QUICKSTART
Over 35 years in Technology Education

INTRODUCTION TO UX & PRODUCT MANAGEMENT

MODULE 1: PRODUCT MANAGEMENT

1. Software quality
2. Software development is teamwork
3. What is Product Management?
4. Product manager responsibilities in the software development lifecycle
5. Product lifecycle
6. Software development models
7. Managing requirements
8. Technical writing

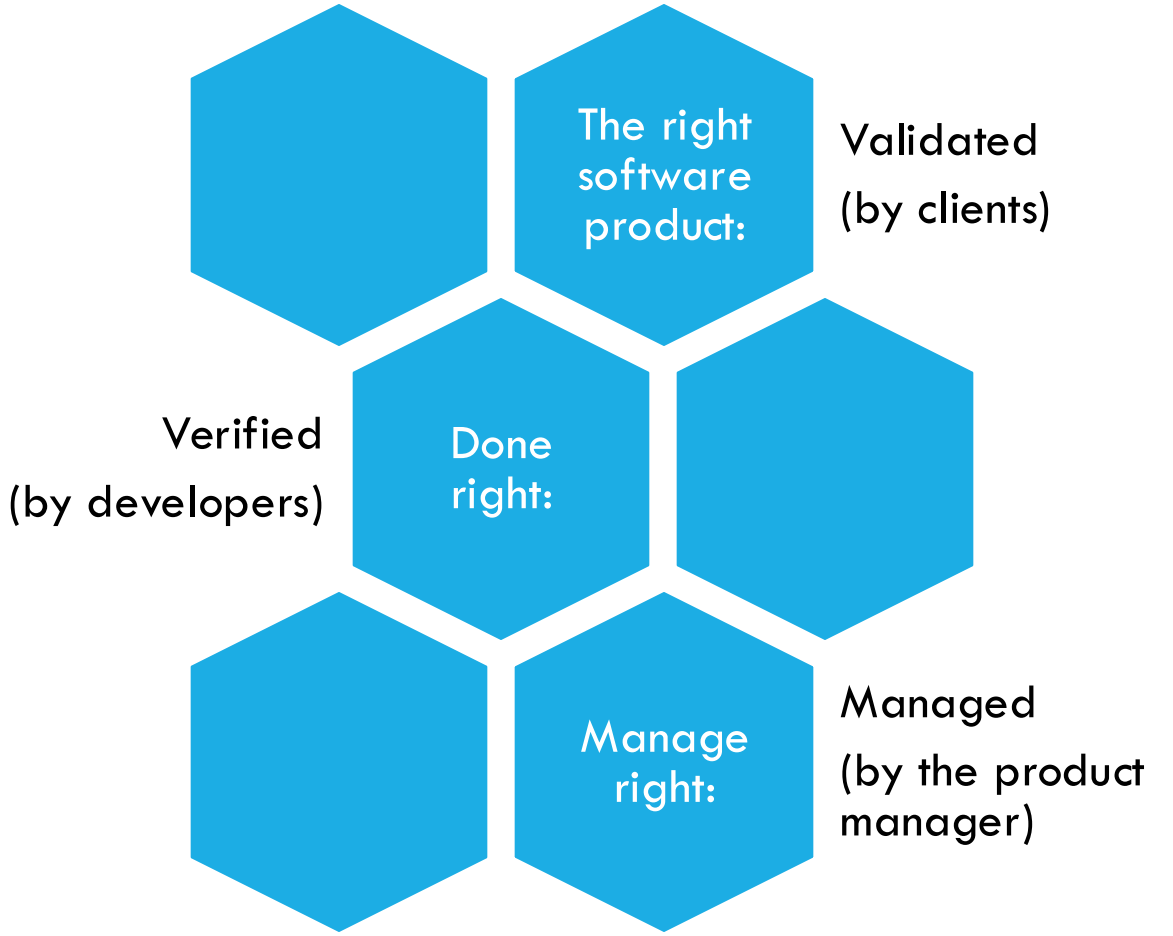
MODULE 2: USER EXPERIENCE DESIGN

1. Introduction: what is user experience design?
2. The UX pyramid and its influence on UX design
3. Developing user personas and value propositions
4. UX prototypes, models, and their evaluation
5. User Stories
6. Wireframes
7. Draw.io example
8. UI Design, component libraries, design systems

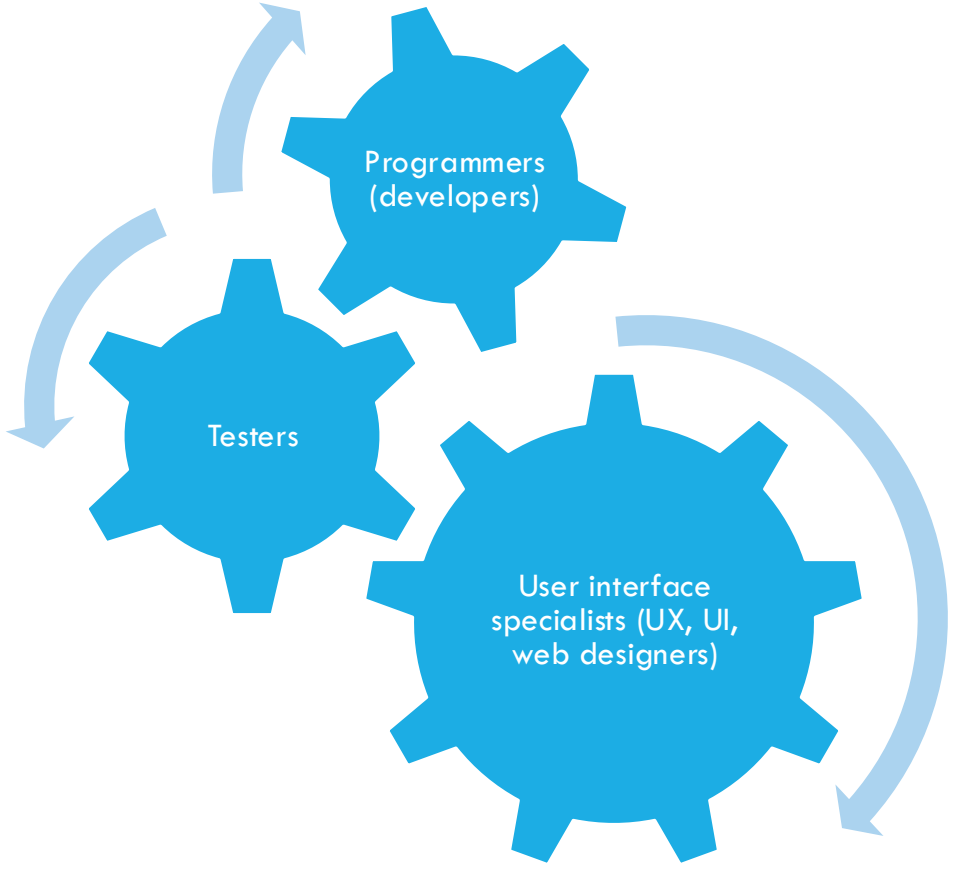
QUICKSTART
Over 35 years in Technology Education

MODULE 1:
PRODUCT MANAGEMENT

SOFTWARE QUALITY



SOFTWARE DEVELOPMENT TEAM

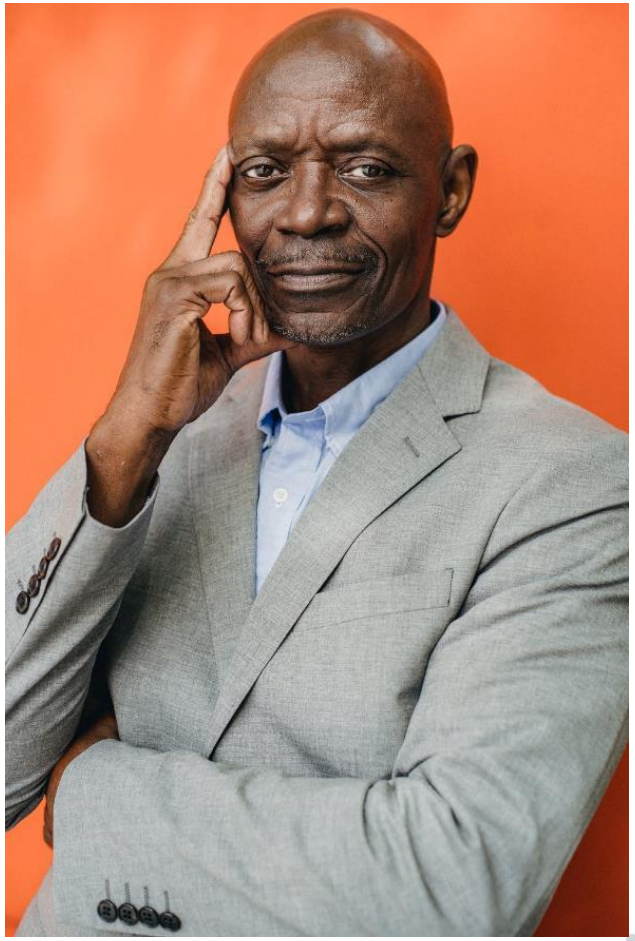


SOFTWARE PRODUCT MANAGEMENT (SPM)

