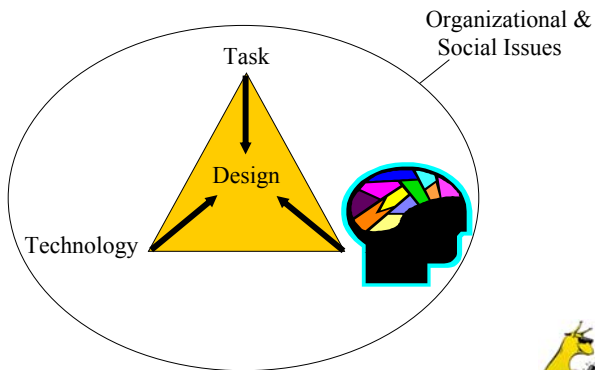


## HCI Foundations: Design



## 1. Different Design Approaches

- ▶ System-centred design
  - What can be built easily on this platform?
  - What can I create from the available tools?
  - What do I (programmer) find interesting to do?
- ▶ Task-centred design
  - structured around specific tasks the user will want to accomplish with the system being developed.
  - Able to get users but not throughout the process.
  - Tasks are chosen early in the design effort to:
    - ▶ Raise issues about the design
    - ▶ Aid in making design decisions
    - ▶ Evaluate the design as it is developed



## 1. Different Design Approaches

- ▶ Goal-centered design
  - Articulates users' goal rather than tasks
  - Tasks change with the technology and can contradict goals. Goals seldom change
  - May result in different task / task sequence which could be better
- ▶ User-centered design
  - Design is based upon a user's abilities and real needs, context, work, tasks; users are involved throughout
- ▶ Participatory design (one of UCD) → Scandinavian, union-agreed model
  - Users are active collaborators in the design process
  - Users are considered subject matter experts



## 1.1. Task Centered Design

- ▶ Phase 1: Identification
  - identify specific users
  - articulate example realistic tasks
- ▶ Phase 2: Requirements
  - decide which of these tasks and users the design will support
- ▶ Phase 3: Design
  - base design representation and dialog sequences on these tasks
- ▶ Phase 4: Walkthrough Evaluations
  - using your design, walk through these tasks to test the proposed interface



## 1.2. Phase 1: Identification

- ▶ Get in touch with real people who will be potential users of your system
- ▶ Spend time with them discussing how the system might fit in, observe them
- ▶ Learn about the user's tasks
  - Articulate concrete, detailed examples of tasks they perform or want to perform that your system should support
  - Ask users to verify



## 1.2. Phase 1: Identification

- ▶ Good task examples
  1. Are very specific
    - say exactly what the user wants to do
    - specify actual inputs and outputs
    - do not assume anything
    - can be used to compare different design alternatives in a fair way
  2. Are real
    - illustrate functionality in user's real work context
    - reflect user's real task interest and needs



## 1.2. Phase 1: Identification

### ▶ Good task examples

#### 3. Describes a complete job

- not just a list of simple things the system should do!
- does more than present a sub-goal independent of other sub-goals
- forces designer to consider how interface features will work together
- contrasts how information input and output is carried through the dialog
  - ▶ where does information come from?
  - ▶ where does it go?
  - ▶ what has to happen next?



## 1.2. Phase 1: Identification

### ▶ Good task examples

#### 5. Are evaluated by users

- Have gone through user's omissions, corrections, clarifications and suggestions

#### 6. As a set, identifies 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 or user with special need = unexpected or specialized tasks



## 1.3. Phase 2: Requirements

### ▶ Which user types will be addressed by the interface?

- designs can rarely handle everyone so
- why are particular users included / excluded?

