1. What is Usabiliyt and what is User Experience?

1.1. Usability

- **ISO-9241 part 11**:
  "Extent to which a product can be used by **specified users to achieve specified goals** with **effectiveness, efficiency** and **satisfaction** on a specified **context of use**."
- **Usability Engineering** is about achieving **a "good" usability**. This means a product is **"user friendly", "easy-to-use" or "intuitive".**

- **Effectiveness:** Accuracy and completeness with which users achieve specified goals.
  Is the user able to achieve his tasks?

- **Efficiency:** Resources expended in relation to the accuracy and completeness with which users achieve goals.
  How man interaction steps are necessary to achieve a task?

- **Satisfaction:** Freedom from discomfort, and positive attitudes towards the use of the product.
  Do the user like or dislike the product?

- **Context of Use:** Users, tasks, equipment (hardware, software and materials) and the physical and social environment in which a product is used.
  Is the product suitable for the environment it is used in?

- **Dependency** (Medical Applications)
- **Reliability** (Safety-critical systems)
- **Productivity** (Office Applications)

1.2. User Experience

- **ISO-9241 part 210:**
  "Person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service."

- User experience includes all the users' **emotions, beliefs, preferences,** perceptions, physical and psychological responses, behaviors and accomplishments that occur **before,** during and **after** use.
- User experience is a consequence of brand image, presentation, functionality, system performance, interactive behavior and assistive capabilities of the interactive system, the user's internal and physical state resulting **from prior experiences, attitudes, skills and personality**, and the context of use.

- **Business** (Shopping, Product configuration, etc.)
- **Communication** (Mobile devices and services)

1.3. Usability Engineering

1.3.1. History:

- **1960s:** Computing time on large mainframes was too costly to waste it with inefficient use → *"User Engineering"* and *"Human Factors"* to improve UIs for scientists and engineers
- **1970s:** Novel vision of *"personal computing"* and of computers as personal *"tools for thought"* → *Psychology* and *cognitive science* are applied on UI design for non-expert users. GUIs are invented.
- **1980s:** "Human-Computer Interaction" (HCI) is established as a joint *scientific and engineering* discipline. GUIs become widespread.
- **1990s:** World Wide Web and e-Commerce boost interest in easy-to-use websites and improving customer rates.
  → User-centered Design (UCD) and Usability Engineering (UE) based on HCI knowledge are increasingly applied by market leaders.

- **Today:** "Web 2.0" platforms, Rich Internet Applications, mobile devices with multi-touch & gesture interaction, infotainment devices in cards an living rooms, … compete in terms of usability and "User Experience" (UX)
  → **Usability Engineering is widely practiced** in the "User Experience" groups of IT manufacturers, service providers, service providers and consultancies (e.g. IBM, Google, Apple, …)
  → Good usability, user experience and a user-centered perspective are crucial today – not only for assuring quality of the user interface, but also for innovating and inventing products.

1.3.2. Definition:

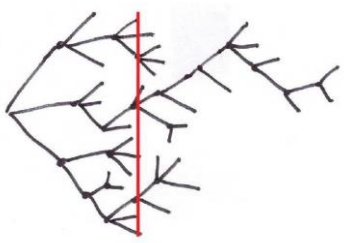
- "Usability Engineering **systematically** focuses on the **user,** her **needs** and her **abilities** during product & design development … and not only on the **technology!**"

- "Usability Engineering is a discipline that provides **structured methods** for achieving usability in user interface design during product development."

- "The term **usability engineering** (in contrast to interaction design and user experience design) implies more of a **focus on assessing and making recommendations to improve usability than it does on design**, though Usability Engineering may still engage in design to some extent particularly **design of wire-frames or other prototypes.**" (Wikipedia)

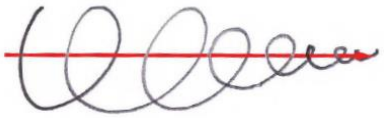
- "The role of **design** is to get the **right design**. The role of **usability engineering** is to **get the design right.**" (Bill Buxton)

Problem setting
"Getting the right design"



- Go for quantity
- Compare alternatives
- Filter out best solution

Problem solving
"… and the design right"



- Go for quality
- Improve with iterations
- Refinement of best design

- Engineers work to **solve problems**
- Therefore, a **clear definition of the problem** to be solved is necessary
- **Design** provides a **problem setting**
- Engineering seeks to **define and solve the problem**

- **Problem Definition** (Requirements Engineering)
  - User Requirements
  - Task Analysis
    → Modeling Activities

