EECS 3461
User Interfaces

Usability Methods

Recap: UCD

1. Define the Context
2. Describe the User
3. Needs Analysis and Task Analysis
4. Function Allocation
5. System Layout / Basic Design
6. Mockups & Prototypes
7. Usability Testing
8. Iterative Test & Redesign
9. Updates & Maintenance

Recap: UCD

Always think from perspective of **user**

No simple, universal **checklists**

- There are **concepts**, **principles**, and **guidelines**
- Focus on *higher level* principles that apply *across* situations, display types, etc.

Dealing with real-world **constraints**

- Should **not cut out steps**
- **Optimize** the efficiency of each step instead

Is UCD always iterative?

Not necessarily

- *in principle* applicable to a 'waterfall' development
- at its best in an *iterative* development environment

Iteration can take place:

- ‘in the small’
  - within each stage
- ‘in the large’
  - the whole development cycle

‘Right first time’ concept is a dangerous myth!

Waterfall Model

The main problem with the waterfall model is that it doesn't let you fix problems in earlier stages

- Typically, the waterfall model is adapted to allow changes to flow backwards to the previous stage
- Often, though, changes need to flow back to much earlier stages, and later stages need to be performed earlier

Finding Problems Early

A major goal in software engineering is to find and fix problems early on in the process. As soon as possible, we want to:

- Make sure the system being delivered matches what the user wants
- Find and fix bugs before the customer gets the system
- Solve usability problems before the system is implemented and before the customer receives it

Iterative Refinement

Refinement of the different stages

- Validate requirements by describing the proposed solution to the user and walking them through the tasks they perform
- Unit-test code to find problems in modules before they're integrated into the system

Internal Function Model

Internal functional model

- Independent of screen layouts or keyboard shortcuts
- A model of the data the user manipulates
- Typically, the class diagram of an object-oriented system representing the important parts of the system

Programmers vs. “real” people

Developers often have an early focus on getting the functional requirements right and finding bugs

- Don’t have an early focus on getting usability right

Much harder and more expensive to make the system usable later in the process than it is earlier

Users

Determine users' needs up front, instead of finding out why the system doesn't meet their needs after the system is delivered

Understand the nature of the problem the system is solving and understand the tasks the **user** will perform using the system

Involving the User

- Describe the proposed interface workflow to the user before doing any screen layouts
- Test interfaces using paper designs
  - Iterate through different versions of paper designs based on feedback from the user
  - Quickly produce alternate interfaces without having to code anything
- Create software prototypes that can be easily modified based on user feedback

Usability Methods

Usability Methods

- Methods to use along the way
  - E.g.
    - Analytic Evaluation
    - Expert (Heuristic) Evaluation
    - Observational Evaluation
    - Survey Evaluation
    - Experimental Evaluation
  - More methods, for different stages
    - NOTE: Not all are *evaluation* methods

Methods Table 1/3 (here and further ©UsabilityNet)

☑ limited time/resources ☐ No direct access to users ☐ Limited skills/expertise

Planning & Feasibility	Requirements	Design	Implementation	Test & Measure	Post Release
Getting started	User Surveys	Design guidelines	Style guides	Diagnostic evaluation	Post release testing
Stakeholder meeting	Interviews	Paper prototyping	Rapid prototyping	Performance testing	Subjective assessment
Analyse context	Contextual inquiry	Heuristic evaluation		Subjective evaluation	User surveys
ISO 13407	User Observation	Parallel design		Heuristic evaluation	Remote evaluation
Planning	Context	Storyboarding		Critical Incidence Technique	
Competitor Analysis	Focus Groups	Evaluate prototype		Pleasure	
	Brainstorming	Wizard of Oz			
	Evaluting existing systems	Interface design patterns			
	Card Sorting				
	Affinity diagramming				
	Scenarios of use				
	Task Anaysis				
	Requirements meeting				

UCD

1. Define the Context
2. Describe the User
3. Needs Analysis and Task Analysis
4. Function Allocation
5. System Layout / Basic Design
6. Mockups & Prototypes
7. Usability Testing
8. Iterative Test & Redesign
9. Updates & Maintenance

Methods Table 2/3

☐ limited time/resources ☑ No direct access to users ☐ Limited skills/expertise

Planning & Feasibility	Requirements	Design	Implementation	Test & Measure	Post Release
Getting started	User Surveys	Design guidelines	Style guides	Diagnostic evaluation	Post release testing
Stakeholder meeting	Interviews	Paper prototyping	Rapid prototyping	Performance testing	Subjective assessment
Analyse context	Contextual inquiry	Heuristic evaluation		Subjective evaluation	User surveys
ISO 13407	User Observation	Parallel design		Heuristic evaluation	Remote evaluation
Planning	Context	Storyboarding		Critical Incidence Technique	
Competitor Analysis	Focus Groups	Evaluate prototype		Pleasure	
	Brainstorming	Wizard of Oz			
	Evaluting existing systems	Interface design patterns			
	Card Sorting				
	Affinity diagramming				
	Scenarios of use				
	Task Anaysis				
	Requirements meeting				

UCD

1. Define the Context
2. Describe the User
3. Needs Analysis and Task Analysis
4. Function Allocation
5. System Layout / Basic Design
6. Mockups & Prototypes
7. Usability Testing
8. Iterative Test & Redesign
9. Updates & Maintenance

Methods Table 3/3

☐ limited time/resources ☐ No direct access to users ☑ Limited skills/expertise

Planning & Feasibility	Requirements	Design	Implementation	Test & Measure	Post Release
Getting started	User Surveys	Design guidelines	Style guides	Diagnostic evaluation	Post release testing
Stakeholder meeting	Interviews	Paper prototyping	Rapid prototyping	Performance testing	Subjective assessment
Analyse context	Contextual inquiry	Heuristic evaluation		Subjective evaluation	User surveys
ISO 13407	User Observation	Parallel design		Heuristic evaluation	Remote evaluation
Planning	Context	Storyboarding		Critical Incidence Technique	
Competitor Analysis	Focus Groups	Evaluate prototype		Pleasure	
	Brainstorming	Wizard of Oz			
	Evaluting existing systems	Interface design patterns			
	Card Sorting				
	Affinity diagramming				
	Scenarios of use				
	Task Anaysis				
	Requirements meeting				

