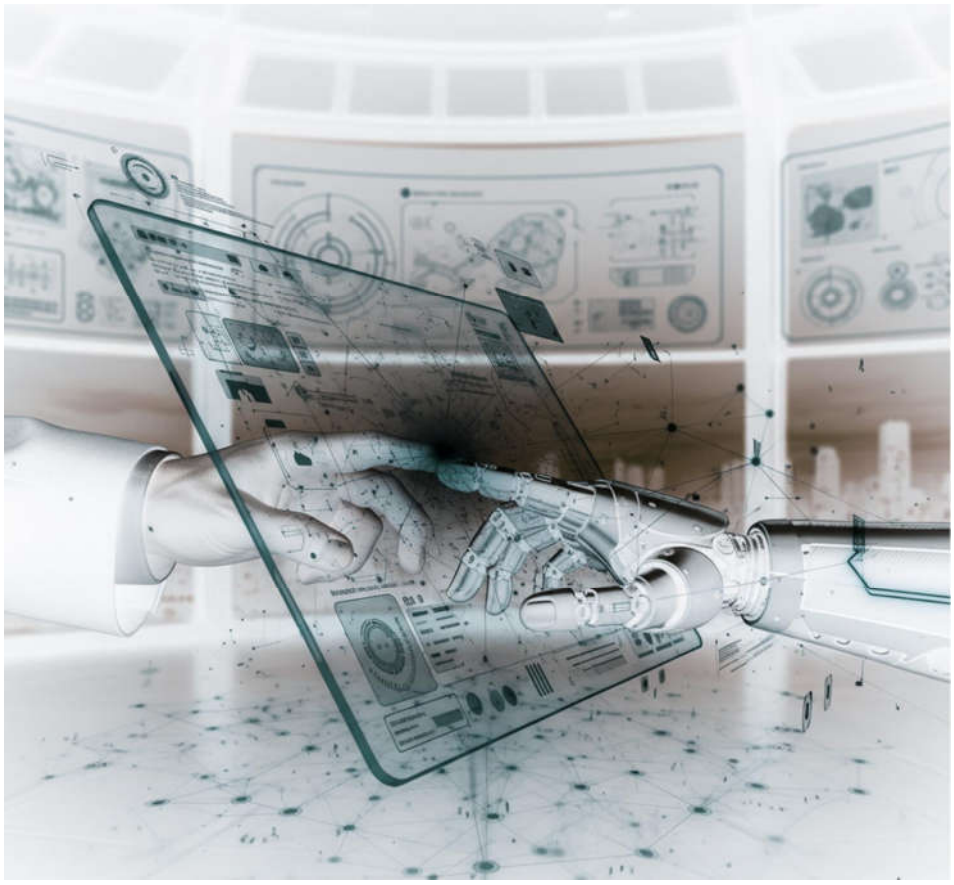


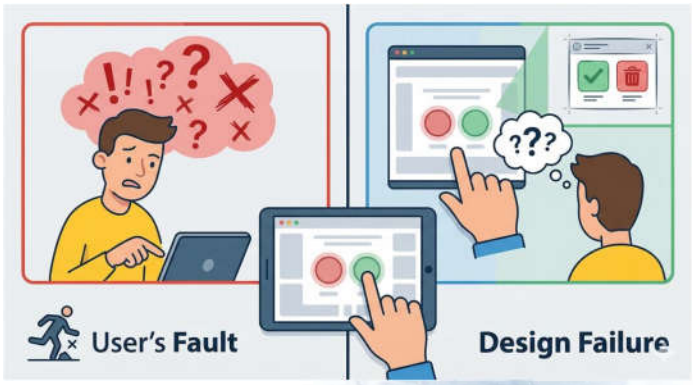
Human-Machine Interaction

User-Centered Design Methods

Dr. Patrick Chan
patrickchan@scut.edu.cn
South China University of Technology

Who's fault?

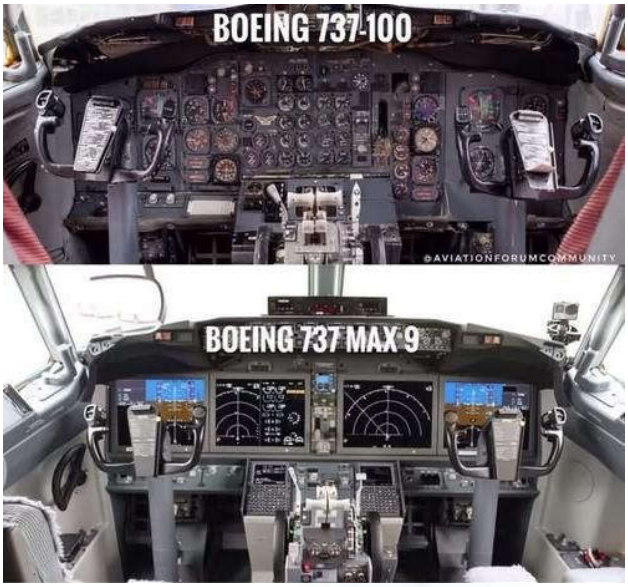
- Imagine users frequently press the wrong button.
- **Is it the user's fault, or Is it a design failure?**



- **Machine-Centered Design**
  User should learn the system
- **Human-Centered Design**
  System should adapt to human behavior

Example: Aircraft Cockpits

- **Machine-Centered Design**
  - Too many instruments and dense information
  - Critical and non-critical data mixed together
- **Human-Centered Design**
  - Critical information (altitude, speed, attitude) is visually emphasized
  - Warnings use multiple modalities:
    - Color coding (red / yellow / green)
    - Visual + auditory alerts



"Technically Correct" Systems Fail

- Why a carefully designed, bug-free, fast, and feature-rich system **fails**?
  - Designers often build for "themselves" or "the average user" (who doesn't exist)

Segway Failure

Performance vs Experience

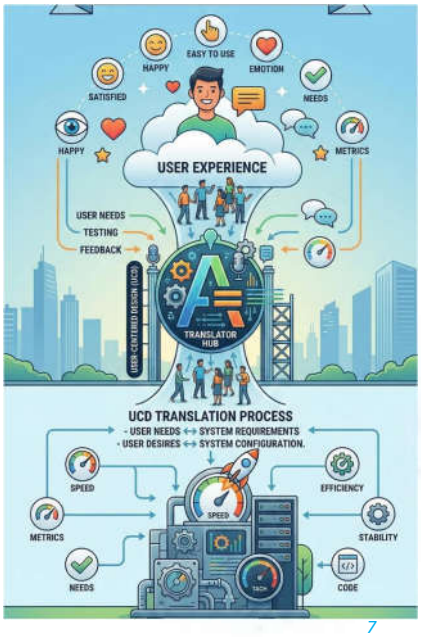
- **System Performance (SP)**
  - Measured by latency, throughput, uptime, and CPU usage
  - e.g., App loads in 0.5 seconds
- **User Experience (EX)**
  - Measured by satisfaction, learnability, and emotional response
  - e.g., App feels overwhelming and confusing
- Unfortunately, excellent system performance is **not necessary** to improve user experience

Example

- **Face Recognition Unlock**
  - Unintended Unlocking
  - No more “forgot password” excuse
- **Smart Autocorrect**
  - Changes intended meaning
  - Embarrassing wrong words
- **Auto-Play Video (Social Media)**
  - Plays sound unexpectedly
  - Feels intrusive