SOFTWARE PRODUCT MANAGER RESPONSIBILITIES

1. Ensuring product quality
2. Managing development process
3. Interacting with clients
4. Dealing with the development team

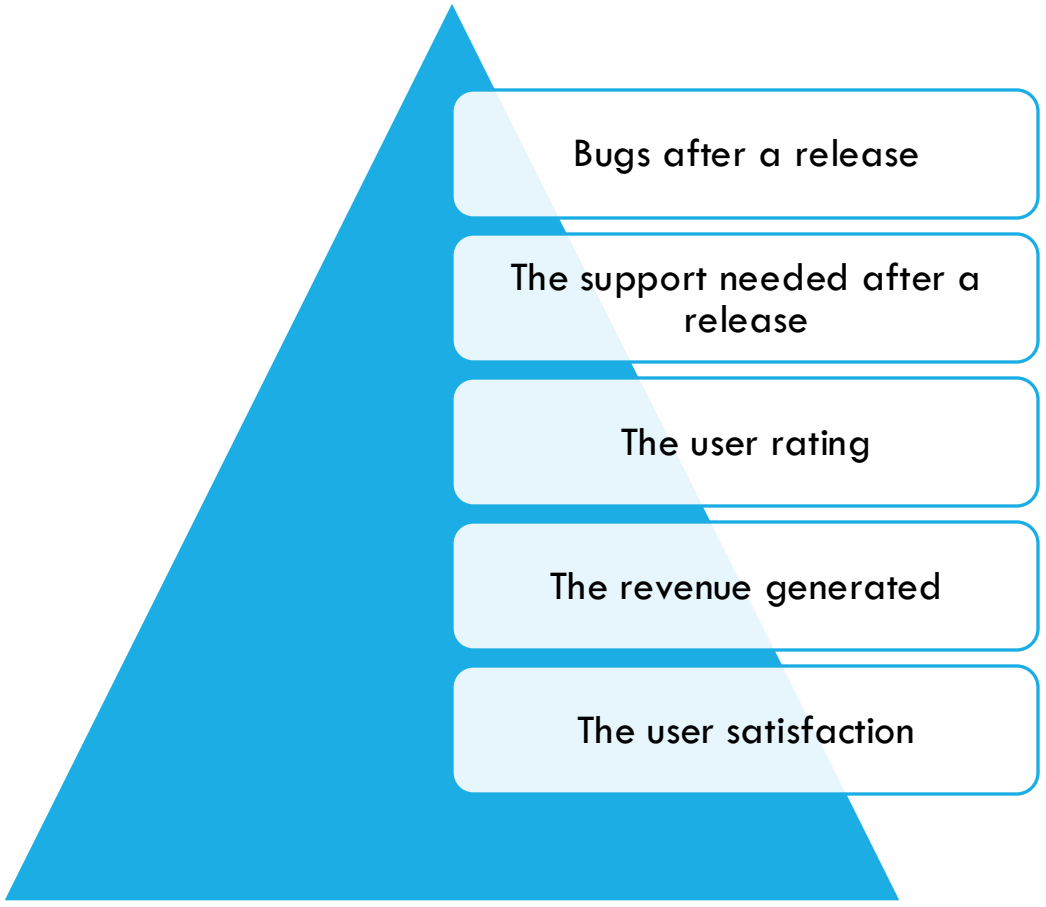
Summary: Solving a customer problem by leading the team and the project



