Introduction to HCI
Fall 2021

Establishing requirements

Mahmood Jasim
UMass Amherst
mjasim@cs.umass.edu
https://people.cs.umass.edu/~mjasim/

Logistics

- Project
  - A working prototype

- Milestones
  - Investigate status quo, come up with several possible solutions, and systematically choose the best
  - Iteratively prototype your solution, gradually increasing its detail and polish
  - Evaluate your solution

Milestones (Report + Presentation)

- Milestone 1:
  - Problem statement, literature review, personas and tasks
- Milestone 2:
  - Features, Low-fidelity prototypes, high-fidelity prototypes
- Milestone 3:
  - Final prototype, evaluation design

Final Report (Report + Demo)

- Abstract
- Introduction
- Literature review
- System description
- Evaluation
- Results
- Conclusion

- Video demo

Example projects and project ideas

- Some sample projects from last years
  - https://tinyurl.com/nu2dcpze
- Some project ideas
  - https://tinyurl.com/5ak7rxps

Learning Goals

- Define and give examples of different types of requirements.
- Give examples of HCI techniques that are suitable / helpful for setting requirements.
- Be able to identify appropriate metrics for a given requirement (and outline what features good metrics have).
- Explain 3 steps for requirements generation.

What are requirements?

Two types of requirements…

- 1. Functional requirements: what the interface must do
  - usefulness -- scope, features…
- 2. Non-functional requirements: constraints that development must live in:
  - Delivery time, maximum cost, delivery platform, supportability, sustainability…
  - Usability / user experience: what it should be like to use (a primary focus of HCI design)
- ALL: clear, specific, defined in a MEASURABLE way

“Usability” focus is rarely independent

- Meeting usability requirements will/should often influence functional requirements. E.g.
  - Needed speed of response from system to user
    - Implementation platform
  - Desired user context – e.g. On-the-go
    - Must work on mobile platform
  - Use in distracted environment – e.g. Hospital tool
    - Voice output, speech input
    - Visual display requirements to support memory chunking

Functional vs. Usability requirements:

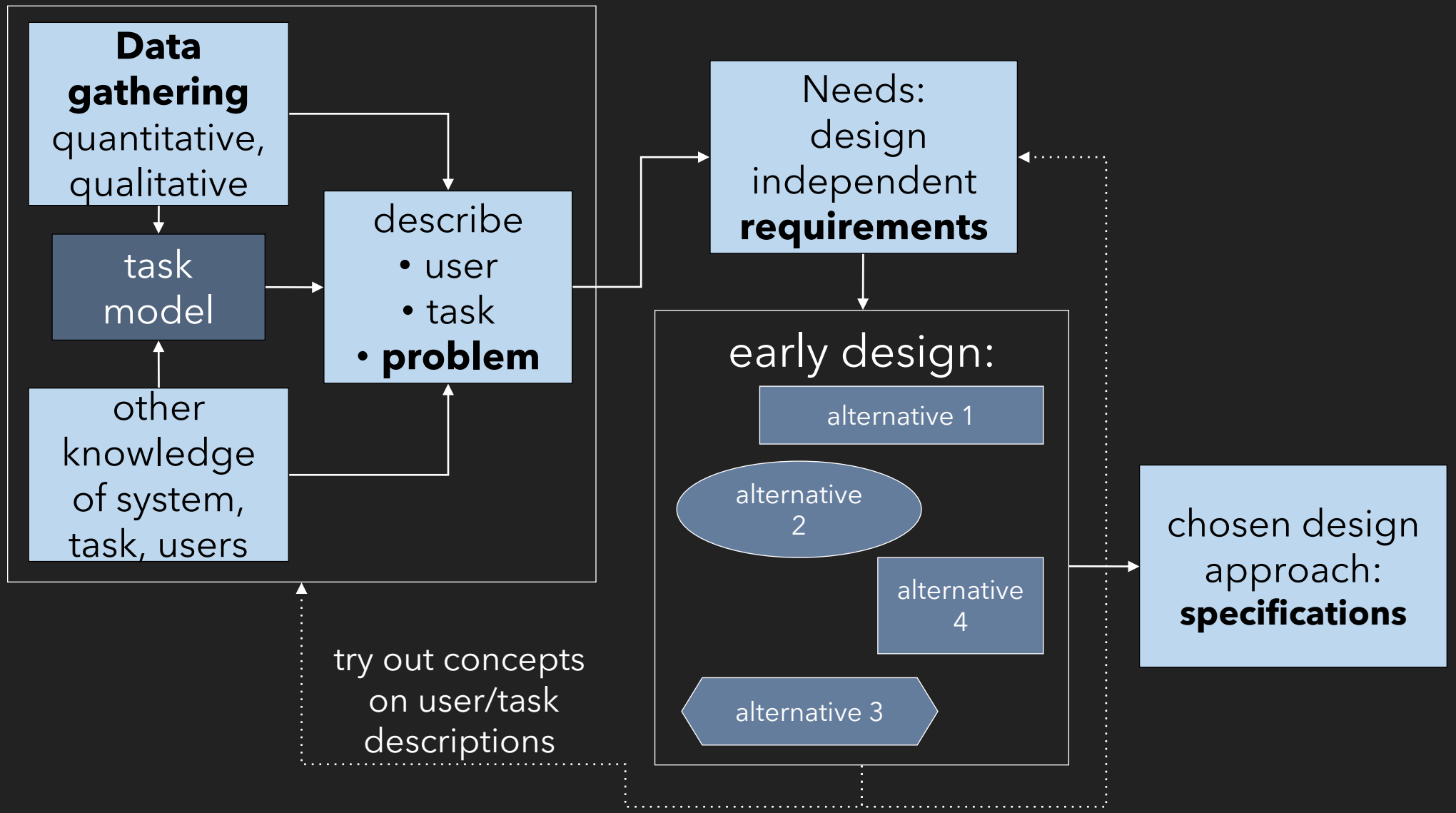
- The following list of requirements includes functional, usability and user experience requirements that could be defined for a movie theatre website. Which requirement is the best example of a functional requirement?
- Users must be able to buy electronic tickets in less than 3 distinct steps.
- Users must feel that the buying of electronic tickets is easy.
- Users must be able to buy electronic tickets using the website.
- Users must be able to learn how to buy electronic tickets on their first attempt.