User-Centered Design (UCD)

- Aim to bridges the gap between user experience and system performance
  - Acts as the **translator** between complex machine logic and human mental models
  - Aims to **minimize** the mental effort required to operate a machine, allowing the user to focus on their primary task



UCD Process Overview

- UCD Process (**ISO 9241-210**) contains
  1. **User Research**
     Understand the users and goals
  2. **Task Analysis**
     Translate user needs into requirements
  3. **Prototyping**
     Develop design ideas and prototypes
  4. **Evaluation**
     Test against requirements with real users

User Research

- Systematic study of target users and their requirements, to add realistic contexts and insights to design processes
  - Uncover Why behind What
- Involves interview, survey, and observations techniques

User Research

Interview

- Conversation to understand users deeply
  - Explore the user's world, their terminology, and their daily "workarounds"
- Goal:
  - Understand behavior
  - Discover needs
  - Reveal mental models
  - Identify hidden problems

User Research

Interview: Type

- **Structured Interview**
  - Fixed set of questions
  - Good for comparing data across many users
  - Lacks depth
- **Semi-Structured Interview**
  - Has a guide/script but allows the interviewer to follow interesting "rabbit holes"
    - E.g. "You mentioned you hate the checkout process. Can you tell me more about the last time you abandoned a cart?"
  - More Preferrable
- **Unstructured Interview**
  - Free conversation
  - Explore ideas and possibility
  - Time-consuming

User Research

Interview: Do

- **Five Whys**
  - Analysis technique developed by Toyota
  - Reach the root cause of a behavior
  - Repeating asking "why" until the root cause is identified
- **Clarify Mental Models**
  - Understand how users think the system works
    - "What do you think this button does?"
  - Find out if there is mismatch between user's mental model and system model

User Research

Interview: Do

- **Observe Behavior**
  - Non-verbal language often reveals more than words
    - Users may say: “It’s fine.” But you might see: hesitation, confusion, repeated clicks.
- **End with Reflection**
  - Give the user space to step back at the end, think as a whole
    - “How would this fit into your daily workflow?”
    - “Would you use this in real life? Why or why not?”

User Research

Interview: Don't

- **Leading Questions**
  - Don't "prime" the user to agree with you
    - ❌ "Don't you think this dashboard is more organized than the old one?" (Leading the user to say "Yes").
    - ✅ "How does this dashboard compare to the way you managed this data previously?"
- **Binary Questions**
  - Binary questions (yes/no) kill the conversation
    - Just use survey!!
  - Open-ended questions is perferable
    - ❌ "Is this feature useful to you?"
    - ✅ "In what specific scenarios would you see yourself using this feature?"

User Research

Interview: Don't

- **Filling the Silence**
  - Designers often jump in to explain the product
  - When the user pauses. Stay silent, let them think.
    - ❌ "Is it clear? Any problems? What would you improve? You like it, right?"
    - ✅ User pauses. You wait. After a while: "What are you thinking?"
- **Selling the Product**
  - You are there to learn, not to pitch your idea
    - ❌ "Actually, this feature is very powerful. Let me show you how it's supposed to work."
    - ✅ "I noticed you hesitated there. What were you expecting to happen?"

User Research

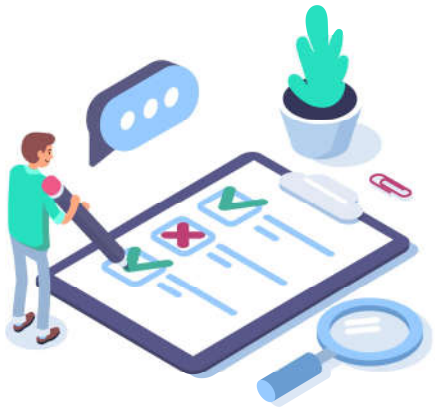
Interview

- **Suitable for**
  - Explore complex workflows
  - Understand decision-making
  - Study trust in automation
  - Investigate safety concerns
  - Discover unknown issues
- **Best for**
  - Early-stage design
  - Problem discovery
  - Safety-critical systems

User Research

Survey

- Structured questions to collect data from many users
- Verify the qualitative insights from a few interviews apply to a larger population
- Goal:
  - Measure opinions
  - Compare preferences
  - Identify patterns
  - Quantify experience



User Research

Survey: Question Type

- **Closed-ended Questions**
  - Yes / No
  - Multiple choice
  - Rating scale
  - Likert Scale
    - Strongly disagree → Strongly agree
    - Example: “Easy to use”
- **Open-ended Questions**
  - Harder to analyze
  - Example: “What frustrated you most?”

User Research

Survey: Do

- **Keep Survey short**
  - Long survey → Low quality data
- **Keep Question Simple**
  - One idea per question
  - As Short as possible
  - Clear language, no technical jargon
- **Pilot testing**
  - Test with 5–10 participants

User Research

Survey: Don't

- **Ambiguous wording**
  - ❌ Was the system fast
  - ✅ How long did the task take to complete
- **Unbalanced Scales**
  - ❌ Strongly disagree, Disagree, Agree
  - ✅ Disagree, No Comment, Agree

User Research

Survey: Don't

- **Leading Question**
  - ❌ "How helpful was this excellent feature?"
  - ✅ "How helpful was this feature?"
- **Assume User Knowledge**
  - ❌ "How does this compare to your previous enterprise workflow system?"
  - ✅ "Have you used a similar system before?" If Yes → "How does this compare to the system you previously used?"

User Research

Survey

- **Suitable for**
  - Measure satisfaction level
  - Compare design alternatives
  - Evaluate trust or workload
  - Identify common pain points
  - Collect data at scale
- **Best for:**
  - Large sample size
  - Statistical comparison
  - Validation of hypotheses

User Research

Observation

- Watching users in real context to understand real behavior
  - Captures real interaction, not reported opinion
  - Users say “It’s fine.” but you may see hesitation
- Goal:
  - See what users actually do
  - Identify hidden pain points
  - Detect mismatches between design and reality

User Research

Observation: Type

- **Direct Observation**
  - Researcher watches user in person.
- **Contextual Inquiry**
  - Observe + ask questions during task.
- **Remote Observation**
  - Screen recording, system logs, video.
- **Shadowing**
  - Follow user through their workflow.

User Research

Observation: Content

- **Behavior**
  - Task sequence, Repeated actions, Errors, Backtracking
- **Time**
  - Task completion time, Delays, Pauses
- **Body Language**
  - Facial expression, Posture, Eye movement
- **Workarounds**
  - **Create their own solutions** → Strong signal of design failure

User Research

Observation: Example

- If you see users sticking Post-it notes on their monitors with instructions, the interface is too complex or lacks necessary information
- Observing how people actually hold a heavy industrial controller led to the redesign of button placement for one-handed use.

User Research

Observation: Do

- **Observe Natural Behavior**
  - Let users work normally
  - Avoid over-instruction
  - Study real workflow
- **Focus on Behavior, Not Opinion**
  - Hesitation
  - Repeated actions
  - Workarounds
  - Errors
  - Pauses
  - Body language

User Research

Observation: Do

- **Capture Context**
  - Design problems often come from context, not interface alone
  - **Observe**: Environment (noise, lighting, interruptions), Stress level, Multitasking, Physical constraints
- **Follow-up Questions**
  - Observe first → Ask later
  - After observation:
    - "What were you thinking at that moment?"
    - "What were you expecting?"

