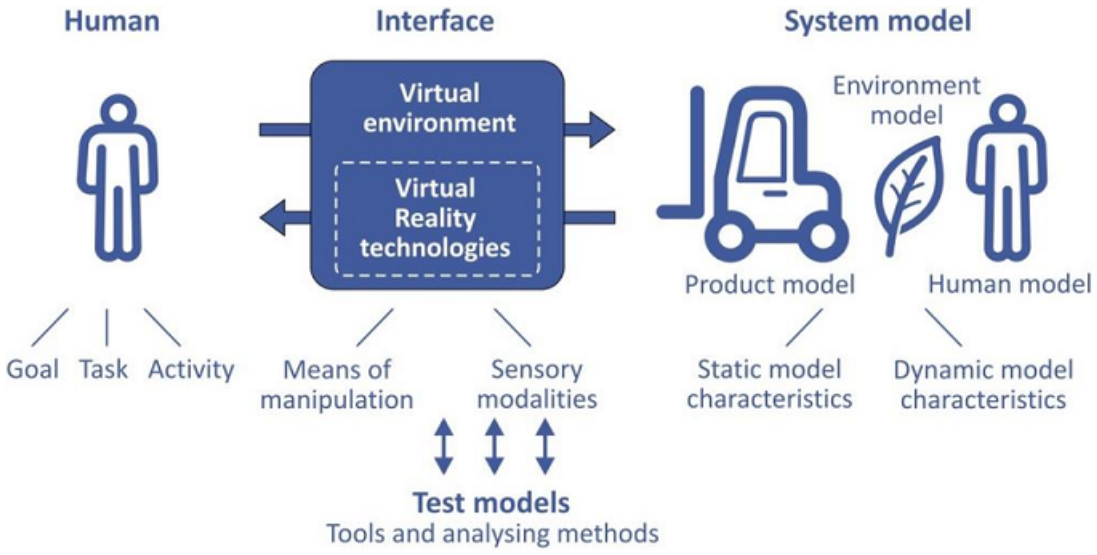
During this stage, detailed analysis of the tasks that users will perform with the HMI system is conducted to inform the design and development process. The main aspects of the task analysis stage:

- Identifying User Tasks: Task analysis begins with identifying and comprehending the specific tasks that users need to accomplish.

- Task Decomposition: This decomposition helps in creating a clear and detailed picture of what users need to do.
- Hierarchical Task Analysis: This hierarchical structure helps in designing an intuitive interface that reflects the user's mental model.
- Task Sequencing: Task analysis also includes determining the order and sequence of tasks, as well as any dependencies between them.
- Usability and Performance: The information gathered during task analysis is used to evaluate and refine the design of the interface to ensure that it supports users in performing their tasks efficiently.
- Iterative Process: It's important to continuously refine the analysis as the design evolves.

2.1.3 Modeling stage

This stage involves creating a model or prototype of the HMI interface based on the findings from the earlier stages:



The modeling stage serves as the bridge between conceptual design and actual implementation. It allows for testing and refining the HMI interface before it is deployed in real-world applications. The iterative nature of this stage ensures that user feedback is incorporated, resulting in an HMI that meets user needs and expectations.

2.1.4 Specification stage

In this stage the detailed requirements and specifications for the HMI system are defined. It lays the foundation for the entire HMI development process and ensures that the system meets the needs of users and the objectives of the project. The main aspects of the specification stage are:

- **Study of needs for the HMI:** It involves a comprehensive analysis to understand the specific requirements and expectations of the users and the project.
- **Conceptual specification:** A conceptual specification for a Human-Mmachine Interface (HMI) involves defining the high-level requirements and design principles before delving into the detailed design and implementation.
- **Functional specification:** Enumerate all the functional capabilities of the HMI system, such as:
  - Real-time monitoring of processes or equipment.
  - Control functions, including start, stop, and adjust.
  - Data visualization, trends, and historical data access.
  - Alarm management and notifications.
  - User authentication and security features.

1. **Syntactic specification:** refers to the definition of rules and grammar that dictate how commands, data, and interactions within the HMI system should be structured and formatted. It includes:

    - **Command Syntax:** defines the syntax for issuing commands to the HMI system, specifying how users should provide instructions.
    - **Data Representation:** HMI systems often display data to users. Syntactic specification outlines how data should be represented visually, specifying formats for numerical values, text, graphics, and other data types.
    - **Message Formatting:** In communication between the HMI and other devices or systems, syntactic specification governs the format of messages. This includes specifying how data is structured within messages, the use of protocols, and data encoding formats.

2. **Lexical specification:** refers to the definition of rules and guidelines for recognizing and interpreting individual words, symbols, and tokens within the HMI system. It can accurately process and understand user inputs and commands. The lexical specification includes:

    - **Reserved Keywords:** HMI systems often have reserved keywords or commands that trigger specific actions. Lexical specification lists and defines these keywords, ensuring that they are recognized as commands when used in user inputs.
    - **Data Types:** It defines the lexicon of data types used in the HMI system. This includes specifying how different data types.
    - **Symbols and Operators:** Lexical specification includes rules for recognizing symbols and operators used for calculations or interactions.
    - **Localization:** defines how different languages or dialects are handled within the HMI.

Chapter 3

Models and Architectures

1. The Dialog Controller (definition and role).
2. Presentation of the Seeheim model
3. Presentation of the PAC model
4. Presentation of MVC model
5. Presentation of agent models

3.1 Introduction

The architectures governing HMI software encompass the fundamental design principles and templates that steer the development, fusion, and upkeep of these applications. Nonetheless, these HMI software architectures exhibit substantial variability contingent on the platform, device, and specific domain of the HMI application. Such disparities pose notable challenges, including issues of inconsistency, suboptimal performance, and the potential for incompatibility when transitioning across diverse HMI systems.

In this chapter, we delve into the exploration of strategies and frameworks aimed at achieving standardization of HMI software architectures across a spectrum of platforms and devices.

3.2 The Dialog Controller

The Dialog Controller on an Human-Machine Interface (HMI) is a component responsible for managing and controlling dialog boxes or windows that appear on the interface. These dialog boxes are typically used to interact with users, display information, and request input. Here's an explanation of the Dialog Controller's role:

Dialog Box Management: The Dialog Controller oversees the creation, display, and removal of dialog boxes.

User Interaction: It handles user interactions within dialog boxes. This includes capturing user input, such as button clicks, text entries, and selections from dropdowns, and processing these inputs accordingly.

Data Exchange: It facilitates communication between the dialog boxes and the underlying software or system. It transfers data entered or selected by the user to the appropriate parts of the HMI application.

Navigation: It manages the flow of dialog boxes, determining which dialog should appear next based on user interactions and application logic.

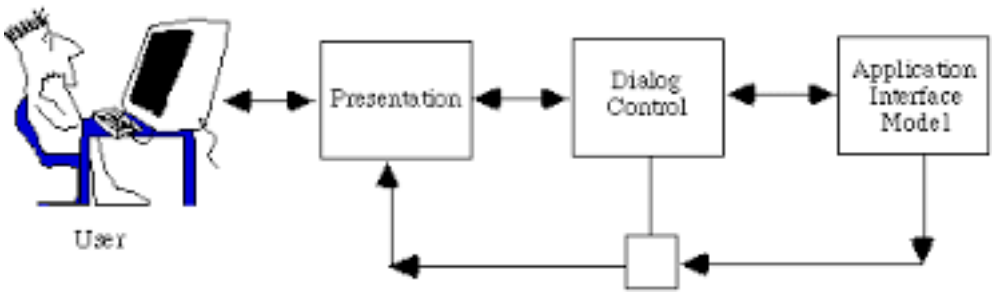
Error Handling: The Dialog Controller can also handle error messages and prompts, ensuring that users are informed of any issues or exceptional conditions.

Appearance and Behavior: It controls the appearance and behavior of dialog boxes, including their layout, styling, animations, and responsiveness to different screen sizes and orientations.

User Guidance: In some cases, the Dialog Controller may provide guidance or assistance to users through tooltips, hints, or context-sensitive help.

3.3 Presentation of the Seeheim model

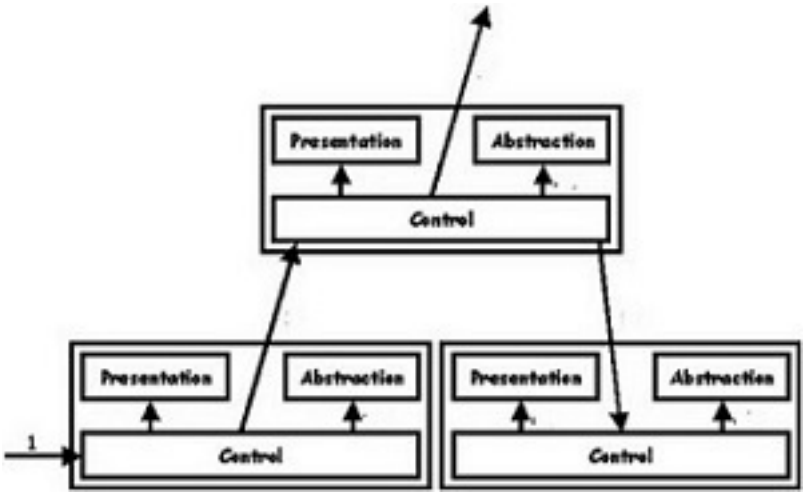
The Seeheim Model is an architectural pattern introduced in 1983 to structure the human-computer interaction in interactive software. It provides a logical framework for designing user interfaces. This model separates the dialogue structure from the presentation of the user interface.



This model was created before direct manipulated interfaces were constructed, and therefore the model has difficulties in capturing the dynamics of a graphical user interface.

3.4 Presentation of the PAC model

The Presentation-Abstraction-Control (PAC) model is an architectural pattern that promotes the separation of concerns in software user interfaces.