- **Problem Setting** (Interaction Design)
  - Sketches
    → Design Activities

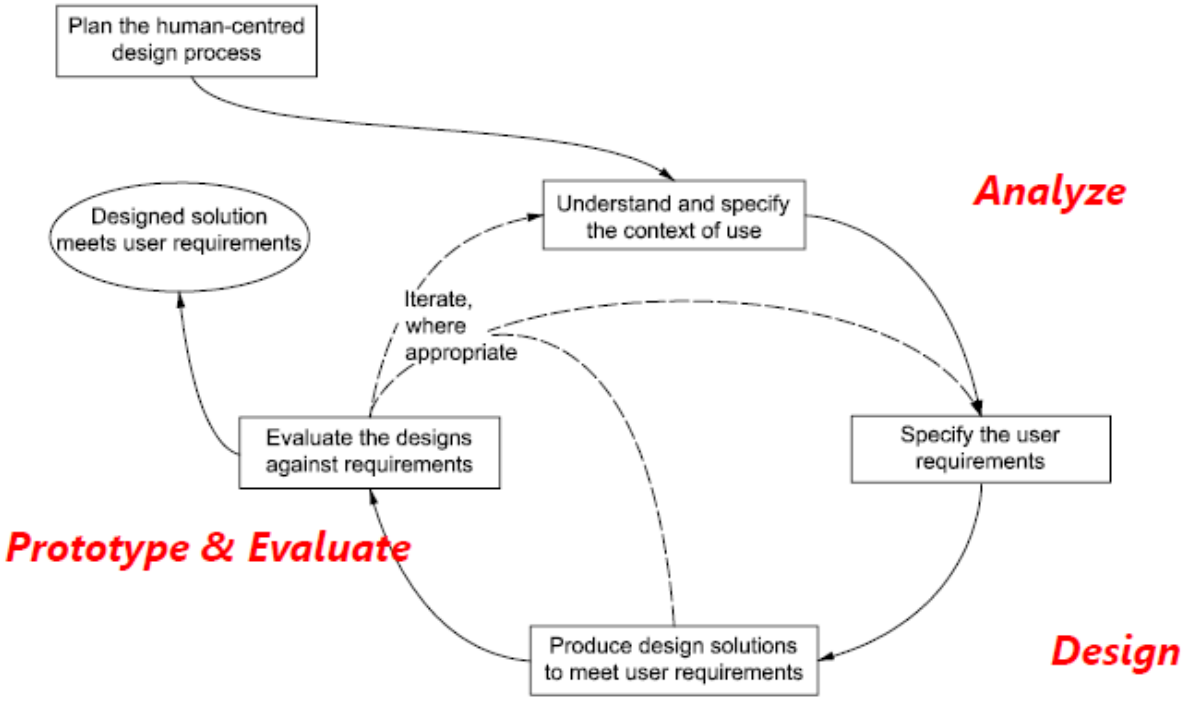
- **Problem Solving** (Usability Engineering)
  - Guidelines
  - Interface (Re)Design
  - Usability Testing
    → Prototyping

1.3.3. Process Models – How to do Usability Engineering?

- All **Usability Engineering processes** typically define a **workflow** or **process model** for achieving good usability
- These **process models** are meant to **organize** the different **activities** of usability engineering

- Typically all user-centered processes contain **3 basic steps** or phases of the process:
  - **Analyze** (or Requirement Analysis)
  - **Design** (or Modeling)
  - **Prototype & Evaluate** (or Development & Testing)

- **ISO 9241 part 210**
  "Human centered design process for interactive systems"



- Iterative improvement
  - Waterfall model is not only unsuitable for code, but especially for UI
  - **Today:** Iterations with successive refinements based on usability inspections and tests

1.4. Usage-Centered Design – 5 Key elements

Based on the book "Software for Use" by Larry Constantine and Lucy Lockwood, 1999

- Promotes "**Usage-Centered Design**" not "**User-Centered Design**"

- "The chief difference from other interface design philosophies is that **user-centered design** tries to optimize the user interface around **how people can, want, or need to work**, rather than forcing the user to change how they work to accommodate the software developers' approach."

- "Usage-Centered Design focuses on **the work that users are trying to accomplish** and on what the software will need to **supply via the user interface.**"

1.4.1. Pragmatic design guidelines

Usability Rules:

- "... being **like a compass**, establishing a **general direction** for greater usability."

- ***Access Rule***
  "The system should be usable, without help or instruction, by a user who has knowledge and experience in the application domain but no prior experience with the system."

- ***Efficacy Rule***
  "The system should not interfere with or impede efficient use by skilled user who has substantial experience with the system."

- ***Progression Rule***
  "The system should facilitate continuous advancement in knowledge, skill and facility and accommodate progressive change in usage as the user gains experience with the system."

- ***Support Rule***
  "The system should support the real work that users are trying to accomplish by making it easier, simpler, faster or more fun or by making new things possible."

- ***Context Rule***
  "The system should be suited to the real conditions and actual environment of the operational context within which it will be deployed and used."

Usability Principles:

- "… resolve **practical problems** in user interface design."

- ***Structure Principle***
  "Organize the user interface purposefully, in meaningful and useful ways based in clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things resemble one another."

- ***Simplicity Principle***
  "Make simple, common tasks simple to do, communication clearly and simply in the user's own language and providing shortcuts that are meaningfully related to longer procedures."