User Research

Observation: Don't

- **Prove You Are Right**
  - Observation is not validation
  - It is discovery
- **Create Artificial Conditions**
  - Lab-only testing may **hide**:
    - Real-world stress
    - Environmental distractions
    - Workflow interruptions

User Research

Observation: Don't

- **Interrupt Immediately**
  - Jumping in destroys authentic behavior
- **Teach or Correct**
  - If you fix the mistake: You remove the evidence of design failure.

User Research

Observation

- **Suitable for**
  - Study workflow
  - Evaluate complex systems
  - Investigate safety-critical tasks
  - Understand physical interaction
  - Study multitasking environments
- **Best for**
  - Medical systems
  - Industrial control rooms
  - Autonomous vehicles
  - Robotics / HRI

User Research

Comparison

Aspect	Interview	Survey	Observation
Main Purpose	Understand **why** users think or feel a certain way	Measure **how much** / **how many**	See **what** users **actually do**
Type of Data	**Qualitative** (opinions, reasoning, emotions)	**Quantitative** (ratings, scores, percentages)	**Behavioral** (actions, timing, errors)
Sample Size	Small	Large	Small to medium
Depth of Insight	High	Low to medium	Medium to high
Reveals Mental Models	Yes	Limited	Indirectly
Reveals Real Behavior	Indirectly	No	Yes
Statistical Comparison	No	Yes	Limited
Time & Effort	High	Low to medium	Medium
Risk of Bias	Interviewer bias	Question design bias	Observer interpretation bias
Best Used When	Exploring complex problems	Validating and comparing designs	Studying workflow and safety

User Research

Example

- **Safety-Critical Example**
  - **Interview** reveals:
    "I don't fully trust the autopilot."
  - **Survey** shows:
    Trust score = 3.2 / 5
  - **Observation** shows:
    Driver keeps hands hovering over wheel
    Frequently checks dashboard
- Each method reveals different layers of truth

User Research

When to Use Which

- **Early Design Stage**
  - Interview + Observation (Discover problems)
- **Mid Design / Prototype Testing**
  - Observation + Interview (Identify usability issues)
- **Late Stage / Validation**
  - Survey (Measure satisfaction, workload, trust)
- Communicate with the users in any stage

User Research

Combined Approach

- **Strong** User Research combines:
  - Observe behavior
  - Interview to understand reasoning
  - Survey to measure at scale

- Discover → Understand → Validate

User Research

User Diversity

- **Diversity**
  Age, technical literacy, physical ability, and cultural background
- **Should We Design for the "Average User"?**

User Research

User Diversity

- **Myth of "Average User“**
  - Designing for the "middle" results means fitting no one
- **Principle**
  - Design for specific clusters of people, not generic
  - Design for the edges, and the middle will follow

User Research

Persona

- A **fictional character** representing a key segment of your target users
- Personas must be built from actual user research data (interviews/surveys)
  - Identify patterns
  - Group by behavior
  - Create a representative profile
- Purpose: To humanize data and make user needs memorable for the development team

User Research

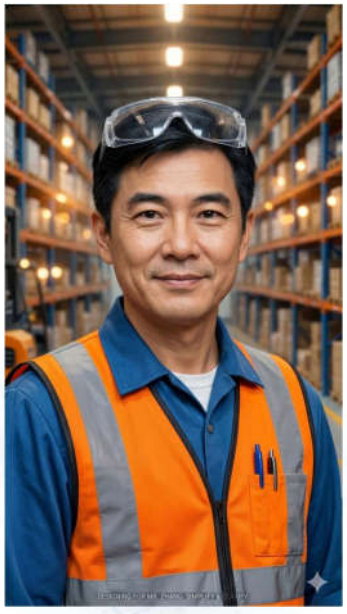
Persona

- Well-developed persona typically contains:
  - Demographic background (age, job, experience)
  - Technical skill level
  - Goals and motivations
  - Pain points and frustrations
  - Behavioral patterns

User Research

Persona: Example

- **Mr. Zhang, 45, Warehouse Operator**
  - 20 years of warehouse experience
  - Comfortable with machinery, not complex software
  - **Goal**: operate safely and efficiently
  - **Pain points**: Too many alarms, Unclear interface hierarchy, and Confusing status indicators
- Design **for Mr. Zhang**, not for engineers



User Research

Scenario

- A narrative describing how a persona interacts with the system to achieve a goal
- Focus on
  - **Context**
    Why are the users starting the task?
  - **Sequence of actions**
    How does the system help them finish?
  - **The environment**
    Where is the user?

User Research

Scenario

- **Purpose**
  - Clarifies when and how the system is used
  - Reveals critical interaction steps
  - Helps designers evaluate usability, timing, and safety
  - Connects user goals to system behavior
- A usage scenario describes **a specific situation of use**, not the user or the system in isolation.

User Research

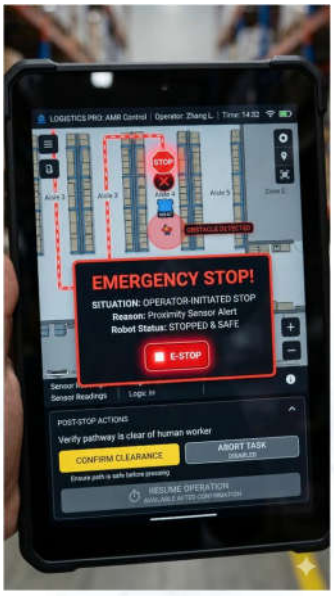
Scenario: Type

- **Normal Scenario**
  - When everything goes right
- **Edge (Failure) Scenario**
  - What if the Wi-Fi drops?
  - What if the user enters wrong data?

User Research

Scenario: Example

- **System:** Warehouse mobile robot control interface
- **User (Persona):** Mr. Zhang, warehouse operator
- **Scenario: Emergency Stop During Operation**
  - Mr. Zhang is supervising a mobile robot transporting goods in the warehouse.
  - The robot is moving autonomously along a predefined route.
  - Suddenly, a human worker steps into the robot's path.
  - The system issues a proximity alert on the control interface.
  - Mr. Zhang notices the alert and presses the Emergency Stop button.
  - The robot immediately stops and switches to a safe state.
  - Mr. Zhang confirms the situation and resumes operation.