The PAC architecture does not have the model as its core component, but a hierarchical structure of PAC components. Each PAC component consists of these items:

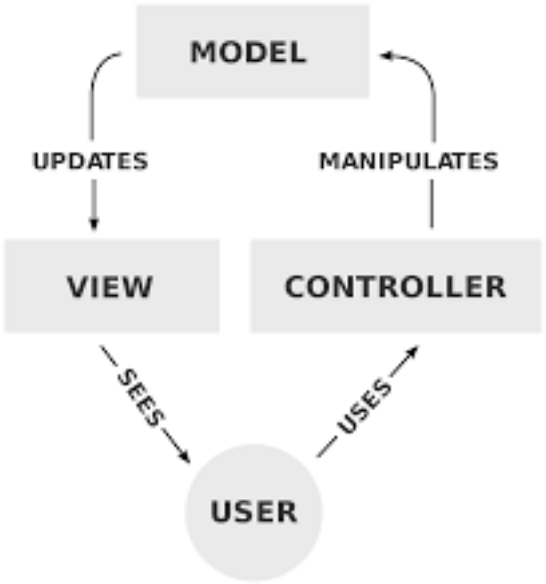
Control: It processes external events and updates the model. It also directly updates the Presentation part.

Abstraction: contains the data, like in MVC. However, it may be just part of the complete data structure of the application, and it does not play an active role in the notification of changes.

Presentation: is exactly like the View of MVC. It displays the information from the Abstraction.

3.5 Presentation of MVC model

The Model-View-Controller (MVC) is a widely used architectural pattern in software development, particularly for designing user interfaces. It provides a structured and organized way to separate an application's concerns into three distinct components: Model, View, and Controller.

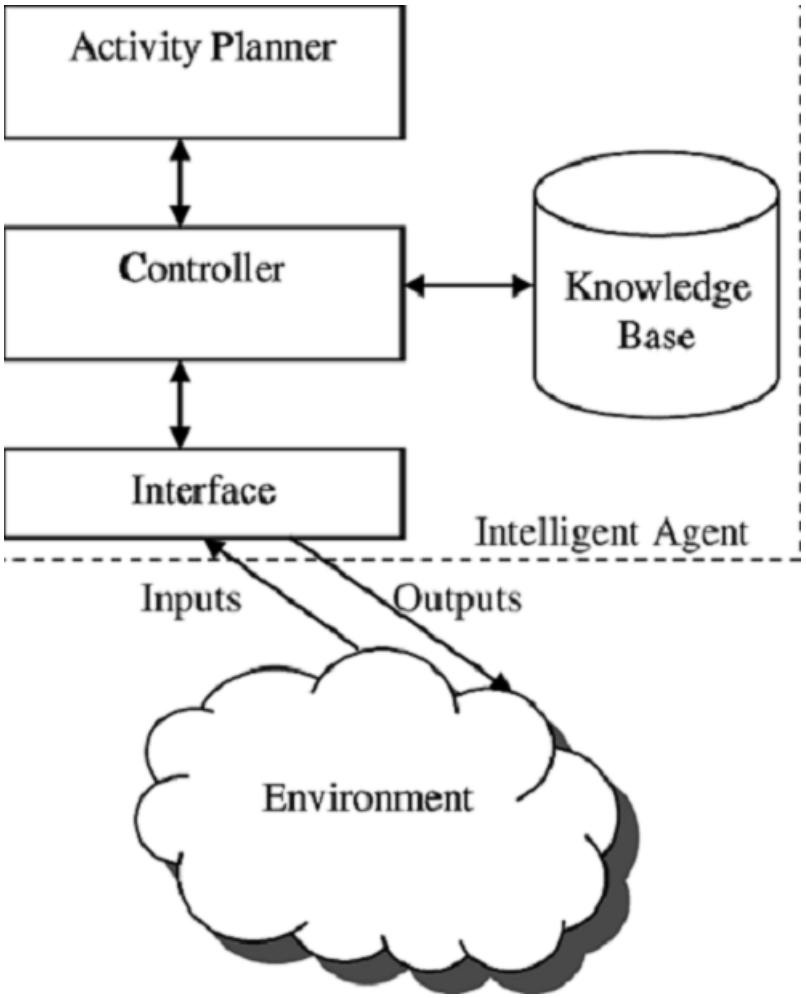


The Model: contains the pure application data and pure logic describing how to present the data to a user.

The View: The View is responsible for presenting the data to the user and managing the user interface components. It displays the information from the Model and presents it in a user-friendly format. **The Controller:** The Controller acts as an intermediary between the Model and the View. It receives user input and processes it to update the Model or the View accordingly. Controllers handle user interactions, making decisions based on the input and updating the Model and View as necessary.

3.6 Presentation of agent models

These models introduce a level of automation, intelligence, and adaptability to HMIs where the agent models involve the use of intelligent software agents. These agents are autonomous, decision-making entities that can perform specific tasks or functions within the HMI system. In many cases, HMI agent models employ multi-agent systems. Multiple intelligent agents work together within the HMI ecosystem to achieve common goals, share information, and coordinate tasks.



Distributed Architecture: Agent-based HMIs often feature a distributed architecture where intelligent agents are distributed across the system. This allows for decentralized decision-making and improved scalability.

Autonomous Task Execution: Intelligent agents can autonomously perform tasks such as system monitoring, fault detection, diagnostics, and even decision-making. This reduces the need for constant human intervention.

Adaptive Behavior: Agent models can exhibit adaptive behavior by learning from user interactions and adjusting their responses or recommendations based on user preferences and historical data.

Human-Agent Collaboration: While agents can handle many tasks independently, they also facilitate human-agent collaboration. Users can interact with agents to request information, receive recommendations, or make decisions jointly.

Self-Organization: Some agent models incorporate self-organizational capabilities, allowing the HMI to reconfigure itself dynamically to optimize user experience and task execution.

Chapter 4

Ergonomic rules in HMIs

1. Nielsen heuristics.
2. Bastien and Scapin ergonomic criteria
3. Coutaz Rules

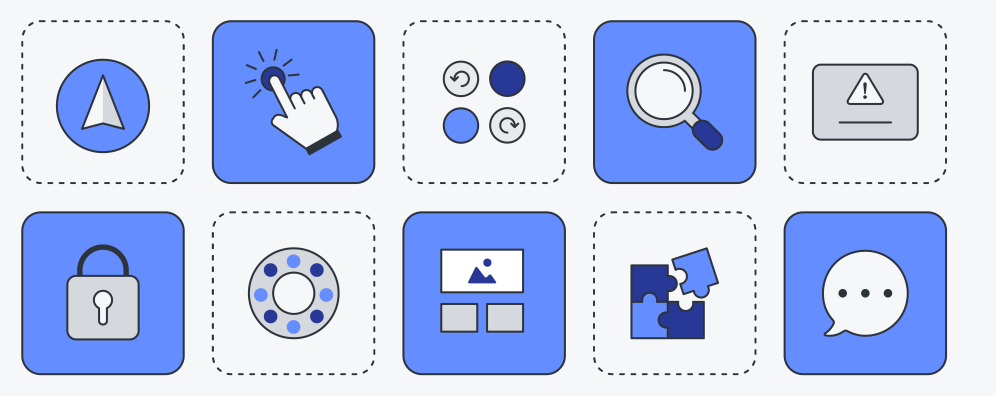
4.1 Introduction

Ergonomics plays a crucial role in the design of Human-Machine Interfaces (HMIs) to ensure they are user-friendly, efficient, and comfortable. these ergonomic rules, HMIs can be designed to enhance user experience, minimize errors, and improve overall productivity and safety in various applications.

Ergonomics is a multidisciplinary field that applies to various industries, including office environments, manufacturing, healthcare, and transportation. By following these ergonomic rules, organizations can create safer, more comfortable, and productive workplaces while reducing the risk of injuries and health issues.

4.2 Rules of User Interface Design

User interface design is much more than pure aesthetics. The foundation of good UI design is user-friendliness, which is achieved through visuals.

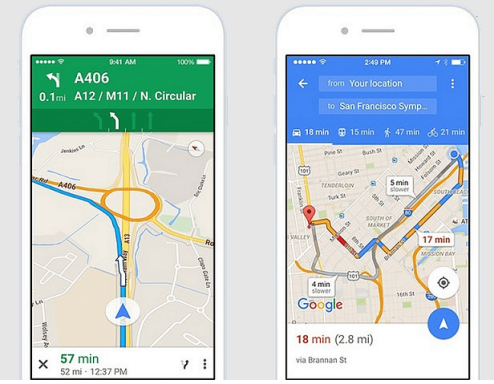


The primary focus of a user interface design is anticipating what a user might need to do to make their experience as intuitive as possible. User interface design is all about usability, utility and desirability. These are 10 rules of thumb for every UI designer. The essential 10 user interface guidelines are:

4.2.1 Visibility of system status

Making the system status visible helps the users understand the outcome of their prior interactions and decide the next steps intuitively.

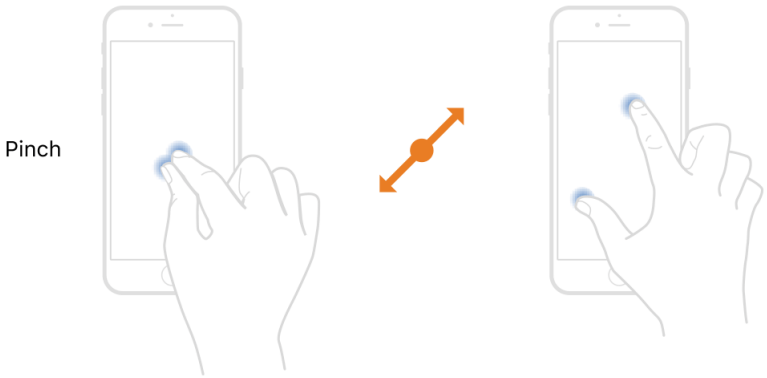
For example, Google Maps uses an arrow, or a car icon, to indicate where the user is in their journey. Additionally, they've placed a status bar at the top of the screen that displays their next step and how far to go until their next step.



4.2.2 Match between the system with the real world

When it comes to UI design, keep it simple. Put information in natural and logical order. Terms, icons and images should correspond to predictable outcomes. Icons like a magnifying glass or arrow are clear to your user. This practice is also called "natural mapping".

For example, The pinch-to-zoom Pinch-to-zoom was invented in 1983 but it wasn't used in consumer devices until 2005 with JazzMutant's product, the Lemur. The Lemur was a multi-touch device that served as a controller for musical devices like synthesisers and mixing consoles.