- ***Visibility Principle***
  "Keep all needed options and materials for a given task visible without distracting the user with extraneous or redundant information."

- ***Feedback Principle***
  "Keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to the user."

- ***Tolerance Principle***
  "Be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing while also preventing errors where possible by tolerating varied inputs and sequences and by interpreting all reasonable actions reasonably."

- ***Reuse Principle***
  "Reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember"

- ***User Interface Style guides***
  "Most user interface designers design screens, windows and widgets; the best ones design user **interface architectures**."
  (e.g. Apple, Microsoft, Android, …)

1.4.2. Model-driven design process

Three core models:

- ***Role model (see Chapter 4)***
  The relationships between users and the system

- ***Task model (see Chapter 5)***
  The structure of tasks that users will need to accomplish

- ***Content model (see Chapter 6)***
  The tools and materials to be supplied by the user interface, organized into useful collections and the interconnections among these collections.

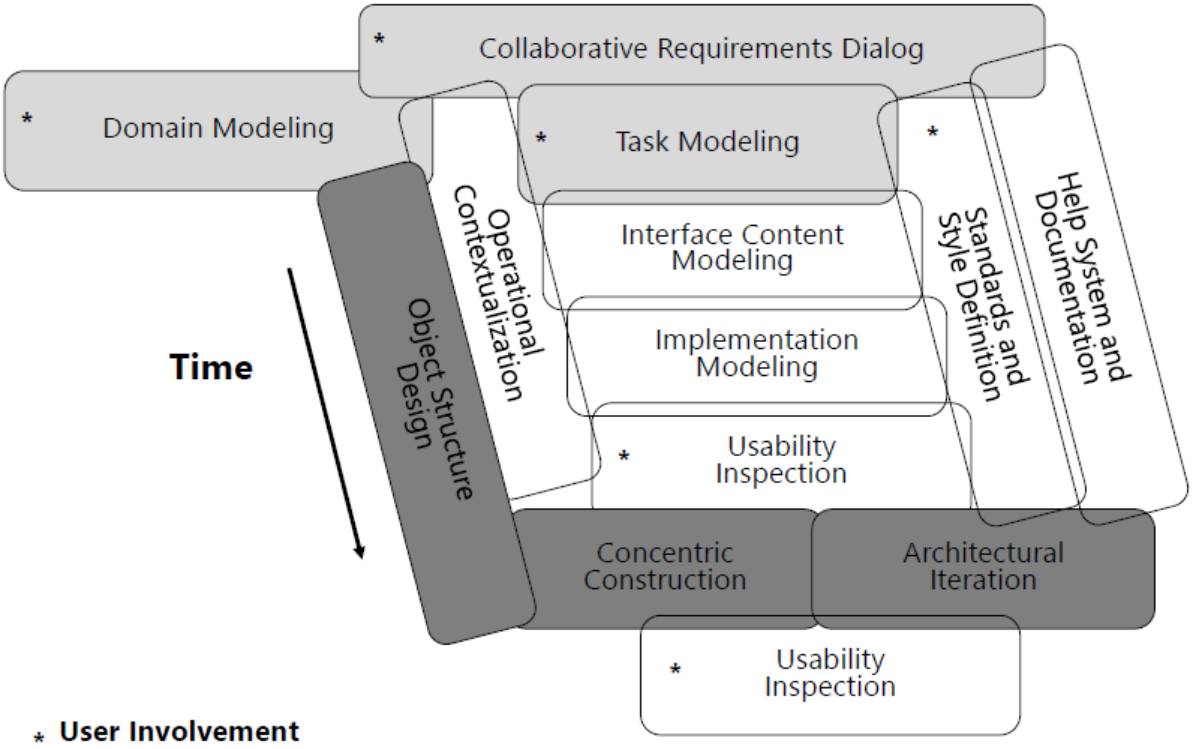
Two additional models:

- ***Operational model***
  The operational context in which the system is deployed and used

- ***Implementation model***
  The visual design of the user interface and description of its operation

- Models in Usage-Centered Design have a "**semi-formal**" nature
- They have a **specific format and notation** (that often reminds of UML but is much less formal)
- "Semi-formal" models serve the design team as a point of **orientation**, means of **communication** and to **manage** and **share** knowledge
- Usage-Centered Design promotes models as **creative tools** for human designers and **not as formal specifications**.

1.4.3. Organized development activities



1.4.4. Iterative improvement

- "Getting a system exactly right the first time is very difficult, if not impossible."
- **Cost** of finding and fixing a defect **increases with time**.
- Testing and **fixing a model** is **much cheaper** than testing and fixing code.