User Research

Journey Map

- Visualize the user experience across steps and time
  - Covers the experience from "Before" to "After" the interaction
  - Shows User's Actions, Thoughts, Emotions, and Pain points
  - Reveal where design decisions impact user workload, stress, and safety over time

User Research

Journey Map: Example

- **System:** Warehouse mobile robot
- **User:** Mr. Zhang, warehouse operator
- **Goal:** Safely complete a delivery task

User Research

Journey Map: Example

- Stage 1

  Start Shift

  - **Action:** Logs into robot dashboard
  - **Thought:** "Is everything working properly?"
  - **Emotion:** Neutral
  - **Pain Point:** Interface loads slowly

- Stage 2

  Assign Delivery Task

  - **Action:** Selects destination and confirms route
  - **Thought:** "Hope there are no obstacles."
  - **Emotion:** Slight concern
  - **Pain Point:** Too many status indicators on screen

User Research

Journey Map: Example

- Stage 3

  During Robot Movement

  - **Action**: Monitors robot progress
  - **Thought**: "Battery looks low… is it enough?"
  - **Emotion**: Increasing anxiety
  - **Pain Point**: No clear battery risk prediction

- Stage 4

  Unexpected Obstacle

  - **Action**: Receives alert and presses emergency stop
  - **Thought**: "Is the system reacting fast enough?"
  - **Emotion**: Stress
  - **Pain Point**: Alarm sound unclear

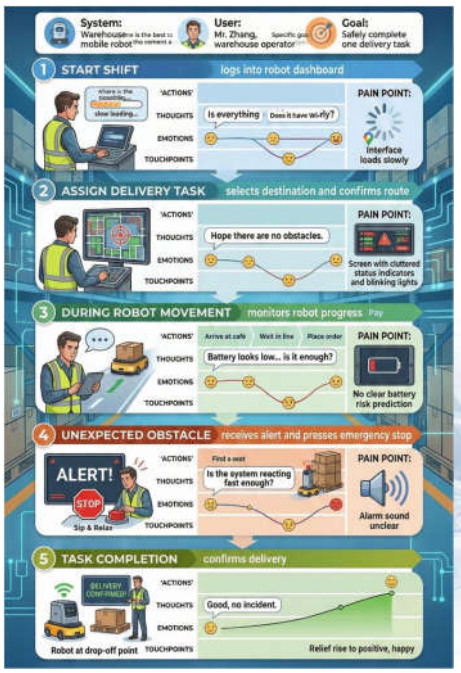
User Research

Journey Map: Example

- Stage 5

  Task Completion

  - **Action:** Confirms delivery
  - **Thought:** “Good, no incident.”
  - **Emotion:** Relief
  - **Opportunity:** Provide performance summary feedback



User Research

Journey Map

- Journey Map provides **insight**
  - **Optimization**
    - E.g. "Step 3 should be removed because it causes a 40% drop-off rate."
  - **Opportunity**
    - E.g. "Users are bored waiting for the machine to warm up; let's show them a progress status."

User Research

Journey Map

- **Common Mistake**
  - Changing the persona's needs midway through design to fit a new feature
  - Only mapping the positive parts of the experience
  - Focus on the graphic design

User Research

Common Mistakes

- **Small/Biased Samples**
  - Only talking to "friendly" users or internal staff
- **Ignoring Negative Feedback**
  - Dismissing user struggle as "user error" rather than a design flaw
- **Analysis Paralysis**
  - Spending long time on research while the development team moves on without you

User Research

Output

- **User Profiles / Personas**
  - Descriptions of typical users
- **User Needs & Requirements**
  - E.g. clear feedback
- **Context of Use**
  - Environment where the system will be used
  - E.g. physical environment, time pressure, tools already used
- **User Goals**
  - What users want to achieve.
  - E.g. reach destination quickly
- **Pain Points and Problems**
  - Problems users face with current systems.
  - E.g. Too many steps
- **Insights for Design**
  - E.g. users prefer voice input when driving

User Research

Discussion

- Users' opinions are very important
- **Should designers always follow what users say?**

User Research

Henry Ford Trap

- "If I had asked people what they wanted, they would have said faster horses." by **Henry Ford**

User Research

Henry Ford Trap

- User cannot provide transformational solutions
- Their opinions can be limiting
  - Constrained by experiences and expectations
  - Describe incremental improvements

User Research

Can vs Can't

- **Can**
  - What problems users are currently facing
  - How they currently complete a task
  - What motivates their behavior
- **Can't**
  - Exactly what features to build
  - Predicting the future
  - Market size or financial viability

Task Analysis

- Systematically breaking down what users do to achieve a goal
  - Steps, Decisions, Information flow, Errors, Cognitive demands
- **Goal**
  - Understand how work is actually performed
  - Improve system design and interfaces

Task Analysis

- Task is analyzed starting at **the goal**
  - Provided by User Research
- Example:
  - Goal: Set navigation destination
    Goal: Diagnose patient condition
    Goal: Adjust exoskeleton assistance level
- Analyze how work is actually performed to achieve the goal

Task Analysis Method

- **Hierarchical Task Analysis (HTA)**
  - Focus on Actions
  - Step sequence
  - Observable behavior
  - "What is done?"
- **Cognitive Task Analysis (CTA)**
  - Focus on Thinking
  - Decision process
  - Mental reasoning
  - "Why and how is it decided?"

Hierarchical Task Analysis (HTA)

- Identify the hierarchical structure of tasks users perform to achieve a goal (not how system should work)
  - Decomposes a high-level goal into progressively smaller sub-goals and actions until they are directly performable
  - Big goal → Smaller goals → Concrete action
- Purpose:
  - Understand task structure
  - Identify complexity
  - Detect error-prone steps

Hierarchical Task Analysis (HTA)

- Identify **task complexity**:
  - Too many steps
  - Unnecessary actions
  - Mode confusion
  - Unsafe transitions
- Important to safety systems:
  Task complexity increases error probability

Hierarchical Task Analysis (HTA)

- **A tree structure**:

Level 0	**Level 1**	**Level 2**	**Level 3** …
Goal	→ Subtask 1	→ Action 1.1	→ Operations 1.1.1
			→ Operations 1.1.2
			→ …
		→ Action 1.2	→ Operations 1.2.1
			→ …
	→ Subtask 2	→ …	→ …

- Each level specifies the subtasks required to achieve the goal at the higher level

Hierarchical Task Analysis (HTA)

- Example: Car Navigation
  - Goal 0: Set destination in car navigation
    - 1: Open navigation system
    - 2: Enter destination
    - 3: Select correct result
    - 4: Confirm route

Decomposing Further

- 2.1 Tap search bar
- 2.2 Type address
- 2.3 Review suggestions
- 2.4 Select correct address

Hierarchical Task Analysis (HTA)

- After Identifying the Task Structure
  - Identify Critical Task Elements
    - **Decision points**: What choices must the user make?
    - **Information needed**: What cues or data are required?
    - **Cognitive load**: How much thinking or comparison is required?
  - Identify Potential Errors and High-Risk Steps
    - where, what, why and how?
      - Complex decisions, Time pressure, High cognitive workload
- Translate Insights into Design Improvements

Hierarchical Task Analysis (HTA)

- **Common Mistakes**
  - Over-simplifying tasks
  - Skipping decision points
  - Not including environmental context
  - Focus on "how system should work"
    (Should focus on "how users actually work")

Task Analysis

Cognitive Task Analysis (CTA)

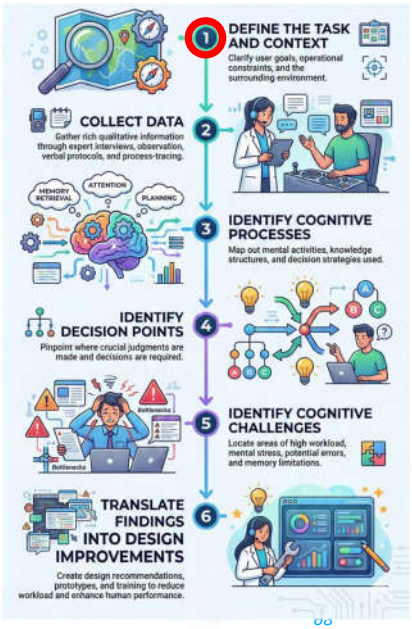
- Study the mental processes behind task performance
  - It maps thinking, not just behavior
  - Focuses on:
    - How users perceive
    - How users interpret
    - How users decide
- Cognitive misalignment → System failure