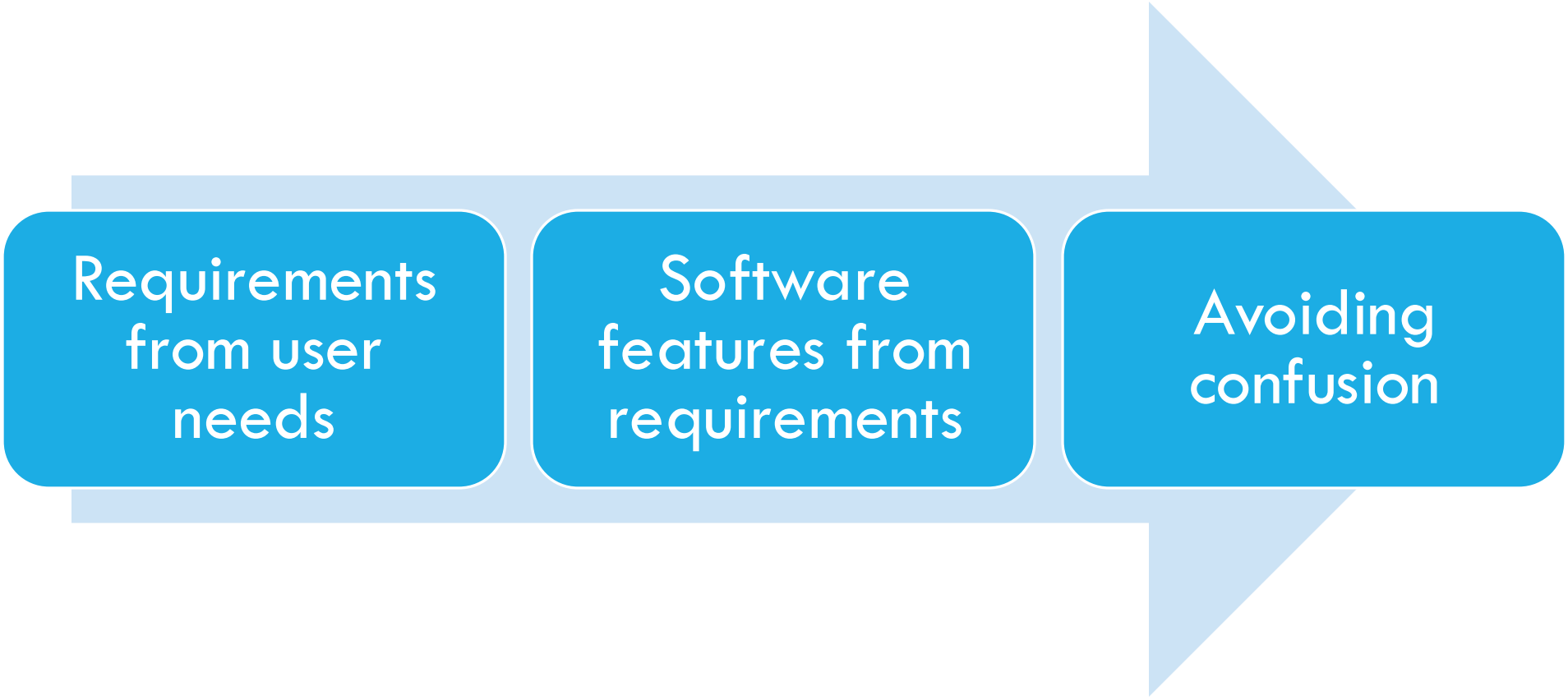
SUCCESSFUL SOFTWARE PRODUCT

1. On Schedule
2. On Budget
3. To Specification

The most important factor: Timing



REQUIREMENTS



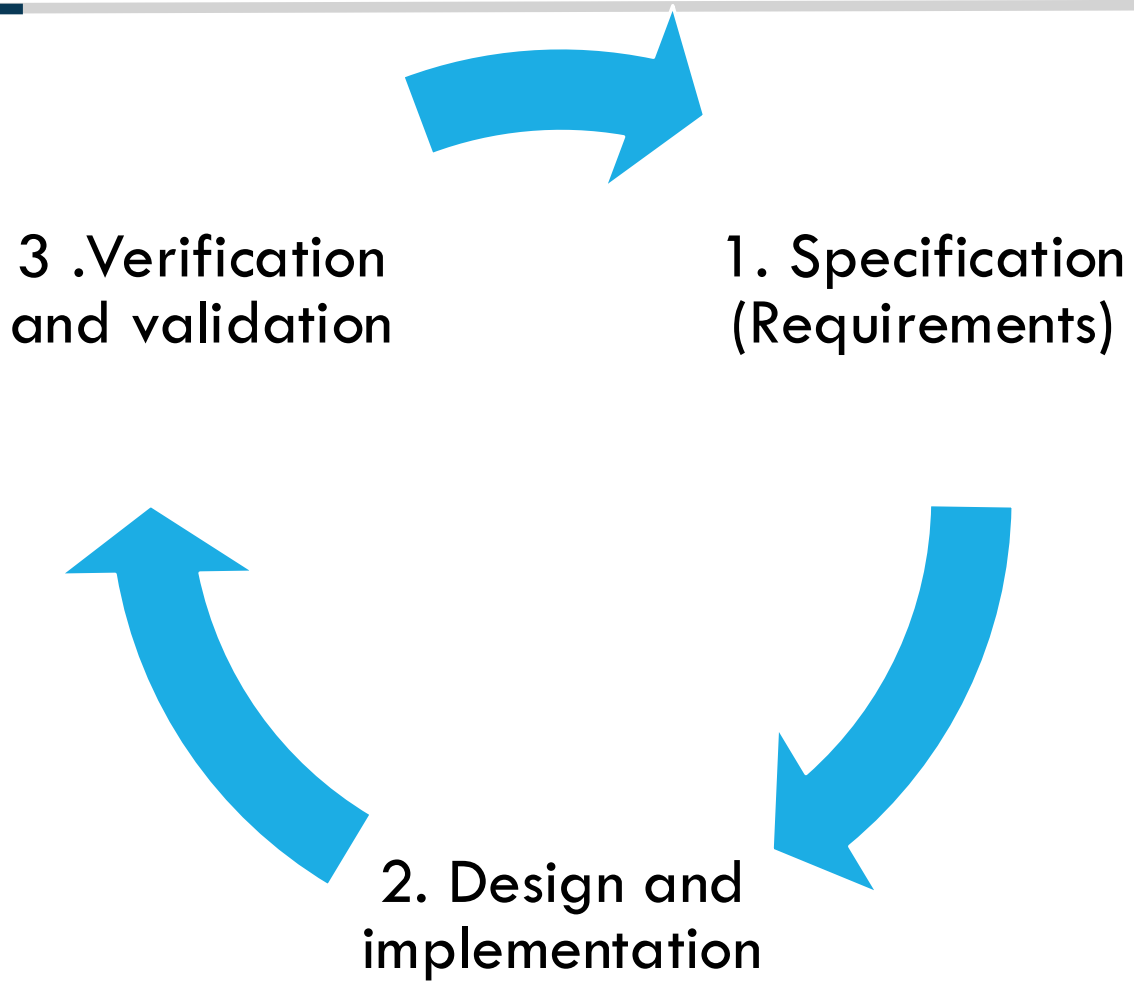
PLANNING

1. Scheduling (manager and developers)
2. Estimations
3. Potential risks

MONITORING

1. Not only the velocity
2. Delivering on time and to specification

PRODUCT LIFECYCLE



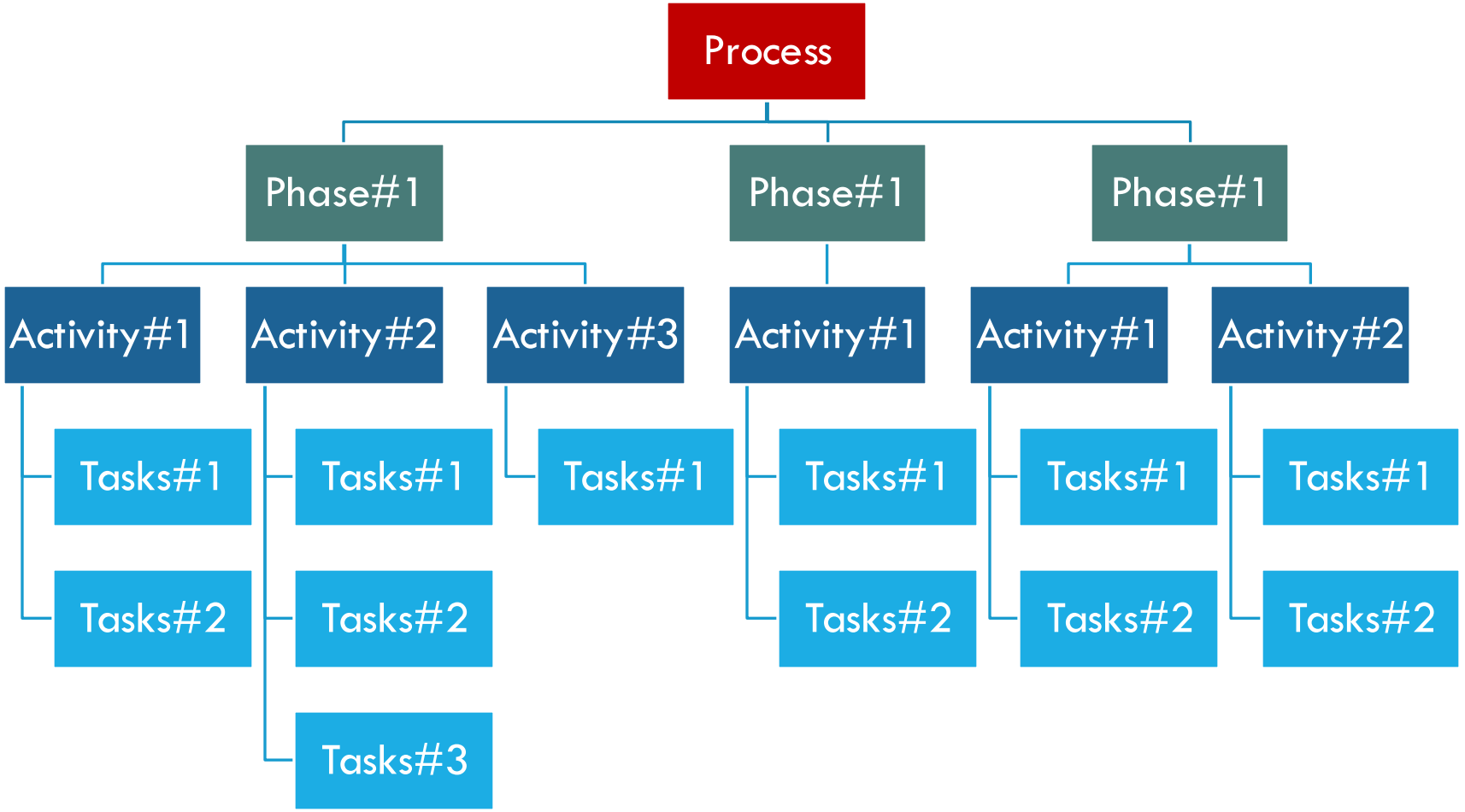
BEST PRACTICES

e.g.

1. Feature selection for releases,
2. Briefing schedules,
3. Time estimations

Methodology: Set of practices (e.g. Scrum is an agile methodology)

FROM PROCESSES TO TASKS



TASKS

1. Activities contain tasks
2. Colleagues (roles) perform tasks
3. Tasks consume resources
4. Tasks produce and use software work products

Dependencies!

TASKS AND ROLES

A role perform a task e.g.

1. Manager,
2. User, client,
3. CTO, CEO,
4. Programmer, coder,
5. Tester

TASKS AND DELIVERABLES

A deliverable is a part of the final product

Deliverables are e.g.

