Introduction to HCI

Establishing requirements

Prof. Narges Mahyar
UMass Amherst

nmahyar@cs.umass.edu
Courses, projects, papers, and more:
http://groups.cs.umass.edu/nmahyar/

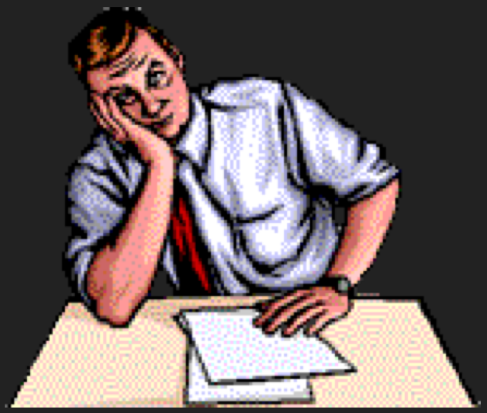
Today

- Task centered design [40 min]
    - Task description: task example and analysis
    - Requirements and metrics
- In class activity [25 min]
    - Task analysis and description
- Project discussion [10 min]

Learning Goals

- Define and give examples of different types of requirements.
- Compare/contrast: task description, need, problem statement, requirement, specifications, metrics, provide proper examples.
- 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.

Task centered system design



vs.

The User
a pretend person who will mold themselves to fit your system

Mary
a real person with real constraints trying to get her job done

Tasks are central to HCI

Recall Cooper’s 3 type of user goal:

- Life goals (reflective)
- End goals (behavioral)
- Experience goals (visceral)

Task description:
task *examples* work together with task *analysis*

- In HCI, establishing requirements typically begins with establishing tasks:
  - ***Task examples*** *describe tasks and (to some extent) users*

- Together with design prototypes, task examples are also a good way to evaluate initial designs at low cost and effort.

USERs ->in this lecture, when you see User, think Persona.

Creating design goals: Scott Klemmer

https://www.youtube.com/watch?v=m92DLyQNoS8

Task examples

- Articulate concrete, detailed examples of tasks users perform or want to perform that your system should support

- It is useful to categorize task examples:
  - Routine
  - Infrequent but important
  - Infrequent and incidental

    - Why?

This is the style of task example you will write for the Ideate milestone.

Grocery list task example

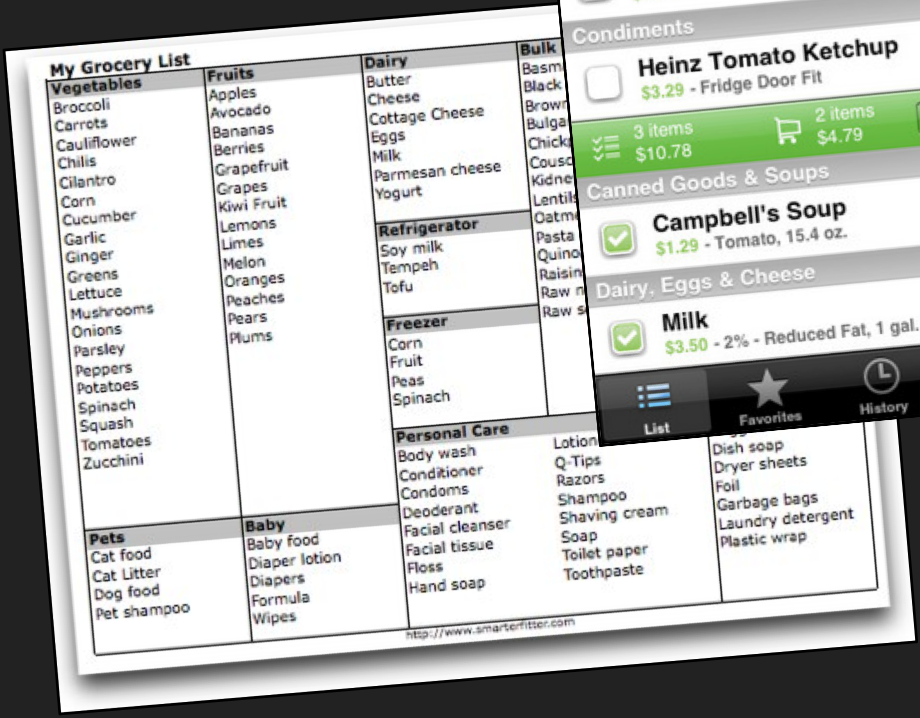
- Melody is doing the weekly menu planning for her family of 4. She chooses a set of recipes that suit the season, available prep time, current individual dietary preferences and her own preference at that moment.
- Many of this week’s choices are regulars. She creates a shopping list of ingredients, ordered by where they can be found in the grocery store. Her partner Cameron, who does the actual shopping and is more familiar with the store, supplies “feedback” on any errors she makes.
- When a recipe requires an ingredient that was already needed for an earlier day’s meal, it is incremented. After getting through the week’s meals, she adds a few regular items like milk, bread, cereal and juice. After Cameron has left with the list, she realizes she’s forgotten to check the pantry for staples like flour and rice.