Task Analysis

Cognitive Task Analysis (CTA)

1. Define the Task and Context

- Clear scope prevents vague analysis
  - What task is being analyzed
  - Who performs the task (novice / expert)
  - Environment and constraints (time pressure, stress, automation)

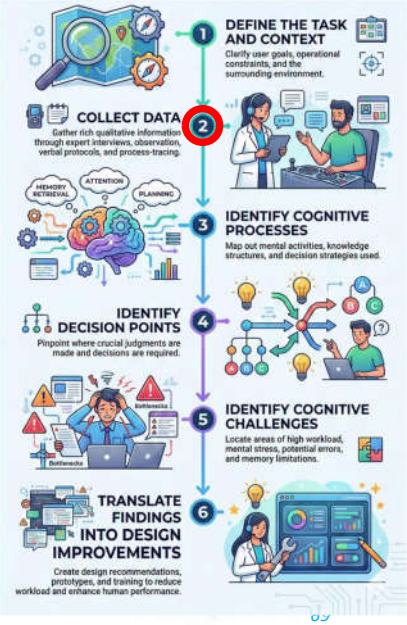


Task Analysis

Cognitive Task Analysis (CTA)

2. Collect Data

- Gather information about how the task is performed
  - Observation
  - Expert interviews
  - Think-aloud protocol
  - Simulation replay
  - Incident reports

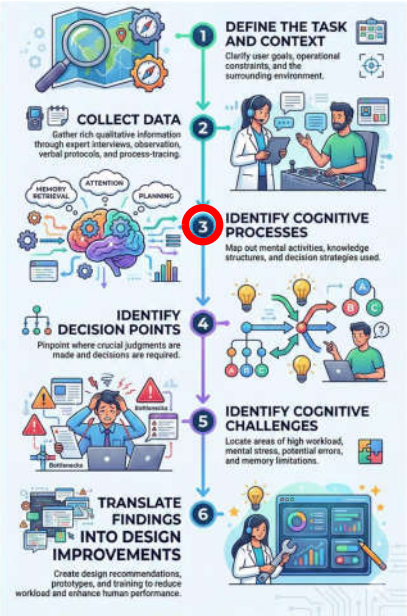


Task Analysis

Cognitive Task Analysis (CTA)

3. Identify Cognitive Processes

- Analyze what happens in the user's mind
  - What information is noticed?
  - What cues are used?
  - What mental model is applied?
  - What assumptions are made?
  - What alternatives are considered?

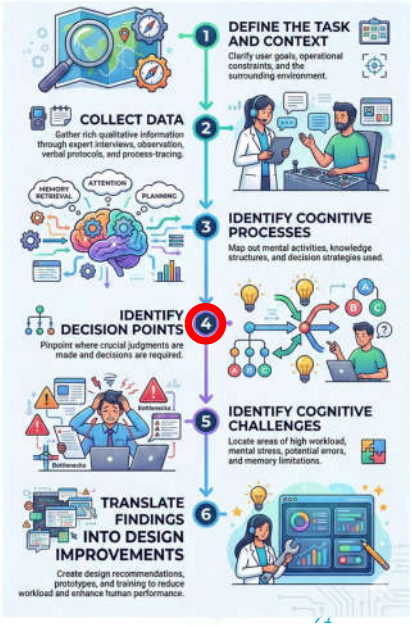


Task Analysis

Cognitive Task Analysis (CTA)

4. Identify Decision Points

- Determine where important decisions occur
  - What information supports it?
  - What uncertainty exists?
  - What alternatives are possible?

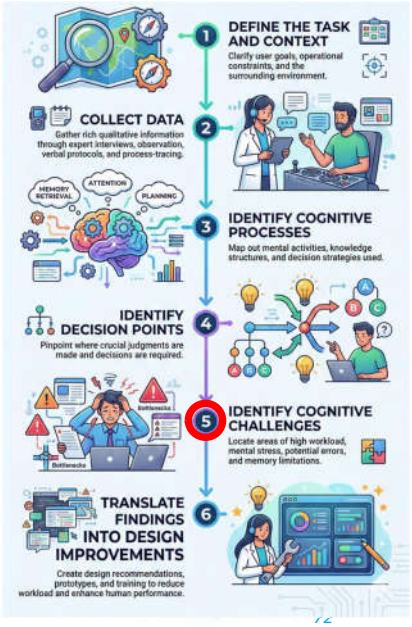


Task Analysis

Cognitive Task Analysis (CTA)

5. Identify Cognitive Challenges

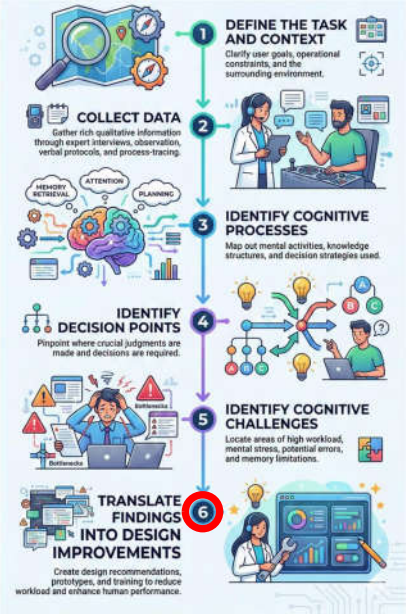
- Look for factors that may cause human error
  - Information overload
  - Poor alarm design
  - Mode confusion
  - Over-trust
  - Time pressure
  - Incomplete information



Cognitive Task Analysis (CTA)

6. Translate Findings into Design Improvements

- Convert CTA insights into
  - Interface redesign
  - Alarm improvements
  - Better information display
  - Automation transparency
  - Training improvements



Cognitive Task Analysis (CTA)

- Example: **Autopilot Handover**
  - **Goal**: Take over control
  - **Cognitive steps**:
    - Detect abnormal situation
    - Interpret warning
    - Assess risk
    - Decide whether to intervene
    - Plan corrective action
  - Physical action is simple but Mental processing is complex

Cognitive Task Analysis (CTA)

- **Common Mistakes**
  - Ignoring mental workload
  - Assuming experts think like novices
  - Only analyzing physical steps
  - Not considering uncertainty
  - Ignoring stress conditions
    - Cognition changes under pressure

HTA + CTA

- **Combined Framework**
  - **HTA** → identifies **task structure**
  - **CTA** → analyzes **decision making and cognition**
- Together they reveal:
  - Decision points
  - Cognitive workload
  - Potential errors
  - Mental model mismatches

Task Analysis

Output

- **Task Hierarchy**
  - Goal → Subgoals → Tasks → Actions
- **Task Flow / Task Sequence**
  - Task performing order
- **Task Plans**
  - The order or conditions of tasks
- **Identification of Critical Task Elements**
  - E.g. decision points, possible user errors, cognitive load
- **Task Documentation**
  - task hierarchy diagrams
  - workflow diagrams
  - task tables
  - descriptions of user actions