4.2.3 User control and freedom

Users need clearly marked exits from accidental or unwanted actions. This saves them from having to redo an entire process. “Emergency exits” create a feeling of freedom for users. The exits need to be clearly labelled and discoverable too.

For example: Undo and redo in Google Docs In most word processors, Google Docs included, you’ll find “undo” and “redo” functions in the toolbar. You might also recognise the common keyboard functions control or command + Z or control and command + Y. This allows users to quickly and easily fix a mistake without experiencing too much frustration.

4.2.4 Consistency and standards

By maintaining consistency, we will avoid making users learn something new. Sticking to industry standards reduces cognitive load and allows users to feel like your app is intuitive.

There are two types of usability; internal and external. Internal usability means that all of your wording, icons, fonts, layouts and actions are uniform throughout your user interface. External usability refers to adhering to other apps’ industry standards for those same components.

For example: Search magnifying glass, A magnifying glass always means “search.” The search function, indicated by the magnifying glass, is usually located at the top of the page on desktop and bottom of the screen on mobile. They are easy to find in these areas and easy to navigate quickly.

4.2.5 Error prevention

To design efficiently, focus on preventing high-cost errors first. Then, make a plan to tackle the more minor frustrations. UI designers can use practical constraints, change default settings or provide confirmation options to prevent slips. They can minimise mistakes by making the cognitive load lighter. For example, Shake to undo confirmation box on iPhone's latest OS Apple introduced shake-to-undo with iOS 13 back in 2019.

4.2.6 Recognition rather than recall

Humans' short-term memory only lasts 20-30 seconds. The designer should ensure that users don't need to remember or transfer information from one part of your interface to another. All key elements, actions and options should be visible or easily retrievable throughout the app. They should also be located in the same place. For example, Security code autofill iOS 13 also introduced a security code autofill function. When a user requests a security code from an app, the keyboard will grab the code from their text and suggest it above the user's keyboard. No short-term memory is required anymore.

4.2.7 Flexibility and efficiency of use

While consistency and uniformity are important to build trust and intuition with users, flexibility can be crucial too. Making your interface efficient by providing shortcuts and customisations can build a different kind of trust with your user.

This can look like a customisable dashboard, keyboard shortcuts or touch gestures that speed up common functions. Alternatively, like most social media platforms, you could offer your user content personalisation. This might be a way to tailor their feed like Pinterest or a hidden algorithm like Instagram or TikTok.

4.2.8 Aesthetic and minimalist design

Designs shouldn't feature something that's rarely needed. Keeping unimportant components in a design reduces the relative visibility and importance of key elements.

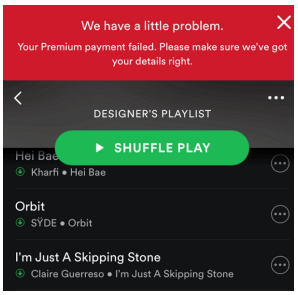
This doesn't mean your design has to be flat design. But it does mean that you should focus your content and visual design on the essentials. Above all else, your interface should support the users' primary goals.

For example, facebook uses a slick black-and-white interface. Buttons are clearly labelled in grey or outlined. Their site is easy to navigate and includes very few menu options.

4.2.9 Help users recognise, diagnose and recover errors

Users want autonomy and control. To avoid this, it's essential to help users recognise, diagnose and recover from mistakes independently.

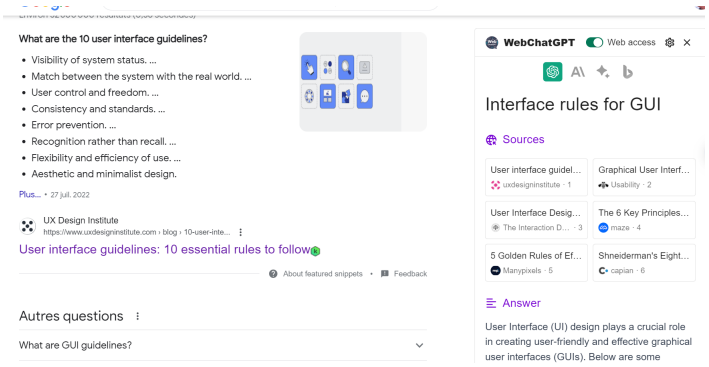
Clear, plain language error messages that offer a constructive solution can accomplish this. Avoid using error codes. Include a graphic or visual representation of the error for even faster recognition.



4.2.10 Help and documentation

Strong error messages are not enough to completely cover all of your bases. Providing help and documentation that is searchable and easy to find is crucial for the long term success of any piece of software or hardware. This should be kept concise. The next steps to solve a problem or learn a function should be listed in a concrete way. Where possible, it's best to present the documentation to the user at the moment that they actually need it.

Chatbots are a prime example of offering help at the moment. Rather than exploring the website for help.

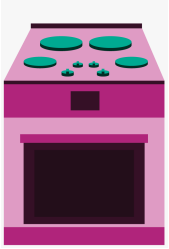


4.3 Nielsen heuristics

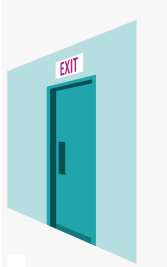
Jakob Nielsen's 10 general principles for interaction design. They are called "heuristics" because they are broad rules of thumb and not specific usability guidelines. Jakob Nielsen's Usability Heuristics are a set of ten principles or guidelines widely used in the field of user interface (UI) and user experience (UX) design to evaluate the usability of software and websites. These heuristics serve as a checklist for designers and usability experts to identify potential usability issues. Here are the ten Nielsen Usability Heuristics:

1. Visibility of System Status: Keep users informed about what's happening through appropriate feedback within a reasonable time. This includes loading indicators and progress bars.

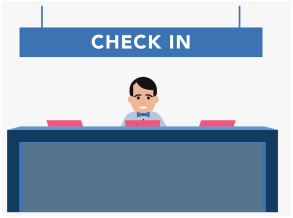
2. Match between System and the Real World: Speak the user's language. Use words, phrases, and concepts familiar to the user. Follow real-world conventions and make information appear in a natural and logical order.



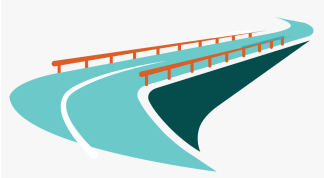
3. User Control and Freedom: Allow users to easily undo actions, exit situations, and navigate without feeling trapped. Provide "emergency exit" options.



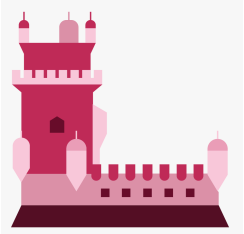
4. Consistency and Standards: Follow established conventions, both internal (within your product) and external (widely accepted industry standards), to create a consistent and predictable user experience.



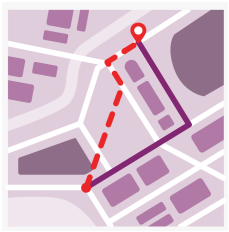
5. Error Prevention: Strive to prevent errors by offering clear instructions, confirming destructive actions, and providing meaningful error messages.



6. Recognition Rather than Recall: Reduce the user's memory load by making objects, actions, and options visible. Users shouldn't have to remember information from one part of the interface to another.



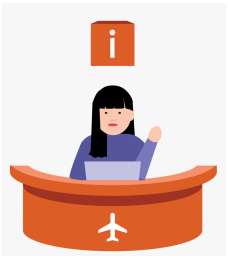
7. Flexibility and Efficiency of Use: Offer shortcuts and accelerators to power users while still accommodating novice users. Don't force users into a single way of doing things.



8. Aesthetic and Minimalist Design: Strive for a clean and visually pleasing design. Only include elements that are necessary for functionality.

9. Help Users Recognize, Diagnose, and Recover from Errors: Provide clear error messages that explain the problem and suggest a solution. Offer guidance to help users recover from errors.

10. Help and Documentation: Ideally, make the system self-explanatory. If necessary, provide concise, accessible documentation and help features.



These heuristics are valuable for assessing and improving the usability of digital interfaces. Designers and evaluators use them as a starting point to identify potential problems and enhance the user experience.

4.4 Bastien and Scapin ergonomic criteria

Bastien and Scapin are two scientists in ergonomic psychology and cognitive ergonomics who have chosen to focus on user experience and human-computer interfaces. The ergonomic criteria presented below were created in the context of a research project in the mid-90s.

These criteria provide a structured approach to assessing the quality of interactive systems. Below are the key ergonomic criteria proposed by Bastien and Scapin:

1. **Guidance:** This criterion evaluates whether the system provides clear and helpful guidance to users, such as instructions, tooltips, and cues. Those are the sub citeria of Guidance criteria

    - **Prompting:** refers to the means available in order to lead the users to making specific actions whether it be data entry or other tasks.
    - **Grouping/Distinction:** concerns the visual organisation of information items in relation to one another. This criterion takes into account the topology (location) and some graphical characteristics (format). The criterion Grouping/Distinction of Items is subdivided into two criteria: Grouping/Distinction by Location and Grouping/Distinction by Format.
    - **Immediate Feedback:** concerns system responses to users' actions. These actions may be simple keyed entries or more complex transactions such as stacked commands.
    - **Legibility:** concerns the lexical characteristics of the information presented on the screen that may hamper or facilitate the reading of this information (character brightness, contrast between the letter and the background, font size, interword spacing, line spacing, paragraphs spacing, line length, etc.).

2. **Workload:** It assesses the cognitive and physical workload imposed on users while interacting with the system. Lowering workload is essential for user comfort. The criterion Workload is subdivided into two criteria:

    - **Brevity:** corresponds to the goal of limiting the reading and input workload and the number of action steps. The criterion Brevity is subdivided into two criteria: Concision and Minimal Actions.
        - **The criterion Concision:** concerns perceptual and cognitive workload for individual inputs or outputs.

  - **The criterion Minimal Actions:** concern workload with respect to the number of actions necessary to accomplish a goal or a task. It is here a matter of limiting as much as possible the steps users must go through.

- **Information Density:** concerns the users' workload from a perceptual and cognitive point of view with regard to the whole set of information presented to the users rather than each individual element or item.