Key point to note

- Task examples are interface independent, or as independent as possible.

- The key distinction between a scenario and a task is that a scenario is design-specific, in that it shows how a task would be performed if you adopt a particular design, while the task itself is design-independent.

People make many kinds of grocery lists



My Grocery List

Vegetables
Broccoli
Carrots
Cauliflower
Chilis
Cilantro
Corn
Cucumber
Garlic
Ginger
Greens
Lettuce
Mushrooms
Onions
Parsley
Peppers
Potatoes
Spinach
Squash
Tomatoes
Zucchini

Fruits
Apples
Avocado
Bananas
Berries
Grapefruit
Grapes
Kiwi Fruit
Lemons
Limes
Melon
Oranges
Peaches
Pears
Plums

Dairy
Butter
Cheese
Cottage Cheese
Eggs
Milk
Parmesan cheese
Yogurt

Refrigerator
Soy milk
Tempeh
Tofu

Freezer
Corn
Fruit
Peas
Spinach

Bulk
Basm
Black
Brown
Bulga
Chick
Cousc
Kidne
Lentils
Oatm
Pasta
Quino
Raisin
Raw n
Raw s

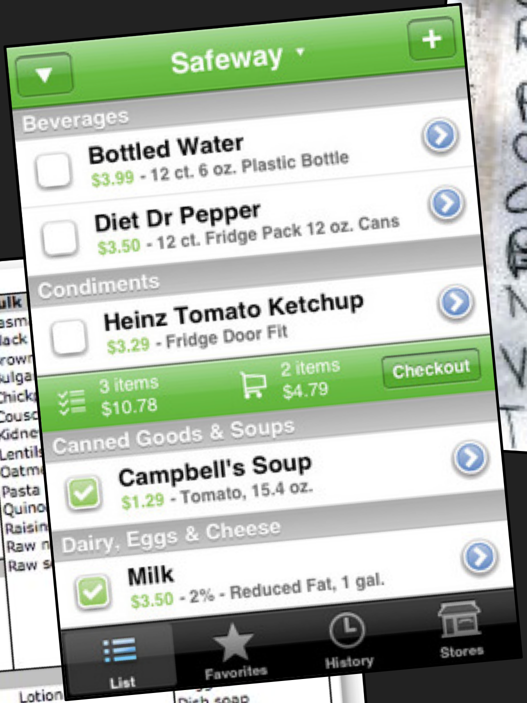
Pets
Cat food
Cat Litter
Dog food
Pet shampoo

Baby
Baby food
Diaper lotion
Diapers
Formula
Wipes

Personal Care
Body wash
Conditioner
Condoms
Deodorant
Facial cleanser
Facial tissue
Floss
Hand soap
Lotion
Q-Tips
Razors
Shampoo
Shaving cream
Soap
Toilet paper
Toothpaste