Prototyping

- Creating an **early version of a system** to explore and test design ideas
- Allows users to experience the design before it is fully built
- Purpose:
  - Visualize design concepts
  - Test usability early
  - Identify problems before development
  - Improve design through iteration

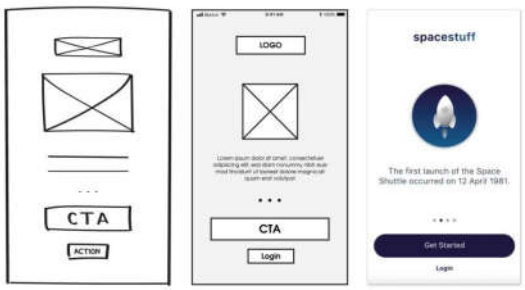
Prototyping

- Moving ideas from **head** to **physical world**
  - "**I think this works**" becomes "**Let users try it**."
- Visualize design concepts quickly
  - Identify problems early and reduce development risk
  - Evaluate usability and interaction
  - Communication

Prototyping

Fidelity

- **Degree** to which a model or prototype accurately represents the appearance, functionality, and interaction of the real system
  - **Low-Fi**
    Low Cost & Fast
  - **High-FI**
    Realistic & Use for validation



Prototyping

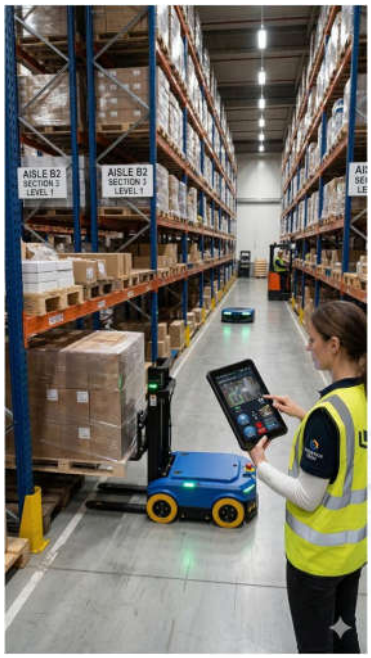
Fidelity

Fidelity	Aim	Characteristics	Pros	Cons
Low-Fi	Explore ideas early	Rough sketches, paper, simple wireframes	Fast, cheap, easy to change	Low realism, limited interaction
Mid-Fi	Test task flow and layout	Structured wireframes, basic interaction	Balanced realism and effort	Some behaviors simplified
High-Fi	Validate usability	Detailed, interactive, close to final system	Realistic, good for testing	Time-consuming, costly to change

Prototyping

Example

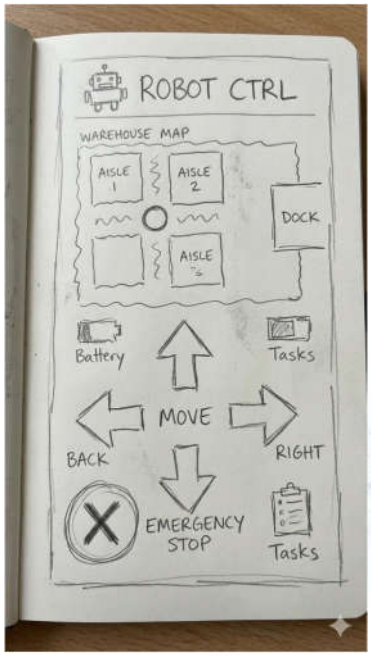
- A **logistics company** wants to design a control interface for operators to manage warehouse robots
- Three prototypes with different fidelity levels are built for Robot Control Interface for a Warehouse Mobile Robot



Prototyping

Example: Low-Fi Prototype

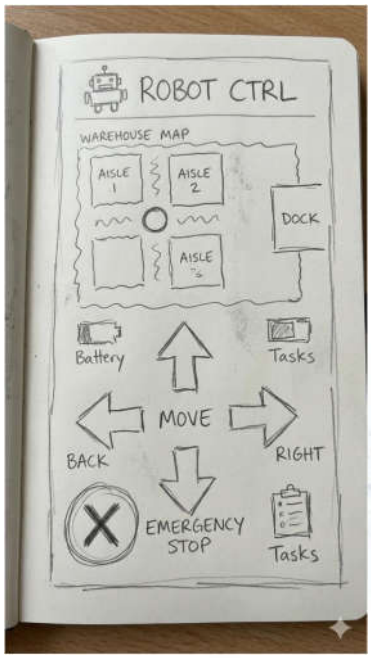
- **Paper sketch** of robot dashboard
- Hand-drawn layout:
  - Robot location
  - Battery level
  - Speed indicator
  - Large emergency stop button



Prototyping

Example: Low-Fi Prototype

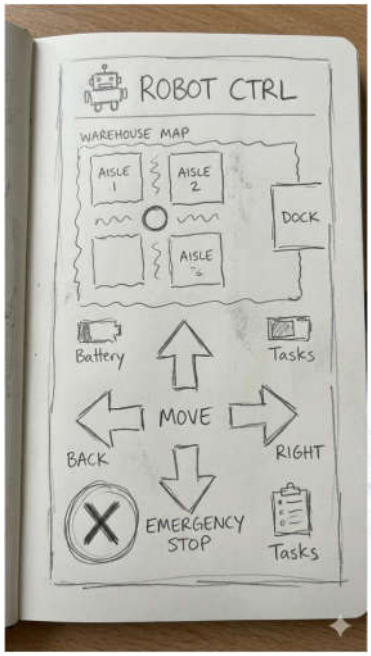
- **How to Test**
  - Role-play interaction
  - Instructor acts as “robot system”
  - User points at buttons to simulate actions
  - Ask user to describe:
    - Where they look first
    - How they would stop the robot



Prototyping

Example: Low-Fi Prototype

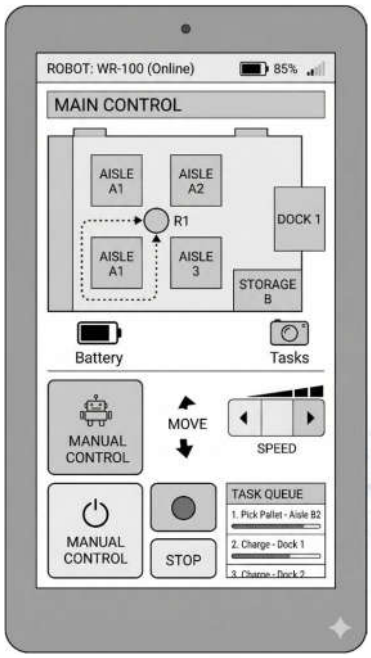
- **Focus at**
  - Is the emergency button easy to find?
  - Is the information layout logical?
  - Does the workflow make sense?
- **Don't focus at**
  - Colors
  - Animation
  - Real performance