3. **Explicit Control:** This criterion examines whether the system gives users explicit control over their actions, allowing them to perform tasks with precision. The criterion Explicit Control is subdivided into two criteria:

   - **Explicit User Action:** refers to the relationship between the computer processing and the actions of the users. This relationship must be explicit, i.e., the computer must process only those actions requested by the users and only when requested to do so.
   - **User Control:** refers to the fact that the users should always be in control of the system processing (e.g., interrupt, cancel, pause and continue). Every possible action by a user should be anticipated and appropriate options should be provided.

4. **Adaptability:** It considers the system's ability to adapt to users' needs and preferences, allowing for customization and flexibility. The criterion Adaptability is subdivided into two criteria:

   - **Flexibility:** The criterion Flexibility refers to the means available to the users to customise the interface in order to take into account their working strategies and/or their habits, and the task requirements.
   - **User Experience:** The criterion User Experience refers to the means available to take into account the level of user experience.

5. **Error Management:** Evaluates how well the system handles errors, including error prevention, error detection, and the recovery process. The criterion Error Management is subdivided into three criteria:

   - **Error Protection** : refers to the means available to detect and prevent data entry errors, command errors, or actions with destructive consequences.
   - **Quality of error messages** : refers to the phrasing and the content of error messages, that is: their relevance, readability, and specificity about the nature of the errors (syntax, format, etc.) and the actions needed to correct them.
   - **Error correction**: refers to the means available to the users to correct their errors.

6. **Consistency:** it refers to the way interface design choices (codes, naming, formats, procedures, etc.) are maintained in similar contexts, and are different when applied to different contexts.

7. **Significance of Codes:** The criterion Significance of Codes qualifies the relationship between a term and/or a sign and its reference. Codes and names are significant to the users when there is a strong semantic relationship between such codes and the items or actions they refer to.

8. **Compatibility:** it refers to the match between users' characteristics (memory, perceptions, customs, skills, age, expectations, etc.) and task characteristics on the one hand, and the organisation of the output, input, and dialogue for a given application, on the other hand. The criterion Compatibility also concerns the coherence between environments and between applications.

4.5 Coutaz Golden Rules

Joëlle Coutaz's "Seven Golden Rules" are a set of principles for interface design that focus on creating user-friendly and ergonomic computer interfaces.

These seven golden rules are essential in creating interfaces that are intuitive, efficient, and user-friendly. They are widely recognized in the field of Human-Computer Interaction (HCI) and are used as guidelines for designing interfaces that enhance the user experience. The 7 golden rules of Coutaz:

1. **Fight for consistency:** Ensure that the interface behaves consistently throughout, so users can predict how it will respond to their actions.
2. **Fight for conciseness:** Keep the interface simple and avoid unnecessary elements or information, making it easier for users to understand and navigate.
3. **Reduce cognitive load:** Minimize the mental effort required by users to operate the interface. This involves simplifying tasks and providing clear guidance.
4. **Place control in the hands of the user:** Allow users to have control over the system and their interactions with it, empowering them to make choices.
5. **Offer informative feedback:** Provide users with timely and meaningful feedback about their actions, helping them understand the system's response.
6. **Design dialogues to yield closure:** Structure interactions in a way that leads to a clear conclusion or endpoint, preventing user confusion or frustration.
7. **Prevent errors:** Anticipate and prevent errors through careful interface design, making it difficult for users to make mistakes.

Chapter 5

Multi-users interface design

1. Steps for multi-user interface design.
2. Comparative study between single-user and multi-user HMI.
3. User-centered design methods.
4. Example of multi-user interface.

Introduction

Designing interfaces for multiple users, often referred to as Multi-User Interface (MUI) design, involves creating interactive experiences that cater to the needs of multiple users simultaneously. It requires careful planning, performance optimization, user control, and consideration of hardware and collaboration elements, among other factors, to ensure a seamless and user-friendly experience. Usability testing and accessibility are also essential aspects of MUI design.

5.1 Steps for Multi-User Interface Design

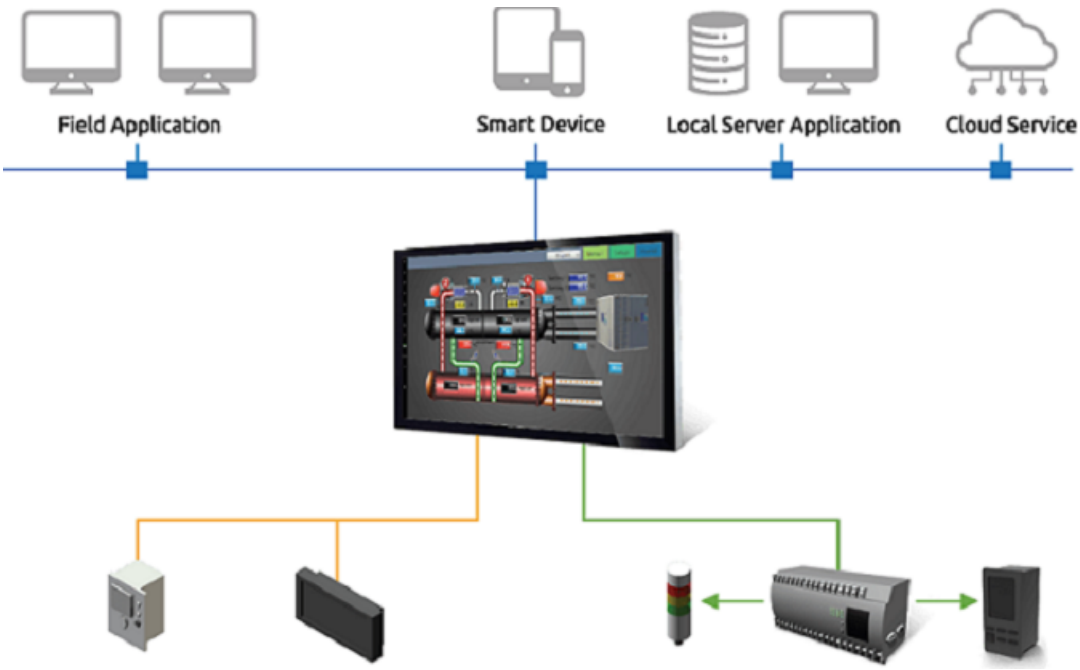
The principles and steps for effective Multi-User Interface design:

1. **User Scenarios:** Start by defining user scenarios that describe how different users will interact with the system or application.
2. **Performance Optimization:** Prioritize performance to ensure a seamless experience for all users. Efficiently handle data, processing, and communication to minimize lag and delays.
3. **User Control:** Empower users by placing them in control of their actions. Provide options for customization and user-specific settings to enhance their experience.
4. **Hardware Consideration:** Design with respect to the hardware on which the interface will run. Optimize the interface for the capabilities and limitations of the devices being used.
5. **Model-Based Design:** Consider employing model-based design techniques, which involve creating abstract representations of the interface and its behavior before implementation.
6. **Collaborative and Social Elements:** If the multi-user interface is collaborative or social in nature, design features that facilitate communication, sharing, and cooperation among users. Incorporate chat, collaboration tools, or social networking components as needed.
7. **Usability Testing:** Regularly conduct usability testing with actual users to gather feedback and refine the interface.
8. **Accessibility:** Ensure that the multi-user interface is accessible to users with disabilities. Incorporate features such as screen readers, keyboard navigation, and alternative text for images to make the interface inclusive.

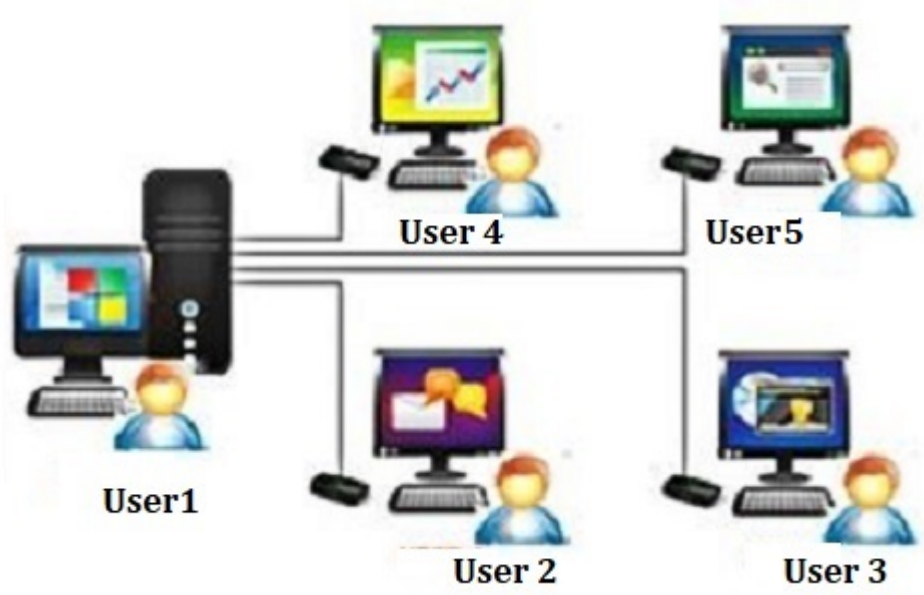
5.2 Single-user Vs Multi-User HMI

Single-user and Multi-User HMI systems refer to how human operators interact with machines and control systems:

- **Single-User HMI:** In a single-user HMI system, there is one operator or programmer responsible for controlling and monitoring the machine or system. This type of HMI setup is suitable for scenarios where only one person needs access to the control interface at a time, such as small-scale machines or standalone equipment.



- **Multi-User HMI:** A multi-user HMI system involves multiple users or teams working simultaneously on the same project or machine, often with shared resources and goals. It allows multiple operators to interact with and control the same system concurrently, which is useful in larger-scale industrial environments where collaboration is essential.



The choice between single-user and multi-user HMI depends on the specific requirements of the application, including the need for collaboration and the complexity of user access management. Here's a comparison of the two:

Property	Single-User HMI	Single-User HMI
Restriction	Single operator or user at a time	Multiple users to access the system simultaneously
Collaboration	Limiting collaborative work	Real-time collaboration among users
Simplicity	Simpler to set up and manage	Complex due to the need for access control and user management
Use Cases	One user interacts with a machine	Several operators interact and control a system together

5.3 User-centered design methods

User-centered design (UCD) is an iterative design process in which designers focus on the users and their needs in each phase of the design process. In UCD, design teams involve users throughout the design process via a variety of research and design techniques, to create highly usable and accessible products for them.

