Defining User Needs:
- task examples
- requirements
announcements

Project Teammate request survey up, due next Tuesday:
https://survey.ubc.ca/surveys/pbucci/344-2017w1-teammate-request-proj/

This week: hand in two slides for peer sharing
Next week: project report due
Two weeks out: midterm (covers up to and including this week in lecture material and the full mini-project)
quiz

Q1: 92% correct
Q2: 86% correct
Q3: 86% correct
Q4: 47% correct
Q5: 78% correct
Q6: 83% correct
task examples

learning goals

• outline the steps in task centered design
  – explain how it may differ from other design process, and describe its strengths and limitations

• list the elements and characteristics of a good task example, and demonstrate these when writing task examples

• be able to “walkthrough” an interface design using a task example
task driven design

where else we’re covering it

by now (W05 pre-readings)
• what task examples are, example
• (optional) task analysis

workshop + upcoming assignments
• (next week) - practice creating task examples from observation and interview data

more examples of task examples:
• course website / resources
the “Double Diamond”
Understand USERS:
- who they are
- their key tasks

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

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

CONFIRM & debug:
- performance in real use

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

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

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

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

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

Evaluate w/:
- task examples
- task analysis

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

Make use of:
- low fidelity prototyping methods

Make use of:
- testable medium-fidelity prototypes

Make use of:
- throw-away prototypes
- design direction
- risk analysis

Make use of:
- real & virtualized users

Release!
the task centered view of process/project

step 1: identification (in pre-design)
  • identify specific users
  • articulate and analyze examples of realistic tasks

step 2: requirements (finishing pre-design)
  • decide which of these tasks and users the design will support

step 3: conceptual design (in early design)
  • base ‘wireframe’ of design representation and interaction sequences on these tasks, emphasizing user “mental model”

step 4: task walkthroughs (earliest evaluations of designs)
  • using your design(s), walk through the tasks to test proposed interface
task description:
task examples work together with task analysis

in HCI, establishing requirements typically begins with establishing tasks:

• task examples describe tasks and users

together with design prototypes, task examples are also a good way to evaluate conceptual design (covered later)
User Interface Design Process: Evolving Iterations

**Understand USERS:**
- who they are
- their key tasks

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

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

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

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

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

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

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

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

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

**Make use of:**
- testable medium-fidelity prototypes

**MID DESIGN**
- field testing

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

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

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

**PRE DESIGN**
- low fidelity prototyping methods

**GOALS**
- Understand DESIGN:

**MATERIALS / METHODS**
- throw-away prototypes
- design direction
- risk analysis

**K MacLean - derived from version by Saul Greenberg (U Calgary)**
task examples
task-centered system design

HCI Requirements Analysis:
exactly who uses the system to do what?

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

vs.

Mary
a real person with real constraints trying to get her job done
step 1: identification

contact real people who will be potential users of system
  • identify specific end users: prototypical categories & extremes

spend time with them discussing how the system might fit in
  • who would be willing to talk to you about this?
  • if you can’t get them interested, who will actually buy/use your system?

learn about the user’s tasks (and potentially model w/ task analysis)

**task examples** articulate **concrete, detailed examples** of tasks they perform or want to perform that your system should support
  • routine
  • infrequent but important
  • infrequent and incidental
step1: identification (cont)

if there are no real users or tasks…

• think again, there probably are!

Jeff Hawkins, the inventor of the Palm Pilot, was said to have carried a small block of wood around in his shirt pocket … As various everyday situations arose, he would take out the block of wood and imagine how he would use the device.¹

¹see Sato and Salvador, Interactions 6(5)

…the same technique can be used to evoke a response from expected end-users.

if all else fails…

• describe your expected set of users, and expected set of tasks
• later, you will need to verify or modify your assumptions.
what is a task example?

A **story** about a *specific user* performing a *specific task* in an interface-independent way is possible.

Includes a **persona** and a **scene**.
what is a task example?

\[ \text{task example} = \text{persona} + \text{scene} \]

**persona**

**Must have**
- Name
- Background
- Goals
- Needs/values
- Real data as input

**Could include**
- job title
- demographics
- relationships
- quote
- pictures
- …
what is a task example?

\[ \text{task example} = \text{persona} + \text{scene} \]

\[ \text{scene} \]

**Must have**
- Setting: a place, time, etc. where action happens
- Action: a problem, task, and set of subtasks the persona enacts
- Real data as input

**Could include**
- artifacts/objects
- setting or environment details
- multiple characters
- …
people make many kinds of grocery lists

lists say a lot about the task – and the people who make the lists.
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?
Lisa 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 eccentricities and her own preference at that moment.