### ▶ Which (sub-) tasks will be addressed by the interface?

- designs can rarely handle all tasks so
- requirements listed in terms of how they address tasks (must, should, could)
- discussion includes why items are in those categories



## 1.4. Phase 3: Design as Scenarios

### ▶ Develop designs around how well they fit users and specific tasks

### ▶ Use tasks to

- get specific about possible designs
- consider how design features work together to help a person accomplish real work
- consider the real world contexts of real users

### ▶ Reconsider how a design scenario handles each task

- what the user would do and see in each step of using the system to perform a task



## 1.5. Phase 4: Walkthrough Evaluation

### ▶ Good for debugging an interface

### ▶ Process:

#### 1 Select one of the task scenarios

#### 2 For each user's step/action in the task:

- Does the story represent all possible actions a user will do?
- Can you rely on user's expected knowledge and training about system?
- If the answer of any of the above is no:
  - ▶ then you've located a problem in the interface
  - ▶ once a problem is identified, assume it has been repaired
- go to the next step in the task



## 1.5. Walkthrough evaluation

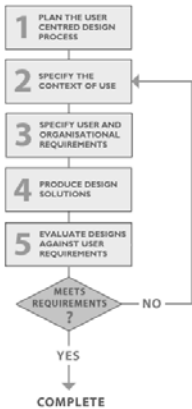
### Task example 1:

Fred, a regular in the store, wants to buy two bags, pays cash, and wants them delivered to his home next week.

The screenshot shows a web form titled 'Cheap Shop Catalog Store'. It is divided into two main sections: 'Purchaser' and 'Catalog Item'.  
The 'Purchaser' section includes fields for Name, Phone, Postal Code, Province, City, Delivery Address, Today's date, and Credit Card No. (with a note 'for dept use: validation id').  
The 'Catalog Item' section includes fields for Number, Quantity (with a dropdown arrow), Cost/Item, and Total.  
At the bottom, there are two buttons: 'Next Catalog Item (PF5)' and 'Trigger Invoice (PF8)'.  
A 'Balance Owing' field is also present at the bottom left.



## 2.1. ISO 13407 – Human-centred design process



## 2.1. User-Centered Design

### Up side of UCD

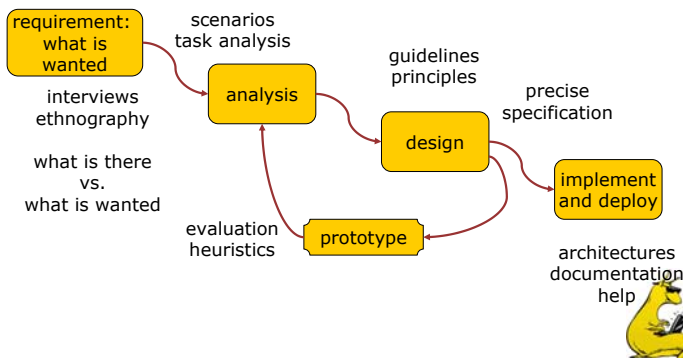
- ▶ users are excellent at reacting/commenting
- ▶ users bring in important "folk" knowledge of use context
  - knowledge may be otherwise inaccessible to design team
- ▶ greater buy-in for the system often results

### Down side of UCD

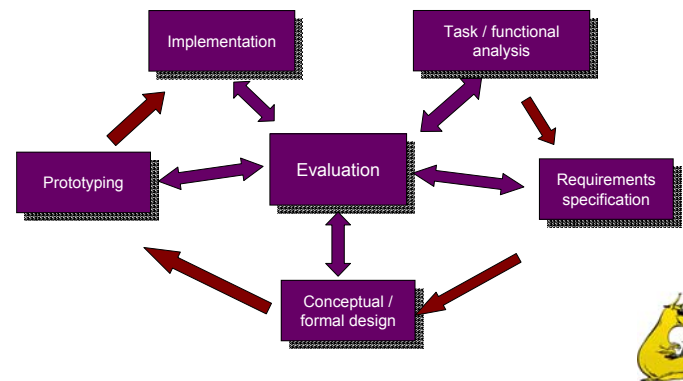
- ▶ hard to get a good pool of users (cost, reluctance)
- ▶ users are not expert designers
  - don't expect them to come up with design ideas from scratch
- ▶ the user is not always right
  - don't expect them to know what they want