Dish soap
Dryer sheets
Foil
Garbage bags
Laundry detergent
Plastic wrap

http://www.smarterfitter.com



Safeway

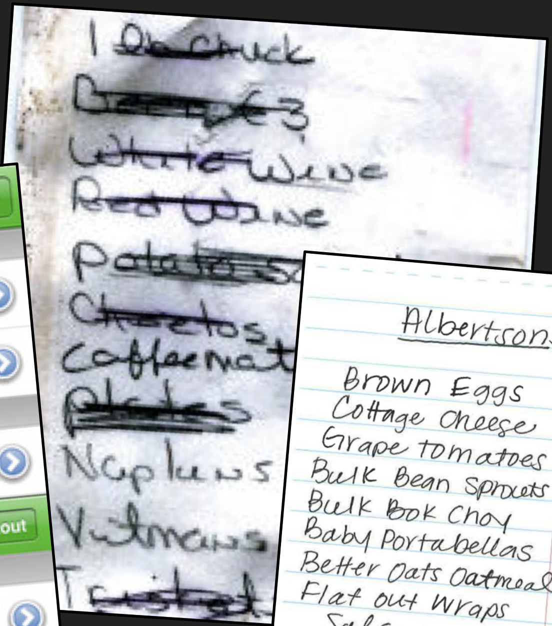
Beverages
- Bottled Water — $3.99 - 12 ct. 6 oz. Plastic Bottle
- Diet Dr Pepper — $3.50 - 12 ct. Fridge Pack 12 oz. Cans

Condiments
- Heinz Tomato Ketchup — $3.29 - Fridge Door Fit

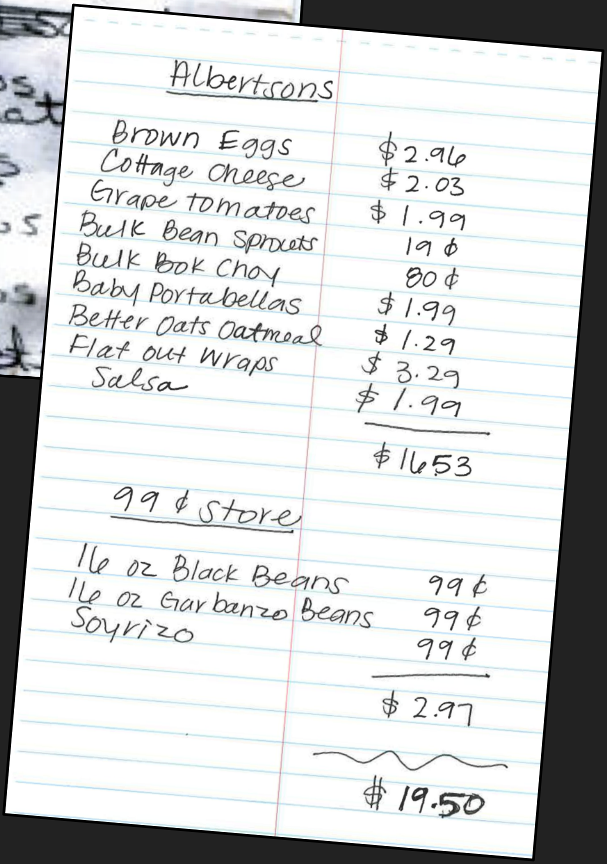
3 items $10.78 | 2 items $4.79 | Checkout

Canned Goods & Soups
- ✓ Campbell's Soup — $1.29 - Tomato, 15.4 oz.

Dairy, Eggs & Cheese
- ✓ Milk — $3.50 - 2% - Reduced Fat, 1 gal.



1 ~~lb chuck~~
~~[illegible]~~ 3
~~White~~ Wine
~~Red Wine~~
~~Potatoes~~
~~Cheetos~~
Coffeemat
~~Plates~~
Napkins
Vitamins
~~Trisket~~



Albertsons	
Brown Eggs	\$2.96
Cottage Cheese	\$2.03
Grape tomatoes	\$1.99
Bulk Bean Sprouts	19¢
Bulk Bok Choy	80¢
Baby Portabellas	\$1.99
Better Oats Oatmeal	\$1.29
Flat out Wraps	\$3.29
Salsa	\$1.99
	\$16.53
99¢ Store	
16 oz Black Beans	99¢
16 oz Garbanzo Beans	99¢
Soyrizo	99¢
	\$2.97
	\$19.50