In UCD, designers use a mixture of investigative methods and tools (surveys and interviews) and generative ones (brainstorming) to develop an understanding of user needs. The term was coined in the 1970s. Later, cognitive science and user experience expert Don Norman adopted the term in his extensive work on improving what people experience in their use of items.

5.3.1 Principles of UCD

ISO standard 9241-210:2019 defines six fundamental principles that form the basis of the user-centered design process:

1. **Design is based on understanding users, their tasks and their environment:** It is not enough to have a vague impression of the product's target group. User-centered design requires deep immersion into the lives of users.

2. **Users are involved throughout the entire development and design process:** This is one of the main differences to other approaches. Users are not just invited to assess a finished product, rather their opinions are the basis for development.

3. **The design process is guided by user ratings:** Users evaluate every prototype and every beta version, and this feedback is used to develop the product.

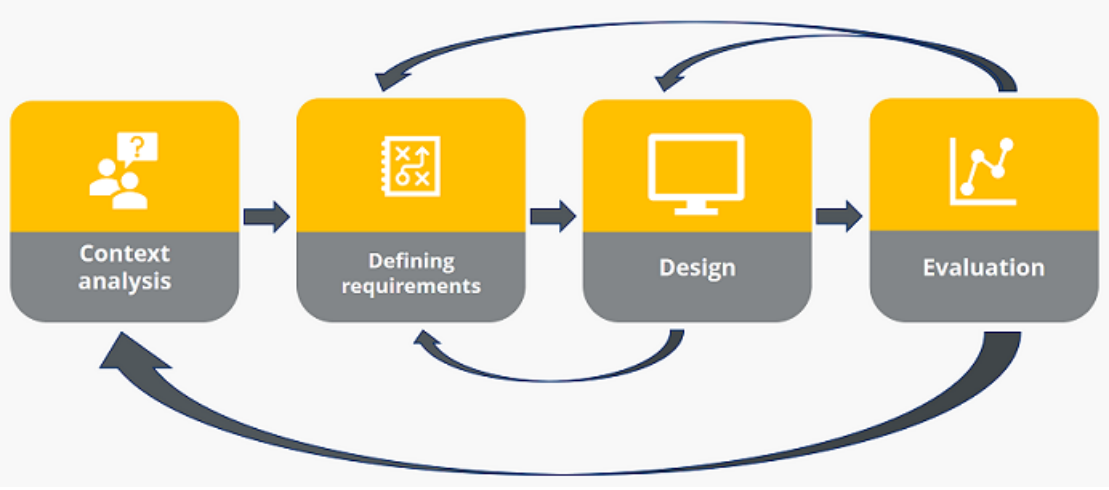
4. **The process is iterative:** The process steps in product development are performed non-linearly and repeatedly. Feedback from users can make multiple iterations of individual phases necessary.

5. **The entire user experience is taken into account:** The aim of user-centered design is not to make using a product as simple as possible. Instead, the process takes a broader view of the user experience. Products should evoke positive emotions, offer genuine solutions and encourage users to use them repeatedly.

6. **The project team is multi-disciplinary:** User-centered design requires close cooperation across disciplines. There is no room for silo mentalities in product development. User requirements can only be implemented optimally if copywriters, graphic designers, and programmers share their different perspectives.

5.3.2 The UCC process

Generally, each iteration of the UCD approach involves four distinct phases:



1. **Context analysis:**

   The first step is to analyze the context in which users will use the product. Who are the future users, and what are their specific applications for the product? Project teams can find answers by observing and surveying potential users.

2. **Defining the requirements:**

   The second step is to define the specific requirements for the new product. This step describes user requirements, taking corporate requirements into account.

3. **Design:**

   The actual design process doesn't start until the requirements have been defined. In the first instance, designers will usually create a simple prototype, e.g. using paper, followed by digital wireframes, and finally produce a finished prototype.

4. **Evaluation:**

   After a prototype has been produced, the project team asks potential users for feedback. For digital applications, this is generally done via extensive user testing and qualitative surveys. Surveys and tests assess effectiveness (can users achieve what they want?), efficiency (how quickly can users achieve their objective?) and general satisfaction.

The project team returns to step 2 or 3 in the design process with the new information to optimize the product. These iterations continue until satisfactory user feedback is achieved, taking into account the corporate frameworks (time and costs).

5.3.3 Advantages of UCD

A consistent focus on user-centered design not only benefits the user, it's also worthwhile for companies.

- **Customer satisfaction:** Close integration of users in the early stages of the development process means that the end product is more likely to meet the customer's expectations. This boosts revenue and reduces customer service costs.
- **Product safety:** The project team develops a product for a specific target group and a specific use case. As this information is considered in detail, the risk of inappropriate usage, which could endanger the user, decreases.
- **Quality:** When developers and designers get to know the needs, fears, and requirements of customers, they develop empathy. This results in more ethical and ergonomic products. Aspects that could have been neglected otherwise – like privacy or usability – become important.
- **Sustainability:** Because the perspective during development is shifted to the needs of potential customers, the resulting products appeal to a broader customer base. In this way, user-centered design also contributes to a company's sustainability goals.
- **Cost efficiency:** The costs for alterations remain relatively low, because user feedback is considered right from the start and not only at the end of the product development phase. This enables developers to incorporate user feedback from the get-go.
- **Competitive advantage:** Since not all companies have made user-centered design their top priority or have had difficulty implementing it, companies that work according to an effective user-centered design process can set themselves apart from the competition.

It remains to be seen whether the marketing sector will be talking about user-centered design, human-centered design or people-centered design in the years to come. Implementation methods are not static; they will continue to change.

However, one thing is already clear: user-centered design approaches aren't just a temporary fad, they're already best practice in the digital sector. And they will grow in importance in the future given the volatility, uncertainty, and complexity of the market.

5.3.4 UCD vs. HCD

User-centered design is very often used interchangeably with human-centered design, but there is a difference in that it is a subset of it. Simply put, all users are humans, but not all humans will be your users. Thus, user-centered design requires deeper analysis of users – your target audience. It is not only about general characteristics of a person; it is about particular habits and preferences of target users to come up with right solutions for specific problems.

User-centered design takes into account age, gender, social status, education and professional background, influential factors, product usage expectations and demands and many other important things that may vary for different segments. What is critical for some may be irrelevant for others. User-centered design is about deep research on users' habits, from their interactions with the product to their vision of how the product should look like and behave.

5.3.5 User centred design examples

- **Focus groups:** involves encouraging an invited group of intended or actual users of a site or digital service to share their thoughts, feelings, attitudes, and ideas on a certain subject. Focus groups are most often used as an input to design. They generally produce non-statistical data and are a good means of getting information about a domain.

- **Usability testing:** Usability testing sessions evaluate a site by collecting data from people as they use it. A person is invited to attend a session in which they'll be asked to perform a series of tasks while a moderator takes note of any difficulties they encounter. Usability testing can be used as an input to design or at the end of a project. It's an excellent way to uncover key usability and digital accessibility issues with a site or digital prototype.

- **Card sorting:** Card sorting is a method for suggesting intuitive structures/categories. A participant is presented with an unsorted pack of index cards. Each card has a statement written on it that relates to a page of the site. Card sorting is usually used as an input to design. It's an excellent way of suggesting good categories for a site's content and deriving its website information architecture.

- **Participatory design:** Participatory design does not just ask users for their opinions on design issues, but actively involves them in the design and decision-making processes. An example would be a participatory design workshop in which developers, designers, and users work together to design an initial prototype. This initial digital prototyping would then feed into a more traditional design process.

- **Questionnaires:** A questionnaire or quantitative survey is a type of user research that asks users for their responses to a pre-defined set of questions and are a good way of generating statistical data.

- **Questionnaire considerations:** Questionnaires allow statistical analysis of results. This data can increase a study's credibility, but it's important that the survey is well designed and asks non-biased questions.
- **Interviews:** An interview usually involves one interviewer speaking to one participant at a time. The advantages of an interview are that a participant's unique point of view can be explored in detail. It is also the case that any misunderstandings between the interviewer and the participant are likely to be quickly identified and addressed.

5.4 Examples of multi-user interface

A multi-user interface refers to a system or software environment that allows concurrent access by multiple users. Here are some examples and contexts in which multi-user interfaces are used:

1. **Multi-User Operating Systems:** Multi-user interfaces are commonly associated with multi-user operating systems, such as Unix, Linux, and Windows Server.
2. **Collaborative Software:** Multi-user interfaces are often found in collaborative software applications. Examples include Google Docs, Microsoft Teams, and collaborative design software.
3. **Database Applications:** In the context of database applications, a multi-user interface allows multiple users to access and interact with a shared database concurrently. This is crucial in enterprise systems where multiple users need access to the same data without conflicts.
4. **Multi-User Interface Development:** There are also tools and environments designed specifically for developing multi-user interfaces, such as video conferencing platforms or online gaming environments.

Chapter 6

Interfaces Adaptatives

1. The Vaudry Model.
2. Study of an example: Agent model.

6.1 Introduction

"Interfaces adaptatives" mean the design and implementation of methods and techniques that allow interfaces to automatically adapt to different users. This concept is particularly relevant in the field of human-computer interaction and user experience design. Adaptive interfaces aim to provide a personalized and user-friendly experience by adjusting elements such as layout, content, or functionality to suit the individual needs and preferences of the user.

6.2 The Vaudry Model

According to Christophe Vaudry, the design of adaptive HMIs has been the subject of a great deal of work. It turns out that this is a very complex problem as evidenced by the borrowings from different branches of information sciences and human sciences such as psychology, artificial intelligence, pedagogy, Human-Machine interfaces, etc.

6.2.1 Aspects of adaptation

Adaptation in human-machine interaction can be considered at least according to the five following aspects:

1. The reason for the adaptation.
2. The object of the adaptation.
3. The trigger for the adaptation.
4. The moment of adaptation.
5. The means used to achieve the adaptation.