UCD

1. Define the Context
2. Describe the User
3. Needs Analysis and Task Analysis
4. Function Allocation
5. System Layout / Basic Design
6. Mockups & Prototypes
7. Usability Testing
8. Iterative Test & Redesign
9. Updates & Maintenance

1 out of 6: Planning and Feasibility

Requirements – Design – Implementation - Testing and measurement - Post release

---

Stakeholder meeting

Analyse the intended context of use

Create a usability *plan* based on ISO 13407

Competitor analysis

Stakeholder meeting

Strategies

- Derive usability objectives from business objectives
- Gain commitment to usability

Benefits

- Collects information about the purpose of the system and its overall context of use
- Identifies all relevant factors before design work starts
- Brings together all the people relevant to the development
  - common vision

Stakeholder Meeting Example Qs

Why is the system being developed? What are the overall objectives? How will it be judged as a success?

Who are the **intended users** and what are their **tasks**? (Why will they use the system? What is their **experience** and **expertise**?)

Who are the **other stakeholders** and how might they be impacted by the consequences of a usable or unusable system?

What are the **technical** and **environmental constraints**? (What types of **hardware** will be used in what environments?)

Stakeholder Meeting Example Qs (2)

What key **functionality** is needed to support the user needs?

How will the system be **used**? What are typical **scenarios** of what the users can achieve?

What are the usability goals? (e.g. How important is ease of use and ease of learning? **How long** should it take users **to complete their tasks**? Is it important to minimise user **errors**? What GUI **style guide** should be used?)

How will users obtain **assistance**?

Are there any initial **design concepts**?

Is there an existing or **competitor system**?

Analyse Context of Use

Fewer people involved

Information on

- Who are the **intended users** and what are their task? (Why will they use the system? What is their **experience** and **expertise**?)
- What are the technical and environmental **constraints**? (What types of **hardware** will be used in what organisational, technical and physical **environments**?)

Maybe arrange for **a field study** to observe users

Analyse Context of Use: Summary

Benefits

- Fewer people involved
- Identifies all relevant factors **before design work starts**
- Provide a basis for designing **later usability tests**

Brainstorming

- One of the oldest known methods for generating group creativity
  - The first phase generates ideas
  - The second phase evaluates them
    - An experienced facilitator is useful
- Benefits
  - Everyone gains understanding of problem space
  - Also, feeling of common ownership of results
- Drawbacks (?)
  - Individuals working alone can generate more and better ideas than when working as a group

ISO 13407
ISO TR 18529

1	**Ensure HCD content in system strategy**
1.1	Represent stakeholders
1.2	Collect market intelligence
1.3	Define and plan system strategy
1.4	Collect market feedback
1.5	Analyse trends in users
2	**Plan and manage the HCD process**
2.1	Consult stakeholders
2.2	Identify and plan user involvement
2.3	Select human-centred methods and techniques
2.4	Ensure a human-centred approach within the team
2.5	Plan human-centred design activities
2.6	Manage human-centred activities
2.7	Champion human-centred approach
2.8	Provide support for human-centred design
3	**Specify the stakeholder and organisational requirements**
3.1	Clarify and document system goals
3.2	Analyse stakeholders
3.3	Assess risk to stakeholders
3.4	Define the use of the system
3.5	Generate the stakeholder and organisational requirements
3.6	Set quality in use objectives
4	**Understand & specify the context of use**
4.1	Identify and document user's tasks
4.2	Identify and document significant user attributes
4.3	Identify and document organisational environment
4.4	Identify and document technical environment
4.5	Identify and document physical environment
5	**Produce design solutions**
5.1	Allocate functions
5.2	Produce composite task model
5.3	Explore system design
5.4	Use existing knowledge to develop design solutions
5.5	Specify system and use
5.6	Develop prototypes
5.7	Develop user training
5.8	Develop user support
6	**Evaluate designs against requirements**
6.1	Specify and validate context of evaluation
6.2	Evaluate early prototypes in order to define the requirements for the system
6.3	Evaluate prototypes in order to improve the design
6.4	Evaluate the system to check that the stakeholder and organisational requirements have been met
6.5	Evaluate the system in order to check that the required practice has been followed
6.6	Evaluate the system in use in order to ensure that it continues to meet organisational and user needs
7	**Introduce and operate the system**
7.1	Management of change
7.2	Determine impact on organisation and stakeholders
7.3	Customisation and local design
7.4	Deliver user training
7.5	Support users in planned activities
7.6	Ensure conformance to workplace ergonomic legislation

Table 1. Human-centred design processes and their base practices

Competitor analysis

- Identifies the strengths and weaknesses of competing products or services before starting work on prototypes
  - A 10 minute tour of each of 4 to 10 of the most popular products
    - The competitive advantages of each product are discussed
    - Short summary of the market position is generated
  - Alternatives: market surveys, lab tests of competitor products

Competitor analysis: Summary

Benefits

- Discover the **strengths and weaknesses** of competing products/services
- Develop list of issues that need to be addressed in order to **compete** effectively
- Gain **consensus** among a group of project stakeholders
- May also result in a list of **desirable features** that the new product could include

2 out of 6: Planning and feasibility – Requirements – Design – Implementation - Testing and measurement - Post release

Ensure that user and usability requirements are 1)well defined and 2)integrated into relevant product requirements specification

- Collect information about the user interface, users, tasks and environments
  - surveys, interviews, contextual inquiry or observation of users in a field study
  - user participation in context of use analysis, focus groups or brainstorming
  - evaluating an existing system
- Structure information
  - card sorting or affinity diagramming, create scenarios of use
- Agree what aspects should be formalised as requirements: requirements meeting

2 out of 6: Planning and feasibility – Requirements – Design – Implementation - Testing and measurement - Post release

Surveys

Interviews

Contextual inquiry

Observation of users

Context of use

Focus groups

Brainstorming

Evaluating an existing system

Card sorting

Affinity diagramming

Scenarios of use

Task analysis

Requirements meeting

User survey for design

How is the software or web site likely to be used by a specific set of users?

Who are these users likely to be?