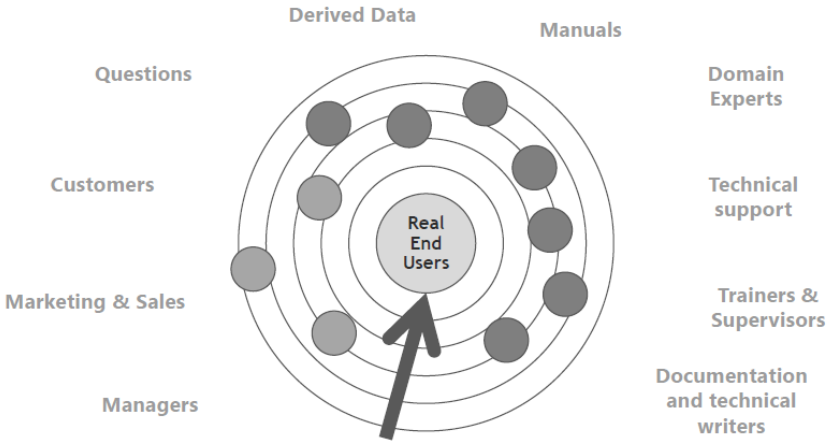
1.4.5. Measure of quality

Topic of UE: Evaluation.

2. Understanding Users and Tasks

2.1. Understanding Users

- "**End users** are the primary and ultimate source of information to guide usage-centered design." (Software for Use, Constantine and Lockwood, 1999)
- **Face-to-face** interviews or meetings with **end-users** have highest bandwidth!



2.2. Field Studies

2.2.1. Data gathering

- **Setting goals** – What do I want to find out?
- **Choosing a method** – How do I get the information I need?
- **Pilot study** – A trial run
- **Approaching the participant** – Who can give me the information I need? Informed consent!

2.2.2. Setting goals

- Understanding the **user**
  - Behavior
  - Reason for behavior
  - Opinion
- Understanding the **task**
  - Responsibilities
  - Steps
  - Workflow
- Understanding the **context**
  - Physical environment
  - Organizational environment
  - Social environment

2.2.3. Choosing the Method

Interviews

- A method for discovering facts and opinions held by potential users of the system being designed.
- "Interviews can be thought of as a conversation with a purpose" (Interaction Design)

- ***Structured***
  - Pre-determined questions
  - Closed questions with answer from pro-determined set of alternatives
  - Fast an standardized (same study for all participants)

- ***Un-structured***
  - Exploratory
  - Open questions (like a conversation)
  - Might unveil unexpected issues

- ***Semi-structured***
  - Script that ensures that the same topics are covered in each interview
  - Start with pre-planned questions and the probe for more information
  - Intended to being broadly replicable

	Structured	***Semi-structured***	***Unstructured***
Strength	• Standardized • Good to analyze • Replicable	• Obvious structure • Largely flexible	• Very flexible • Not biased • Subjective data
Weaknesses	• Little subjective data	• Situation not well graspable • Only partially good to analyze	• Hard to analyze • Time-consuming • Not replicable

- ***Procedure:***
  - Develop set if questions/topics to be covered
  - Work out structure of interview
  - Collate the necessary documents (e.g. consent form)
  - Check recording equipment
  - Pilot the interview
  - Organize time and place

- ***Summary:***
  - Good for Exploring issues
  - Some quantitative but mostly qualitative data
  - Advantages:
    - Interviewer can guide interviewee if necessary
    - Encourages contact between developers and users
  - Disadvantages:
    - Time consuming
    - Artificial environment might intimidate interviewee

Focus Groups

- ***Description:*** Assembly of users (usually 3-10) whose opinions are requested about a specific topic
- ***Goal:*** elicit perceptions, feelings, attitudes and ideas of participants about the topic
- ***Participants:*** representative sample of the target (diversity is useful)
- ***Structure:*** Pre-set agenda to guide the discussion but flexibility to follow unexpected issues
- ***Advantage:*** Participants develop opinions within a social context by talking with each other
- ***Challenge:*** manage and guide quiet and verbose participants

- ***Summary:***
    - Good for collection multiple viewpoints
    - Some quantitative but mostly qualitative data
    - Advantages:
        - Highlight areas of consensus and conflict
        - Encourages contact between developers and users
    - Disadvantages:
        - Possibility if dominant characters

Questionnaires

- ***Purpose:*** Collect demographic data and users' opinions
- ***Structure:***
    - Demographic information (age, gender)
    - Relevant experience
    - Specific question addressing the study goal
    - Questioned aspects need to be relevant for the study goal
- ***Challenges:***
    - Develop clearly worded and specific questions
    - Provide clear instructions on how to complete the questionnaire
    - Efficient and meaningful way of analysis

- "Well-designed questionnaires are good at getting answers to specific questions from a large group of people, and especially if that group of people is spread across a wide geographical area, making it infeasible to visit them all" (Interaction Design)

- ***Questions is Questionnaires:***
    - It is important to select the most appropriate question and response format
    - Closed questions require an answer from a set of possibilities
        - Many answers can be chosen
        - One answer can be chosen
        - Users have to locate their answer within range
    - Open Questions: unrestricted answers