Prototyping

Example: Mid-Fi Prototype

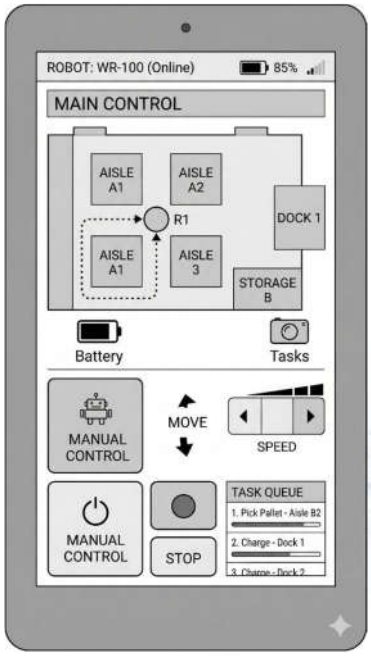
- **Clickable wireframe** (e.g., Figma)
- Basic interactive layout
- Simulated alerts



Prototyping

Example: Mid-Fi Prototype

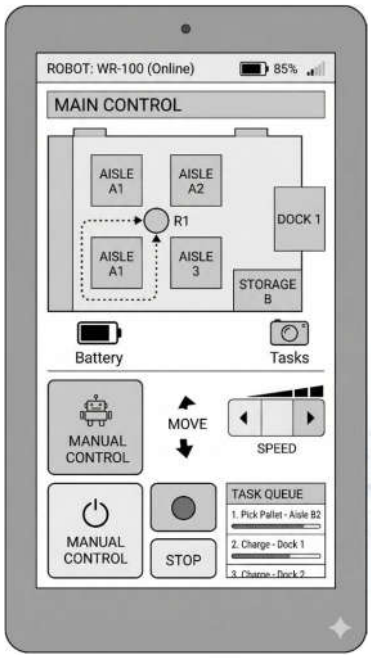
- **How to Test**
  - Give **scenario**:
    - “Robot detects obstacle”
    - “Battery low warning”
  - **Measure**:
    - Time to find emergency stop
    - Navigation path
    - User confusion



Prototyping

Example: Mid-Fi Prototype

- **Focus at**
  - Is navigation intuitive?
  - Are alerts understandable?
  - Is information grouped correctly?
- **Don’t focus at**
  - Final visual polish
  - Hardware latency
  - Real robot behavior



Prototyping

Example: Hi-Fi Prototype

- **Fully functional robot interface**
- Connected to simulated or real robot
- Real-time alerts and motion

Prototyping

Example: Hi-Fi Prototype

- **How to Test**
  - Run real-time scenario under time pressure:
    - Robot moving
    - Sudden obstacle
    - Alarm triggered
  - Measure:
    - Reaction time
    - Error rate
    - Mental workload (NASA-TLX)

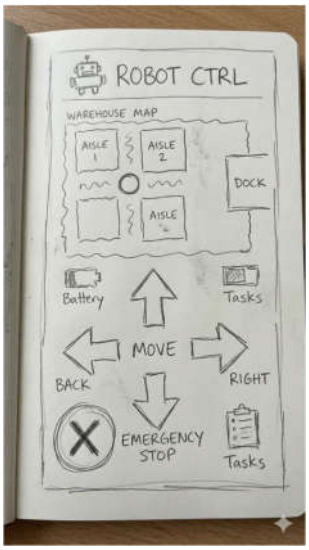
Prototyping

Example: Hi-Fi Prototype

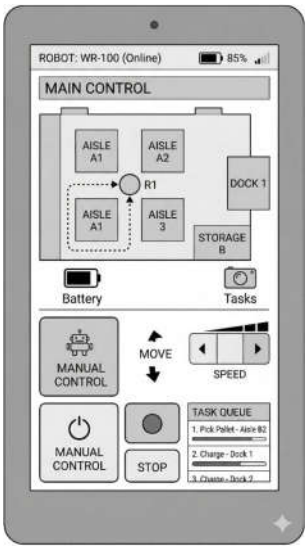
- **Focus at**
  - Can user act under stress?
  - Are there accidental slips?
  - Is workload manageable?
  - Does system support situation awareness?
- **Now we test:**
  - Real performance
  - Stress response
  - Error recovery

Prototyping

Example



Low-Fi Prototype



Mid-Fi Prototype

Hi-Fi Prototype

Prototyping

- **Common Mistakes**
  - Making prototypes too complex too early
  - Skipping user testing
  - Ignoring user feedback
  - Treating the prototype as the final product
    - A prototype is a learning tool

Prototyping

Output

- **Prototype Artifacts**
  - Early versions of the interface
- **Identified Design Issues Problems**
  - E.g. confusing layout, unclear feedback, inefficient task flow
- **Usability Feedback**
  - Insights from users
- **Improved Design Concepts**
  - Refined interface design based on feedback and testing

Usability Testing

- **Evaluating** a system by **observing** real users **performing tasks**
- Users attempt realistic tasks while researchers observe
- **Goal:**
  - identify usability problems
  - measure ease of use
  - understand user behavior
  - improve system design

Usability

- Formal **scientific measure** of effectiveness, efficiency, and satisfaction (ISO 9241-11)
  - More than Beyond "User Friendly“
- It is **context-dependent,** not depend only on the interface
  - **Contextual Triangle**
    - **User:** Skills, experience, fatigue, expectations
    - **Task:** Goal, urgency, complexity, consequences of error
    - **Environment:** Noise, lighting, time pressure, distractions, mobility

Usability Testing

Usability

- Usability as an **Emergent Property**
  - It emerges from the interaction between the user and the system
- The system should adapt to humans, not force humans to adapt to the system
- Respect Human Limits
  - A usable system should consider human capabilities

Usability Testing: Usability

Characteristics

- **Ease of Use**
  - How easily a **typical user** can operate the system correctly **without training**.
  - Reduces hesitation, errors, and training time
    - Clear labels instead of codes
    - Obvious primary action
    - Minimal steps for a task
    - Consistent behavior
- **Learnability**
  - How quickly a first-time **user** can complete basic tasks and understand how the system works
  - **Predictability:** Good designs make actions obvious: expectation
  - **Affordances:** Less need for manuals or help menus.

Usability Testing: Usability

Characteristics

- **Efficiency**
  - How fast and accurately an user can complete tasks **after learning**
  - **Expert paradox**
    - Low learnability **VS** High efficiency
    - Power users perform complex actions with minimal steps
- **Error Rate and Recovery**
  - Reduce
    - **Slips**: accidental action
    - **Mistakes**: wrong decisions
  - Provide
    - **Recovery:** Undo / cancel actions
    - **Prevention:** Forcing functions to block dangerous actions

Usability Testing: Usability

Characteristics

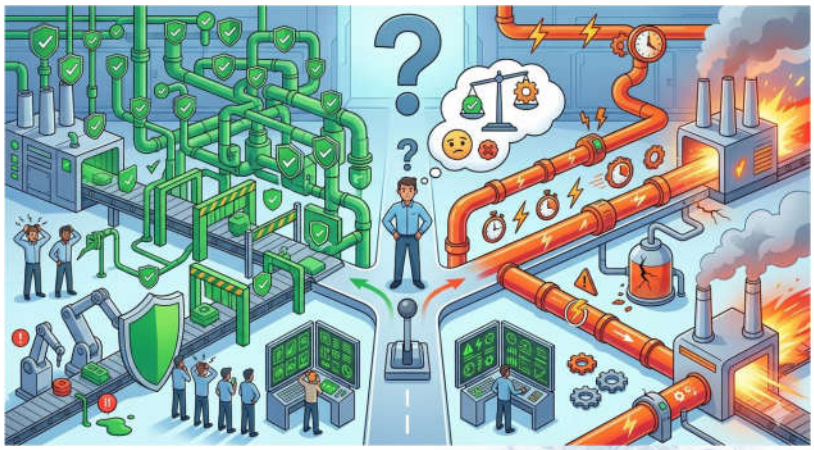
- **User Satisfaction**
  - Emotional response: sense of control, frustration, and overall enjoyment
  - **Aesthetics matter**
    - Clean and attractive designs feels easier to use
    - Users tolerate small issues more when the interface looks professional

Trade-off

- **Safety vs. Speed**
  - **More Safety**
    Adds friction → slower work
    - e.g., confirmations
  - **More Speed**
    Removes checks → higher error risk
- **Simplicity vs. Power**
  - **Beginner-friendly designs** → improve learnability
  - **Expert-focused designs** → increase capability
  - Provide Basic and Advanced modes

Discussion

- Should **safety-critical systems** always prioritize **safety** over **efficiency**?
- Can **too much safety** also create **problems**?