The answers user surveys provide must be relevant to the issues that are important to the *design* team

- Traditionally carried out by post
- Now, over the internet

Interviews

Discovering facts and opinions held by potential users of the system being designed

Difficulties – Time

- usually one interviewer speaking to one informant at a time
- reports of interviews have to be carefully analysed and targeted to ensure they make their impact
  - Otherwise the effort is wasted

Benefits

- one-to-one nature: can address directly the user's concerns
- mistakes and misunderstandings can be quickly identified

Eliciting Information

The user doesn’t know what interface they need, otherwise they wouldn't be asking you to design it

The user may not know the technical limitations and feasibility of what they want

- Sometimes, what they want is too demanding technically
- Often, though, they fail to recognize what technology is capable of doing

Determining what the user really wants from what the user says they want

Why People Use Software

Find some fact or learn about something

Perform a transaction (pay a bill, buy a book)

Control or monitor something

Create something

Converse with people

Find entertainment (play a game, watch a video)

NOT Why People Use Software

Don’t use software to fill out a form

- They are trying to accomplish something (buying something online, reading an article)

If there is a way to let the user achieve the goal without that form, the interface is better without it, no matter how well designed the form is

What is the User Trying to Accomplish?

Focus on how and why a user is using the software

Are you now solving the right problem?

Asking the right questions is important

When a client wants a certain feature, ask why. Then ask why again. Keep going. (Goal)

Users' Level of Interest

A user's level of interest might be low, respect that

For example, they might use the software only once or only for a short period of time periodically

- kiosk
- online store
- printer installer

User's Motivation to Learn

It is easy to overestimate how much effort users are willing to spend to learn your interface

They often only learn enough to get by

On the spectrum from beginner to intermediate to advanced users, some may remain perpetual beginners

Contextual inquiry

Specific type of interview for *field data* from users

Usually done by one interviewer speaking to one interviewee (person being interviewed) at a time

Gather as much data as possible from the interviews for later analysis

Benefits

- Interviewees are interviewed in their context, when doing their tasks, with as little interference from the interviewer as possible
- Data should be gathered during interviews with little or no analysis, interview should result in raw data

User observation/field studies

Investigator viewing users as they work; taking notes on the activity that takes place

- Direct observation (investigator is actually present during the task)
- Indirect observation (task is viewed by some other means such as through use of a video recorder)

Useful early in user requirements stage for obtaining qualitative data

Also useful for studying currently executed tasks and processes

User Groups

Different pieces of software have different target audiences. Consider these examples:

- Facebook users
- Software Developers
- Nanotechnology Researchers
- Admins vs. the rest of the users

Some software has distinct user groups.

Each users within a group is unique (computer skills, domain experience, etc.), though.

Find out what is generally true about your users.

Information about Users

What you really want to learn about user groups:

- Their goals in using the software
- The language they use to describe what they're doing
- Their skill at using software similar to what you are designing
- Their attitudes toward the kind of thing you are designing

Remember to focus on how and why they are using your software

Understanding the User

Describing each class of target users

- Education, skill level
- Computer proficiency
- Knowledge of the domain
- Experience with similar systems
- Demographics (age, gender, social and cultural background)
- Language(s)
- Unique characteristics (e.g. vision problems, wearing gloves, hands full)
- Importance of interface to user
- Frequency of use of the system
- Implications of system failures

Personas

For each major user group, create a fictional person that captures the most important aspect of users in that group

Give the personas names and detailed information about skills, personalities, and attitudes

Encompass average and extreme users in the group

User observation/field studies (2)

Benefits

- View what users actually do in context
- Direct observation: focus attention on specific areas of interest
- Indirect observation: captures activity that would have gone unrecorded or unnoticed

Drawbacks

- Observation can be obtrusive
- Observer effect
- Co-operation of users is vital, so the interpersonal skills of the observer are important
- Notes and videotapes need to be analysed *by the note-taker*:
  - time consuming
  - prevents the task being split up for analysis by a number of people.

Analyse context of use (again – for *this* stage)

Collect and agree detailed information about

- Who are the intended users
- What are their tasks?
  - Why will they use the system? What is their experience and expertise?

What are the technical and environmental constraints?

- What types of hardware will be used in what organisational, technical and physical environments?

Essential input to requirements and the planning of other usability methods

May be collected at an early stage during planning and feasibility, or in more detail as part of the usability requirements.

Benefits

- Ensure that all factors that relate to use of the system are identified before design work starts.
- Provide a basis for designing later usability tests

Focus groups

Informal assembly of users whose opinions are requested about a specific topic

The goal is to elicit perceptions, feelings, attitudes, and ideas of participants about the topic

Focus groups are generally NOT appropriate for evaluation

Benefits

- Individuals come together and express diverse views on the topic
  - find the range of views, but also for the participants to learn from each other, and to generate a sense of social cohesion.

Brainstorming…

See earlier

Evaluate existing system

Evaluation of an earlier version or competitor system to identify usability problems and to obtain **baseline measures** of usability

Benefits

- Identifies problems to be avoided in the design of the new system
- Provides measures of effectiveness, efficiency and satisfaction which can be used as a baseline for the new system

Method

- Select the most important tasks and user groups to be tested (based on the context of use study). If possible, evaluate the system using the method for usability testing

Card sorting

Discover the **latent structure** in an unsorted list of statements or ideas

The investigator writes each statement on a small index card and requests six or more informants to sort these cards into groups or clusters, working on their own

The results of the individual sorts are then combined and if necessary analysed statistically

Benefits

- If the informants are representative of the user population for whom the application is being designed, then the result will reflect the structure in which the users expect the ideas or concepts should be presented

Affinity diagramming

Sort large amounts of data into logical groups

Existing items and/or new items identified by individuals are written on sticky notes which are sorted into categories as a workshop activity

Affinity diagramming can be used to

- analyse findings from field studies
- identify and group user functions as part of design
- analyse findings from a usability evaluation

Affinity diagramming: Summary

Benefits

simple and cost effective technique for soliciting ideas from a group and obtaining consensus on how information should be structured

Method: Planning

Arrange a meeting of participants with the relevant expertise that will last one to two hours

Write any existing items on sticky notes.

