chapter 6

HCI in the software process

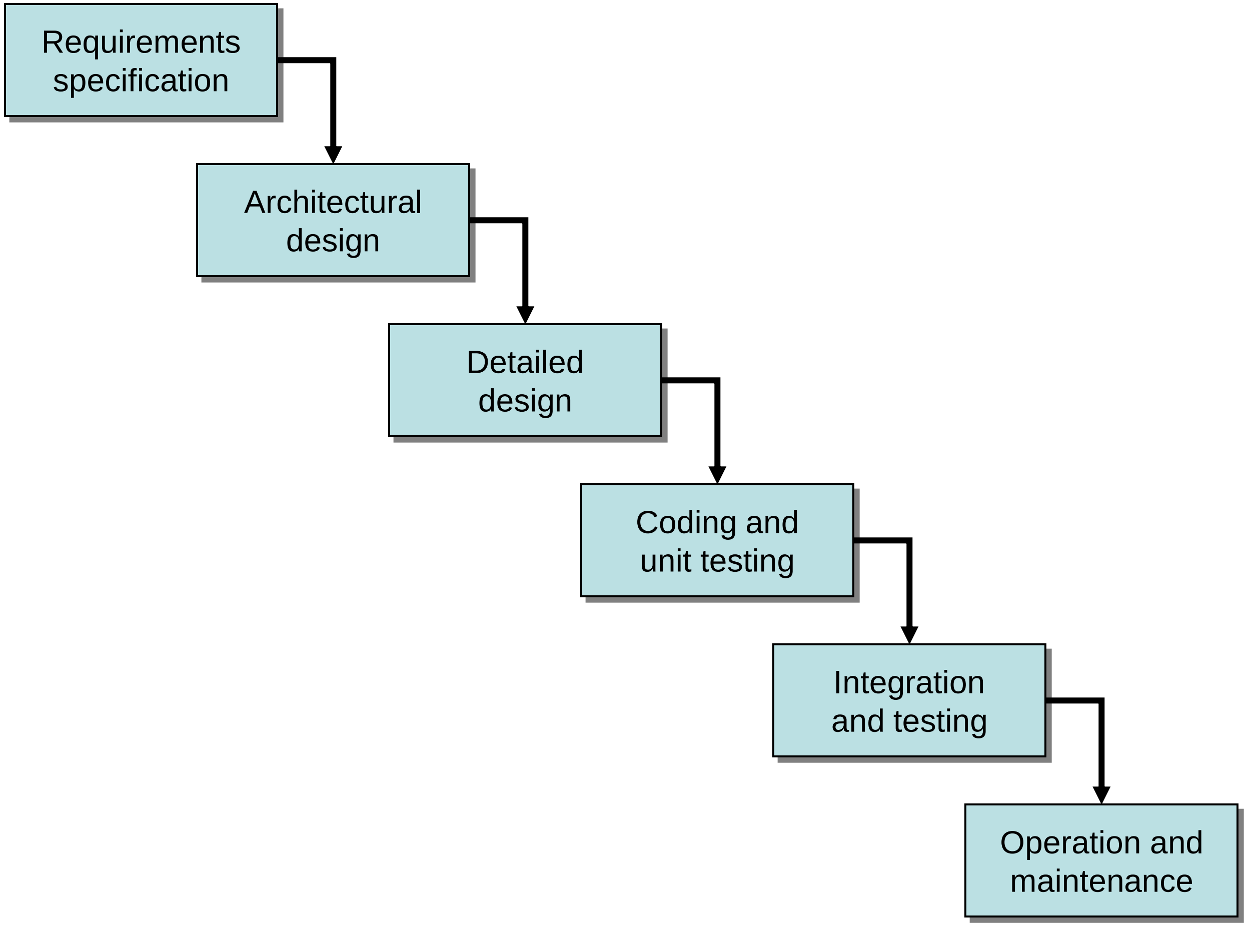
HCI in the software process

- Software engineering and the design process for interactive systems
- Usability engineering
- Iterative design and prototyping
- Design rationale

the software lifecycle

- Software engineering is the discipline for understanding the software design process, or life cycle
- Designing for usability occurs at all stages of the life cycle, not as a single isolated activity

The waterfall model



Activities in the life cycle

Requirements specification

designer and customer try capture what the system is expected to provide can be expressed in natural language or more precise languages, such as a task analysis would provide

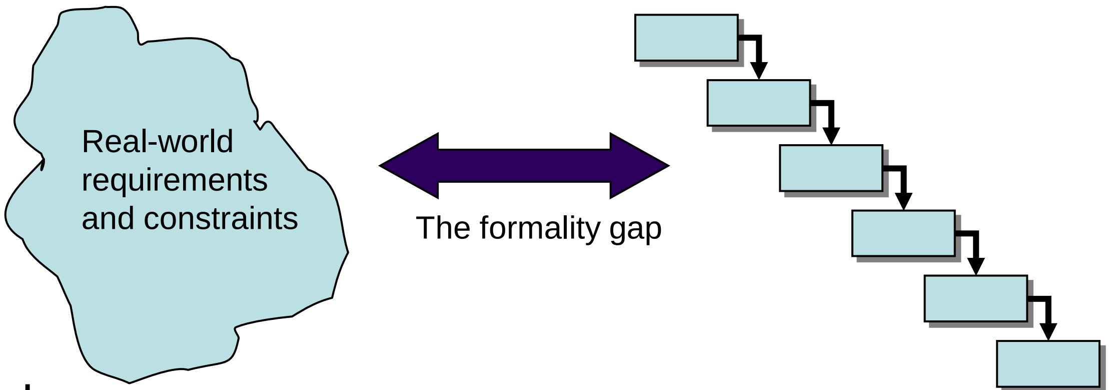
Architectural design

high-level description of how the system will provide the services required factor system into major components of the system and how they are interrelated needs to satisfy both functional and nonfunctional requirements

Detailed design

refinement of architectural components and interrelations to identify modules to be implemented separately the refinement is governed by the nonfunctional requirements