1. Source code,
2. Tests,
3. Documentation,
4. Design,
5. Requirements

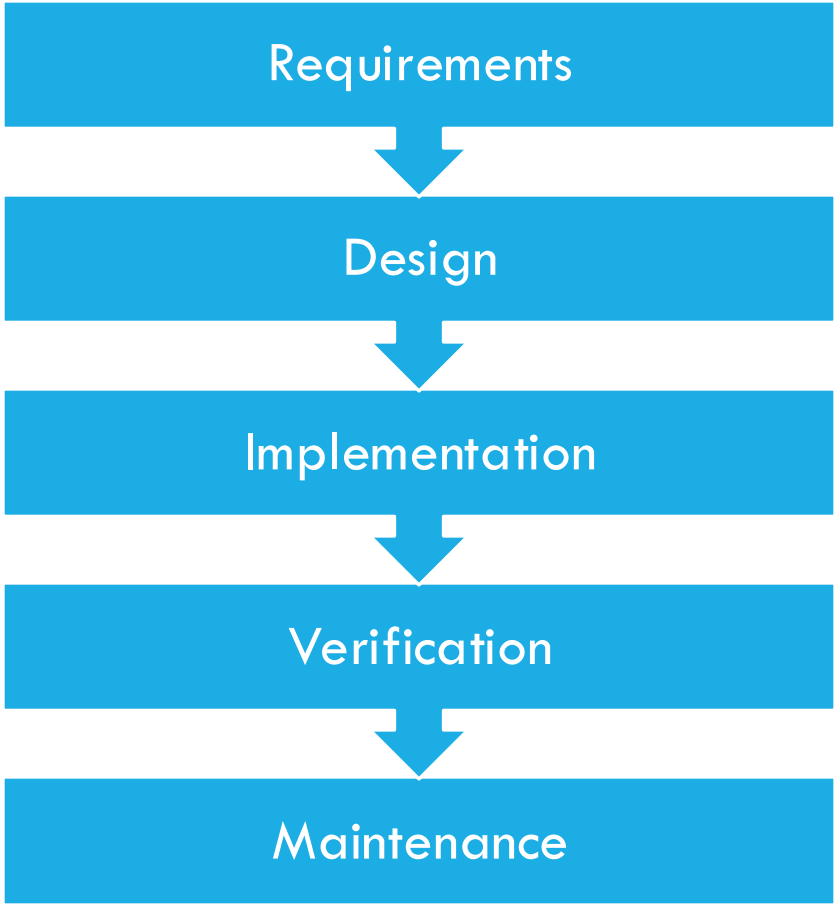
Tasks sometimes use software work products

TASKS AND RESOURCES

1. Money
2. Time
3. Know-how
4. Human resources
5. Technology

LINEAR MODELS – WATERFALL MODEL

1. Not very adaptable to changes
2. Emphasis on documenting
3. Delivered to the client only at the end of the full process



AGILE MANIFESTO

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

SCRUM

Scrum Master / Product Owner (Stakeholders) / Team (3-9)

Sprints (1-4 weeks) / **Product Backlog** (goal selection) / **Demo**

Verbal **communication**

Adaptation

Sprint meetings (sp. planning, daily sc. , sp. review, sp. retrospective)

Working prototype

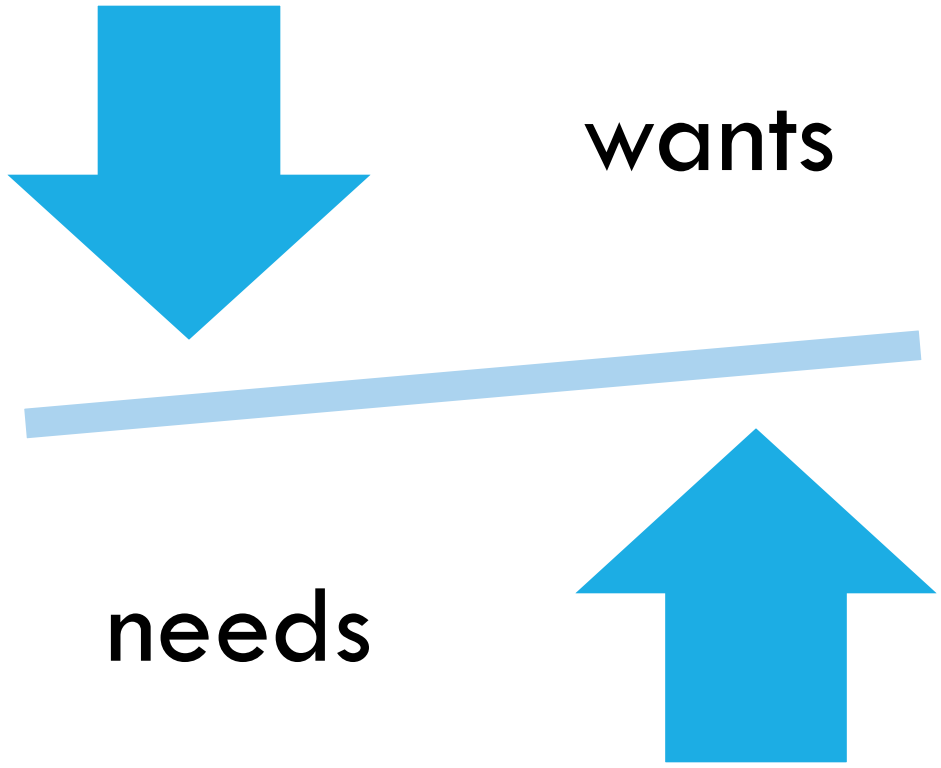
CLIENT NEEDS AND REQUIREMENTS

Difference between “wants” and “needs”

Wants: Desired function in the product

Needs: Required functions to solve specific problem

Analogy: wants/needs -> do things right/do the right things

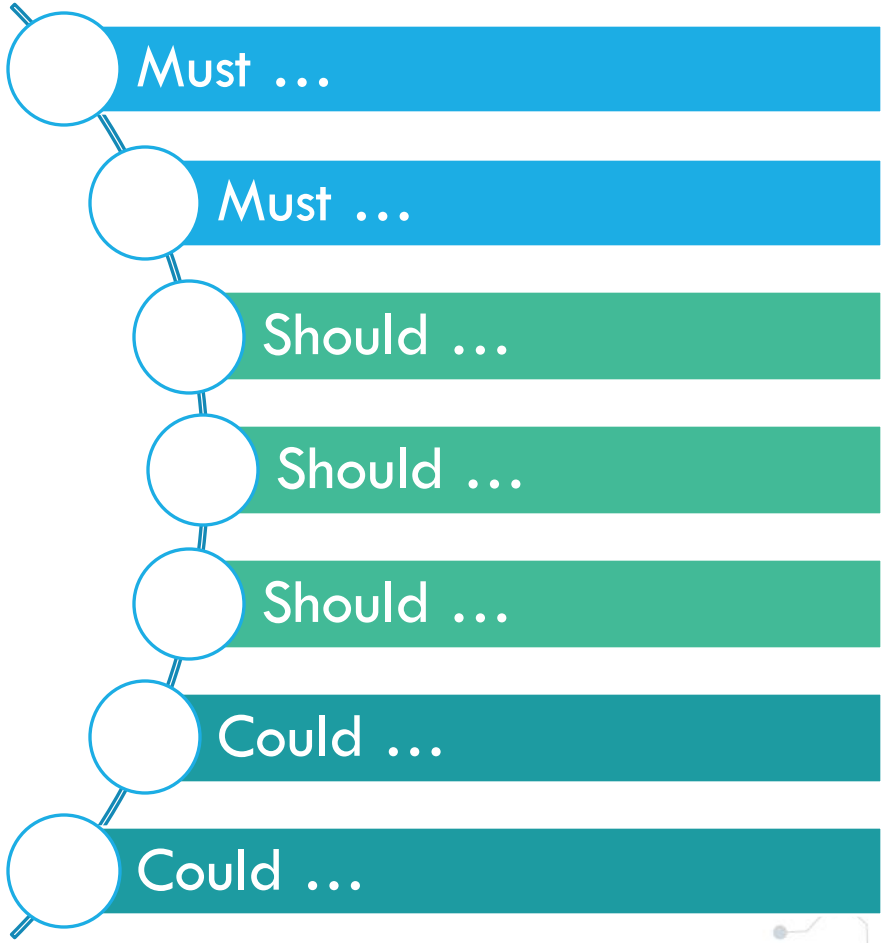


PRIORITIZING AND ANALYZING REQUIREMENTS

1. List of requirements

2. Three importance categories
   - Must be done
   - Should be done
   - Could be done

3. Make sure that requirements are **reflect the product**



REQUIREMENTS

Business requirements and rules e.g.

- Brand uniformity
- Privacy policy
- Regulations and standards

User requirements

- **Use cases**: Show the relationship between software and user (later)
- **User stories**: Who? What? Why? (later)

REQUIREMENTS

Functional requirements

- What should the product do?
- Translate user requirements to system requirements
- e.g. Information flow diagram

Non-Functional (Quality) requirements

- How well the product should do?
- Quality factor of the product (e.g. safety, security, accuracy, dependability, usability, efficiency, performance and maintainability)

TECHNICAL WRITING

Principles

- Non-ambiguous, meant to be boring
- Repeat the nouns (no it, no that/which)
- Simple sentences (Hamingway app)
- Based on a defined glossary (one term per concept)
- Common language for all stakeholders, development team, management etc.