Use a room where you can fix flip chart paper to the wall using Blue Tack

Different colours of sticky notes

Scenarios of use (Use cases) 1

Specify **how** users carry out their tasks in a specified context

Provide **examples** of usage as an input to design

Provide a **basis** for subsequent **usability testing**

User- and task-oriented **use cases**

Scenarios of use (Use cases) Benefits

Encourages **designers** to consider the characteristics of the intended **users**, their **tasks** and their **environment**

Usability issues can be explored at a very **early stage** in the design process (before a commitment to code has been made)

Can help identify **usability targets** and likely task completion times

Promotes **developer buy-in** and encourages a UCD approach

Scenarios can also be used to generate **contexts for evaluation studies**

Only **minimal resources** are required to generate scenarios

Can be used by developers with little or no human factors expertise

Scenarios of use (Use cases) Method

- An experienced moderator is recommended for the sessions in which the scenario is explored
- Gather together the development team and other relevant stakeholders
- Identify intended users, their tasks and the general context
  - This information will provide the basis for the scenarios to be created by the development team
- Functionally decompose user goals into the operations needed to achieve them
- Consider which activities should be performed by the user and which by the computer

Scenarios of use (Use cases) Methods

Create an outline of the users' activities, goals and motivations for using the system being designed, and the tasks they will perform

To maintain design flexibility, scenarios **should not** specify what product features are used

Assign task time estimates and completion criteria as usability targets

The session can be videotaped for later review or transcribed for wider distribution

The results from scenario building sessions can be used to plan user-based evaluations

3 out of 6: Planning and feasibility – Requirements – Design – Implementation - Testing and measurement - Post release

Design guidelines

Create and develop design ideas using multidisciplinary input.

If necessary allocate tasks between humans and machines

Visualise design ideas using sketches, models and simulation/dynamic prototypes

Consider using parallel design.

Evaluate design ideas with a few typical users. Get them to carry out typical simulated/real tasks, using methods that may include storyboarding or wizard of oz.

Expert or heuristic evaluation may also be used.

Feed the results back into the design process quickly.

Iterate the process of design - evaluation until design objectives are fulfilled.

Design Guidelines

"Visibility, Feedback, Constraints, Consistency, Affordances"

Heuristic rules (e.g., from B. Shneiderman or from J. Nielsen)

Design guidelines for the Web

Accessibility guidelines

…

Design Guidelines for the Web

- Site Structure and content
  - What information content does the user need at what level of detail?
  - Use terminology familiar to the user
  - Card sorting to design appropriate structure
- Support Navigation
  - Follow conventions, consistency
  - Home links, buttons, colour for links
  - Meaningful page titles(!)

Design Guidelines for the Web

Page Design

- Good home page, no scrolling
- Minimize file sizes (min. load time)
- Set image dimensions (easier page rendering)
- No flashing, no animation, no sound on loading (!)
- Check how it prints on “US Letter” and on A4 paper sizes

Accessibility guidelines: Web

In detail: http://www.w3.org/TR/WAI-WEBCONTENT/full-checklist.html

Images & animations: Use **alt** attribute to describe each visual

Image maps. Use the client-side **map** and text for hotspots

Multimedia. Provide captioning and transcripts of audio, and descriptions of video

Hypertext links. Use text that makes sense when read out of context. For example, avoid "click here"

Page organization. Use headings, lists, and consistent structure. Use **CSS** for layout and style where possible.

Graphs & charts. Summarize or use the **longdesc** attribute

Scripts, applets, & plug-ins. Provide alternative content in case active features are inaccessible or unsupported

Frames. Use the **noframes** element and meaningful titles

Tables. Make line-by-line reading sensible. Summarize

Paper prototyping

(1)Clarify requirements and (2)Enable draft interaction designs and screen designs to be rapidly simulated and tested

Up to 4 stages

- **concept design**: to explore different metaphors and design strategies
- **interaction design**: to organise the structure of screens or pages
- **screen design**: for initial design of each individual screen
- **screen testing**: to refine the screen layout

Paper prototyping: Summary

Benefits

- Usability problems can be detected at a very early stage in the design process before any code is written
- Communication between designers and users is promoted
- Paper prototypes are quick to build and refine
  - rapid design iterations
- Only minimal resources and materials are required(!)

Heuristic evaluation

Specialists judge whether each element of a user interface follows a list of established usability heuristics

- Usually 2–3 analysts evaluate system, noting down their observations and often ranking them in order of severity
  - The analysts are usually experts in human factors or HCI
  - But ‘non-experts’ have also been shown to report valid problems
- Can be conducted at various stages of the development
  - preferable to have already performed context analysis to help experts focus on actual / intended usage
  - beneficial on early prototypes before actual users are brought in to help with further testing
- Usability problems found are normally restricted to aspects of the interface that are reasonably easy to demonstrate: use of colours, lay-out and information structuring, etc.

Expert Evaluation – Guidelines (1)

From Shneiderman (*Designing the User Interface*):

1. Strive for consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialogues to yield closure
5. Offer simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Expert Evaluation – Guidelines (2)

From Nielsen:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Heuristic evaluation: Summary

Quick and relatively cheap feedback to designers; good ideas for improving UI

Good estimate of how much the user interface *can* be improved

Problems found by inspection methods and by performance measures overlap to some degree, although both approaches *will find problems not found by the other*

NOTE: The method can seem overly 'critical'

- designers may only get feedback on the problematic aspects of the interface
- method is normally not used for the identification of the 'good' aspects

Parallel design

Alternative designs, often interface designs, created by 2–4 design groups at the same time

The design groups work independently of each other

- No communication until after each presents a draft design in a design workshop

Final design may be one of the designs or a combo of designs, taking the best features from each

Limitations

- Requires a number of design team members to be available at the same time to produce the concepts
- Requires a lot of time to be invested over a short period for the design work to be carried out
- Time must be allocated to compare parallel design outputs properly

Parallel design (2)

Benefits

- Might *seem* like an expensive approach
  - BUT the many ideas are generated *without implementing* them
  - Allows a range of ideas to be generated quickly and cost effectively
- Parallel nature of the approach allows several approaches to be explored at the same time, thus compressing the concept development schedule
- Final solution can have benefits from *all* ideas proposed
- Only minimal resources and materials required to convey product feel
- The technique can be utilised by those with little or no human factors expertise