## 2.1 Traditional design sequence



## 2.2. Star method of UCD



## 2.2 Star method

- ▶ **Components:**
  - Requirements - finding out what people need from the system
  - Evaluation - checking that you've got it right
  - Conceptual design - creating the overall idea for the new system
  - Physical design - filling in the details of what the new system will be like and how it will work
  - Prototyping and envisionment - bringing ideas to life
- ▶ **Features of star method:**
  - Evaluation is central to everything
  - Activities can happen in 'non-orderly' manner



## 2.2. Re-decorating your room

- ▶ **Requirements** - I need a space to work in. I want to get rid of some clutter. I want the room to be lighter, fresher, cleaner...
- ▶ **Conceptual design** - need to create an area for working in; need to build a cupboard to store things in; paint the walls a lighter color.
- ▶ **Physical design** - a partition up in that corner; that cupboard I saw in Ikea could be used to store things in my flat; I am going to paint the walls 'apple-white'
- ▶ **Envisionment** - look at this model room in this magazine; here's a sketch of my ideas for a cupboard; you know the colour of Rod's bedroom...
- ▶ **Evaluate** - that partition would be too expensive, that cupboard would get in the way; light-colored wall would get dirty very quickly...

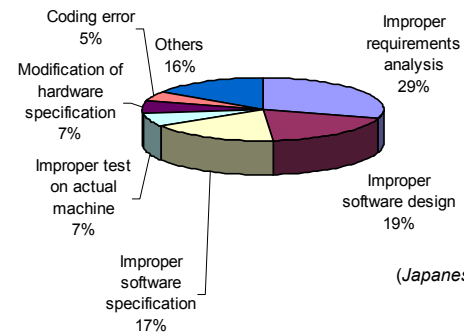


## 2.3. Requirements: Why

- ▶ To understand what we are going to be doing
- ▶ We build systems for others, not for ourselves
- ▶ Requirements definition: the stage where failure occurs most commonly
- ▶ Getting requirements right is crucial
- ▶ Understanding problem space
  - Are there problems with the existing product or way people do things?
  - Why do you think there are problems?
  - How do you think your proposed design ideas might overcome these?
  - How will the proposed design extend or change current products or ways of doing things?



## 2.3. Requirements: Development costs



(Japanese METIA, 2004)



## 2.3. Requirements

### Functional:

- ▶ Historically requirements
- ▶ Features, functions that the system can/cannot do

### Non-functional

- ▶ 'The other issues'
- ▶ "-ilities" (quality, accessibility, evolveability, flexibility, etc.)
- ▶ Constraints
- ▶ Cost, aesthetics, etc.
- ▶ Usability requirements



## 2.3. Typical Real-World Constraints

- ▶ Elapsed time to market
- ▶ Cost/effort to design and implement
- ▶ Size/footprint/weight/power/price
- ▶ Computer power/memory (related to cost and power)
- ▶ Consistency with overall product line
- ▶ Backward compatibility
- ▶ Differentiation from competitive products



## 2.3. NF Requirements

- ▶ **Accessibility** = ensuring that people with special needs are not at a disadvantage.
- ▶ **Usability** = quality of the interaction in terms of parameters such as time taken to perform tasks, number of errors made and the time to become a competent user.
- ▶ **Acceptability** = fitness for purpose in the context of use. It also covers personal preferences that contribute to users 'taking to' an artefact, or not.
- ▶ **Engagement** = designing for great, exciting and riveting experiences.



## 2.3. NF Requirement: Usability

- ▶ Basic ideas: humans are emotional, are not interested in putting a lot of effort into, and generally prefer things that are easy to do vs. those that are hard to do. (McQuillen, 2003).
- ▶ ISO 9241 defines usability to consist of three components:
  - *effectiveness*: the accuracy and completeness with which specified users can achieve specified goals in particular environments
  - *efficiency*: the resources expended in relation to the accuracy and completeness of goals achieved
  - *satisfaction*: the comfort and acceptability of the work system to its users and other people affected by its use.