Verification and validation



Verification

designing the product right

Validation

designing the right product

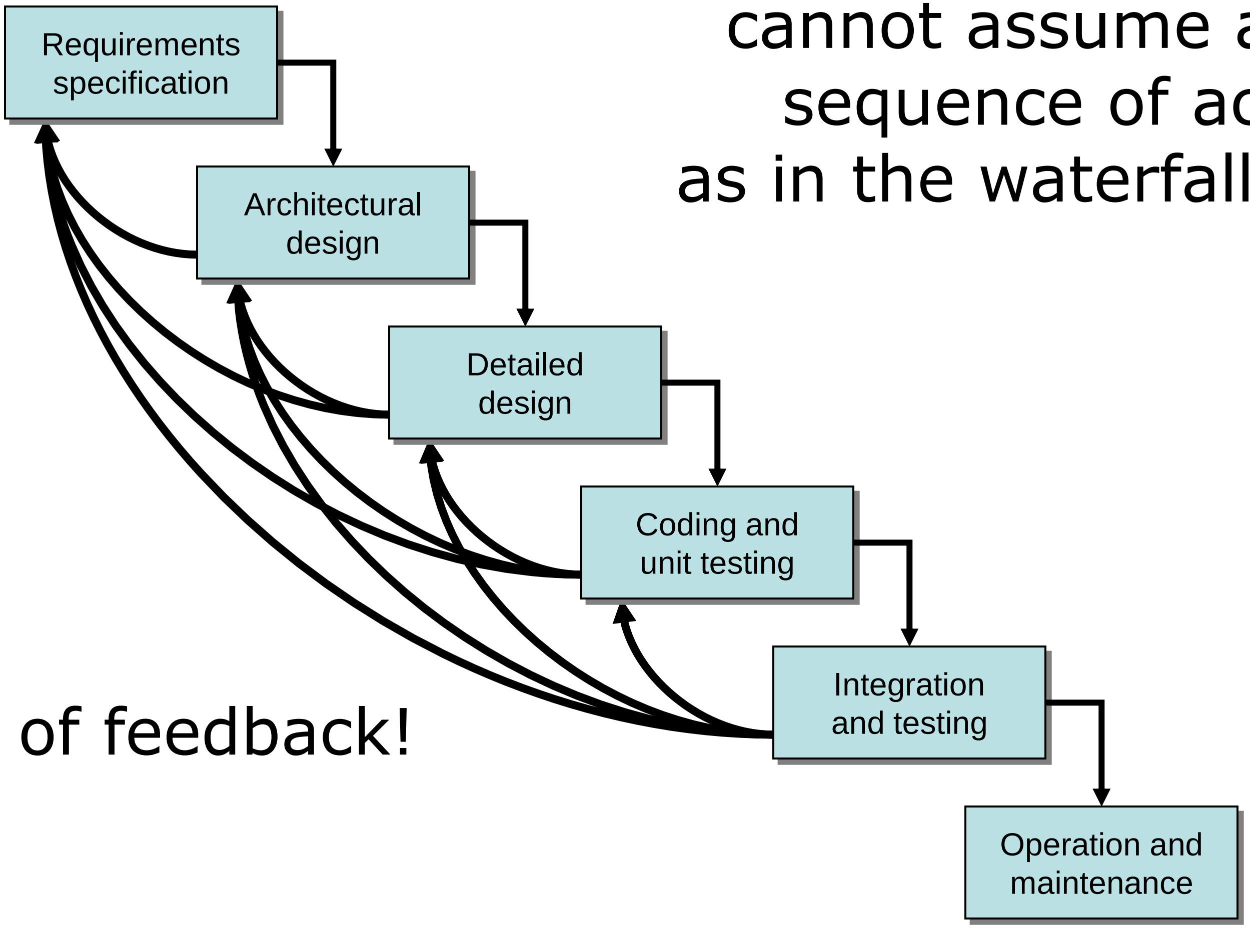
The formality gap

validation will always rely to some extent on subjective means of proof

Management and contractual issues

design in commercial and legal contexts

The life cycle for interactive systems



cannot assume a linear sequence of activities as in the waterfall model

lots of feedback!

Usability engineering

The ultimate test of usability based on measurement of user experience

Usability engineering demands that specific usability measures be made explicit as requirements

Usability specification

- usability attribute/principle
- measuring concept
- measuring method
- now level/ worst case/ planned level/ best case

Problems

- usability specification requires level of detail that may not be
- possible early in design satisfying a usability specification
- does not necessarily satisfy usability

part of a usability specification for a VCR

Attribute:	Backward recoverability
Measuring concept:	Undo an erroneous programming sequence
Measuring method:	Number of explicit user actions to undo current program
Now level:	No current product allows such an undo
Worst case:	As many actions as it takes to program-in mistake
Planned level:	A maximum of two explicit user actions
Best case:	One explicit cancel action

ISO usability standard 9241

adopts traditional usability categories:

- effectiveness
  - can you achieve what you want to?
- efficiency
  - can you do it without wasting effort?
- satisfaction
  - do you enjoy the process?

some metrics from ISO 9241

Usability objective	Effectiveness measures	Efficiency measures	Satisfaction measures
Suitability for the task	Percentage of goals achieved	Time to complete a task	Rating scale for satisfaction
Appropriate for trained users	Number of power features used	Relative efficiency compared with an expert user	Rating scale for satisfaction with power features
Learnability	Percentage of functions learned	Time to learn criterion	Rating scale for ease of learning
Error tolerance	Percentage of errors corrected successfully	Time spent on correcting errors	Rating scale for error handling

Iterative design and prototyping

- Iterative design overcomes inherent problems of incomplete requirements

- Prototypes
  - simulate or animate some features of intended system
  - different types of prototypes
    - throw-away
    - incremental
    - evolutionary

- Management issues
  - time
  - planning
  - non-functional features
  - contracts

Techniques for prototyping

Storyboards
- need not be computer-based
- can be animated

Limited functionality simulations
- some part of system functionality provided by designers
- tools like HyperCard are common for these
- Wizard of Oz technique

Warning about iterative design
- design inertia – early bad decisions stay bad
- diagnosing real usability problems in prototypes….
  - …. and not just the symptoms

Design rationale

Design rationale is information that explains why a computer system is the way it is.