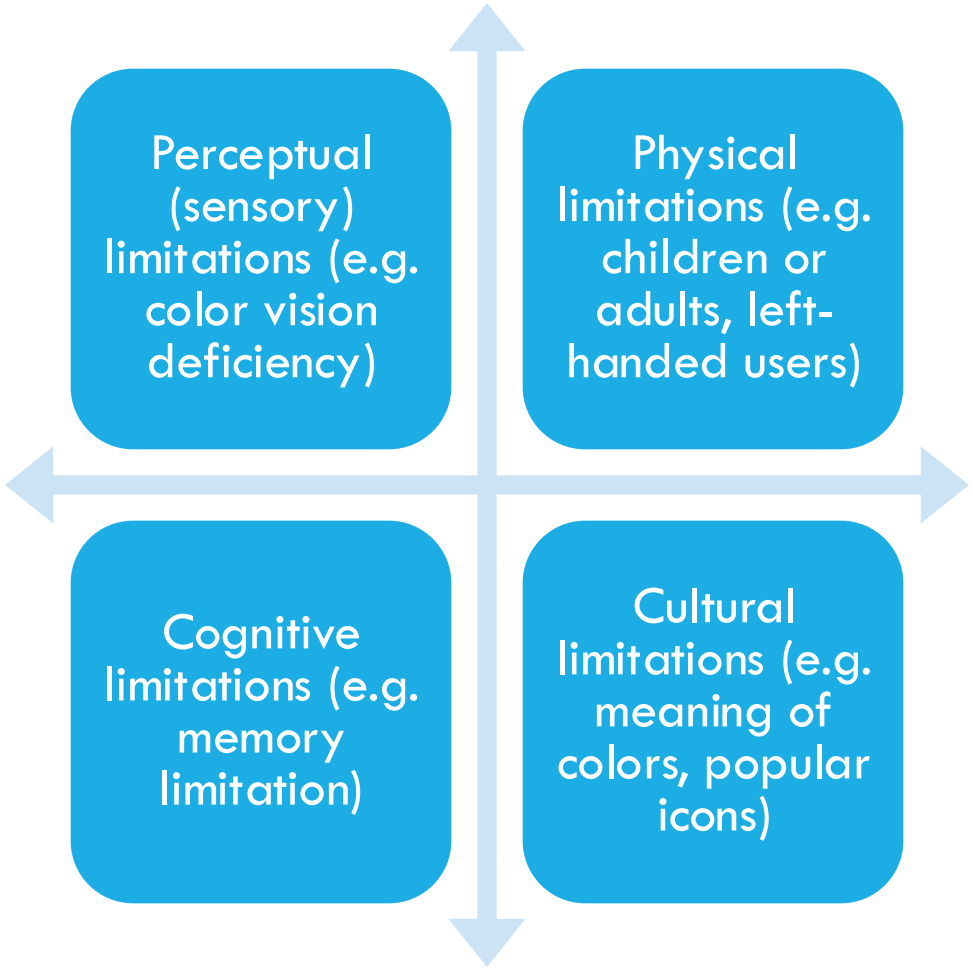
Markdown

- Version control (GitHub-GitLab)

QUICKSTART
Over 35 years in Technology Education

MODULE 2:
USER EXPERIENCE DESIGN

LIMITATIONS OF USERS



DESIGN VERSUS REQUIREMENTS

Requirements: What?

Design: How?

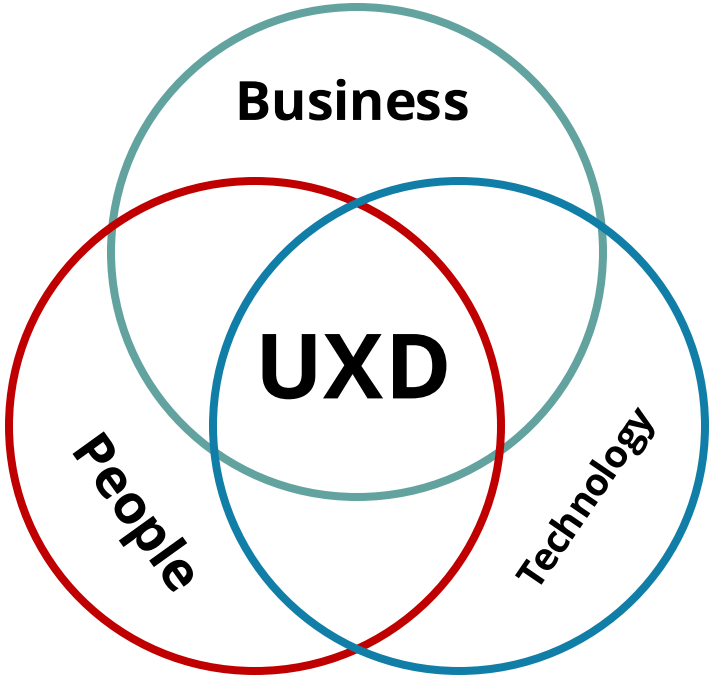
USER EXPERIENCE DESIGN

Interfaces that are usable and useful

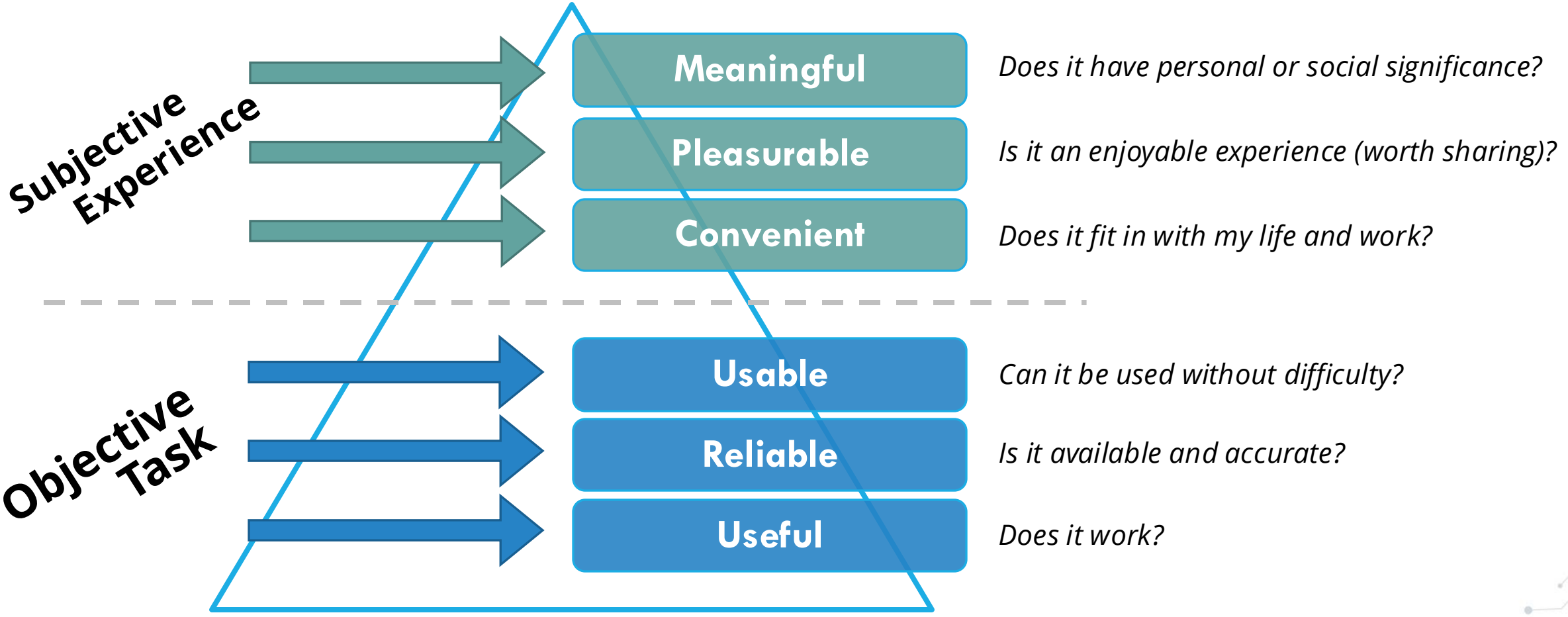
- **Usable**: efficiency and satisfaction during usage
- **Useful**: The user can complete a task

Design cycle

- **Requirements**: understand how users are completing tasks now
- **Design**: develop new interfaces to complete the task
- **Prototypes** and mockups
- **Test** and evaluation: usability and usefulness



UX PYRAMID


Subjective Experience
Meaningful
Pleasurable
Convenient
Does it have personal or social significance?
Is it an enjoyable experience (worth sharing)?
Does it fit in with my life and work?
Objective Task
Usable
Reliable
Useful
Can it be used without difficulty?
Is it available and accurate?
Does it work?