## 2.3. Requirements: Steps

1. Gather data
  - Interviews, observation, ethnographic study, surveys/questionnaires, documentation, immersion, etc
2. Organize data
  - Notes, cards, affinity diagrams, recording, brainstorming, computer tools
3. Represent data
  - Lists, outlines, matrices
  - Scenarios, personas, storyboards, use cases



## 2.3.1 Requirements: Gather

- ▶ Look at competitors → good and bad ideas
- ▶ Investigate: Activities and Artifacts (not just artifacts)
- ▶ Study related processes and objects in the environment that people may use
  - Office environment - papers, whiteboards, ...
  - Phone calling - phone book, note pad, dial, ...
- ▶ Focus on *observable behaviors*
  - What are the practices, methods, steps, objects, ..., used?
- ▶ Learn *what* users do, *why* they do it, *how* they do it, *when and where* they do it, with what *tools* or *people* they do it



## 2.3.1. Ethnography

- ▶ Immerse oneself in situation you want to learn about (has anthropological and sociological roots)
  - Investigating people in their cultural context
- ▶ Has 2 elements:
  - Combine observation, interview and document review
  - Grounded in theory
- ▶ Understanding the user
  - Understand goals and values
  - Understand individual's or group's interactions within a culture
  - Try to make tacit domain knowledge explicit in an unbiased fashion
  - For UI designers: improve system by finding problems in way it is currently being used



## 2.3.1. Ethnography Steps

- ▶ 1.1. Preparation
  - Familiarize yourself with the system and its context
  - Set initial goals and prepare questions
  - Gain access and permission to observe & interview
- ▶ 1.2. Field study
  - Establish rapport with users
  - Observe/interview users in workplace and collect all information (actions, words, environment, other people, interruptions)
- ▶ 1.3. Analysis
  - Compile, analyze and interpret information
- ▶ 1.4. Reporting

Rose et al '95



## 2.3.1. Semi-Structured Interviews

- ▶ Predetermine data of interest - know why you are asking questions – but keep them open (don't lead the users)
- ▶ Plan for effective question types
  - How do you perform task x?
  - Why do you perform task x?
  - Under what conditions do you perform task x?
  - What do you do before you perform...?
  - What information do you need to...?
  - Whom do you use system y with?
  - What do you use to...?
  - What happens after you...?
  - What is the result or consequence of... (or NOT)...?



## 2.3.2. Requirements: Organize

- ▶ Affinity Diagram
  - Write down each quote/observation on a slip of paper
  - Put up on board
  - Coalesce items that have affinity (similar things about an issue)
  - Give names and colors to different groups
  - Continue grouping subgroups → indicate relationships
  - A hierarchy will be formed



### 2.3.3. Req. Represent: Personas

- ▶ A model of key user attributes and goals
- ▶ Distilled from observing/interviewing real people
- ▶ Presented as a vivid, narrative description
- ▶ Of a single "person" who represents a user segment
- ▶ Used to guide the design of products, channels, and messaging
- ▶ Personas need to have goals (what they are trying to achieve w. the product), inclinations, capabilities.
- ▶ Personas represent behavior patterns, not job descriptions
- ▶ It is best to develop a few (not too many) concrete personas who have hard characteristics such as name, computer experience, etc
- ▶ Try to bring the character alive - perhaps include a picture or two

<http://research.microsoft.com/research/coet/Grudin/Personas/Pruitt-Grudin.pdf>



### 2.3.3 Reasons for Personas

- ▶ There is no such thing as an average user
- ▶ A compromise design pleases no-one
  - The broader you aim, the more likely you miss the bulls-eye
  - 50% of the people 50% happy doesn't work: car: soccer mom, carpenter, dot-com exec
  - "Every time you extend functionality to include another constituency, you put another speed bump of features and controls across every other user's road."
- ▶ A targeted design can achieve
  - 10% people 100% ecstatic
  - Examples: pickup truck, paro



### 2.3.3 A family of AOL users





Al, 47, Software developer, visually impaired, wants fast connection  
 Mary, 45, architect, needs GB data transfers, novice Internet user