Measure

- **Quantitative**
  Performance Data
  - Task success rate
  - Task completion time
  - Help usage frequency
  - Number of errors
- **Qualitative**
  Experience Feedback
  - Think-aloud protocol
  - Interviews
  - Observation

Quantitative Measure

- **Task Completion Rate**
  - Percentage of users who finish a task without help
- **Time-Based Efficiency**
  - **Time on task**: how long a task takes
  - **Time to learn**: improvement across repeated trials

Usability Testing

Quantitative Measure

- **Types of errors**
  - **Slip**: wrong action (pressed wrong button)
  - **Mistake**: wrong decision (misunderstood system)
  - **Critical error**: task failure or unsafe state
  - **Recoverable error**: user corrected it
- **Error frequency**
  - Measure how often errors occur:
    - Errors per task
    - Errors per user
    - Errors per 100 interactions (target on a function)

Usability Testing

Quantitative Measure

- **Visual attention**
  - Eye-tracking
    - **Heatmaps** where users looked
    - **Gaze plots** order of attention Reveals missed warnings or confusing layouts
- **Biometric stress markers**
  - Measure real-time workload:
    - Heart rate variability
    - Pupil dilation
    - Skin conductance

Quantitative Measure

- **System Usability Scale (SUS)**
  - 10 questions
    → score from 0 to 100
  - Quick overall usability benchmark
- **Task Load Index (NASA-TLX)**
  - Measures workload across 6 factors:
    - Mental demand
    - Physical demand
    - Time pressure
    - Performance
    - Effort
    - Frustration

Quantitative Measure

- **Limitation** on Pure Numbers
  - **"Average" fallacy**
    - An average score can hide extremes
      - half love it, half hate it → average looks fine
  - **Context gap**
    - Real users are distracted, tired, multitasking
      - Lab results ≠ real use

Usability Testing

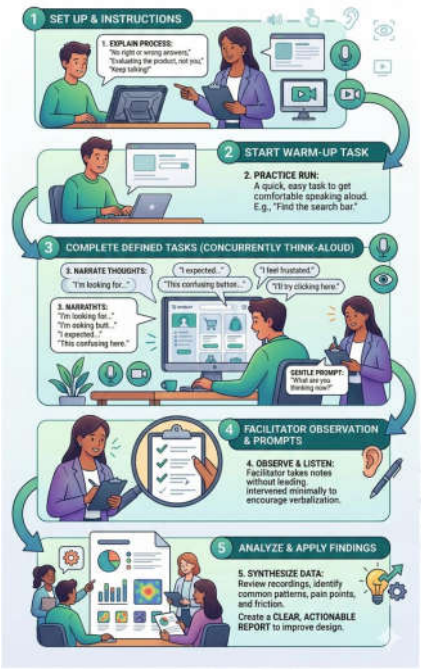
Qualitative Measure

- **Observation**
  - Watch users perform tasks and note hesitation, confusion, and workarounds
- **Interviews**
  - Ask users what they expected vs what actually happened

Usability Testing

Qualitative Measure

- **Think-Aloud Protocol**
  - Users explain what they are thinking, expecting, or trying to do.
  - Help researchers understand user reasoning during interaction.
  - Example:
    - “I’m clicking this button because I think it will open the settings.”



Ethical Considerations

- **Informed consent & safety**
  - Explain risks (e.g., VR motion sickness)
  - Participants can stop anytime without penalty
- **Protect sensitive data**
  - Anonymize recordings and logs
  - Store data securely, especially in medical or enterprise systems

Common Mistakes

- **Leading the witness**
  - Giving hints when struggle → hides real design problems
- **Testing too late**
  - Only testing after full implementation → fixes become costly or impossible
- **Clear conclusion**
  - ❌ "Users cannot effectively find 'Emergency Stop' "
  - ✅ "90% of users find 'Emergency Stop' less than 2s"

Usability Testing

Output

- **Usability Problems**
  - Issues users encounter while performing tasks.
    - E.g. confusing navigation, unclear icons, difficult task steps
- **User Feedback**
  - Qualitative insights from users
- **Performance Metrics**
  - Quantitative measures of usability
    - E.g. task completion rate/time
- **Design Improvement Recommendations**
  - Suggestions to improve the system based on test results

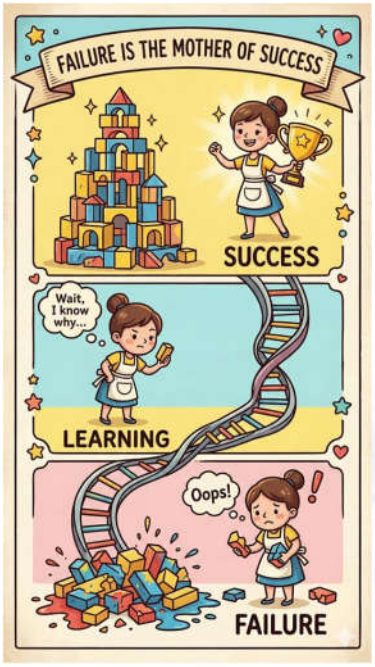
Iterative Design Loop

- **Continuous Improvement**
  - Design a version
    → Test with users
    → Analyze failures
    → Redesign
- Design is never "finished," only "released"

Continuous Improvement

Learning from Failure

- A **"failed" user test** is **a success for the designer**
  - It prevents a failed product
- **Don't just fix the symptom**;
  fix the underlying mental model error



Continuous Improvement

Feedback Integration

- **Prioritize**
  - You can't fix everything
  - Focus on "Must-haves" and safety issues first
- **Neutrality**
  - Filter out user "opinions"
  - Focus on user "performance"

Continuous Improvement

Iteration Strategy

- **Parallel Prototyping**
  - Designing different versions and testing them **against each other** at the same time
- **Evolutionary**
  - Gradually adding detail to the same concept.

System Development

- **Fix problems as early as possible**
  - **1:10:100 Rule**
    - A change costs $1 in requirements
    - A change costs $10 in prototyping
    - A change costs $100 after release

System Development

- **User-centered design** tries to:
  - discover problems early
  - avoid costly changes later
  - ensure the system meets user needs
- **System development** typically starts after successful usability evaluation of the prototype