DESIGN

Usability

- Effective
- Efficient
- Satisfying

Affordance: properties that determine how the software could be used

Signifiers: signals that communicate what actions are possible and how they should be done

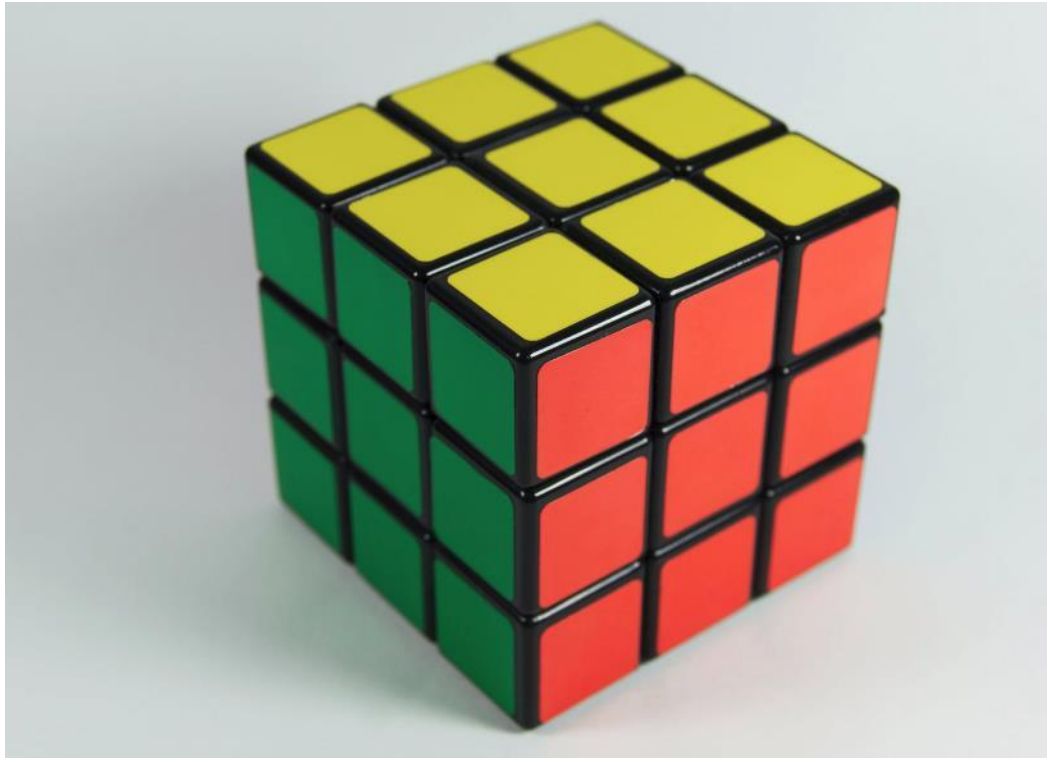
Feedback: information to the user about input has occurred

PROBLEM SPACE

Understand users and tasks

1. Naturalistic observation
2. Surveys
3. Focus group method: directed conversation with 5-7 users
4. Interview: one-on-one conversation between designer and user

Primary results: user characteristics and **personas**



PERSONAS

"Who are we designing for?“

Fictional characters based on real people

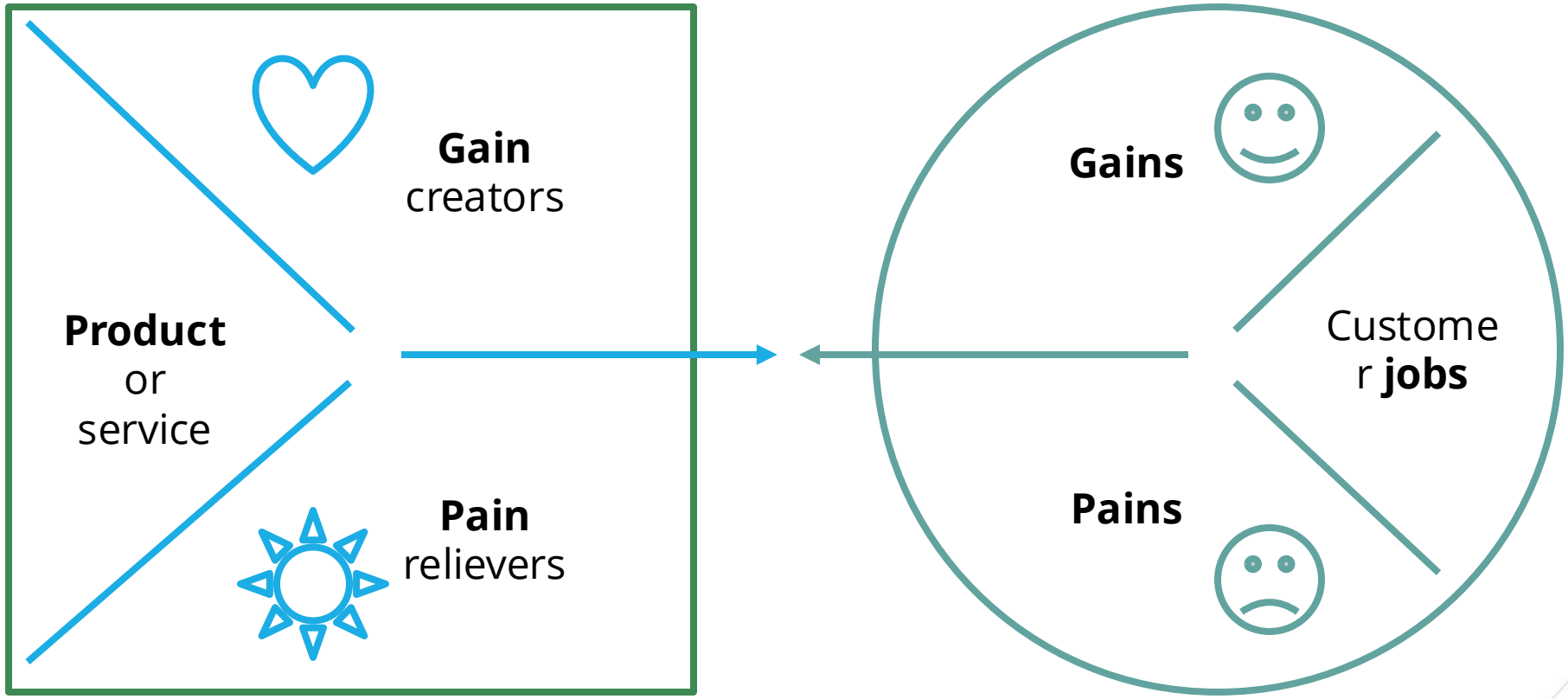
Represent a user group based on preliminary research

Properties

- Optional name
- Environment (e.g. social)
- Responsibilities (e.g. job)
- Demographic features
- Pains
- Gains
- Attitudes
- User quote to represent user’s features above

VALUE PROPOSITION CANVAS

1. The **pain or gain** the product addresses
2. The customer **segment** it serves
3. How the product **compares** to competitors