Parallel design – Method

- Define the boundaries for the parallel design
  - Goal of system, tasks that it should support, user characteristics, etc
  - Each design team should receive the same set of requirements before start
  - Agree on the criteria by which the designs will be assessed
- Design teams should have roughly equivalent skills
- Each design teams may use whatever media they prefer to present their designs
  - Low level of prototyping is recommended
  - No extra points should be given for ‘sophisticated’ prototypes

Parallel design – Method (2)

- Decide beforehand how much time to allocate to the design work
  - set a clear time limit. 10–20 hours per group is often sufficient
- Allow sufficient time to carry out a fair comparison of the designs produced
  - Design workshop
  - Discuss each design separately
  - Then discuss how different aspects of the designs may be combined

Storyboarding

Low fidelity prototype consisting of a series of screen sketches

Used by designers to illustrate and organize their ideas and obtain feedback

Particularly useful for multi-media presentations

Benefits

- Provides an overview of the system
- Demonstrates the functionality of the storyboard elements
- Demonstrates the navigation scheme
- Can check whether the presentation is accurate and complete
- Can be evaluated by users

Storyboarding – Method

Use context of use and scenarios as input

Brainstorm ideas, this may include lists, charts, doodles, and quick notes

Select the best ideas

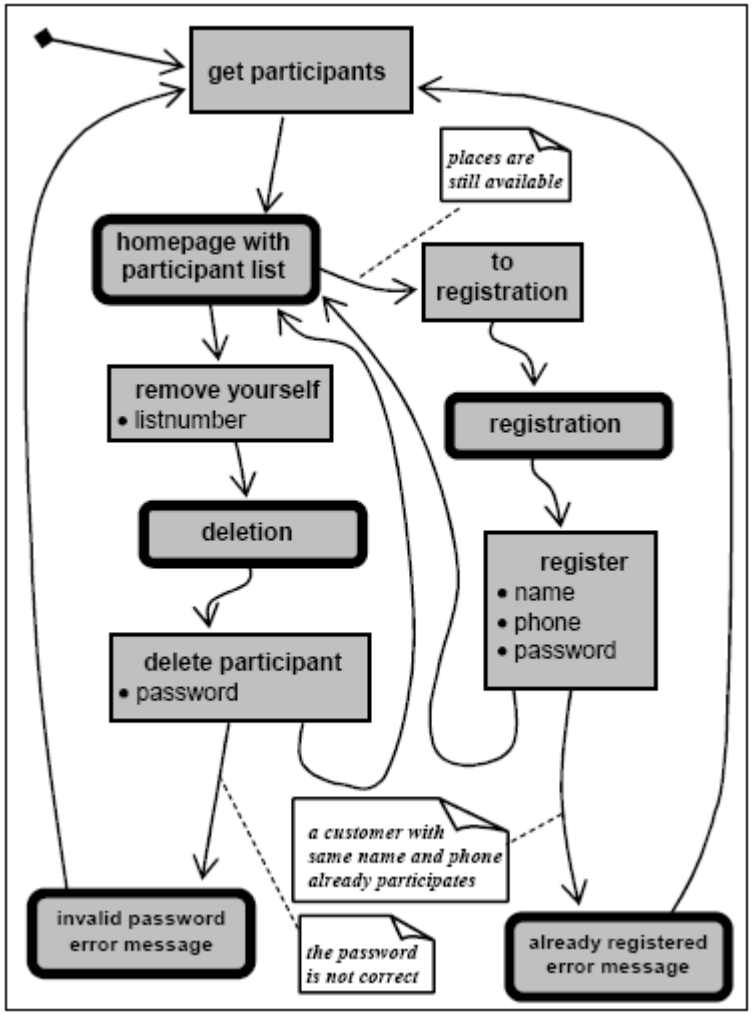
- re-consider the project requirements, time&resource constraints, and the target audience and end users

Select the top ideas and try to get feedback from others involved

Sketch each screen, and describe any pictures, images, animations, sound, music, video or text

Storyboarding

State diagrams can also be used to illustrate the interface



Evaluate prototype

Participative user-based evaluation of a paper or machine prototype to identify usability problems, where the user is probed to explain their expectations and problems

Storyboards or wizard of oz prototypes can also be evaluated

Benefits

- Potential usability problems can be detected at an early stage before development is complete
- A deeper understanding of the users' expectations and impressions of the system

Evaluate prototype – Method

Before

- Select most important tasks and user group(s) to be tested (e.g. the most frequent or the most critical); 3–5 users are sufficient to identify the main issues
- Consider using user-defined tasks
- Produce task scenarios and input data and write instructions for the user
- Invite developers to observe the sessions if possible OR videotape the sessions

Evaluate prototype – Method

During

- **Do not** give any hints or assistance unless the user is unable to complete the task
- Observe the interaction and note any problems encountered
- For a paper prototype, as the user selects options on each screen, the designer explains what happens, and either points to the next screen or presents the next screen to the user
- The user may be prompted for their impressions of a page design, what they think different elements may do, and what they expect the result of their next action to be. The user may also be asked to suggest how individual elements could be improved
- Interview the user to gain general opinions, and to ask about specific problems encountered

Evaluate prototype – Method

After

- Produce a list of usability problems, categorised by importance (use sticky notes to sort the problems), and an overview of the types of problems encountered

Wizard of Oz

Enables unimplemented technology to be evaluated by using a human to simulate the response of a system

Method

- The "wizard" sits in a back room, observes the user's actions, and simulates the system's responses in real-time
- For input device testing the "wizard" will typically watch live video feeds from cameras trained on the participant's hand(s), and simulate the effects of the observed manipulations
- Often users are unaware (until after the experiment) that the system was not real
- The "wizard" has to be able to quickly and accurately discern the user's input, which is easiest for simple for voice input or hand movements. The output must also be sufficiently simple that the "wizard" can simulate or create it in real time

Wizard of Oz: Summary

Benefits

- This technique can be used to test device concepts and techniques and suggested functionality before it is implemented
- Find out the kinds of problems people will have with the devices and techniques
- Investigate aspects of the products form such as visual affordance (whether the product shows how it can be used)

Task allocation

Allocation of tasks between humans and machines

Identify which tasks can only be allocated to either the machine or human (mandatory allocation)

Then provisionally allocate tasks on either a permanent and dynamic basis

- Provisional allocation should then be evaluated and revised if necessary

Benefits

- Tasks should be allocated to humans and machines in a way that best combines human skills with automation to achieve task goals, while supporting human needs.