Many of this week’s recipe choices are regulars. She creates a shopping list of ingredients ordered by where they can be found in the grocery store. Her husband, Nick, 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 Nick has left with the list, she realizes she’s forgotten to check the pantry for staples like flour and rice.
good task examples:

1. say **what** the user wants to do but **does not** say **how** they would do it
   - no specific assumptions made about the interface
   - can be used to compare different design alternatives in a fair way

2. are very **specific**
   - says exactly what the user wants to do
   - specifies actual inputs the user would eventually want
   - *tasks can be hierarchical*
good task examples:

3. describe a complete job
   - **not just a list** of simple independent goals!
   - forces designer to consider **how different steps will work together**

4. say who the users are ** personas**
   - design success strongly influenced by what users know
   - use real(ish) names
   - reflect real interests of real users
   - helps find tasks that illustrate functionality in a person’s real work context
good task examples:

5. as a set, identify a broad coverage of users and task types
   • the typical ‘expected’ user   typical routine tasks
   • the occasional but important user  infrequent but important tasks
   • the unusual user  unexpected or odd tasks

6. ARE EVALUATED  (final and very important step)
   • circulate descriptions to users, and rewrite if needed
     → ask users for
       omissions  corrections  clarifications  suggestions
Lisa 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 eccentricities and her own preference at that moment.

Many of this week’s recipe choices are regulars. She creates a shopping list of ingredients ordered by where they can be found in the grocery store. Her husband, Nick, 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 Nick has left with the list, she realizes she’s forgotten to check the pantry for staples like flour and rice.
step 2: test design using TEs

develop *designs* to fit specific users and tasks

use *task examples* to:

- make design candidates concrete & debug them
- consider how design features work *together* to help a person accomplish real work
- consider the real world contexts of real users

how? *scenarios* show how each *task* is handled by *design*:

- what the user would see / do step-by-step when performing the task

    task example + design = *scenario*

    a scenario is design-specific;
    a task example is design-independent.
step 3: walk-through evaluation (week 09)

good for: developing / debugging an interface

process:

- select one of the task examples
- for each user’s step/action in the task and using your current interface prototype:
  - ask a preset of questions (about: mental models)
  - ask: is there a believable story that motivates the user’s actions?
  - if you locate a problem, mark it & pretend it has been repaired
  - then go on to next step.

This is called a Cognitive Walkthrough.
activity – part I (subgroups)

Critique 3 task examples on one design

Three TE’s have been provided to you.
→ See how well they support figuring out how well that design supports the person in the TE, in doing their task.

1. Walkthrough
2. Critique

Human activity to support: Time Tracking
Designs to compare: two web-based time tracking tools
activity – part I

Create two task examples

1. Build a persona.
   Remember the real people from your pilots/studies.
   Brainstorm and write down as many details as you can remember about who they were:

   - what were their backgrounds?
     *were they students? Upper/lower year? Have families?*

   - what did they need/value?
     *did they care about expensive groceries/restaurants? Why?*
Create two task examples

1. Build a persona.
   Make up (at least) two names. Start to organize the personal details from your real users into (at least) two personas. Group similar traits together. Make sure that each persona tells a very different story to ensure full coverage.
activity – part I

Create two task examples

2. Add a scene.

For each of your personas, choose a high-level interface-independent task that they would like to perform. Build a story around that task. Include details such as setting, important other people and relationships, etc., but don’t include any details that aren’t directly relevant to your task or interface.

Shoot for about 1-2 paragraphs.
Create two task examples

3. Critique.
   Find someone who didn’t help write the task examples. Get them to read yours and critique it. Then trade.
right now:

subgroups: walkthroughs of task examples.
~5min / task example
right now:

1) demonstrate time tracking site to your other teammates

2) go through each task example and decide which SITE best fits its needs. (about 3 min per TE)
activity – part II (full groups)

How well does each site support each TE?
limitations of task-centered design

tasks almost always embody a process even if they are not specific to a specific technological implementation

• may not encourage consideration of alternate ways to do tasks

• may be hard to produce ‘pure’ ‘system’ or ‘process’ independent tasks; must specify where to “freeze” the design vs. allow it to vary
The professor arrives late for class, hurriedly turns on her laptop, then searches for a wireless connection to the projector to display her slides.