- ***Summary:***

- Good for answering specific questions
- Some qualitative but mostly quantitative data
- Advantages:
  - Can reach many people with low resource
- Disadvantages:
  - The design is crucial (requires expertise and experience)
  - Response rate may be low
  - Responses may not be what you want

Observation

- "It can be very difficult for people to explain what they do or to even describe accurately how they achieve a task. So, it is very unlikely that an interaction designer will get a full and true story by using interviews or questionnaires. Observation in the field can help fill in detail and nuances that are not elicited from the other forms of investigation " (Interaction Design)

- ***Description:*** Observe users' behavior in their natural environment
  - Note taking (pictures?)
  - Video recording
- ***Goal:*** Understand users' **context, task and goals**

- ***Observation framework:*** Important to have a clear focus in an observation setting
  - Space: What is the physical space where the activity is carried out?
  - Actors: Who are the people involved?
  - Activities: What do they do, and why?
  - Objects: What physical objects are present?
  - Acts: What are specific actions?
  - Events: Is it part of a special event?
  - Time: What is the sequence of events?
  - Goals: What are people trying to accomplish?
  - Feelings: What is the mood of the actor(s)?

By Robson, 2002

- ***Role of observer:*** passive or participant
- ***Handling the data:*** Take time after each observation for going through your notes and other records

- ***Summary:***
  - Good for understanding context of user activity
  - Mostly qualitative data
  - Advantages:
    - Observing actual work gives insights that other techniques can't give
  - Disadvantages:
    - Very time-consuming
    - Huge amounts of data

Ethnography

- ***Principles*** of context inquiry
  - *Context:* experience work context by yourself
  - *Partnership:* understand the work cooperatively with users
  - *Interpretation:* verify common understanding
  - *Focus:* concentrating on important aspects

- ***Combines***
  - Direct observation
  - Interviews
  - Questionnaires
  - Study of artifacts

- Companies e.g. as Intel want to "develop a deep **understanding of how people live and work**" and rely on ethnographic studies, such as:
  - US citizens' motivation for adopting a "green" lifestyle
  - The role of cash in Middle East's commerce
  - Use of mobile technology in Japanese clubbing and nightlife

2.2.4. Data Analysis

- **Quantitative** data analysis
  - Data in form of numbers (e.g. averages and percentages)
  - Important to decide how you want to present the results
  - Are the numbers meaningful?
  - Divide the data into sub-sets (e.g. distinguish between female and male users)

- **Qualitative** data analysis
  - Focuses on the nature of something and can be represented by themes, patterns and stories
  - Identifying recurring patterns or themes
  - Categorizing data
  - Looking for critical incidents

- **Interviews and Focus Groups**
  - Raw data: audio/video recording and/or notes
  - Transcription: whole or in part
- **Questionnaires**
  - Closed questions result in quantitative data
  - Open questions result in qualitative data
- **Observation**
  - Wide variety of data (observer's notes, photographs, video- and audio recordings)
  - Most likely to be analyzed with a qualitative approach
  - Patterns and themes in the data might present themselves – analysis according to the observation framework

2.2.5. *Practice*

- **Plan** your field study
  - **Set** appropriate **goals**
  - **Define the focus** of your investigation
  - **Choose** suitable **methods** (consult your books!)

- **Test** your methods
  - **Prepare** materials
  - **Pilot** the study
  - **Check** the data
- **Conduct** study
  - Approach participants
  - **Collect** data
  - **Analyze** data

3. Conceptual Modeling

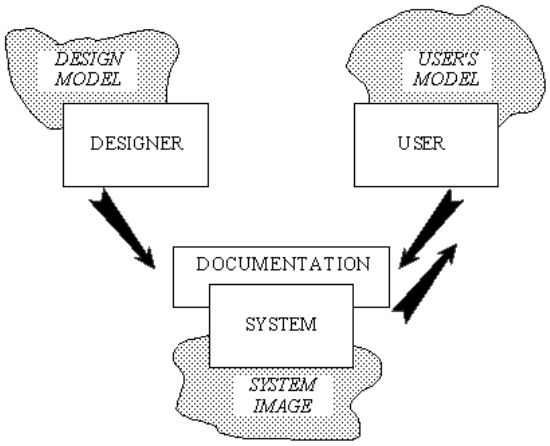
3.1. Basics

- "… a **model** is anything used in any way to **represent anything else**. Some models are physical objects […] however a **conceptual model,** may only be drawn on paper, described in words, or imagined in the mind. They are used to help us **know and understand the subject matter** they represent." (Wikipedia)
- Everyone has a **mental model** of the working of a **car**. Without such a mental model *we would not be able to drive it*!
- We **built** such models every day **during interacting** with the world or observing it (e.g. taking driving lessons, crashing, …)
- Models are **based on our observations**, out experiences and the conclusions we drawn from them.
- Modeling **reduces the complexity** of the real world and leads to a **simplified abstract view**
- Modeling makes our life in the world easier, because we can **focus on the important aspects** and make better decisions
- **Abstract models** can help us to **apply** a model to **classify, understand or handle** things, we have not known before
- There is **not one single "true" model** of something
- The model is dependent on many things, such as the **modeling person** or the **context** and the **purpose** of the model

3.2. Modeling in Software Design

- Every user interface designer or programmer has his/her **individual model (or view) of the users, their tasks, the objects** and the objectives

- **Designing** a user interface, writing code or deciding on a data model is a direct **translation** of the **individual models** (or views) **into the design** of a piece of software, hardware (or some other technological artifact)
- **Understand** and observe **tasks**, talk with **users**, create **models** and **share & discuss models** with users and other designers **before and while** creating the system!