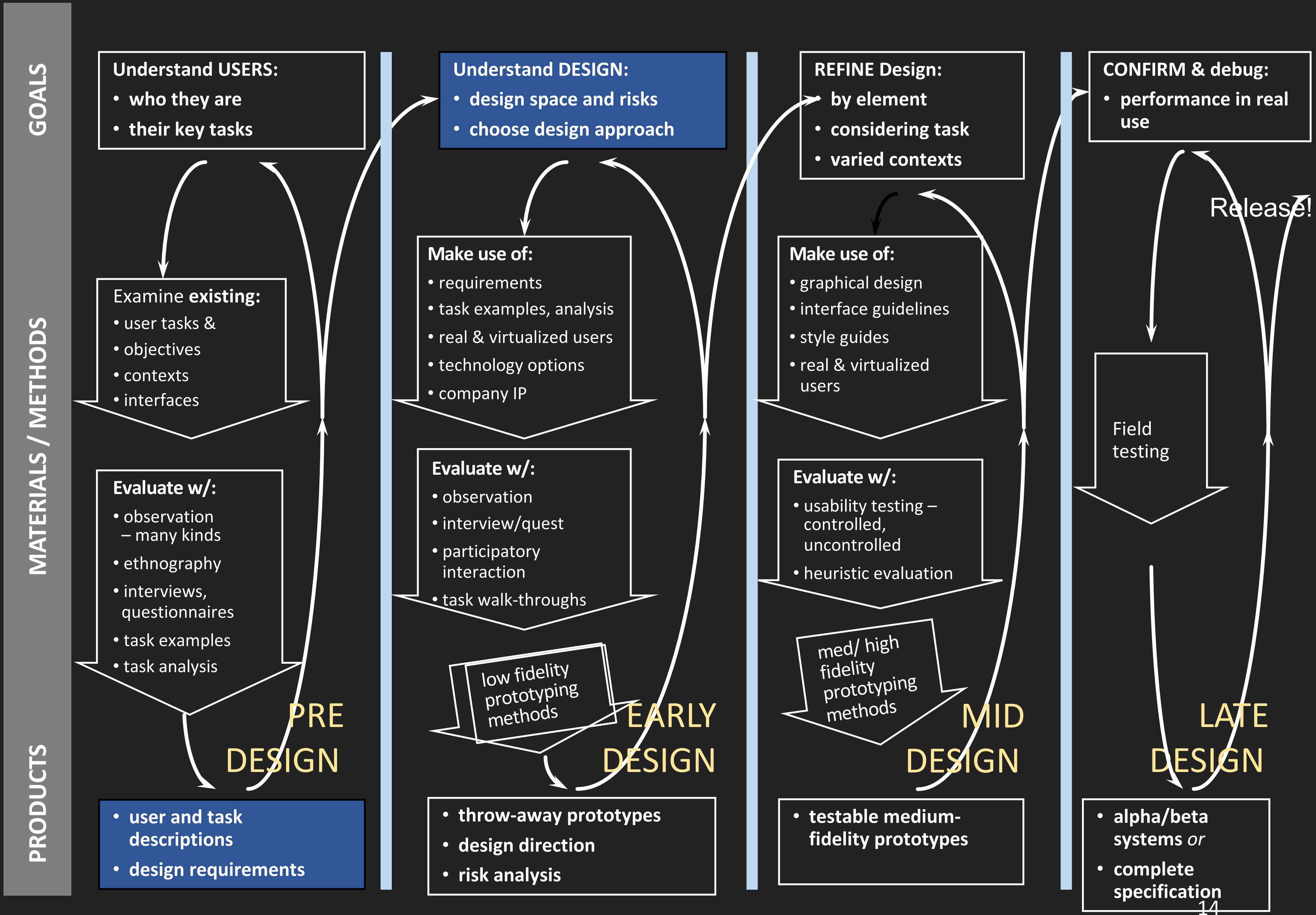
And lists are just one of the ways that people vary in how they do their meal planning.

- plan ahead, or last minute??
- alone, or as a family?
- does list maker also shop?
- improvisation allowed?

Rich information collected for “Empathize” …

- … is used for creating personas AND for generating task examples.

Big Picture – WHEN do task examples and requirements happen?