Benefits of design rationale

- communication throughout life cycle
- reuse of design knowledge across products
- enforces design discipline
- presents arguments for design trade-offs
- organizes potentially large design space
- capturing contextual information

Design rationale (cont'd)

Types of DR:

- Process-oriented
  - preserves order of deliberation and decision-making
- Structure-oriented
  - emphasizes post hoc structuring of considered design alternatives

- Two examples:
  - Issue-based information system (IBIS)
  - Design space analysis

Issue-based information system (IBIS)

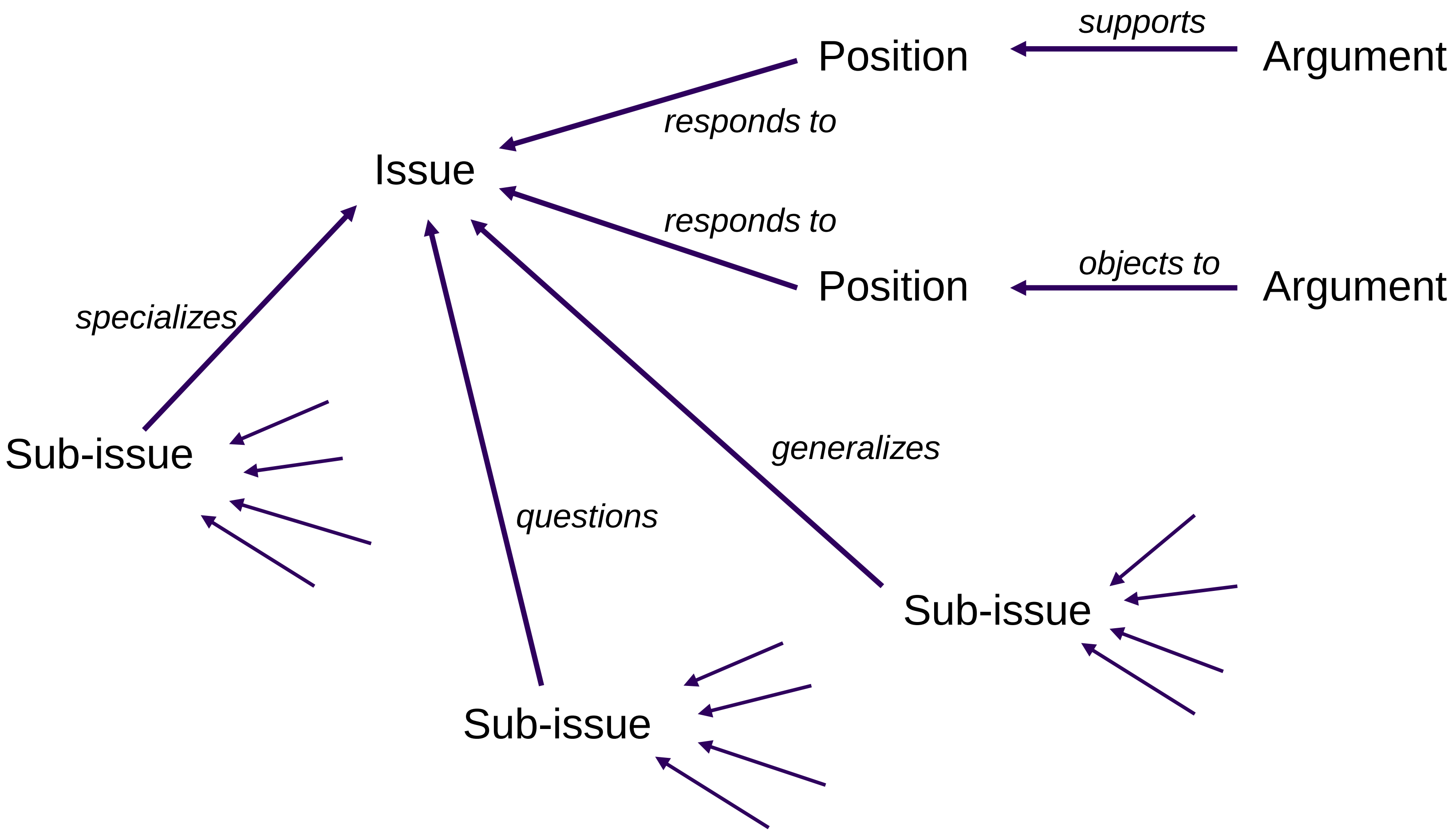
- basis for much of design rationale research
- process-oriented
- main elements:

  issues
    - hierarchical structure with one ‘root’ issue

  positions
    - potential resolutions of an issue

  arguments
    - modify the relationship between positions and issues
- gIBIS is a graphical version

structure of gIBIS



Design space analysis

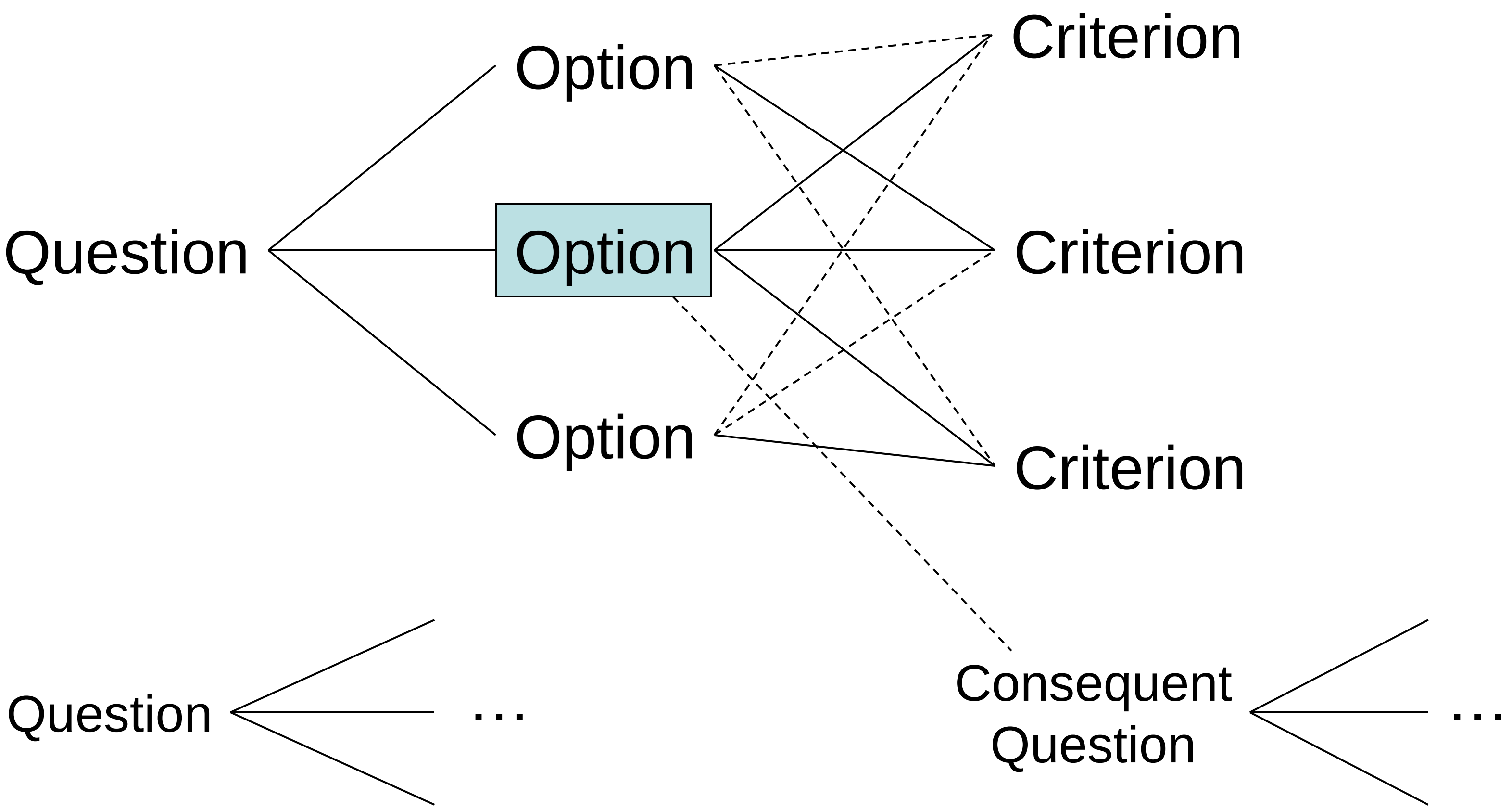
- structure-oriented
- QOC – hierarchical structure:

  questions (and sub-questions)
  - represent major issues of a design

  options
  - provide alternative solutions to the question

  criteria
  - the means to assess the options in order to make a choice
- DRL – similar to QOC with a larger language and more formal semantics

the QOC notation



Psychological design rationale

- to support task-artefact cycle in which user tasks are affected by the systems they use
- aims to make explicit consequences of design for users
- designers identify tasks system will support
- scenarios are suggested to test task
- users are observed on system
- psychological claims of system made explicit
- negative aspects of design can be used to improve next iteration of design

Summary

The software engineering life cycle

- distinct activities and the consequences for interactive system design

Usability engineering

- making usability measurements explicit as requirements

Iterative design and prototyping

- limited functionality simulations and animations

Design rationale

- recording design knowledge
- process vs. structure

Classroom Problem Statement

Title: ***Improving the Usability of UNIPLEX: A Human-Computer Interaction Design Challenge***

Scenario:
You are part of a student software development team at MIST tasked with redesigning the user interface of **UNIPLEX**, the academic portal currently used for course registration, result viewing, and administrative communication.

Many students have expressed difficulties using UNIPLEX, including:
•Confusing navigation and layout
•Lack of feedback after actions (e.g., course registration)
•Inefficient steps to complete simple tasks
•Low mobile usability
•Aesthetic and consistency issues

Your task is to **analyze the current design problems** and propose **usability-centered improvements** using principles of **HCI and usability engineering**.

Your Tasks

Part A: Analysis and Usability Engineering (30 minutes)

1. Identify **three major usability problems** in UNIPLEX based on your experience or user observations (or assumptions, if you haven't used it).
2. For each problem, define:
   - **Usability attribute** (effectiveness, efficiency, satisfaction)
   - **Measuring concept and method** (e.g., time to complete a task, number of clicks)
   - **Current and target levels** (worst case, planned, best case usability)

Part B: Iterative Design & Prototyping (30 minutes)

3. Choose **one** of the usability problems and:
   - Describe an **iterative design** approach to address it
   - Choose a **prototyping technique** (e.g., storyboard, throw-away, Wizard of Oz)
   - Sketch or outline your prototype (rough sketches or verbal description accepted)

Part C: Design Rationale (30 minutes)

4. Use **QOC (Questions, Options, Criteria)** or **IBIS** to structure your design rationale:
   - Formulate one design **question** related to your prototype (e.g., "How should course registration feedback be presented?")
   - List at least two **options** and compare them using relevant **criteria** (e.g., clarity, speed, learnability)
   - Briefly justify your chosen option with an argument

Deliverables:

•A short write-up (1–2 pages or slides) including:

•Identified problems and usability specs

•Description/sketch of prototype idea

•Structured design rationale (QOC or IBIS)

Evaluation Criteria:

•Depth and clarity of usability analysis (30%)

•Relevance and creativity of prototype solution (30%)

•Logical and structured design rationale (30%)

•Presentation and communication (10%)

Case Study: Smart UNIPLEX - AI Chatbot Integration for Academic Services

MIST is planning to upgrade **UNIPLEX**, its student academic portal, by integrating an **AI-powered virtual assistant** (similar to ChatGPT). The assistant will help students with routine academic tasks such as course registration, checking exam schedules, and answering policy-related questions.

During initial testing, students gave mixed feedback:

•Some appreciated the 24/7 availability.

•Others found it **confusing**, **too robotic**, or **not intuitive**, especially when accessing core services.

•Concerns were raised about **trust**, **transparency**, and **lack of feedback** when tasks are completed.

The development team is tasked with ensuring the system is **usable**, **engaging**, and **effective** for a diverse group of students with varying levels of digital literacy.

Your Task:

As an HCI consultant, analyze the above situation and provide a structured response that addresses the following:

•Identify and discuss **three usability challenges** specific to AI chatbot integration in academic systems like UNIPLEX.

•Propose how you would apply **iterative design and prototyping** to improve the chatbot's interface and experience. Mention a **prototyping technique** suitable for early testing.

•Explain how **design rationale** can be used to capture and justify important design decisions (e.g., whether the chatbot should display visual confirmations for completed actions).

•Suggest **measurable usability criteria** (based on ISO 9241) that the team should use to evaluate the chatbot before full deployment.

Observing users

Chapter 12

Observation

- Why? Get information on..
  - Context, technology, interaction
- Where?
  - Controlled environments
  - In the field (where the product is used)
- Observer:
  - outsider
  - participant
  - ethnographers

Frameworks to guide observation

- \- *The person*. Who?
  \- *The place*. Where?
  \- *The thing*. What?

- The Goetz and LeCompte (1984) framework:
  \- *Who* is present?
  \- What is their role?
  \- *What* is happening?
  \- *When* does the activity occur?
  \- *Where* is it happening?
  \- *Why* is it happening?
  \- *How* is the activity organized?
- Checklist can also help (p. 369).

Data collection

- Notes:
  - not technical, writing speed may be a factor, hard to observe and write at the same time, laptop is faster but intrusive and cumbersome, two people work better than one.
- Still camera:
  - images are easily collected, allows evaluators to be mobile.

Data collection cont.