Task allocation – Method

Before

Context analysis and task analysis

- task structure and demands, the knowledge needed to perform the tasks, environmental constraints, functional and safety requirements etc.

Mandatory allocation

- to humans due to technical infeasibility or ethical or safety considerations
- to machines due to demands exceeding human capabilities or a hostile environment

Task allocation – Method (2)

Provisional allocation

- *Permanently allocate* tasks based on factors such as task criticality, cost, training or knowledge requirements, or task unpredictability
- *Dynamically allocate* tasks based on factors such as human workload, the need for cognitive support, individual differences in users, changing capacity of the user, or learning

Jobs must be designed from the tasks based on factors such as responsibility, task variety, interference between and within tasks, user communication, and individual capability

More on Provisional allocation

The provisional allocations and jobs should be evaluated based on factors such as: **safety**, system performance, usability, cost, job **satisfaction** and human well-being, **acceptance** by users, management and **society** and **social impact**. The evaluation findings should be used to review and revise the provisional allocations which should then be re-evaluated.

4 out of 6: Planning and feasibility – Requirements – Design – Implementation - Testing and measurement - Post release

Style guides and design guidelines

Ensure that rapid prototyping activities incorporate usability

Style guides

Style guides are used to provide a consistent look & feel

Should be defined as part of usability requirements

Conformance should be monitored during development

How

- Handbooks (e.g., by A. Cooper)
- Style Guides (e.g., from Microsoft)
- Design Principles (e.g., from J. Nielsen)

Benefits

- Style guides embody good practice in interface design
- Following a style guide will increase the consistency between screens
- Using a style guide can reduce the development time
- Following general usability guidelines will improve the quality of the interface

Prototyping

Paper prototyping

Creation of an artefact that will eventually be discarded rather than becoming part of the final delivered software

Benefits

- Quick
- Possible to evaluate many designs

More on Paper

Paper Prototyping

- Why create paper prototypes?
  - To find usability problems early
    - They help you gather usability data as early as possible
  - They're easy, simple, fast, and cheap to create (a *low-fidelity* prototype)
  - They can usually be created faster than software equivalents (including HTML)

Paper Prototyping

Why create paper prototypes (cont'd)?

- They're a great way of visualizing a design
- They help you describe that design to others
  - They let you visualize different possible designs
- Sample users are willing to suspend their disbelief
- Users know it's not the final version, so they don't feel bad being critical about it

Paper Prototyping

Why create paper prototypes (cont'd)?

- Allows technical and non-technical people to collaborate on the interface design (development, marketing, support, documentation, training)
- More than one person can work on the paper design at once
- Can mix screenshots of an existing system with paper prototypes to test changes to the system

Paper Prototyping

Why don’t people use them paper prototypes more often?

- Too simple, too fast, too obvious?
- Too “low-tech”?
- People don’t think they’ll get much out of a method that’s so cheap
- It feels like you’re cheating!

Paper Prototyping

How to use paper prototypes

- Create the design(s) on paper
- Test different designs with different users, with you pretending to be the computer
  - The user uses a pen as a mouse and writes into textboxes
  - You emulate “drop down” menus, select tabs, display different windows, display data in labels, etc.
  - The “computer” isn’t allowed to give hints to the user
- Evolve the designs based on user feedback, sometimes immediately

Paper Prototyping

What's in a paper prototyping "kit"?

- Paper ☺, transparency film, markers, scissors, glue, sticky notes
- Printouts of buttons, textboxes, dropdown menus, tabbed sheets, dialog box outlines, etc.

Paper Prototyping

When shouldn’t you use paper prototyping?

- When evaluation of detailed screen elements is required
- When you’re in a late stage of the design cycle

5 out of 6: Planning and feasibility – Requirements – Design – Implementation - Testing and measurement - Post release

Diagnose usability problems

user-based methods such as participatory evaluation, diagnostic evaluation, and critical incident analysis should be used when possible, supported by questionnaires to assess attitudes

these can be supplemented by expert or heuristic evaluation.

These methods should be used to improve early machine prototypes.

Evaluate if usability objectives have been achieved

requirements for user performance and satisfaction can be evaluated by use of performance testing, cognitive workload and attitude questionnaires.

other usability objectives can be assessed by expert evaluation.

Formative vs. Summative Evaluation

- Formative evaluation
  - When the interface is being developed
  - Can test how usable the interface is
  - Use the feedback to modify the interface's design and make it more usable
- Summative Evaluation
  - After the interface is complete or nearly complete
  - Little opportunity to make the system more usable
    - Much harder to make changes at this point
  - Still useful for comparing two existing products with one another

Formative Evaluation

- Should begin as early as possible
  - Time to make major design modifications
    - Try to have something that can be evaluated 10% of the way through the project!
- Aim for an average of 3 major cycles of formative evaluation, each followed by an iterative redesign
  - Most of the helpful usability information will come from the first cycle

Kind of Data Obtained

- Quantitative
  - usability information vs. your usability specification
    - number of user errors
    - time to complete tasks
- Qualitative feedback
  - open-ended tasks: suggestions about what can be improved

Logs, Notes, Video, Audio

- Track mouse clicks and keypresses
  - Sometimes it's hard to process this later
- Writing notes
- Video, Audio
  - Take very long time to analyze (>> rec. length)
    - Notes with the **time codes** to review help
  - Video: good backup of events
  - Video clip: good demo of a problem

Participatory evaluation

Evaluation of a paper or machine prototype to identify usability problems

User is probed to explain their expectations and problems

Users (*participants,* old: *subjects*) are an essential part

- See on selecting participants later

Diagnostic evaluation

User based evaluation of a *working system*

- primary objective is to identify usability problems
- 3-5 users, 8: better results, more: complex systems
- most frequent/important tasks, tasks from specification
- Protocol: hints? questions? Speaking aloud when doing tasks?