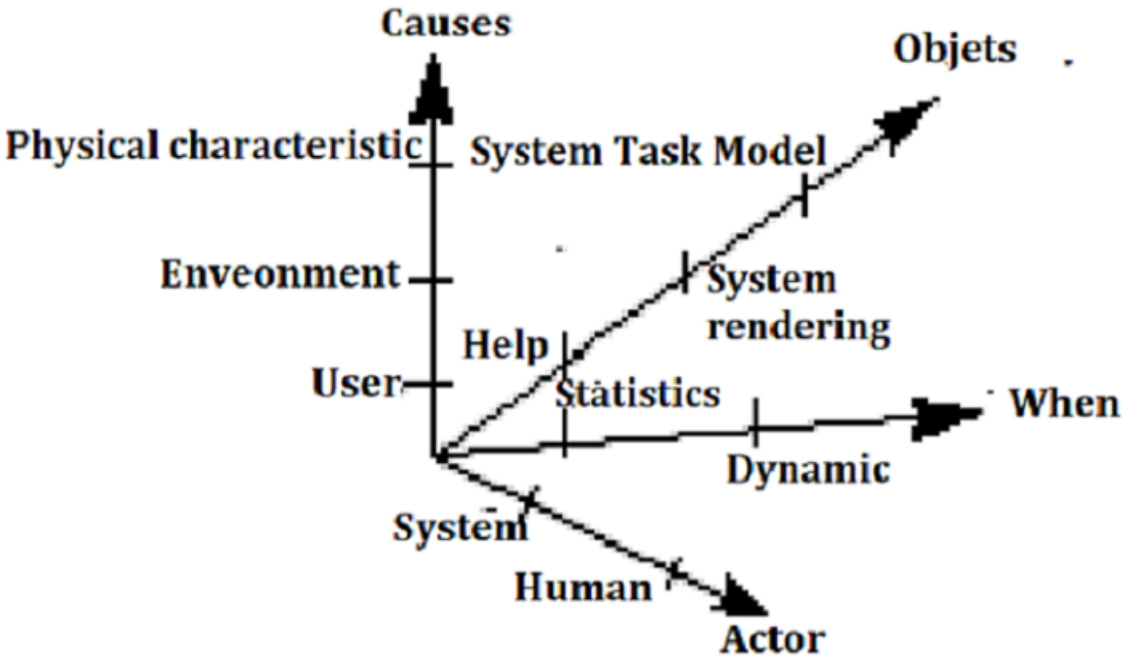
6.2.2 Principle of Adaptation in HMI(s)

The different tasks carried out on the adaptation system made it possible to identify different types of adaptations. The adaptation of an application is the ability to satisfy users in their dialogue situations. Remember that the interaction (or use) situation is a triple <User, use, context> and that the context summarizes the platform or reading device and the environment used.

The system is considered adaptable if it can meet the user's customization request. From a design perspective, adaptability is paramount. A system is said to be adaptable if it can automatically adapt to the user or other usage factors. From a design perspective, adaptability is paramount.

Therefore, adaptation can be associated with many aspects of interactive applications:

1. The Human-Machine interface is divided into three sub-elements:
   - Interface display (position, size, image, sound, color, structure),
   - Interaction techniques and styles (menu, multiple selection, etc.).
   - Navigation
2. Filtered and/or reorganized data or information.
3. Modifiable services (additions, restrictions, configurations).



Depending on the axis of causes, it is adaptation to the physical characteristics of the target platform or adaptation to environmental conditions.

Depending on the when axis, the adaptation is static, that is to say that the constraints can be fixed at the design, and the interface is generated accordingly, or the adaptation is dynamic, that is to say say that during execution, the interface is adapted according to variations in constraints.

Depending on the object axis, plasticity is an adaptation of the task tree and/or the rendering technique.

Depending on the axis of the actors, the system or the human (designer or end user) ensures the adaptation.

6.2.3 Adaptability Vs Adaptivity

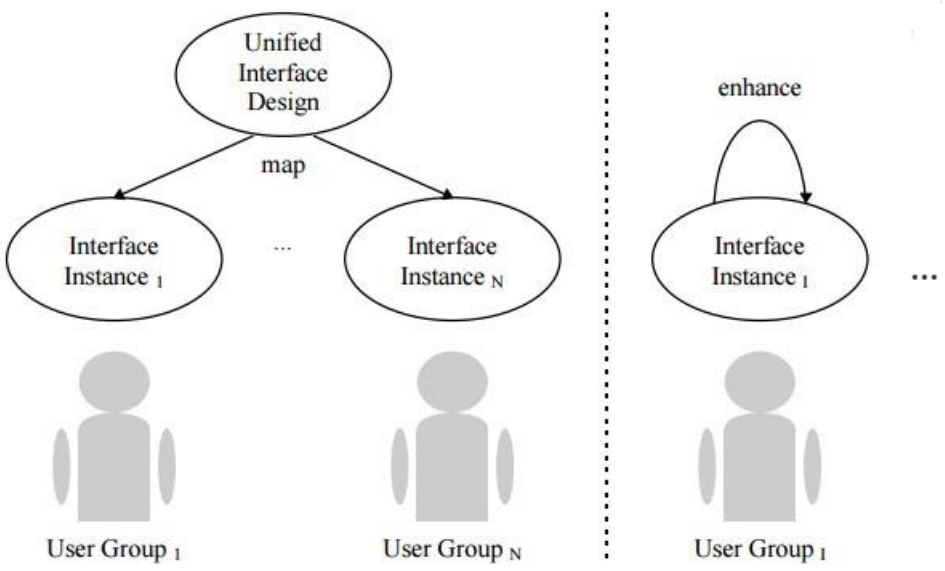
Interface customization is characterized by two characteristics: adaptability and adaptability. Adaptability means that the user can change the interface, and adaptability means that the interface can be changed automatically without any explicit user interaction.

Adaptability and adaptivity are terms used to describe "how" an intelligent mechanism can achieve its goal of adapting a user interface to a particular user.

Adaptability in this context is multiple instantiations of that user interface with a combination of components and attributes to tailor the user interface to a particular user group.

Adaptivity is a continuous extension of the user interface that is produced at runtime (based on feedback methods). There are not multiple predefined user interfaces. Only one (the user interface that interacts with real users) is constantly evolving.

Simply put, adaptability modifies the UI at instantiation (or earlier), and adaptivity modifies the UI at runtime when the user interacts with it.



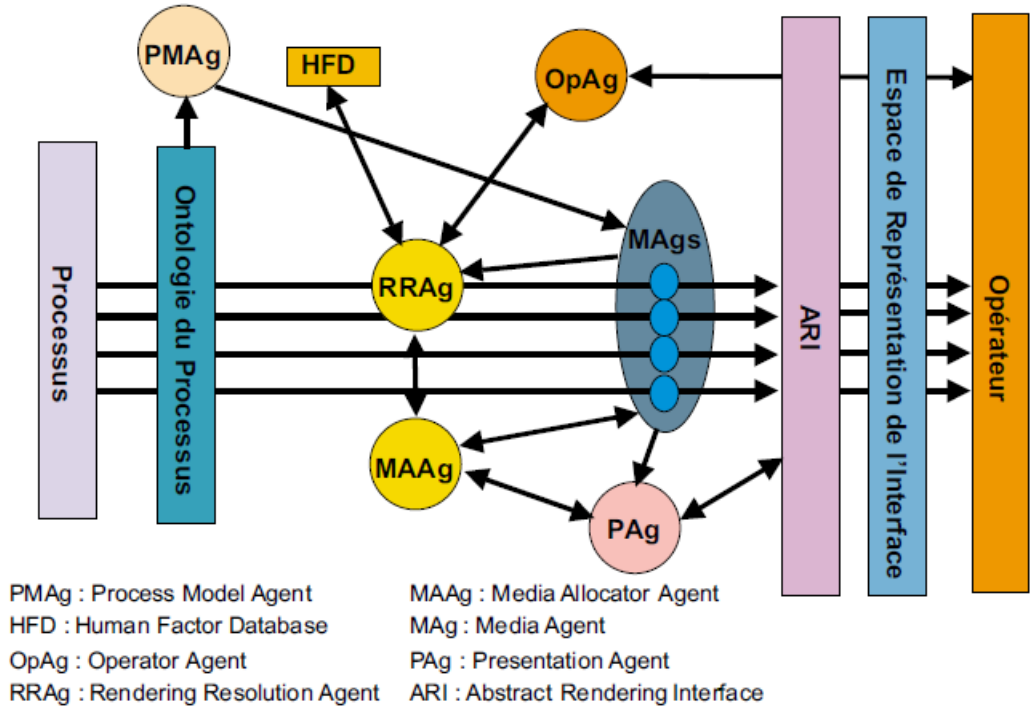
The following table summarizes the difference between adaptability and Adaptivity:

Adaptability	Adaptivity
User issues an explicit request, the system adapts to respond	System adapts itself thanks to user observation
Before user/system interaction	During user-system interaction
Static	Dynamic
Depends on the user's goal	Depends on the goal of the action

6.3 Study of an example: Agent model

The AMEBICA (**Auto adaptive Multimedia Environment Based on Intelligent Collaborating Agents**) project aimed to model, create and validate an architecture based on agents to achieve an adaptive composition applied to industrial supervision.

This architecture should make it possible to quickly generate applications specific to different industrial domains by specifying only the application context.



The AMEBICA architecture is made up of rational agents of the BDI (Belief-DesireIntention) type which follow a notion of weak agent. Their characterization defines a rational agent as possessing the following characteristics:

- Reactive. It reacts to the actions of other agents.
- **Proactive:** It is directed by its own internal goals.
- **Autonomous:** He does not undergo direct and continuous supervision.
- **Social:** He interacts with other agents.

6.3.1 The Agent Presentation free space calculation algorithm

This algorithm is based on the use of a scan line. The function of the Presentation Agent is to calculate the rectangular areas available on the screen, based on knowledge of the rectangular areas already occupied by elements of the interface (which correspond to the information displayed by Media Agents. The information on the free rectangular areas is then given to the Media Allocator Agent which deduces an appropriate placement of the different information.