Charlotte, 16, Nursing student, chats for 2 hrs per day  
 Jane, 2, loves Sesame Street online games  
 Rufus, 10, dog, barks on every animations



### 2.3.3 Sample persona

<p>Mary Jones, 45, architect for SCArc, an urban architecture company based in Santa Cruz</p>  <p><b>Name, face, job</b></p>	<p><b>Formatted as a narrative</b></p> <p>She took Computing and Internet class at Cabrillo College last Fall. When working at home, she normally starts working at 09:00, taking a break 12:00-14:00 to pick up the children, then go back to work until 18:00. Have to handover computer to Charlotte at 18:00 sharp. Most often her remote work involves discussing with clients over skype, sending sketches in jpg (typical size 1GB per file), and updating her office on the discussions.</p>	<p><b>Focused on enabling design decisions</b></p> <p>computer                  Impatient when file transfer is slow                  Bad at multitasking (cannot send file and chat at the same time)</p> <p><b>Key goals:</b>                  Sending 1GB in less than 1 minute                  Reliable connection, especially for video chat                  Secure connection</p> 
--	--	---

### 2.3.4. Req: Represent: Scenarios

- ▶ Scenarios are stories about people undertaking activities using technologies in contexts
- ▶ Most often narrative but can be complemented with pictures
- ▶ Develop conceptual scenarios that cover the main activities that the technology has to support
  - Pete logs onto the computer
- ▶ Develop concrete versions of these for specific designs of the technology
  - Pete clicks on the key icon in the File toolbar



### 2.3.4 Example of a scenario

1. The user selects Add a Note from the menu. A new window appears.
2. From the list box at the top of the window she selects the name of the client.
3. A list of campaigns appears in the list box below, and she selects a particular campaign.
4. A list of adverts appears in the next list box, and she selects a specific advert.
5. She types a few paragraphs into a text box to describe her idea for the advert.
6. She fills the space on screen and a vertical scrollbar appears and the text in the text box scrolls up.
7. She enters her initials into a text box, and the system checks that she is allocated to work on that campaign.
8. The date and time are displayed by the system, and the Save button is enabled.
9. She clicks on the Save button and the word Saved appears in the status bar.
10. The text box, the text field for initials and the date and time are cleared.



## 2.3.5. Use Cases

- ▶ Interaction between a user (actor) and a given or assumed user interface
- ▶ Encompass a set of usage scenarios, bound to the same goal of the primary actor
- ▶ Organized into a Main Success Scenario and a set of Extensions
- ▶ Contain attributes such as goal, primary actor, precondition, level of abstraction
- ▶ Essential use case: A simplified, abstract, generalized use case defined in terms of user intentions and system responsibilities → no technological constraint



## Use Case: Login

### Properties

- **Primary Actor:** [Customer](#)
- **Goal:** [Customer](#) logs into the program.
- **Level:** Sub-function

### Main Success Scenario

1. Primary Actor indicates that he/she wishes to log-in to the system.
2. Primary Actor performs the following in arbitrary order.
  - 2.1 The Primary Actor provides the user name.
  - 2.2 The Primary Actor provides the password.
3. Primary Actor confirms the provided data.
4. System authenticates the Primary Actor.
5. System informs the Primary Actor that the Login was successful.
6. System grants access to the Primary Actor based on his/her access levels.

### Extensions

- 4a. The provided username or/and password is/are invalid:
  - 4a1. System informs the Primary Actor that the provided username or/and password is/are invalid
  - 4a2. System denies access to the Primary Actor.