3.3. Modeling Users

- A user role is an **abstraction not a real person**, not a job title, not a position, not a functions, …
- It is an **abstract class of people** defined by a particular kind of **relationship** to a system
- One **real person** can take on any number of **different roles** with the same system
- One **user role** can be played by any number of **different persons**
- **"**Users are of interest to developers, **not as people**, but because **of the roles** they will play in relationship with the system to be designed and built.**"** (Software for Use)

User Roles vs. Personas

- "Personas describe user while **user roles** describe the relationships between users and systems. **Personas are figurative** models rather than abstract models, that is, they are constructed to **resemble real users**, even down to photos, background, information, and personal history. […] many people find that the creative process of construction personas to be **engaging and energizing**. Personas are fun." (Constantine, 2005)
- "By contrast, **user roles do not resemble real people** nor are they intended to; roles are Spartan **abstractions** narrowly focused on **those aspects** of the relationship most likely to be **relevant** to presentation and interaction design. Compared to personas, user roles are a **more technical** and **formally structured** model."

- ***Personas:***
  - *Pro*: realistic, natural, appealing, memorable
  - *Con*: complicated, distressing, too much details
  - *Con*: "What matters and what dos not?"
  - *Con*: easily cross the boundaries from user description into user interface design!
- ***User Roles:***
  - *Pro:* capture most relevant (usage-related) things
  - *Pro:* concise form (a single page or a single index card per role)
  - *Pro:* cover a lot of different users
  - *Con:* tend to be harder to user for unexperienced designers and remain more abstract

- "The **user role map** is a way of capturing the big picture; it reveals how all the **various roles fit together** in defining **who** will use the system and **how.**" (software for use)
    - *Affinity:* similar
    - *Classification:* specializes
    - *Composition:* includes

3.4. Modeling Tasks

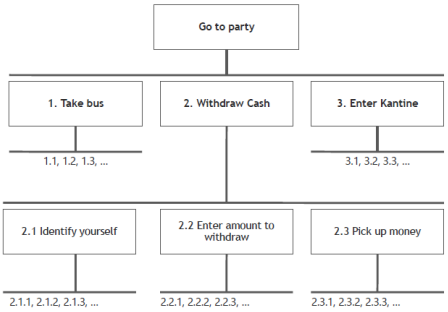
3.4.1. Scenario

- "Scenarios are **narrative descriptions** of an **activity** or **activities**, taking the form of a **story**, a vignette, or an **episode** bound in **time** and taking **place** with a given **context**." (Software for Use)
- "A **scenario** can say what happens in a **particular situation** without committing to details of precisely how things happen. **Much of the richness of a scenario is in the things that are not said.** A scenario specifies its **actors' goals** and **behaviors**, and these can be arbitrarily detailed. Yet a scenario narrative is **easily modified** or **elaborated** and can be made deliberately **incomplete** to help developers' **cape with uncertainty**." (Rosson & Carroll: Usability Engineering)
- "In our experience, conventional **scenarios** have some serious **limitations** when it comes to **user interface** design. Their **emphasis on realism and rich detail** can **obscure broad issues** and general organization. Because scenarios comprise plausible combinations of individual tasks or activities, **they can make it more difficult to isolate and understand the basic kernels** of interaction." (Software for Use)
- **One solution could be visual notations instead of text to reduce detail and focus on the interaction.**

3.4.2. Flowcharts

- "This **sequential approach** to modeling human work is **detailed** and **concrete**, resembling in many respects, computer programming. Consequently, it **tends to appeal to programmers** and traditionally trained system analysts. As might be expected, flowcharting and its kin max generally **be better suited to modeling computer tasks** than human tasks." (software for Use)

3.4.3. Hierarchical Decomposition - Hierarchical Task Analysis (HTA)



- "Functional decomposition can **produce elegant models** that have **little to do with how work is actually accomplished** or how users think about it. […] Not surprisingly this approach seems to hold special appeal for **academics and others with a research or scientific bent.**" (Software for Use)
- "Strength of HTA is the **step-by-step transformation** of a **complex activity space** into an **organized set** of **successive choices** and actions. However, too **much emphasis on tasks decomposition can be** problematic. […] HTA documents the **functional goals** of a workspace, BUT may direct attention away from an organization' social relationships goal."