Functional: usefulness not usability
specifies a feature or capability

Can requirements change?

- Requirements should be stable if based on good data
- But not rigid
  They may shift over time, in particular as design reality dictates what is possible / feasible given other constraints.

Requirements establishing primarily during pre- and early design



Three steps to requirements

- Identify the human activity
  which the proposed interactive system will support:
  task, goals, conditions; current problems and strengths
- Identify all the users & other stakeholders
  who perform or will perform the activity:
  groups, capabilities, motives, needs
- Set focus and levels of support
  which the system will provide (the system's usability):
  constraints on the product's performance, to support specific user-stakeholders you have targeted.

Identify the human activity

- What outputs might you have at the end of this?
  - Goals
  - Task descriptions, task examples
  - Task models; normal steps and process; common breakdowns

Example: scheduling meetings

- Informal problem definition:
  - Hard to learn everyone's schedule & find a common free time
  - Participants respond slowly or incompletely to request
  - Complicated to respond in adequate detail
  - Individual schedules change → time no longer available
  - Shared calendars: privacy and system incompatibility
- Result: too much iteration; non-convergent
- Course of action:
  - Ideas?

Online scheduler is one obvious one which has taken hold; others?

Scheduling example

Some possible task goals:

- Identify who needs to be @ meeting
- Find common empty spaces in calendars
- Identify a subset of empty spaces to suggest
- Choose one » tell everyone
- Receive confirmation that everyone still avail
- If no, iterate
- Identify location

NEXT, break one of these down (many possible ways)

- Find common empty spaces in calendars:
  - Ask all to communicate avail during a block; OR suggest times, get responses
  - Examine, manually or automatically
  - Find common openings, if any
  - If no, iterate with different time blocks or suggestions

How might this go wrong?

- What if people respond very slowly?
- What if people respond incompletely?
- What if there are no solutions?

Dependencies among tasks: Task objects

- "Task objects" are resources required by tasks
  - Artifacts (files, lists, databases)
  - People (special expertise, authority, or knowledge)
  - Other processes, equipment or tasks
- Low-level tasks typically focus on a single resource
  - Task cannot be accomplished without the resource
  - Once resource is available the task can be completed
- (Higher-level / composite) tasks have multiple dependencies
  - The focus shifts as different sub-tasks are performed
  - Activity is suspended when resources are not available

Signs of task dependencies?