- Audio:
  - less intrusive than video, allows evaluators to be mobile, inexpensive, lack of visual records, hard to transcribe data.
- Video:
  - both visual and audio data, can be intrusive, can be inexpensive with small cameras, can allow evaluators to be mobile, attention is focused on what is seen through the lens, analysis can be time consuming.

Data collection cont.

- Interaction logging (transcripts & replay):
  - logs everything you do in the system, easy to generate detailed analysis, transparent to the user, facial expression etc. is not logged.

- Techniques may be used individually or combined => requires coordination.

Data analysis

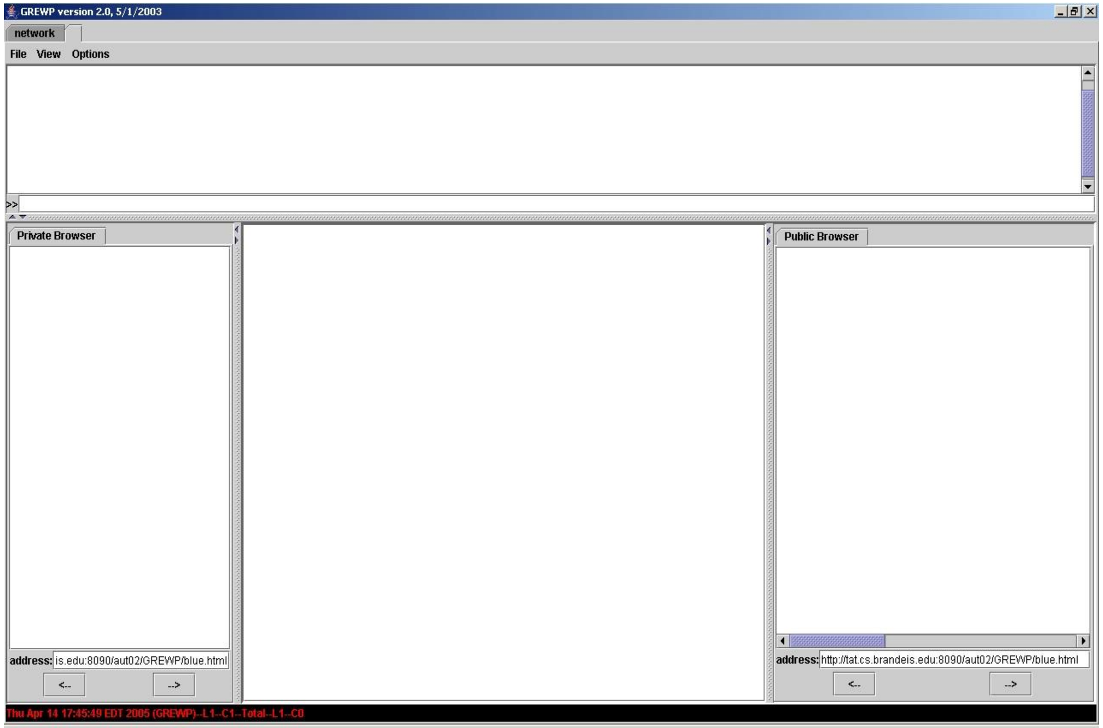
- *Qualitative data* - *interpreted* & used to tell the 'story' about what was observed.

- *Qualitative data* - *categorized* using techniques such as content analysis.

- *Quantitative data* - collected from interaction & video logs. Presented as values, tables, charts, graphs and treated statistically.

CS111 Experiment

- Create a presentation of a world country and its culture.
- ***GREWP tool provides users with:***
  - ***a shared workspace online,***
  - ***chat to communicate,***
  - ***public and private browsers***
  - ***Generates transcripts for replay***