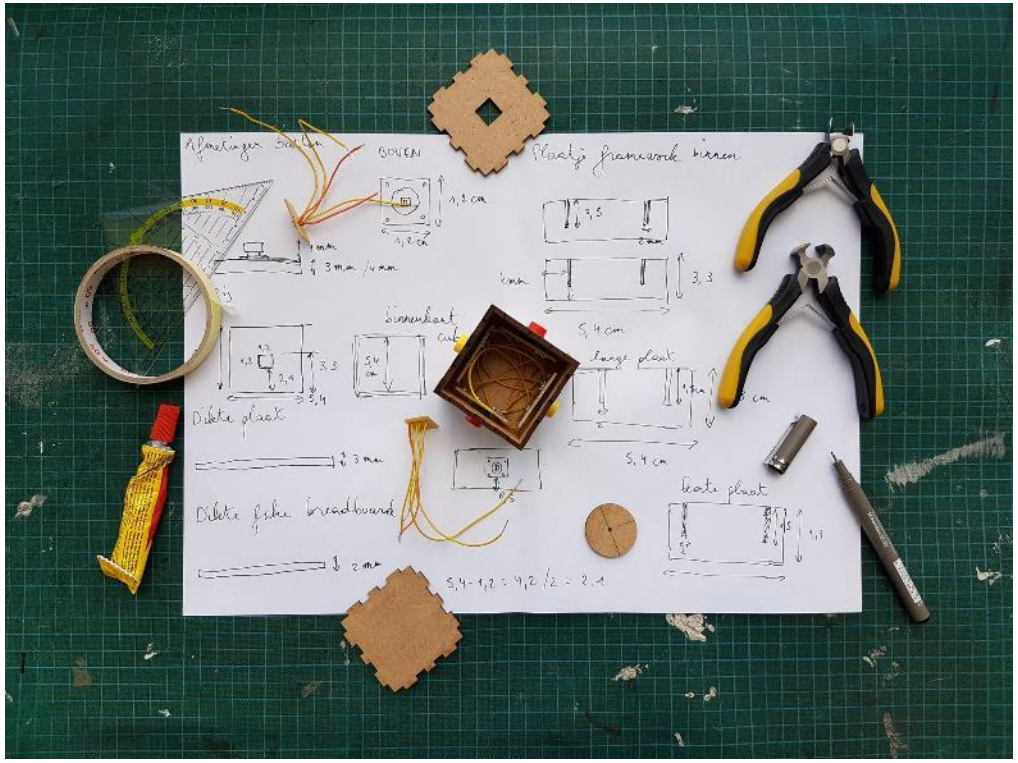


UX PROTOTYPES

An **early model** of new design

Low- and high resolution or fidelity prototypes

Why? Save resources, help design



EVALUATION

Goal: **usefulness** and **usability**

- Learnability
- Memorability

Measure

- Amount of time
- Number of clicks
- Number of errors
- Mental effort
- Affective status

Heuristic evaluation

Cognitive walktrough

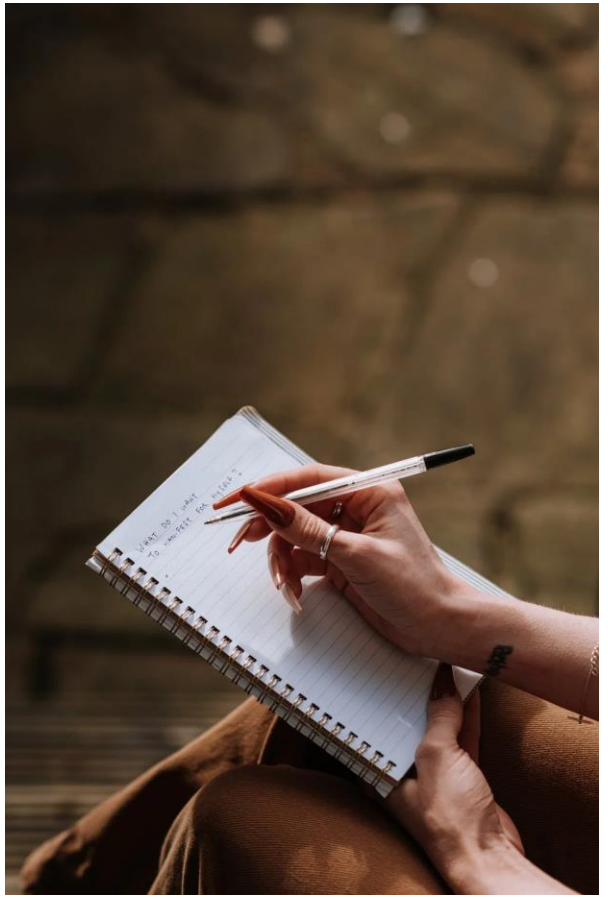
HEURISTIC EVALUATION

A heuristic evaluation is a usability inspection method for software that helps to identify usability problems in the user interface

It involves evaluators examining the interface and judging its **compliance with recognized usability principles (the heuristics)**

e.g. Nielsen's heuristics

- *e.g. The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.*
- *e.g. Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions*

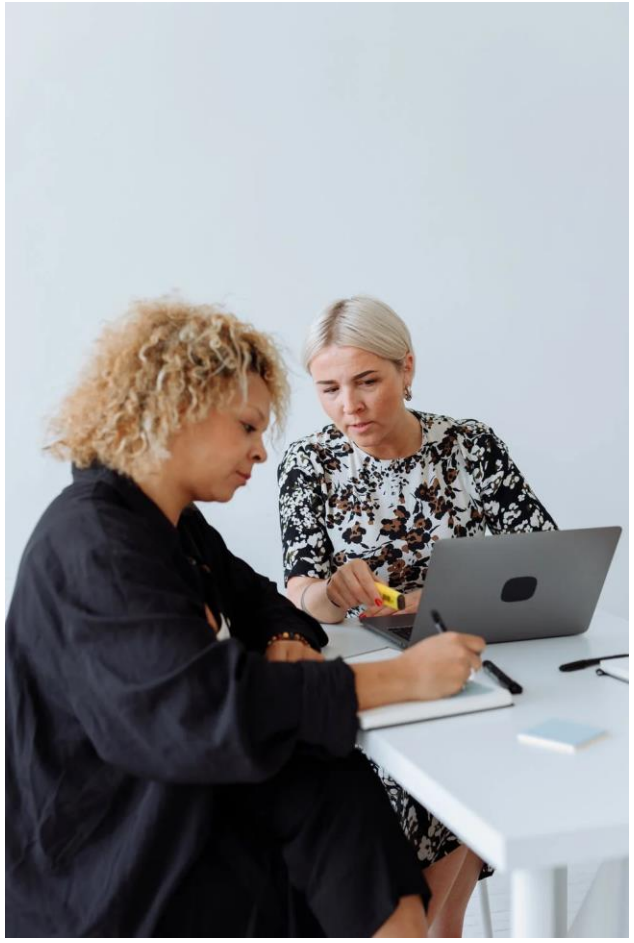


COGNITIVE WALKTROUGH

The cognitive walkthrough method is a usability inspection method used to identify usability issues in interactive systems, focusing on **how easy it is for new users to accomplish tasks with the system**

Task-specific, whereas heuristic evaluation takes a holistic view to catch problems

Users typically prefer to **learn a system by using it** to accomplish tasks, rather than, studying a manual



GATHER INFORMATION

Interview users and the client

- Using “Why?” open questions (not “yes or no questions”)

Observe users using the software

USER STORY

US: Clearly outlines a specific requirement

Format

1. As a … (stakeholder/user role)
2. I want to … (user tasks or functions)
3. So that I can … (reason why the user do the task)

e.g.

1. As a customer
2. I want to select an ingredient from a list
3. So that I can order a special pizza

GOOD USER STORIES

Independent

Not too specific with many technical details

Valuable to the client

Estimatable development time

A not too big chunk of development work

Verifiable

EPIC

An epic is a huge user story that is too large to fit into a sprint

An epic is a collection of user stories with a unified goal

Breaking up epics into multiple user stories

WIREFRAME (OR MOCKUP)