- Joint use of task objects by different tasks
  E.g. Access to shared files or databases
- Communication between people
  may be direct (phone call) or indirect (memo)
- Synchronization
  with real-world physical and mechanical processes
- Suspension
  blocking when resources (information, people, real-world processes) are not available

Scheduling example

- Objects for Scheduling task?
- Conflicts in their use suggest dependencies…
  - Calendars
  - Communication mechanisms (email, phone, cooler)
  - "Leader" – meeting leader, secretary, program

- Other signs of task dependencies?
  - Can't find time until have heard from all participants
  - Participants can't give feedback on times until told their choices

The user

We have used Personas for this step.

- Need to understand general human needs:
  - Physical and cognitive abilities
  - Social and cultural environments
  - Use models of human behavior to test ideas
- Need to understand specific human needs:
  - Individuals have different skills and requirements
  - They have responsibilities and authority in organizations
  - They are expected to have a certain level of training
  - They have specific access to tools and resources

Do not assume the user is “like you”, or “normal”

Scheduling example

- The USER
- Examples of general needs:
- Social / cultural environments (are people more comfortable with email, telephone or just running into each other?)
- Are some users more overwhelmed with information than others, more than they can humanly process?
- Examples of specific needs:
- Do all have laptops? are some reliant on mobile devices?
- Is there variation in how responsive they are?
- Do they have control over their time – i.e. are they permitted to decide what meetings they should / should not go to?

Focus: which users will you support?

- Usually the intersection of
  - Greatest need and thus opportunity for business
  - Feasibility
  - For example: you might choose NOT to support student users who do not have mobile devices

How does this relate do the different types of Personas?

Metrics: how do we know if we have succeeded?

- Quantitative metrics: measured (countable) indicators of people's use of the interface:
  - Speed of performance, Incidence of errors, Ease of learning the system, User satisfaction
- Qualitative metrics: descriptive accounts that shed light on quantitative measures:
  - E.G. Stories about user impressions and frustrations, and their change pre- and post design

Choosing usability targets

- Choice of usability metrics affects the solution:
  - Prioritize most important facets based on design goals:
    E.g. Is speed most important, or is it very bad to make errors?
  - Ease of learning can be important, especially for novices
- Levels of performance need to be quantified:
  - Must know baseline performance first (pre-redesign)
  - Then establish realistic target levels
  - Make sure we can measure the changes → iterate

Putting it all together: Stating requirements

- No single right way to write requirements; lots of companies have specific methods, tools, etc.

- One approach – list out and then prioritize each of:
  - Supported activities (tasks and steps)
    - Tasks and processes involved that support the activity.
  - User(s)
    - Who does the task and what are their characteristics?
  - Level of support
    - What usability properties are important?

Where does technology come in?

- Tools support tasks
  - New tools should improve performance of a task
  - Tools are often specific to the tasks they support
  - Tools must be acceptable / desirable to users
- Systems support processes
  - Systems have to support links between tasks
  - Often tasks are automated using technology
  - Tasks have to be supported in a consistent manner
  - Desirable to reduce dependencies
  - Desirable to reduce task complexity

One way (of many) to use your task description…help you find solutions!

In-class activity

- Work in teams
- Discuss and identify users' tasks for your project
- Keep the following question in your mind:
  - What “task objects” (i.e. resources, parts of process, even things generated by the process) might be required to meet this task?
  - Are there dependencies on these objects that could lead to conflicts or breakdowns?

Additional Information

The form of the solution (finally, the design!!):

- But, design starts by adding constraints
  - Cost (time, money, expertise)
  - Compatibility with specific hardware or software
  - Market pressures (standards, “look and feel”)
  - Many of these don’t come from the designers!
- Multiple levels in the description (and prototypes)
  - The social/cultural/physical environment
  - The user interface
  - The application software
  - The operating system
  - System resources (storage, networking, peripherals)

Other factors in choosing a solution

- Existing intellectual property
  - Technology owned or licensed by the organization
  - Unique skills or knowledge in the organization
  - Market share or reputation
- Innovation
  - Technology becomes obsolete quickly
  - R&D requires time and effort
  - Often incremental improvements are good enough
  - Significant changes may be required sometimes

Optional reading

- Interaction Design: beyond human-computer interaction, 3rd Edition
  - Please read Chapter 10: Task Description, Task Analysis.
  - https://learning.oreilly.com/library/view/interaction-design-beyond/9780470665763/