3.4.4. Use Cases

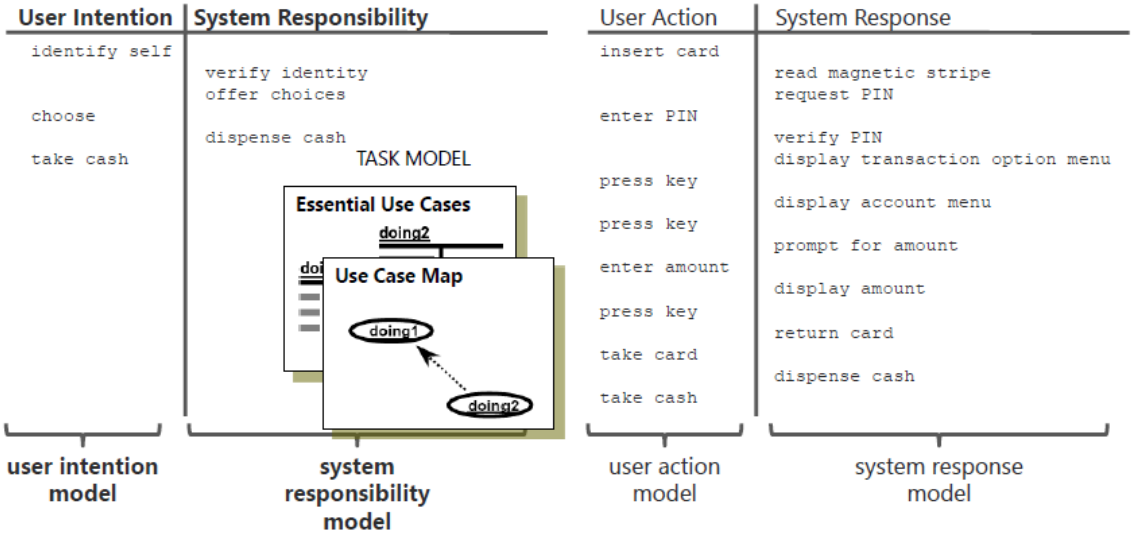
- "As the term is employed in object-oriented software engineering, **a use case is a narrative description of interactiom between a user – in some role – and some system**." (Software for Use)
- Usages Centered Design proposes a certain convention for use cases: a **two-column sructured form** that clearly divides the **responsibilities** and interest of the **user** from those if the **system** and its developers.

Use Cases vs. Scenarios

- Use cases were intended for use in object-oriented software engineering.
- Scenarios were intended for use in usability engineering.
- **Differences:**
  - **Use cases** are typically **more general**, e.g. they include multiple possible responses to an input.
  - **Scenarios** can be regarded as **one instance of a use case.**
  - **Scenarios** are deliberatetly unspecified about the details of the the design, but focus on the **overall design rationale.**
  - **Use cases** are used as **funtional specifiactions**, while **scenarios** raise and consider their **usability implications.**

Concrete Use Cases

- "Conventional use cases typically contain too **many built-in, premazire assumptions**, often hidden implicit, **about the form of the user interface** to be designed." (Software for Use)



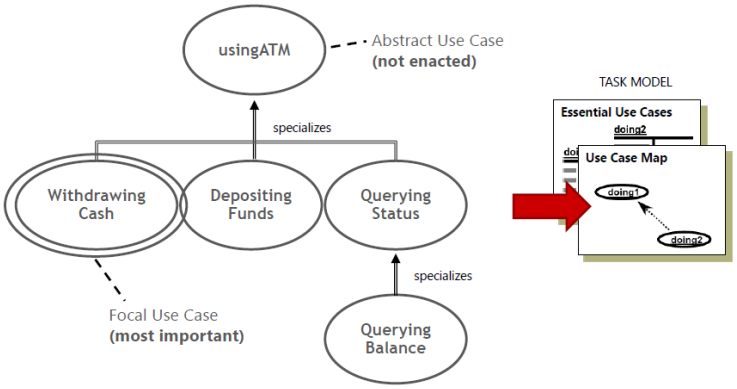
Essential Use Cases

- "An **essential use case** is a **structured** narrative, expressed in the language of the application domain and of users, comprising a **simplified, generalized, abstract, technology-free** and implemtation-independent description of one **task** or **interaction** …" (Software for Use)

- **Modeling:**
    1. **Identify use cases** from needs and intensions of **user roles**
    2. **Compile** a collection of **candidate** use cases by asking questions like:
        - What are the users trying to accomplish?
        - To fulfil this role, what do the users need to be able to do?
        - What are the main tasks of the users in this role?
    3. **Give** the use case a simple, pruposeful, goal-directed **name**
    4. **Eliminate** overlapping use cases by **combining** similar use cases
    5. **Write narratives** for each use case
    6. **Smiplify** an **generalize** use case