Basic **visual** (schematic) **representation of the product**

Focus on **basic functions** and **main user tasks**

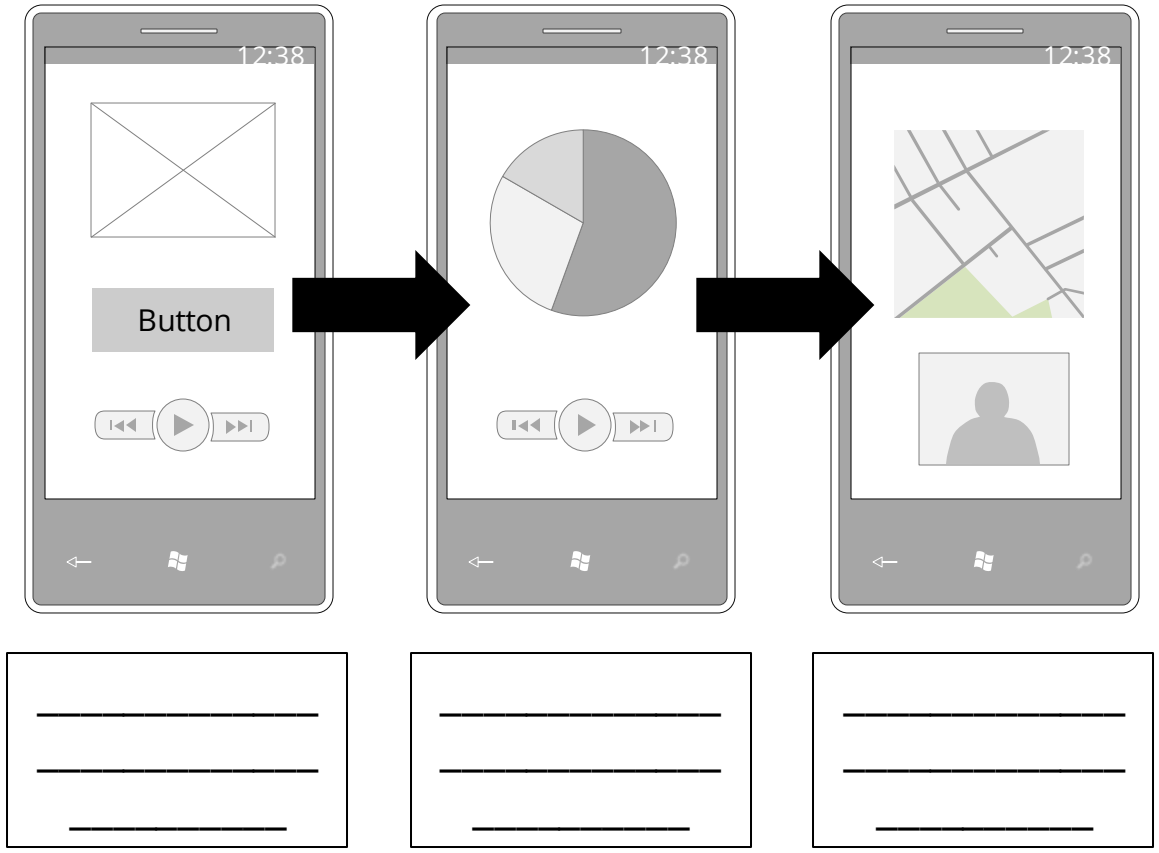
For specifying **requirements** (not only for **UX/UI design**)

STORYBOARD

Sequential visual representation of a user-software interaction

Every storyboard cover a **specific** situation (and specific user)

Combination of **wireframes** and **scenarios** from use cases



WIREFRAMING

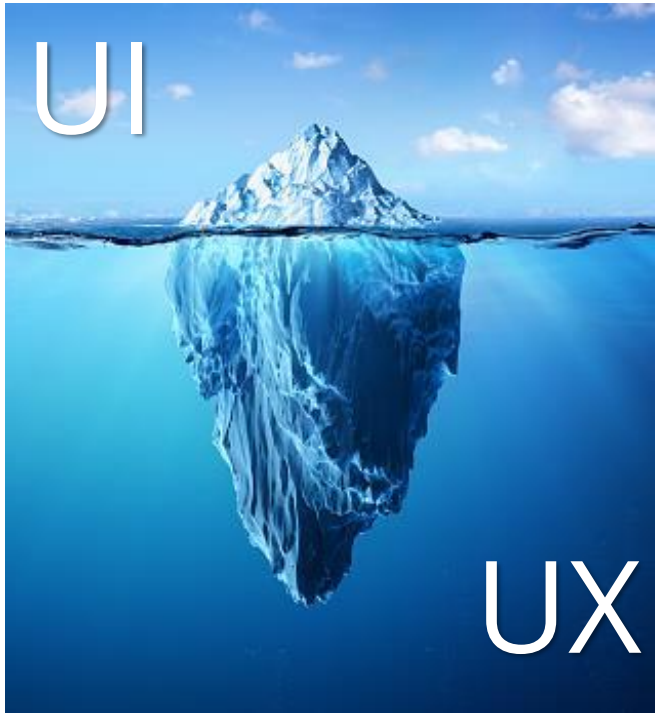
Tool: https://www.diagrams.net/

+ demo + file

UX VERSUS UI

UX

- Interaction design
- Wireframe and prototypes
- Information architect
- User research
- Scenarios
- Accessible, desirable
- Convenient, satisfiable



UI

- Visual design
- Color schemes
- Graphic designer
- Layouts
- Typography
- Images, form, buttons
- Content

DESIGN SYSTEMS AND COMPONENT LIBRARIES

A **Design System** is a set of interconnected patterns and shared practices coherently organized.

Design Systems aid in digital product design and development of products such as apps or websites.

Design Systems may contain, but are not limited to, pattern or **component libraries**, design languages, style guides, coded components, brand languages, and documentation.

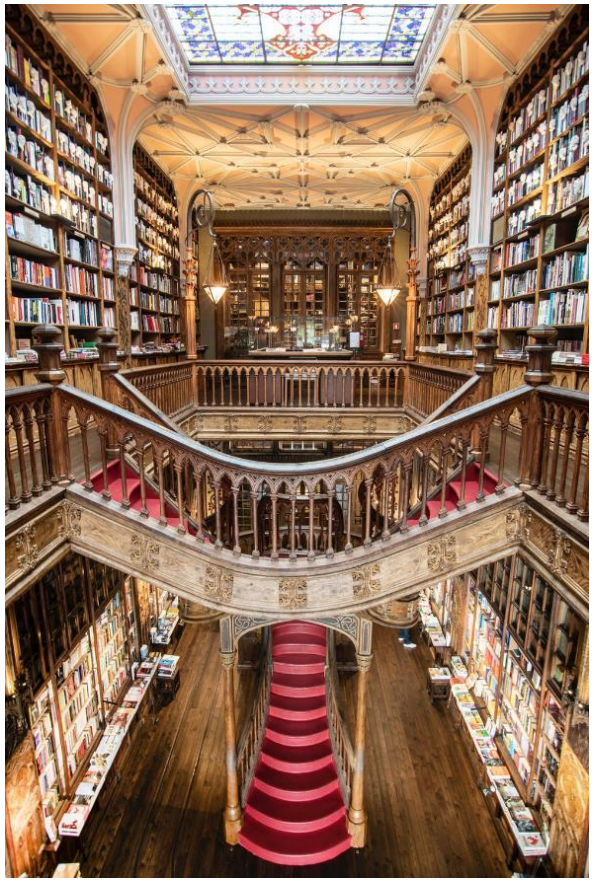
Component Libraries: An organized set of related, reusable component, containing code examples, design guidelines and use cases

COMPONENT LIBRARIES

A **component library** is a set of re-usable components.

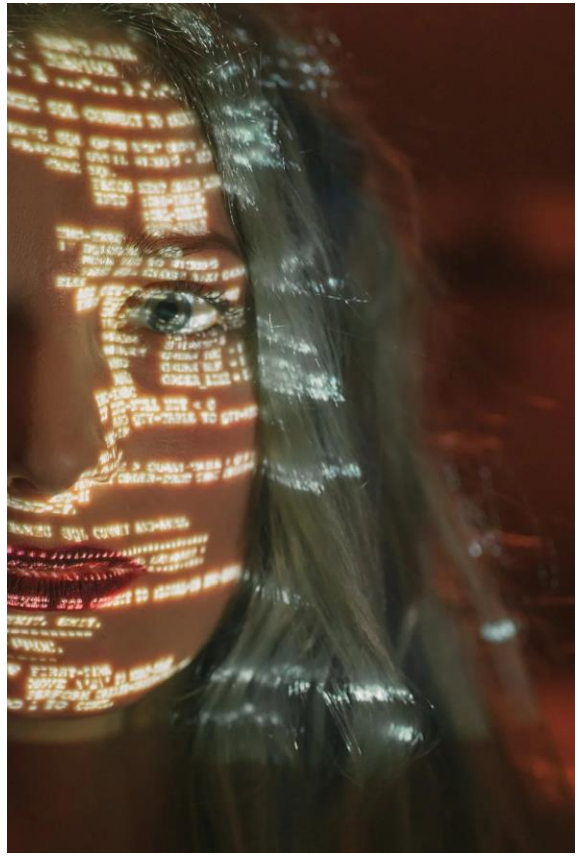
Examples:

- Vaadin: https://start.vaadin.com/welcome
- Wired Elements: https://wiredjs.com/
- UI5 Web Components: https://sap.github.io/ui5-webcomponents/
- Patternfly: https://www.patternfly.org/v4/



DESIGN SYSTEMS

A **design system** is a collection of documents, examples, code snippets, screenshots, design guidelines, components, and other assets for a product design company.



Examples:

- Google Material: https://material.io/
- Apple Human Interface Guidelines: https://developer.apple.com/design/
- Microsoft Fluent: https://www.microsoft.com/design/fluent
- Shopify: https://polaris.shopify.com/