The two main data structures of this algorithm are:

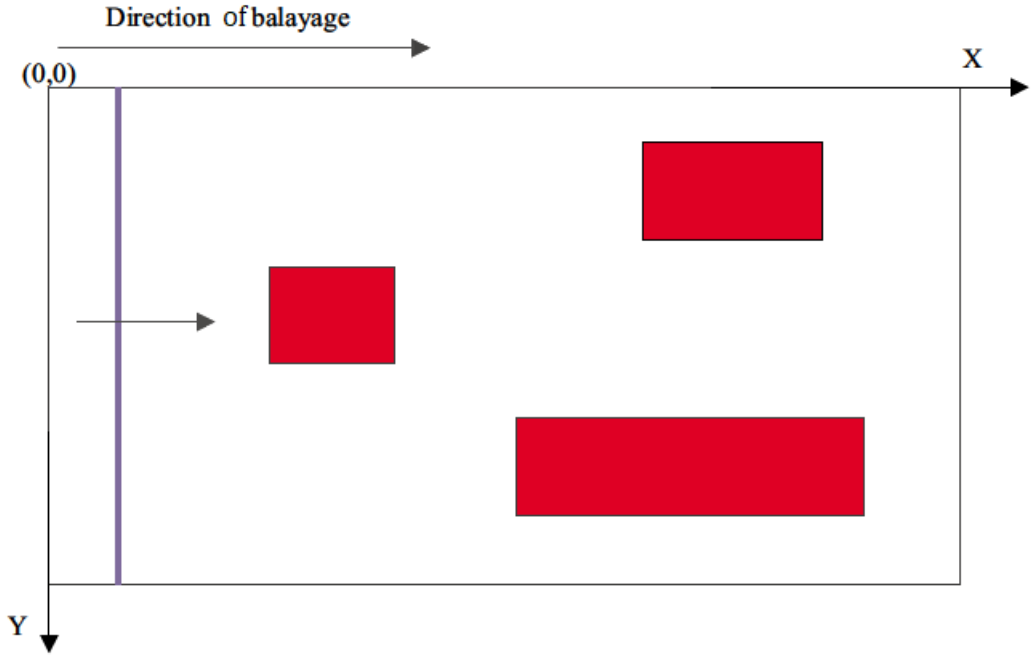
1. A data structure for representing scanlines. This data structure includes the start and end ordinate of the segment, the current abscissa and an associated available rectangle.

2. A data structure to represent available or occupied rectangles. This data structure includes the coordinates of the upper left point of the rectangle, its width, its length, and a Boolean indicating whether it is available or busy.

6.3.2 Example of screen splitting

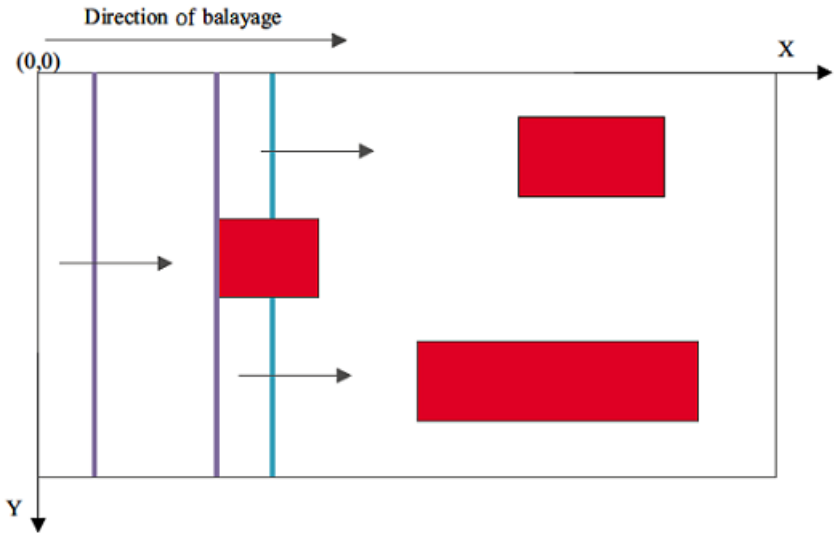
We consider that the scanning is carried out in a rectangle representing the screen. A list of occupied rectangles is given as input to the algorithm. The algorithm outputs a list of available and occupied rectangles that make up the screen in question.

The scanning is carried out along the abscissa axis. The algorithm maintains a list of scanlines. Indeed, each time a scan line encounters an occupied rectangle during its course, the scan line is deleted or fragmented into one or two scan lines.

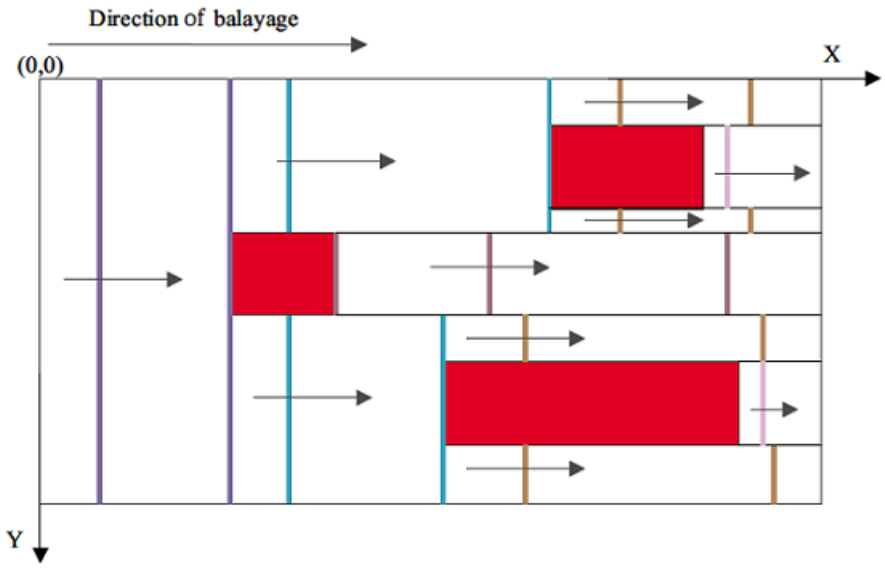


In the opposite situation, when the scan leaves an occupied rectangle, a new scan line is created. The length of this line is equal to the height of the occupied rectangle, the starting ordinate of the scanning line having the value of the ordinate of the point Upper right of occupied rectangle.

Each scan line constructs a rectangle marked as available as it passes through the screen. Before destroying a scan line, the associated available rectangle is added to the list of rectangles making up the screen.



During scanning, two cases can occur: a scanning line is fragmented or destroyed and a new scanning line is created:



Chapter 7

Multimodal Interfaces and Future Interfaces

1. Advanced interaction techniques, Augmented Reality, Tangible, Interface, 3D projection, Movement Analysis).

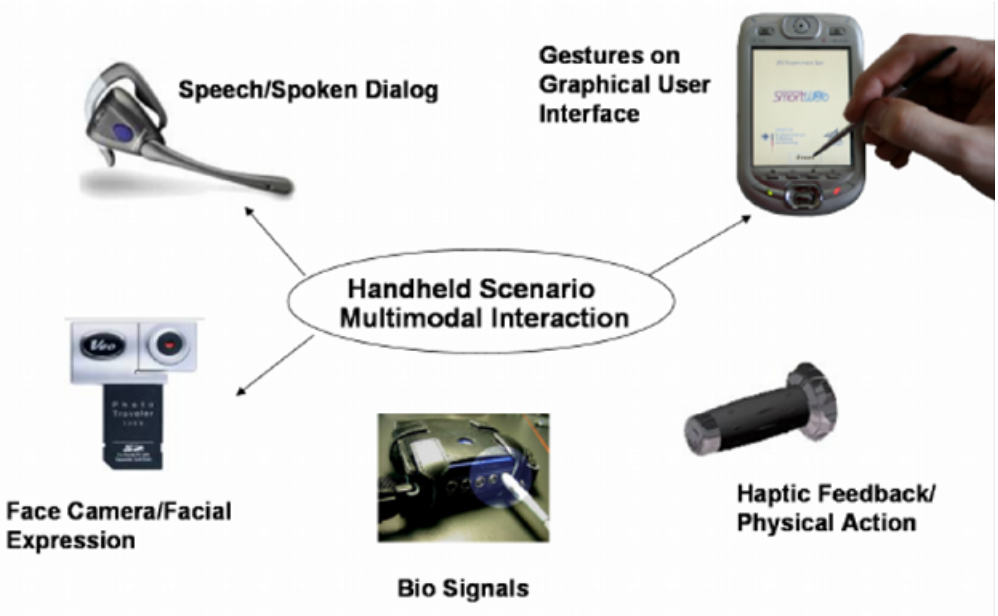
2. Visual Programming.

7.1 Introduction

Multimodal interfaces combine multiple modes of interaction, such as touch, voice, gesture, and gaze, to create a more intuitive and efficient user experience. These interfaces are designed to accommodate diverse user preferences and abilities. Examples of multimodal interfaces include smartphones with touchscreens and voice recognition, as well as virtual reality systems that incorporate hand gestures and eye tracking.

7.2 Multimodal interfaces

They are a new class of emerging systems that aim to recognize naturally occurring forms of human language and behavior, with the incorporation of one or more recognition-based technologies (e.g., speech, pen, vision). Multimodal interfaces represent a paradigm shift away from conventional graphical user interfaces.

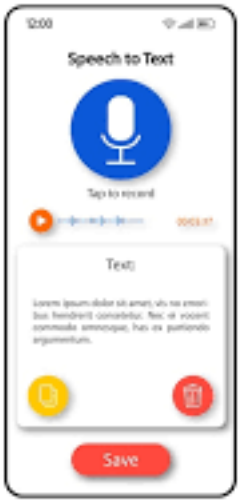


They are being developed largely because they offer a relatively expressive, transparent, efficient, robust, and highly mobile form of human–computer interaction. They represent users' preferred interaction style, and they support users' ability to flexibly combine modalities or to switch from one input mode to another that may be better suited to a particular task or setting.