## 2.4. Prototyping

- ▶ How do we express early design ideas?
  - Minimal development effort at this stage
- ▶ Designs need to be visualized
  - to help designers clarify their own ideas
  - to enable users to evaluate them.
- ▶ The medium needs to be appropriate for
  - the stage of the process,
  - the audience,
  - the resources available and the questions that the prototype is helping to answer.
- ▶ Usually designers create prototypes of increasing complexity

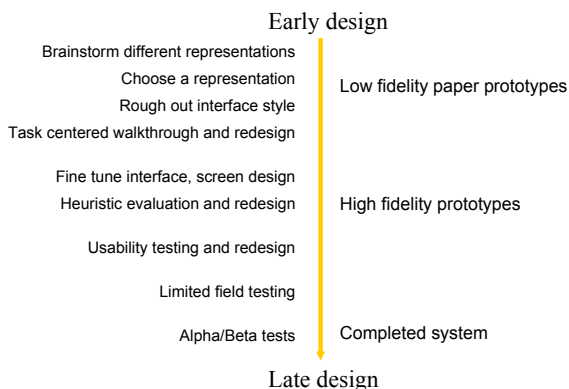


## 2.4 Prototyping Dimensions

- ▶ 1. Representation
  - How is the design depicted or represented?
  - Can be just textual description or can be visuals and diagrams
- ▶ 2. Scope
  - Is it just the interface (mock-up) or does it include some computational component?
- ▶ 3. Executability
  - Can the prototype be "run"?
- ▶ 4. Maturation
  - Revolutionary – Throw out old one
  - Evolutionary – Incorporate design changes
  - Incremental – Modular development



## 2.4. Prototyping



## 2.4.1 Low fidelity prototypes

- ▶ Paper-based prototypes
  - a paper mock-up of the interface look, feel, functionality at the high level
  - "quick and cheap" to prepare and modify
  - For early feedback on conceptual design ideas
- ▶ Issues
  - Robustness – handled by many people
  - Scope – focus on high level only
  - Instructions – designer intervention when users evaluate it
  - Flexibility – can users redesign it 'on-the-fly'?



## 2.4.1 Low-fidelity prototypes

### 1. Sketches

- drawing of the outward appearance of the intended system
- hard to envision a dialog's progression

**Computer Telephone**

Last Name:  
First Name:  
Phone:

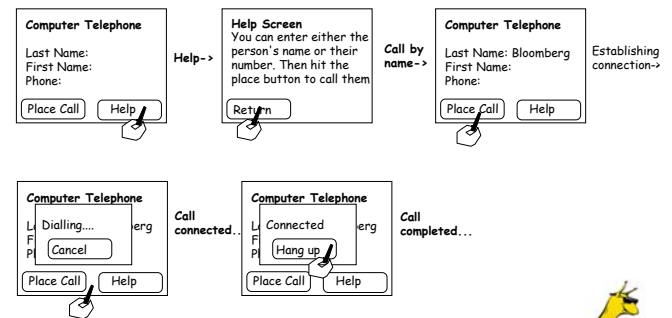
Place Call      Help

### 2. Storyboarding

- a series of key frames
  - originally from film; used to get the idea of a scene
  - snapshots of the interface at particular points in the interaction
- users can evaluate quickly the direction the interface is heading



## 2.4.1 Storyboard of a computer based telephone



## 2.4.2 Prototyping: Wizard of Oz

- A method of testing a system that does not exist



What the user sees



The Wizard



## 2.4.2 Prototyping: Wizard of Oz

- Human simulates the system's intelligence and interacts with user
- Uses real or mock interface
  - "Pay no attention to the man behind the curtain!"
- User uses computer as expected
- "Wizard" (sometimes hidden):
  - interprets subjects input according to an algorithm
  - has computer/screen behave in appropriate manner
- Good for:
  - adding simulated and complex vertical functionality
  - testing futuristic ideas → think about real implementation, though



## 2.4.3 High fidelity prototypes

- Prototyping with a computer
  - simulate or animate some but not all features of the intended system to engage end users
- Purpose
  - provides a sophisticated but limited scenario to the user to try out
  - provides a development path (from crude screens to functional system)
  - can test more subtle design issues
- Danger
  - user's reactions are about small things
  - users reluctant to challenge / change the design



## 2.4.4 Limiting prototype functionality

- vertical prototypes
  - includes in-depth functionality for only a few selected features
  - common design ideas can be tested in depth
- horizontal prototypes
  - the entire surface interface with no underlying functionality
  - a simulation; no real work can be performed