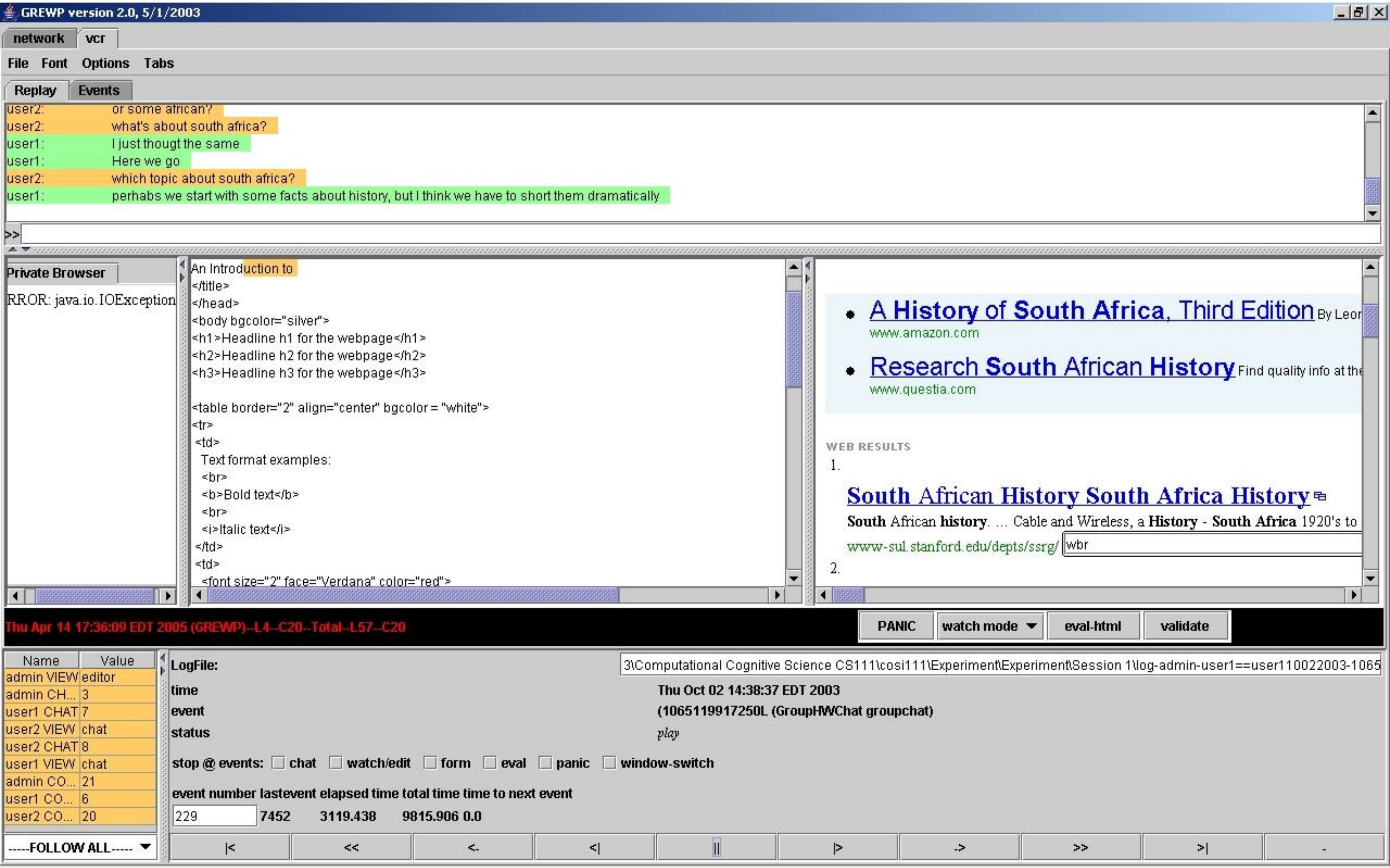


Observing the users

- Where was the study performed:
  - Controlled environment
    - We had everything set up before participants arrived
    - Tested the software
    - etc.
- Data collection:
  - Note taking: Important issues noted on paper and coordinated with transcripts later.
  - Replayed transcripts that the tool generated.

Analyzing the data

- Coordinated notes with transcripts
- Replayed the transcripts
  - Qualitative data (Categorization)
    - Looking for incidents or patterns.
    - How was a certain task completed?
    - How did the users use a certain component in the system?
    - One user frequently got stuck in the HTML coding. Why is that?
    - Analyzing the discourse (Alex Feinman)


GREWP version 2.0, 5/1/2003
network
vcr
File Font Options Tabs
Replay
Events
user2: or some african?
user2: what's about south africa?
user1: I just thougt the same
user1: Here we go
user2: which topic about south africa?
user1: perhabs we start with some facts about history, but I think we have to short them dramatically
>>
Private Browser
RROR: java.io.IOException
An Introduction to
</title>
</head>
<body bgcolor="silver">
<h1>Headline h1 for the webpage</h1>
<h2>Headline h2 for the webpage</h2>
<h3>Headline h3 for the webpage</h3>
<table border="2" align="center" bgcolor = "white">
<tr>
<td>
Text format examples:

Bold text

<i>Italic text</i>
</td>
<td>
<font size="2" face="Verdana" color="red">
A History of South Africa, Third Edition By Leon
www.amazon.com
Research South African History Find quality info at the
www.questia.com
WEB RESULTS
1.
South African History South Africa History
South African history. ... Cable and Wireless, a History - South Africa 1920's to
www-sul.stanford.edu/depts/ssrg/
wbr
2.
Thu Apr 14 17:36:09 EDT 2005 (GREWP)--L4--C20--Total--L57--C20
PANIC
watch mode
eval-html
validate
Name
Value
admin VIEW editor
admin CH... 3
user1 CHAT 7
user2 VIEW chat
user2 CHAT 8
user1 VIEW chat
admin CO... 21
user1 CO... 6
user2 CO... 20
-----FOLLOW ALL-----
LogFile:
3\Computational Cognitive Science CS111\cosi111\Experiment\Experiment\Session 1\log-admin-user1==user110022003-1065
time
Thu Oct 02 14:38:37 EDT 2003
event
(1065119917250L (GroupHWChat groupchat)
status
play
stop @ events: chat watch/edit form eval panic window-switch
event number lastevent elapsed time total time time to next event
229 7452 3119.438 9815.906 0.0
|< << <- <| || |> -> >> >| -

Redesign

user2:	look where I'm in the screen
user2:	title is only in the head
user2:	not in the normal text
---------	
user2:	look how I to a table
user2:	you only put title only in the head

- Proposal 1:
- ***Automatically add reference to a line in the code window to the chat.***
- ***Help users stay coordinated***

Redesign

user2: how is the work?
user2: how far is your table??
----
user1: where are you now?
user1: are you finished with the food?

- Proposal 2:
- **Provide a way to write down a plan and review or modify it visually.**
- **Helps users be aware of each others work.**
- **Automatic update of the plan as work progresses.**

What is UNIPLEX (MIST EMS)?

UNIPLEX at MIST is an integrated **Education Management System** used for tasks like:

- Course registration
- Attendance tracking
- Result publishing
- Assignment submission
- Faculty-student communication
- Class routine management

How Task Analysis Applies to UNIPLEX ???

1. Task Decomposition (HTA) in UNIPLEX
2. Knowledge-Based Task Analysis
3. Entity/Object-Based Analysis

Task Decomposition (HTA) in UNIPLEX

Task: Register for courses

Hierarchical Task Analysis (HTA):

Main Goal: Register for the semester

Subtasks:

1. Log in to UNIPLEX
2. Navigate to course registration panel
3. Select courses as per credit requirement
4. Confirm and submit
5. Download or verify course list

Plan: Do 1 → 2 → 3 → 4 → 5 in that order.

This breakdown helps system designers ensure the interface supports the task sequence logically and clearly.

Knowledge-Based Task Analysis

Task: Register for courses

To successfully register or use UNIPLEX, a user (student) needs to **know**:

• Their credit requirements

• Course prerequisites

• How to navigate the UNIPLEX dashboard

• Course codes and titles

• Deadline for registration

Designing the system should consider this knowledge – e.g., providing tooltips, course descriptions, credit summaries.

Entity/Object-Based Analysis

Task: Register for courses

In UNIPLEX, key **entities (objects)** and **actions** include:

This helps map the system's data models and understand how users interact with features.

Entity/Object	**Associated Actions**
Student profile	View, update, submit
Courses	Browse, select, drop
Assignments	Upload, view feedback
Results	View, download, contest
Attendance	Check, notify instructor

Classroom Example for Discussion

Example Task: Submitting an Assignment via UNIPLEX

Asking:

•What steps do you follow?

•What knowledge is required?

•What objects and actions are involved?

Expected Answers:

•**Steps:** Login → Go to course → Find assignment → Upload → Confirm

•**Knowledge:** Deadline, file format, assignment title

•**Objects/Actions:** Assignment (upload), Course (navigate), Notification (receive)

Then, let students **draw an HTA tree** or flow diagram of this process.

Summary Table

Task Analysis Method	**UNIPLEX Example**
Task Decomposition	Registering courses, uploading assignments
Knowledge-Based	Knowing course codes, deadlines
Entity/Object-Based	Student, course, assignment, result

UNIPLEX is the official **Education Management System (EMS)** used by the **Military Institute of Science and Technology (MIST)** to manage academic activities such as course registration, result publication, attendance tracking, and communication between students, faculty, and administration.