Use Case Map

- "The **use case map** for a given problem **partitions** the total functionality of the system into a collection of **interrelated essential use cases.**" (Software for Use)
- **Specialization**
    - "is a kind of …"
    - Inhertitance
- **Extension**
    - "extends"
    - Inserted or alternative tasks
- **Composition**
    - "includes"
    - "composed of"
    - "uses"
- **Affinity**
    - Unclear, resembling or ambiguius relationships



3.5. Content Model

3.5.1. Interaction Context

- Work has a contextual nature: We carry out **different tasks in different places.** In this place we find the necessary **tools** and **materials.**
- **Efficient performance** takes place in an interaction context in which the necessary **tools** and **materials** are **immediately available.**
- Good user interfaces are **organized** to provide a set of distinct **interaction contexts** (or "interaction spaces") **appropriately equipped** for carrying out the user **tasks.**
- In implementations, **interaction spaces** often become a **recognizable part** of the user interface
- A user interface is a **network** of **interconnected** interaction spaces

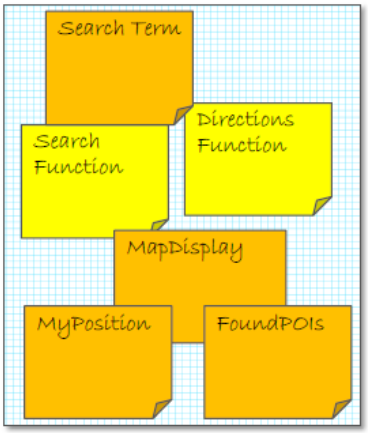
3.5.2. Content Model

- A **content model** represents **one interaction space**
- The navigation map **interconnects** different interaction spaces
- **Identify** interaction contexts based on **essential use cases**

Interaction Context

- **Identify** the necessary **interaction contexts** (or interaction spaces):
  - Start with **one** interaction context **per use case**!
- For **closely related** use cases try to **build one common** interaction context.
- For more **lengthy** or complex use cases **split** them across more than one interaction context.
- Take a sheet of paper for each context and start with **interface content model** for each one.

Content Model



- Identify **tools and materials** for each use case
- Use **post-it** notes of **different colors and sizes** to populate each interaction space
- Think of all necessary **tools and materials** that the user interface **will have to provide** by examining the use case line by line **without** thinking about **specific interface components!**
- Think of the **tools** as **functions** and **active capabilities** (e.g. hot colors)
- Think of the **materials** as **data, containers, displays** or **work areas** for the tools (e.g. cool colors)
- Stay **abstract!** Don't try to look real! DO not commit to a particular widget yet!

- **Benefits** of abstract models:
  - By focusing on **content** we focus on the **actual problem without** getting **distracted** by unnecessary details.
  - **Reorganization** of the whole user interface architecture is **easy,** which encourages **experimentation** and **exploration**

Context Navigation Map

- Define the **overall architecture** in a **context navigation map**
  - Collect all interaction spaces **in a single diagram**
  - Think of the necessary **transitions** between interaction spaces based on use cases
  - **Connect** the interaction space with **arrows** representing the **type** of transition and the **conditions** under which transitions occur

- **Clean up** models:
  - Every **user role** must be supported by **one or more use cases**

    - Every use case must support one or more user roles
    - Every **use case** must be fully supported within the **content model**
    - Every interaction context must be used in one or more use cases
    - All **tools** and **materials** must support one or more **use cases**
    - Every **transition** between use contexts should be used enacting some **use cases**

- Use navigation map to **identify flaws** in overall architecture
- Add direct transitions between closely related interaction contexts

3.6. Operational Model

3.6.1. Operational Context

- **Screen** size, **resolution, distance, movement** and **vibration** must fit operating conditions!
- **Intercultural** aspects are also important
- **Proficiency** and **frequent** of use
- "Truly usable software [or systems] is **highly attuned** to its **environment**. We cannot approach the design of a system's user interface with a one-size-fits-all mentality." (Software for Use)

3.6.2. Operational Model

- Collection of **aspects** and **factors** that may play a role in suability
    - Characteristics of **user roles**
    - **Physical** work environment
    - **Limitations** of employed hardware devices
    - General or specific operational **risk factors**
- The operational model consists of **textual** descriptions which are called **profiles**

- **Incumbents** - characteristics of the actual **user** who will play a given role
- **Proficiency** – how **usage** proficiency is distributed over **time** and among **users** in a given role
- **Interaction** – characteristic **pattern of usage** associated with a given role, use cases, or set of use cases
- **Information** – nature of the **information** manipulated by users or **exchanged** between users and system
- **Functional support** – specific **functions, features** or facilities needed to support users in a given role or for a specific use case or set of use cases
- **Usability criteria** – relative **importance** of specific **usability objectives** for a given role or for a specific use case or set of use cases
- **Operational risk** - type and **level of risk** associated with a given role or for a specific use case or set of use cases
- **Device constraints** – **limitations** or constraining characteristics of the **physical equipment**
- **Environment** – relevant factors of the **physical environment**