Given the task of

... projecting slides wirelessly from a laptop:

Q1: This brief is a scenario, not a task example.

... displaying information to students in a room:

Q2: This brief is a scenario, not a task example.
your mini-project

this week: you will evaluate for understanding
observe and/or interview someone doing a task(s)
analyze and draw conclusions

next week: you will analyze this task(s),
and **richly describe it** with a task example

then, generate a list of requirements.

Later: re-use these task examples in design stages.

As you run your evaluation, think about the TE you will write.

• What should go into it?
• How would a TE about your participant differ from one you’d write for yourself?
Reference / Example:

Cheap Shop
example:
Cheap Shop Catalog Store

old way:

• people shop by browsing paper catalogs scattered around the store.

• when a customer sees an item she wants, she enters its item code from the catalog onto a paper form.

• customers give this form to a clerk, who brings the item(s) from the back room to the front counter.

• customers then pay for the items they want.
developing task examples: Cheap Shop

TASK: at Cheap Shop, people browse a catalog and then order goods.

task example 1: note: design-independent

Fred, who is caring for his demanding toddler son, buys a stroller (red is preferred, but blue is acceptable), pays for it in cash, and uses it immediately.

Fred is a first-time customer to this store and has little computer experience.
Cheap Shop: same task... but lots of variations when you look closely

**TASK:** at Cheap Shop, people browse a catalog and then order goods.

**task example 2:**

Iris, an elderly arthritic woman is price-comparing the costs of a child’s bedroom set, consisting of a wooden desk, a chair, a single bed, a mattress, a bedspread, and a pillow.

she takes the description and total cost away with her, to check against other stores.

two hours later, she returns and decides to buy everything but the chair.
fast forward:
there’s a first Cheap Shop design.

Try out this task example on the design.

Later, we’ll use a formal method for this (cognitive walkthrough).

Now, let’s just use common sense.
Cheap Shop: 1\textsuperscript{st} prototype specifications

to create an order
– on screen 1, shoppers enter their personal information and their first order
– text is entered via keyboard
– the tab or mouse is used to go between fields.

further orders
– shoppers go to the 2nd screen by pressing the Next Catalog Item button

order completion
– shoppers select ‘Trigger Invoice’.
– the system automatically tells shipping and billing about the order
– the system returns to a blank screen #1

to cancel order
– shoppers do not enter input for 30 seconds (as if they walk away)
– the system will then clear all screens and return to the main screen

input checking
– all input fields checked when either button is pressed.
– erroneous fields will blink for 3 seconds, and will then be cleared.
– the shopper can then re-enter the correct values in those fields.
what were those?

requirements or specifications?

what’s the difference?
Cheap Shop

1st prototype of a new design

how is Fred doing?

Screen 1

Screen 2
Cheap Shop: there are other roles, too…

TASK: at Cheap Shop, people browse a catalog and then order goods.

task example 3:

a “Cheap Shop” clerk, the sole salesperson in the store, is given a list of 10 items by a customer who does not want to use the computer.

the items are:

4 pine chairs, 1 pine table, 6 blue place mats, 6 “lor” forks, 6 “lor” table spoons, 6 “lor” teaspoons, 6 “lor” knives, 1 “tot” tricycle, 1 red ball, 1 “silva” croquet set

after seeing the total, the customer decides to take all but the silverware, and then adds 1 blue ball to the list.

the customer then changes his mind about paying by credit card, and decides to pay cash. The customer wants the items delivered to his home the day after tomorrow.

while this is occurring, 6 other customers are waiting for the salesperson.
[break]
requirements & metrics
requirements & metrics

describe the problem we are solving
  → what is required to fix it
  → arrive systematically at a solution approach
requirements & metrics

where else we’re covering it

by now (W05 pre-readings)

• establishing requirements
• data gathering for requirements
requirements & metrics – today:

• what: functional vs. usability requirements
• when and how
• three steps to requirements
  1. ways to describe tasks and users
  2. identify focus users and metrics
  3. set level of support
learning goals

- define and give examples of different types of requirements

- compare/contrast: task description, need, problem statement, requirement, specifications, metric? Provide examples you can defend.