Benefits

- Major usability problems are identified
- An understanding is gained of why the user has difficulties with the system
- Approximate measures can be obtained for the users' effectiveness, efficiency and satisfaction

Critical Incident Technique Analysis

CIT: end users asked to identify specific incidents they experienced personally and which had an important effect on the final outcome

- Emphasis is on incidents rather than vague opinions
- The context of the incident may also be elicited

Data from many users is collected and analysed

Critical Incident Technique Analysis

Method

- Get a subjective report
  - minimize interference from stereotypical reactions or received opinions
- User is asked to focus on one or more critical incidents which they experienced personally
  - A critical incident is defined as one which had an important effect on the final outcome
  - Critical incidents can only be recognised retrospectively
- Then Content Analysis technique to summarise the experiences of many users or many experiences of the same user

Critical Incident Technique Analysis

Benefits

- Open-ended retrospective method of finding out what users feel are the critical features are
- More flexible than a questionnaire or survey
- Recommended in situations where the only alternative is to develop a questionnaire or survey from the start
- Focuses on user behaviour, can be used in situations where video rec. is not practicable

Subjective Assessment (testing & post-release)

Tells the evaluator how the users **feel** about the software being tested

- Distinct from how efficiently or effectively they perform with the software
- The usual method of assessment: standardised opinion questionnaire
  - to avoid criticisms of subjectivity

Subjective Assessment (testing & post-release)

Benefits

- In a *discretionary* use scenario, user satisfaction is most probably the largest single key factor which will influence the users' decision whether or not to continue with the software
  - other factors: price, technology, brand loyalty
- In a *mandatory* use scenario, poor satisfaction leads to absenteeism, fast staff turnover, and unrelated complaints from the workforce
- Subjective Assessment complements data from efficiency and effectiveness measures
- Usually produces a list of satisfying and unsatisfying software features which is especially useful if testing is taking place during development(!)

Survey Evaluation (Questionnaires)

Open questions (Can you suggest any improvements?)

Closed questions (How useful is this particular feature?)

- Checklists
- Multipoint scales, including Likert Scale
- Semantic differential scale
- Ranked order

GOOD

- cheap to administer to a large number of users
- easy to analyze - unless unstructured responses are allowed

BAD

- time and skill required to develop the questionnaire (or commercial set can be used – but this is expensive)
- will only uncover what is looked for

Survey Evaluation - Interviews

Interview users of the system

Can be structured (planned list of things to ask),

or unstructured (topics to cover, but no fixed sequence)

GOOD

- can obtain in-depth response from user
- can enable new issues to emerge

BAD

- time consuming (expensive)
- requires training and skill to carry out

Heuristic evaluation

Use heuristic rules

Quick and (relatively) cheap

Expert evaluation is similar, but does not use specific heuristics

Performance Testing/Usability Testing

Rigorous usability evaluation of a working system under realistic conditions

- Identifies usability problems; compares measures such as success rate, task time and user satisfaction with requirements

Usability Testing

- Quantitatively measure usability for the different iterations of the interface
- Can be based on usability goals

Usability Characteristics

What to usability characteristics could be measured?

Common ones are:

- Initial performance
- Long-term performance
- Learnability
- Retainability
- Advanced feature usage
- First impression
- Long-term user satisfaction

Performance testing

Method

- Select the most important tasks and user groups to be tested
  - (e.g. the most frequent or the most critical)
- Select users who are representative of each user group
  - 3-5 users, 8: better results / more reliable measures.
- Produce a task scenario and input data and write instructions for users

Performance testing

Benefits

- Major usability problems are identified that may not be revealed by less formal testing
- Measures can be obtained for the users' effectiveness, efficiency and satisfaction

Analytic Evaluation

An analysis of a definition of the user interface, sketched in a natural language or some semi-formal language, e.g., Command Grammar Language or GOMS

GOOD

- no need to build prototype
- no need to arrange user testing

BAD

- time consuming
- requires specialists with background in psychology
- doesn’t tell us anything about errors or learning behaviour

Analytic Evaluation (2)

Cognitive Walkthrough

- A type of analytic evaluation
- A check for identified psychological criteria during a “walkthrough”
- Evaluate how well the designed software supports the user in learning to use it
- Performed by expert in cognitive psychology as applied to interface design

GOMS (Goals, Operators, Methods, and Selection rules)

Eyes/ears perceive information

Information enters perceptual processor

Information enters the visual/auditory image store

Information is stored in the working memory and long term memory

Information is analyzed in the cognitive processor and a desired reaction (motor function) is chosen

Desired motor function is activated in the motor processor

Desired motor function is applied by user's body

GOMS MTM (Methods-Time Measurement)

Eye fixation = 230[70, 700] milliseconds

Eye movement = 30 milliseconds

Perceptual Processor = 100[50, 200] milliseconds

Cognitive Processor = 70[25, 170] milliseconds

Motor Processor = 70[30, 100] milliseconds

- Can also apply Fitts' Law

KLM (Keystroke Level Model)

Simpler and faster to use than GOMS

Average times as measured by Card, Moran and Newell:

- Press a key or button
  - Best typist = .08 seconds
  - Good typist = .12 seconds
  - Average skilled typist = .20 seconds
  - Average non-secretary = .28 seconds
  - Typing random letters = .50 seconds
  - Typing complex codes = .75 seconds
  - Worst typist = 1.2 seconds

Point with a mouse (excluding click) = 1.1 seconds

Move hands to keyboard from mouse (or vice-versa) = .4 seconds

Mentally prepare = 1.35 seconds

Experimental Evaluation

Utilizes the scientific method with a controlled experiment (i.e., testing an hypothesis by measuring attributes of subject behaviour)

Includes

- Testing of hypothesis
- Independent variables (varied by experimenter)
- Dependant variables (performance measurements)
- Controlled variables (fixed by experimenter)
- Can the hypothesis be stated in a way that can be tested?
- Statistical analysis to check reliability of results
- Pilot studies

Experimental Evaluation (2)

GOOD

- reliable results

BAD

- need specialist knowledge
- resources needed to set up experiment
- can't be used for every design decision
- works best for narrow questions

Participants

Formerly “subjects”

Participants should match the user population

- Age
- Education
- Experience with computers
- Experience with systems of that type
- Experience of the task domain

Generally at least 10 participants required

Independent / Dependent Variables

- Independent variables
  - Manipulated through the design of the experiment; e.g.,
  - interface style (e.g. GUI vs command-line)
  - level of help (e.g., tool tips vs F1)
  - number of menu items (e.g., 4, 8, 16)
  - icon design (e.g., static vs. animated)
- Dependant variables
  - Performance measurements; e.g.,
    - time to complete a task
    - number of errors made
    - user preferences
    - quality of users performance
  - Must be measurable
  - Must be effected by the independent variable
  - As far as possible, must be unaffected by other factors

Hypothesis

A prediction of the outcome of an experiment.