Recently, concerns have been raised by both students and faculty regarding the usability and user experience of the system. Users report difficulties in navigating key features, system delays, and a lack of intuitive feedback during task execution (e.g., course dropping, downloading grade reports). Additionally, some components (such as result views and advisor approvals) require frequent clarification from support staff, indicating possible design or interaction inefficiencies.

To address these challenges, the UNIPLEX development team decided to conduct a structured **user observation study**, following industry-standard approaches. The objective is to evaluate how users actually interact with the system in both **controlled environments** (lab-based usability testing) and **real-world usage** (in the field), using methods and frameworks drawn from **qualitative and quantitative research**.

As part of this initiative, you are tasked with solving the following problem:

How can structured user observation and methods like interaction logging, note-taking, audio recording, and ethnographic observation be effectively applied to uncover usability challenges in the UNIPLEX system at MIST?

Your analysis should focus on:

- Selecting appropriate observation environments and techniques.
- Defining a structured observation framework.
- Collecting and analyzing qualitative and quantitative user data.
- Identifying key usability issues.
- Recommending evidence-based improvements to the system's design and interaction model.

chapter 18

modelling rich interaction

18.1 Introduction

Main idea: Traditional HCI models - such as Norman's execution - evaluation cycle - focus on discrete actions like "click a button" or "press a key." But in modern systems, interaction is:

- **Continuous** (gesture movements, real-time video editing).
- **Multimodal** (voice + gesture, touch + haptic feedback).
- **Context-aware** (adapts to location, time, or user state).

Why modelling is important:

- Breaks down complex behaviour into understandable components.
- Predicts potential problems before development.
- Supports communication between different stakeholders (designers, engineers, managers).

Example scenario:

- Designing an **AR navigation app**:
  - User points phone to street.
  - GPS + camera + orientation sensors combine input.
  - App overlays arrows in real-time.
  - Context: Sunlight glare, crowd movement, network speed — all impact interaction.

18.2.1 The nature of status–event analysis

Definition: A way to model a system as a set of **statuses** (conditions) and **events** (changes).

Why this works well:

- Fits the way humans think about cause and effect.
- Encourages designers to identify every **state** the system can be in.
- Makes **feedback gaps** visible.

Key principle: Every event should lead to a **clear and observable status change** for the user.

Design note: Missing or delayed feedback = confusion.

Example:

- Status: *Printer ready*, *Printer busy*, *Error*.
- Event: *Print command*, *Print completed*, *Paper jam detected*.

Scenario: Brian checks his watch repeatedly. Alison checks her **calendar**

Abstracted Properties

Status:

Watch → continuous status (time changes smoothly).

Calendar → discrete status (specific dates/events).

Events:

7:35 arrives (time event). Brian leaves (user action event). Watch alarm rings (system event). Alison notices birthday (perceptual event).

Polling:

Brian repeatedly checks his watch. Turning a **status** (time) into an **event** (7:35 reached).

Actual vs. Perceived:

Actual event = 7:35 reached., Perceived event = when Brian notices (7:36).

Granularity:

Watch → minutes scale.

Birthday → yearly scale.

Key Idea for HCI

Status and events interact differently depending on **continuity, perception, and granularity**. Users (and systems) may **poll, perceive, or miss** actual events.

18.2.2 Design Implications

Idea: Applications create **events for users**. The **presentation technique** must **match the timescale** of the event.

Examples

Fast timescale – Critical events:

Coolant failure in nuclear plant → needs **immediate alarm** (seconds).

Email would be too slow → catastrophic.

Slow timescale – Routine events:

Stock of 6mm bolts low → reorder within **days/weeks**.

Red flashing alarm would be **overkill**.

Everyday mismatch:

Brian's cinema reminder (short-term) → alarm at 7:35 works well.

Alison's birthday reminder (long-term) → alarm at noon is disruptive.

A calendar notification fits better.

Key Lessons for HCI

Wrong timing can be **dangerous** (too slow) or **annoying** (too fast).

Designers must **predict event timescales** and choose suitable techniques:

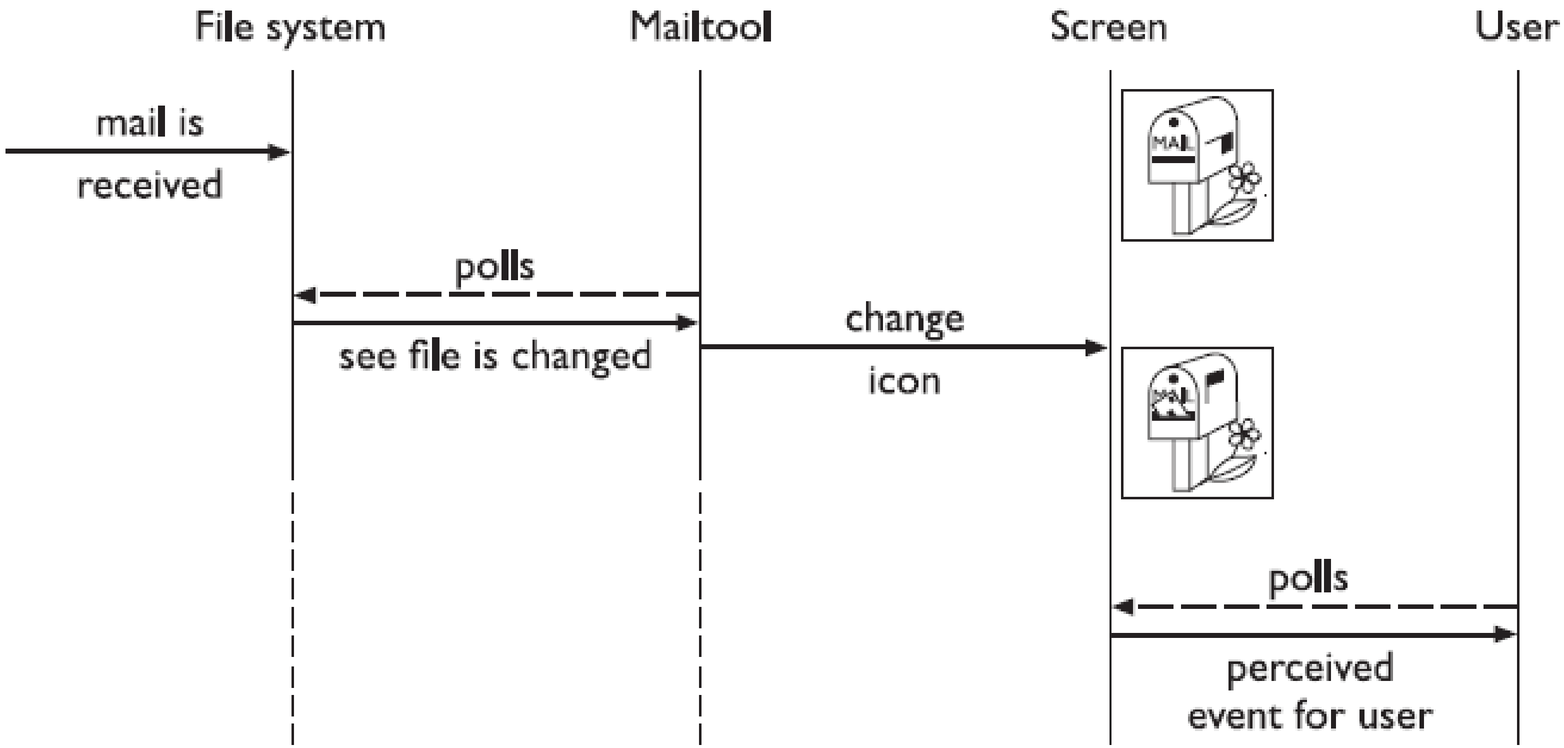
Alarms/alerts → short timescales.