- give examples of HCI techniques (e.g. evaluation, prototyping, modeling) 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
2. what are requirements?

functional requirements: what the interface must do

• usefulness -- scope, features …

non-functional requirements: many kinds.

usually, 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 pure 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
example: defining a human robot interaction:
real project with General Motors

take automation robot “out of the cage:”
robot must be aware of and communicate “naturally” with user
→ HRI design first

HRI → functional requirements:
• vision system capability
• robot mobility, speed, responsiveness … many more
functional vs. usability requirements:

Q4 (pre-reading quiz):
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; e.g. from business model
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

eval results
quantitative, qualitative

task model
describe
• user
• task
• problem

other knowledge of system, task, users

needs
→ design
independent
requirements

early design:

alternative 1
alternative 2
alternative 3
alternative 4

try out concepts on user/task descriptions

chosen design approach: specifications
the “Double Diamond”
Big Picture – WHEN do requirements happen?

**GOALS**
- Understand USERS:
  - who they are
  - their key tasks

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

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

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

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

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

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

- Make use of:
  - low fidelity prototyping methods

- Evaluate w/:
  - testable medium-fidelity prototypes

**PRODUCTS**
- PRE DESIGN
  - user and task descriptions
  - design requirements

- EARLY DESIGN
  - throw-away prototypes
  - design direction
  - risk analysis

- MID DESIGN
  - testable medium-fidelity prototypes

- LATE DESIGN
  - alpha/beta systems or complete specification

**Field testing**

K MacLean - derived from version by Saul Greenberg (U Calgary)
3. three steps to requirements

1. identify (and model) the **human activity**
   which the proposed interactive system will support:
   task, goals, conditions; current problems/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.
1. identify (and model) the human activity

what outputs might you have at the end of this?

• goals
• task descriptions, tasks examples
• task models; normal steps and process; common breakdowns
• lists of problems
• things that work well
• etc.

⇒ together describe our *situation of concern*
the situation of concern is …

altogether the **context** for the design problem -

*something is wrong* that we want to change, or

*something could be improved upon*

some **course of action** is required

- that will result in a **change** that resolves the situation

⇒ this course of action (i.e. the solution!) is what needs to be specified in requirements!
example: scheduling meetings

situation of concern:
• 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?

Doodle is one obvious one which has taken hold; others?
activity – part I

activity goal: 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?
   • Brainstorm all aspects of the task.
   • remember design 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 tasks:
• 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?
a flow diagram of meeting task

**Flow Diagram**
(from meeting creator perspective)

- **Invitée**
  - responds to invites

- **List of possible times**
  - (email, phone, paper, software)
  - may have no available times

- **List of available times**
  - (email, phone, paper, software)

- **Meeting invitation**
  - (email, phone, paper, software)

- **Meeting confirmation**
  - (email, phone, paper, software)

- **Meeting reminder**
  - (email, phone, paper, software)

- **Meeting creator**
  - propose meeting time and place
  - books meeting room
  - sends meeting invitations
  - sends meeting reminders

- **Confirmation**
  - (email, phone, in-person)

- **Meeting room reservation**
  - request for specific time
  - (email, phone, in-person)

- **Space manager**
  - manage meeting room availability
  - check availability

- **Meeting room calendar**

**Advantages:**
- captures flow of information between different people, some artifacts
- loops and other structure is implied
- a different view, which might reveal problems not encountered in other diagrams

**Limitations:**
- only from one perspective (task flow might look different from invitee perspective)
- highlights certain problems, but might hide others (diagramming other aspects may be valuable)
- breakdowns point to places where more support for iteration on times/locations may be necessary (sequence diagram might help)
activity – part I

activity goal: 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?
   • Brainstorm all aspects of the task.
   • remember design 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!
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

most tasks focus on a single resource

- task cannot be accomplished without the resource
- once resource is available the task can be completed

processes have multiple dependencies

- the focus shifts as different 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 (email/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

1. What “task objects” (i.e. resources, parts of process -- e.g., generated by the process-- , etc.) 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.
the user

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

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
activity – part 3

user needs

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?
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?

eexamples 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 (metrics)**
   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.
METRICS: how we know if we have succeeded

**quantitative metrics:**
what happens when people use the interface.
- you can count them
- more in 444 – *but we’ll list some here.*

**qualitative metrics:** reveals why it happened.
- descriptive.
  harder to *directly* count and measure – but really important
- our focus in 344 – *covered in earlier classes*
- e.g. *stories* about user impressions and frustrations, and their *change* pre- and post design; critical incidents; product “buzz” …
choosing usability targets

choice of usability metrics affects the solution:

• prioritize most important facets based on goals:
  e.g. is speed most important, or is it very bad to make errors?
  e.g. learnability can be important, especially for novices, but may impact effectiveness or utility

• establish metrics by deciding on what constitutes success.
  e.g. if most (what is most??) participants feel that this is learnable, and 90% (exactly??) can complete our task efficiently (within 10 mins??), we have succeeded.

note: don’t get too caught up in expressing qualitative metric quantitatively…use what’s right for your interface!
some **quantitative** usability metrics

some examples of things that can be counted:

- speed of performance
- incidence of errors
- ease of learning the system
- user satisfaction
some **quantitative** usability metrics repeat in more detail ....

some examples of things that can be counted:

- speed of performance
- incidence of errors
- ease of learning the system
- user satisfaction
quantitative usability metrics

performance
  • speed
    – how long it takes to perform an activity
    – how many people it requires to complete
  • error rate
    – how often will errors happen
    – how critical will they be to success

survey on 1st day of class:
% respondents that followed instructions?
quantitative usability metrics

learning to use the system

• how much time will it take
• what/ how much background is required
• what proportion of users require extra help
• how much speedup do experts ultimately obtain

satisfaction (subjective)

• do users like the system
**quantitative usability metrics**

*these can be both rated and described*

- recovery from errors
  - can users fix mistakes
  - how difficult is it

- retention of learned skills
  - do users continue to perform well after breaks

- ability to customize
  - can users control how they use the system (how well?)

- ease of reorganization of activities
  - can the system be used to do other things
  - does the system support changing needs
set metrics and levels $\rightarrow$ can you achieve them?

needs $\rightarrow$ design independent requirements

early design:

alternative 1
alternative 2
alternative 3
alternative 4

chosen design approach: specifications

describe
- user
- task
- problem

task model

other knowledge of system, task, users

eval results quantitative, qualitative

try out concepts on user/task descriptions

of system, task, users

other

quantitative, qualitative

user

quantitative,

qualitative

task

model

needs

early design:
activity – part 4

\textit{rough set of (usability/ux) requirements}

1. **specify 2-3 high level problems** a new system/interface could address.
   - \textit{e.g., time can’t be set until participants respond to invitations}

2. List **non-functional properties** that a solution will need to have to address those above issues
   - \textit{e.g., responding to invitations should be fast}

3. Add **quantitative and qualitative metrics** to these properties, that can be used to evaluate design success
   - \textit{e.g. time for respondent to complete task on the first try.}

4. Add **values for metrics** that would be important to achieve, to provide value.
   - \textit{e.g., time to respond to invitations should be < 1 minute}
which of these apply to the scheduling system?

• speed of performance
• incidence of errors
• ease of learning the system
• user satisfaction
• recovery from errors
• retention of learned skills
• ability to customize
• ease of reorganization of activities
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.
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?
putting it all together

example – meeting scheduling

supported task
• locating a jointly available meeting time

users
• people with tight schedules who need to participate in meetings of 3 or more participants
• people with “frequent” online access

level of support
• users can provide all requested information within 1 minute of their time
• require no iteration
• respect privacy (e.g. posting shared calendars)
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**
we don’t explicitly tell you to list the subset of users you are going to support.

but, they must clearly reflected in the requirements that you come up with

your requirements should generally:

- be written in full sentences
- make sense in relation to your findings (you’ll have to justify them!)
the one-line problem statement
a succinct high-level requirements statement

INCLUDES

1. supported activity
   • what tasks or processes are involved and how do they support the activity?

2. user(s)
   • who does the activity and what are their characteristics?

3. level of support
   • what usability factors will we consider important?
the one-line problem statement example: scheduling meetings

supported activity

- locating a jointly available meeting time

users

- people with tight schedules who need to participate in meetings of 3 or more participants
- people with “frequent” online access

level of support

- users can provide all requested information with 1 minute of their time
- require no iteration
- respect privacy (e.g. posting shared calendars)
the **one-sentence** problem statement encapsulates a high-level requirement

*i.e. course of action*

Design a tool that supports

- easy (1 minute effort), zero-iteration scheduling of meetings
- supported activity by three or more participants

level of support quantified!

users

a lot of detail behind this!
how does this **problem statement**
relate to:

• task *examples*
• task *descriptions or models*

put it all together …
the form of the solution (finally, the design!!):
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