GOALS

MATERIALS / METHODS

PRODUCTS

PRE DESIGN

Understand USERS:
- who they are
- their key tasks

Examine **existing:**
- user tasks &
- objectives
- contexts
- interfaces

Evaluate w/:
- observation – many kinds
- ethnography
- interviews, questionnaires
- task examples
- task analysis

- **user and task descriptions**
- **design requirements**

EARLY DESIGN

Understand DESIGN:
- **design space and risks**
- **choose design approach**

Make use of:
- requirements
- task examples, analysis
- real & virtualized users
- technology options
- company IP

Evaluate w/:
- observation
- interview/quest
- participatory interaction
- task walk-throughs

low fidelity prototyping methods

- **throw-away prototypes**
- **design direction**
- **risk analysis**

MID DESIGN

REFINE Design:
- **by element**
- **considering task**
- **varied contexts**

Make use of:
- graphical design
- interface guidelines
- style guides
- real & virtualized users

Evaluate w/:
- usability testing – controlled, uncontrolled
- heuristic evaluation

med/ high fidelity prototyping methods

- **testable medium-fidelity prototypes**

LATE DESIGN

CONFIRM & debug:
- **performance in real use**

Release!

Field testing

- **alpha/beta systems** ***or***
- **complete specification**

K MacLean - derived from version by Saul Greenberg (U Calgary)

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?*

a) users must be able to buy electronic tickets in less than 3 distinct steps.

b) users must feel that the buying of electronic tickets is easy.

c) users must be able to buy electronic tickets using the website.

d) users must be able to learn how to buy electronic tickets on their first attempt.