States that variation in the independent variable will cause a difference in the dependent variable.

Experiment is designed to disprove the null hypothesis (i.e., that there is no difference in the dependant variable between levels of the independent variable)

Experimental Design

Beyond the scope of this course…

in real-life, an expert is hired to design and perform the experiment

But, to whet your curiosity:

between-groups vs. within-groups
  ordering of conditions in each (e.g. *Latin square*)

Outliers

Pilot testing

statistics & statistical analysis (*T-tests*, *ANOVA*s, etc.)

interpreting results

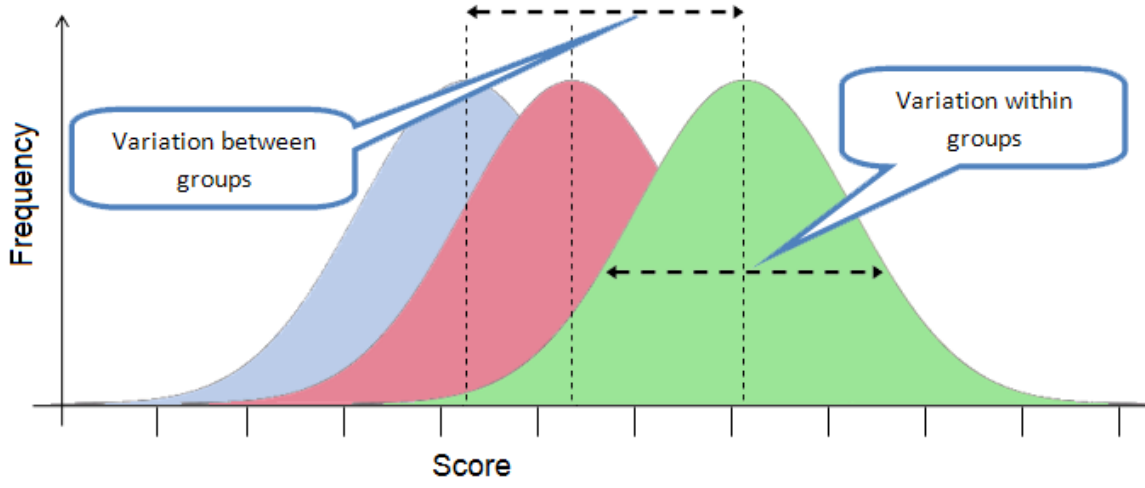
Science!

Data Analysis: ANOVA etc.

How much of the variation due to chance

ANOVA report (e.g., from Excel) “F(2,11) = 6.2, p < 0.05” means

- Variation bw groups 6.2 times larger than within
- More than 19/20 probability it’s not due to chance



6 out of 6: Planning and feasibility – Requirements – Design – Implementation - Testing and measurement - Post release

Monitor the usability of the system after release to ensure that it meets user needs in the field

Can be used as an input to requirements for a new version or release

Techniques for collecting user feedback

- post release tesing
- questionnaires to survey user satisfaction
- remote testing
- analysis of help desk calls
- observing users

Post release testing and measurement

- Measurement involves sampling
  - Small subset of a large and usually indefinable population
- During development, user sample sizes are extremely small
  - order of 3 to 10 users
  - Testing procedures will usually focus on a small number of user tasks
    - BUT Product will often support large number of tasks
  - Task sample sizes will also be extremely small
    - Lack of confidence in measured values
- Can fix this later, with more users!

Remote evaluation

Method where the evaluator and user participant are not in the same location

- **Moderated,** with the evaluator observing the participant in real time
- **Automated** or **unmoderated** with the participant working without direct observation or

Wide number of detailed methods that collect a range of data

- One extreme, there is little difference from in-person *task*-based lab testing, except that the *moderator* and participant are not in the same place
- Other extreme, there are no user tasks at all, and the data collected is aggregated analytics

Remote evaluation

Qualitative (moderated) methods using remote screen-sharing and audio: a participant and moderator work together in real time

- Adobe Connect, GoTomeeting, NetMeeting, LiveLook, UserVue, Skupe, WebEx, Glance

Quantitative (unmoderated)

- Testing on live sites/apps. Tools include UserZoom, RelevantView, WebEffective
- Testing wireframes. Tools include Chalkmark, Usabila
- Testing conceptual artifacts. Tools include online *card sorting*, OptimalSort, WebSort

Quantitative

- User analytics on live sites. Tools include ClickTale, ClickHeat
- A/B/C testing on live sites
- Surveys

User observation/field studies

Involve an investigator viewing users as they work in a field study, and taking notes on the activity that takes place

- Direct
  - investigator is actually present during the task
- Indirect

Planning & Feasibility	Requirements	Design	Implementation	Test & Measure	Post Release
Getting started	User Surveys	Design guidelines	Style guides	Diagnostic evaluation	Post release testing
Stakeholder meeting	Interviews	Paper prototyping	Rapid prototyping	Performance testing	Subjective assessment
Analyse context	Contextual inquiry	Heuristic evaluation		Subjective evaluation	User surveys
ISO 13407	User Observation	Parallel design		Heuristic evaluation	Remote evaluation
Planning	Context	Storyboarding		Critical Incidence Technique	
Competitor Analysis	Focus Groups	Evaluate prototype		Pleasure	
	Brainstorming	Wizard of Oz			
	Evaluting existing systems	Interface design patterns			
	Card Sorting				
	Affinity diagramming				
	Scenarios of use				
	Task Anaysis				
	Requirements meeting				