Calendars/schedules → long timescales.

Simply showing info ≠ user perceives it at the right time.

18.2.3 Naïve psychology

where the user will be looking, Position of mouse, screen, text insertion point



18.2.5 Example – screen button feedback – Home Work

Problem Statement:
As an HCI analyst, you are tasked to model **rich interaction in MIST UNIPLEX** using **status–event analysis** and apply **design implications** for different timescales of events.

Tasks to do:
Identify statuses and events in UNIPLEX for at least **two scenarios**:
(a) **Assignment submission workflow**
(b) **Exam result publication**

Draw a status–event diagram for each scenario.
Example statuses: *Assignment Pending*, *Assignment Submitted*, *Deadline Passed*.
Example events: *Student uploads file*, *Deadline reached*, *Teacher releases grade*.

Analyze polling vs. event notification:
Do students need to **poll** (check repeatedly) for results/assignments?
Or does the system generate **events/alerts** (e.g., push notifications, emails)?

Discuss timing mismatches (Design implications):
What happens if notifications are **too slow** (e.g., assignment reminder comes after deadline)?
What happens if they are **too fast or disruptive** (e.g., repeated alerts for every minor update)?

Propose improvements in UNIPLEX:
Suggest **better event presentation techniques** for short-term (urgent) events vs long-term (routine) events.

Problem Statement:

Expected Outcomes

Students will demonstrate how **status–event analysis** simplifies complex workflows.
They will understand the importance of **matching notification techniques** to the **timescale of events**.
They will propose design improvements to make MIST UNIPLEX more **user-friendly and efficient**.

(a) Assignment Submission Workflow in UNIPLEX

Statuses

1. Assignment Pending – Teacher has uploaded assignment instructions, awaiting student submission.
2. Assignment Submitted – Student has uploaded file before deadline.
3. Deadline Approaching – Time is nearing submission cut-off.
4. Deadline Passed (No Submission) – Student missed submission.
5. Assignment Graded – Teacher evaluates and uploads marks/feedback.
6. Grade Released – Student can view result.

Events

- Teacher uploads assignment → Status changes from *No Assignment* → *Assignment Pending*.
- System sends reminder (X days before deadline) → Status remains *Assignment Pending*.
- Student uploads assignment → *Assignment Pending* → *Assignment Submitted*.
- Deadline reached → *Assignment Pending* → *Deadline Passed*.
- Teacher grades submission → *Assignment Submitted* → *Assignment Graded*.
- Teacher publishes grade → *Assignment Graded* → *Grade Released*.

Status–Event Diagram (Text Form)

No Assignment
↓ (Teacher uploads assignment)
Assignment Pending
↓ (Student uploads file) ----------------------→ Assignment Submitted
↓ (Deadline reached without submission) → Deadline Passed
Assignment Submitted
↓ (Teacher grades)
Assignment Graded
↓ (Grade released)
Grade Released

Polling vs. Event Notification

•Current practice in UNIPLEX: Students often have to poll (log in repeatedly) to check assignment deadlines, submission confirmation, and grading updates.
•Ideal design: Introduce event notifications (push notification, SMS/email reminders).

- Before deadline → Reminder alert.
- After submission → Confirmation alert.
- After grading → Grade release alert.

Design Implications (Timing Issues)

•Too slow: Reminder arrives after deadline → Student misses submission.
•Too fast/disruptive: Repeated alerts every few minutes → Annoying and distracting.
•Balanced approach:

- Short-term/urgent → Strong alerts (reminder 24h and 2h before deadline).
- Long-term/routine → Calendar integration (assignment release dates, grading release).

Proposed Improvements for UNIPLEX

1.Push notifications & emails for deadlines and results.

2.Submission confirmation receipt (so student knows file uploaded successfully).

3.Calendar sync with student device (Google/Outlook Calendar).

4.Progress tracker dashboard showing assignment lifecycle (Pending → Submitted → Graded).

Classroom Problem Statement: Attendance Workflow in UNIPLEX

You are asked to analyze how **attendance** is managed in UNIPLEX using **status–event analysis**.

Tasks:

1. Identify key **statuses**
2. Identify **events**
3. Draw a **status–event diagram** showing the workflow.
4. Discuss whether attendance updates should rely on **polling** or **notifications**.
5. Suggest **improvements** to make the attendance process more efficient and user-friendly.

Classroom Problem Statement: Attendance Workflow in UNIPLEX

You are asked to analyze how **attendance** is managed in UNIPLEX using **status–event analysis**.

Tasks:

1. Identify key **statuses** (e.g., *Class Scheduled, Attendance Pending, Attendance Marked, Attendance Missing*).
2. Identify **events** (e.g., *Teacher opens attendance sheet*, *Student marked present/absent*, *Attendance submitted*).
3. Draw a **status–event diagram** showing the workflow.
4. Discuss whether attendance updates should rely on **polling** (students checking manually) or **notifications** (system alerts).
5. Suggest **improvements** to make the attendance process more efficient and user-friendly.

Expected Outcome:

Students will understand how to apply status–event analysis to attendance and propose practical design enhancements.