functional: usefulness not usability

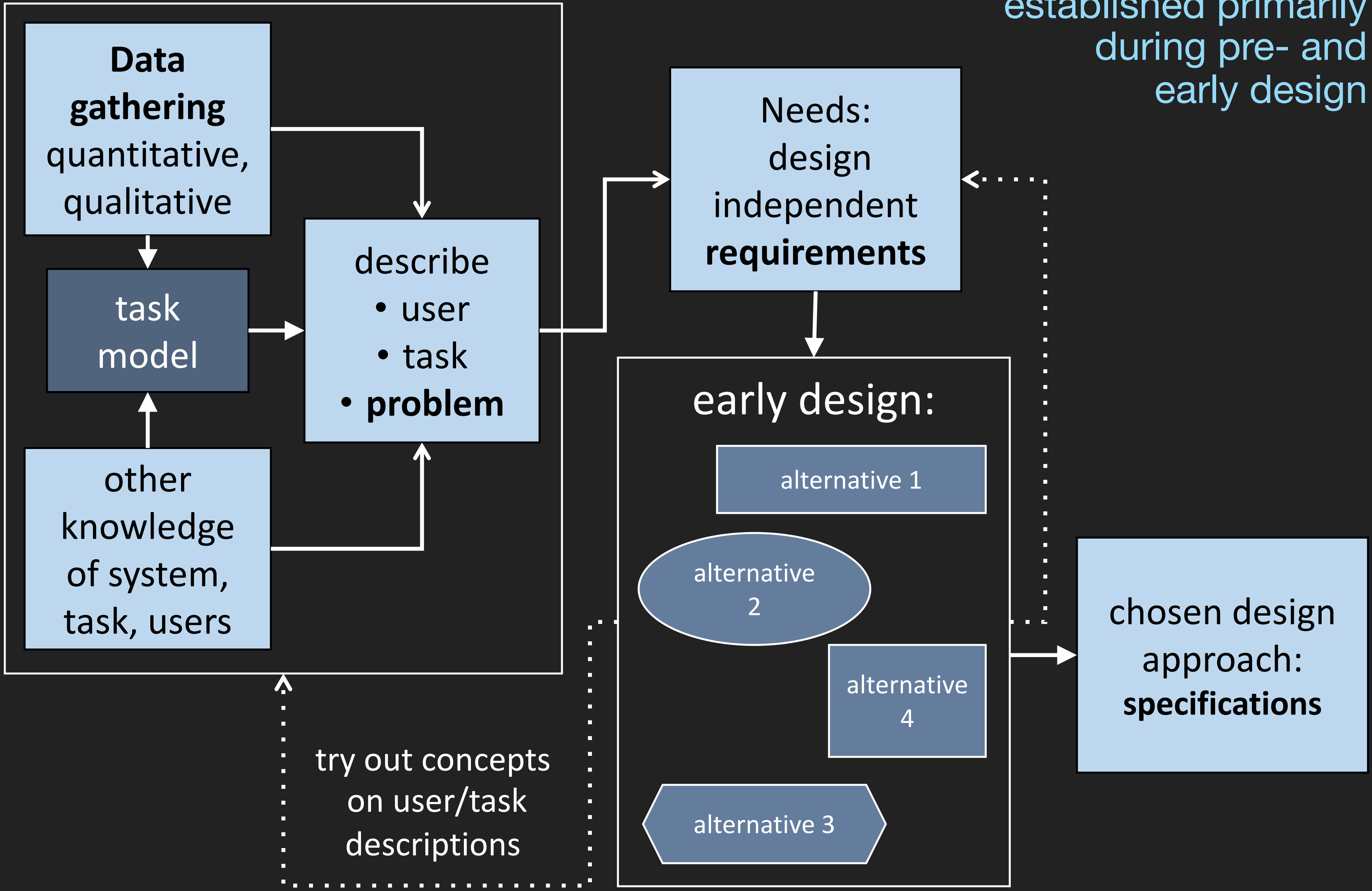
specifies a feature or capability

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

established primarily during pre- and early design


Data gathering quantitative, qualitative
task model
other knowledge of system, task, users
describe
• user
• task
• problem
Needs: design independent requirements
early design:
alternative 1
alternative 2
alternative 4
alternative 3
chosen design approach: specifications
try out concepts on user/task descriptions

Three steps to requirements

1. identify (and model) the human activity
   which the proposed interactive system will support:
   task, goals, conditions; current problems and strengths
2. identify all the users & other stakeholders
   who perform or will perform the activity:
   groups, capabilities, motives, needs
3. 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 (and model) 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?

Activity – part I [10 min]

•Activity goal: In team of 2 practice breaking down and analyzing a human activity to start to generate requirements.

Human activity: scheduling meetings
task: schedule a meeting between project team members.

1. What steps are involved in this task?
   - Try stating goals for the steps.
   - Remember *design/interface independence!* These goals should apply for … doodle poll, email coordination, in meeting, etc.
2. How can these steps go wrong?

•Create a diagram of this task to help answer these questions

scheduling example, my list…

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:**
    1. ask all to communicate avail during a block; OR suggest times, get responses
    2. examine, manually or automatically
    3. find common openings, if any
    4. 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:

Activity – part 2
task dependencies [5 min]

- 1. What “task objects” (i.e. resources, parts of process, even things generated by the process) might be required to meet this task?

- 2. Are there dependencies on these objects that could lead to conflicts or breakdowns?

•Use your diagram to identify, or draw an entirely new diagram if required!

Scheduling example, my list

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

Three steps to requirements

1. identify (and model) the human activity
   which the proposed interactive system will support:
   task, goals, conditions; current problems and strengths
2. identify all the users & other stakeholders
   who do or will perform the activity:
   groups, capabilities, motives, needs
3. 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.

We have used Personas for this step.

The user

- 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”

Activity – part 3
user needs [10 min]

1. Identify potential users in this situation (are they all the same? different?)

2. Brainstorm a list of general human needs or needs specific to these users that could apply to this task?

For developing Personas – What are the relevant variables? What can their range of values be?

Scheduling example, my list

OBJECTS for Scheduling task?
conflicts in their use suggest dependencies…

- calendars
- Comm. Mech. (email, phone, cooler)
- “leader” – mtg leader, secretary, program

Other signs of task dependencies?

- can’t find time until heard from all participants
- participants can’t supply time until told when to look

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?

Three steps to requirements

1. identify (and model) the human activity
   which the proposed interactive system will support:
   task, goals, conditions; current problems and strengths
2. identify all the users & other stakeholders
   who do or will perform the activity:
   groups, capabilities, motives, needs
3. 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.

FOCUS: which users will you support?

- Usually the intersection of
  - Greatest need
  - and thus opportunity for business

Plus

- Feasibility
  we can identify a way to help

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:
  1. **Supported activities** (tasks and steps)
     Tasks and processes involved that support the activity.
  2. **User(s)**
     Who does the task and what are their characteristics?
  3. **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!

Extra slides

Reflection on 1st milestone

- What did you learn?
- What was the biggest challenge??
- How satisfied are you with your result/process?

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