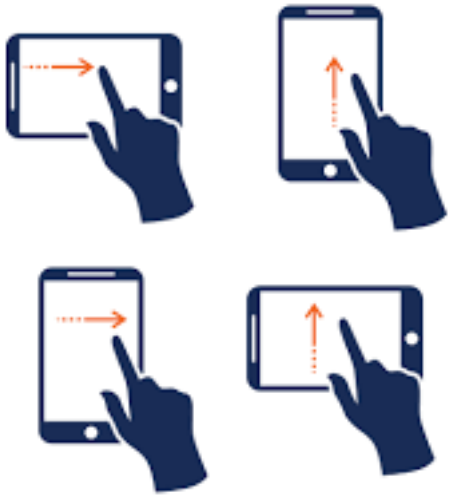
7.2.1 Type of multimodal interfaces

Multimodal interfaces are user interfaces that allow users to interact with a system or device using multiple modes of communication. These interfaces are designed to enhance the user experience by combining various input and output methods. Here are some common types of multimodal interfaces:

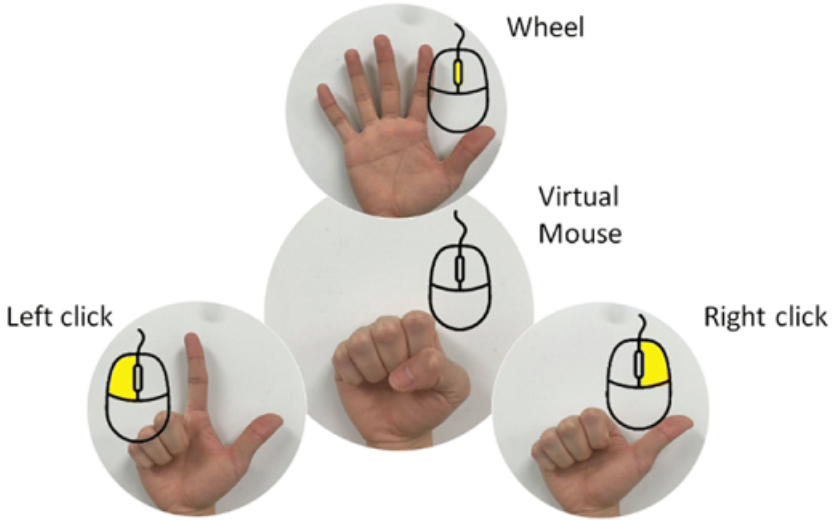
1. **Text and Speech Interface:** Users can interact with the system through both text and speech, such as voice commands and text input.



2. **Gesture and Touch Interface:** These interfaces combine gestures and touch-based interactions, commonly used in touchscreens and motion-sensing devices.



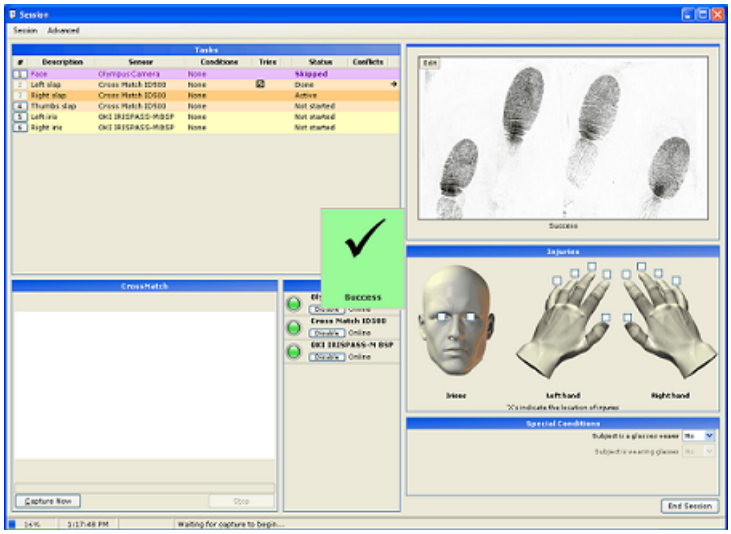
3. **Speech and Gesture Interface:** Users can interact with the system using both spoken commands and physical gestures, often used in gaming and virtual reality applications.



4. **Voice and Visual Interface:** This combines voice recognition with visual elements, like on-screen displays, making it common in smart home devices and virtual assistants.

5. **Touch and Visual Interface:** Users can use touch inputs in conjunction with visual elements, often seen in mobile devices and tablets.

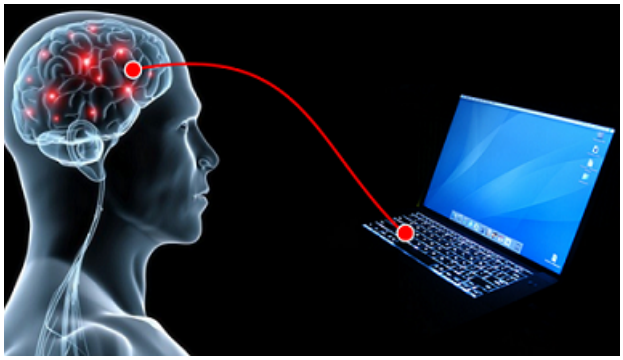
6. **Multimodal Biometric Interface:** Combines various biometric authentication methods, such as facial recognition, fingerprint scanning, and voice recognition, to provide a secure and user-friendly interface.



7. **Haptic and Visual Interface:** This type combines haptic feedback (tactile sensations) with visual information, enhancing user interaction in virtual reality and gaming.

8. **Eye Tracking and Gesture Interface:** Utilizes eye tracking technology in combination with gestures for a more natural and immersive user experience, often found in gaming and augmented reality applications.

9. **Brain-Computer Interface (BCI):** BCI systems allow users to control devices or interact with systems using brain signals, which can be combined with other modalities for more versatile interactions.



10. **Audio and Braille Interface**: Designed for users with visual impairments, this type combines audio feedback with Braille displays for text-based communication.

7.3 Advanced interaction techniques

Methods and approaches used to enable more complex and sophisticated interactions between humans and computers. These techniques are commonly used in user interface design, human-computer interaction, and user experience (UX) design. There are some advanced interaction techniques:

7.3.1 Augmented Reality

Augmented Reality (AR) and Virtual Reality (VR) technologies provide immersive, 3D environments that users can interact with. Augmented reality is an interactive experience that enhances the real world with computer-generated perceptual information. Using software, apps, and hardware such as AR glasses, it overlays digital content onto real-life environments and objects.

7.3.2 Tangible Interface

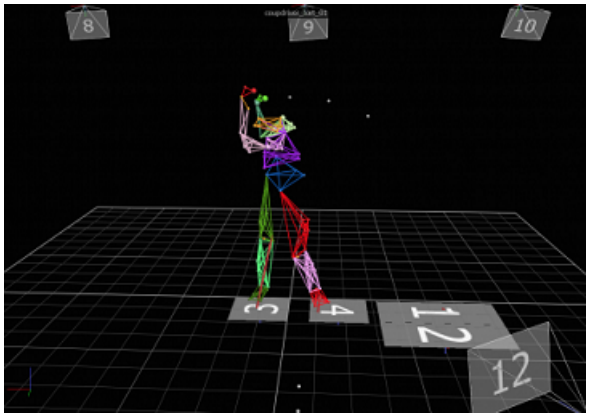
Is a type of user interface that allows users to interact with digital systems using physical objects or tangible elements. These interfaces bridge the gap between the physical and digital worlds, enhancing the user experience and providing more intuitive ways of interacting with technology.

7.3.3 3D projection (Hologram)

the technique of representing three-dimensional objects or scenes in a two-dimensional space, like a screen or a canvas. It's widely used in various applications, including cinema, events, and video games.

7.3.4 Movement Analysis

Movement analysis is an assessment of an individual's motion. It may combine the assessment of biomechanics by a trained individual or the use of technology such as video analysis. Neurophysiology should also be considered in movement analysis.



7.4 Visual Programming

Visual programming is a programming paradigm that utilizes graphical elements, such as icons, symbols, and diagrams, to create and manipulate code. It is often used to make programming more accessible to individuals who may not have a strong background in traditional text-based coding.

7.4.1 Characteristics of Visual Programming

Visual programming is a powerful approach for teaching programming concepts, prototyping, and creating applications with a focus on ease of use and accessibility. It can be a valuable tool for both beginners and experienced developers, depending on the specific use case. It hvae the followig aracteristics:

- **Visual Interface:** Visual programming environments provide a graphical user interface (GUI) where users can drag and drop visual elements to create, edit, and organize code.
- **Blocks or Nodes**: Visual programming often involves using blocks or nodes as building blocks. Each block represents a specific action or

function, and they can be connected to form a flowchart-like structure that defines the program's logic.

- **Connections and Flow:** Users connect blocks or nodes to define the flow of the program. This flowchart-like structure makes it easier to understand the program's logic.
- **Events and Triggers:** In visual programming, you can often define events or triggers visually, which then lead to specific actions or functions being executed. This is commonly used in event-driven programming.
- **Data Flow:** Data can be visually represented and manipulated through connections between blocks. You can see the flow of data within the program, making it easier to understand data processing.
- **Code Generation:** Underlying code is generated automatically based on the visual elements and their connections. Visual programming environments may generate code in various programming languages, making it versatile.
- **Debugging Tools:** Visual programming environments typically include debugging tools to help identify and resolve issues in the code. Users can often set breakpoints, inspect variables, and step through the program visually.
- **Extensibility:** Many visual programming environments allow for the creation of custom blocks or nodes, enabling users to extend the functionality of the platform.
- **Applications:** Visual programming is used in various domains, including game development, robotics, web development, and data analysis. Some well-known visual programming languages and environments include AgentCubes, Blockly, LabVIEW, and TouchDesigner,

7.4.2 Elements of Visual Programming

Visual Programming, also known as Visual Programming Language (VPL), involves creating software applications using visual elements such as diagrams, flowcharts, symbols, and graphical components, rather than traditional lines of code.

The general goal of VPLs is to make programming more accessible to novices and to support programmers at three different levels: Syntax, Semantics, and Pragmatics. The main elements of Visual Programming include:

1. **Graphical Components:** Visual Programming utilizes graphical elements to represent different functionalities, making it easier for users to understand and design software.
2. **Text and Symbols:** Text and symbols are used to provide labels, descriptions, and context to the graphical elements within the program, enhancing readability and user-friendliness.
3. **Spatial Arrangements:** Visual expressions and spatial arrangements of text and graphic symbols are used as elements of syntax or secondary notation in the program, aiding in the logical organization of code.
4. **Flowcharts and Diagrams:** Visual Programming often involves the use of flowcharts and diagrams to represent the flow of the application, making it visually intuitive and accessible.