QUICKSTART
Over 35 years in Technology Education

Principles of Software Engineering 1

Topics covered week 1 / session 1

- Module 1: Software Engineering
  - Lesson 1: Software
  - Lesson 2: Engineering
  - Lesson 3: Development process
- Module 2: Software Development Lifecycle
  - Lesson 1: Lifecycle Overview & Requirement
  - Lesson 2: Specification
  - Lesson 3: Design & Implementation
  - Lesson 4: Testing & Maintenance

Recap: Software Engineering

What is Software ?

- Software development is an engineering process.
- Unlike tangible products, software is intangible and abstract.
- Software = Code + Documentation + hardware environment + continuous maintenance + quality control + support + security rules + service level agreements
- Continuous maintenance, quality assurance, and security measures are integral to software development.

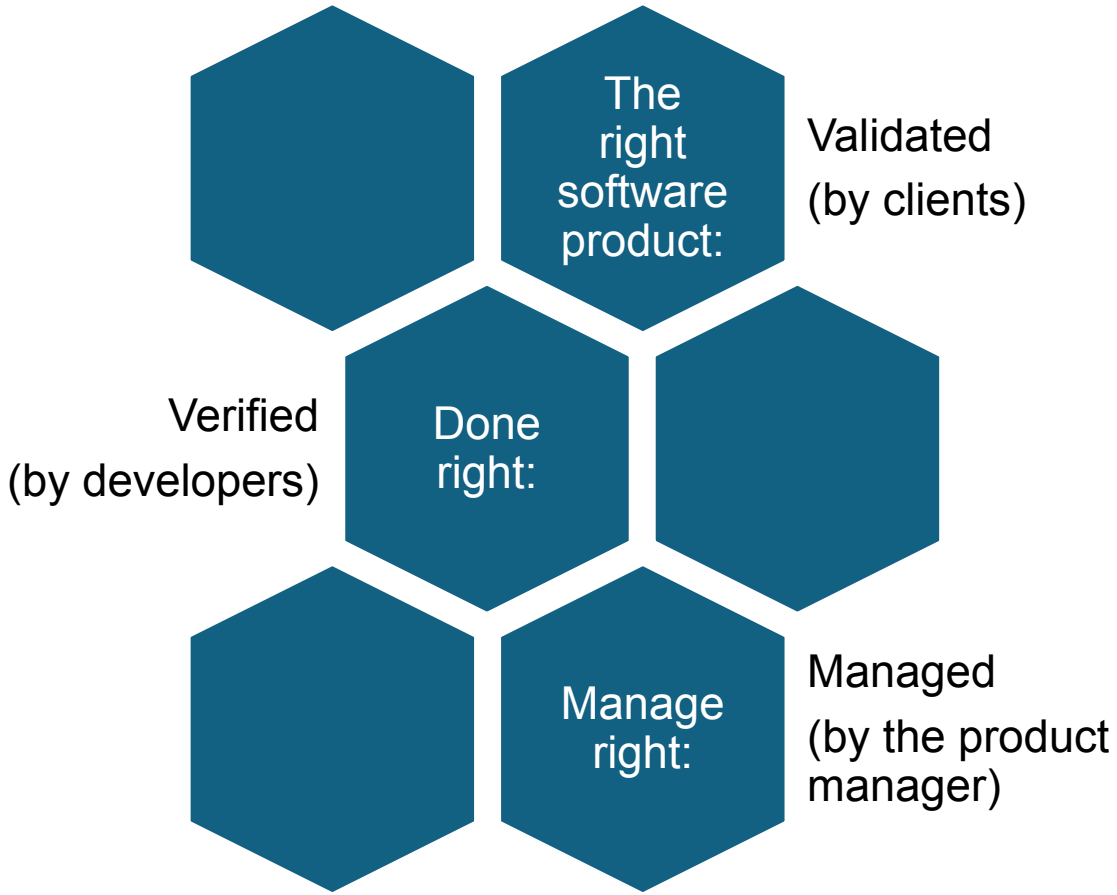
Recap: Software Engineering

Software Quality

Continuous Maintenance:
Regular updates for new features, bug fixes, and performance improvements.

Quality Assurance:
Quality of the development process determines the quality of the end product.

Security Measures:
Authentication, authorization, and encryption protect user data.

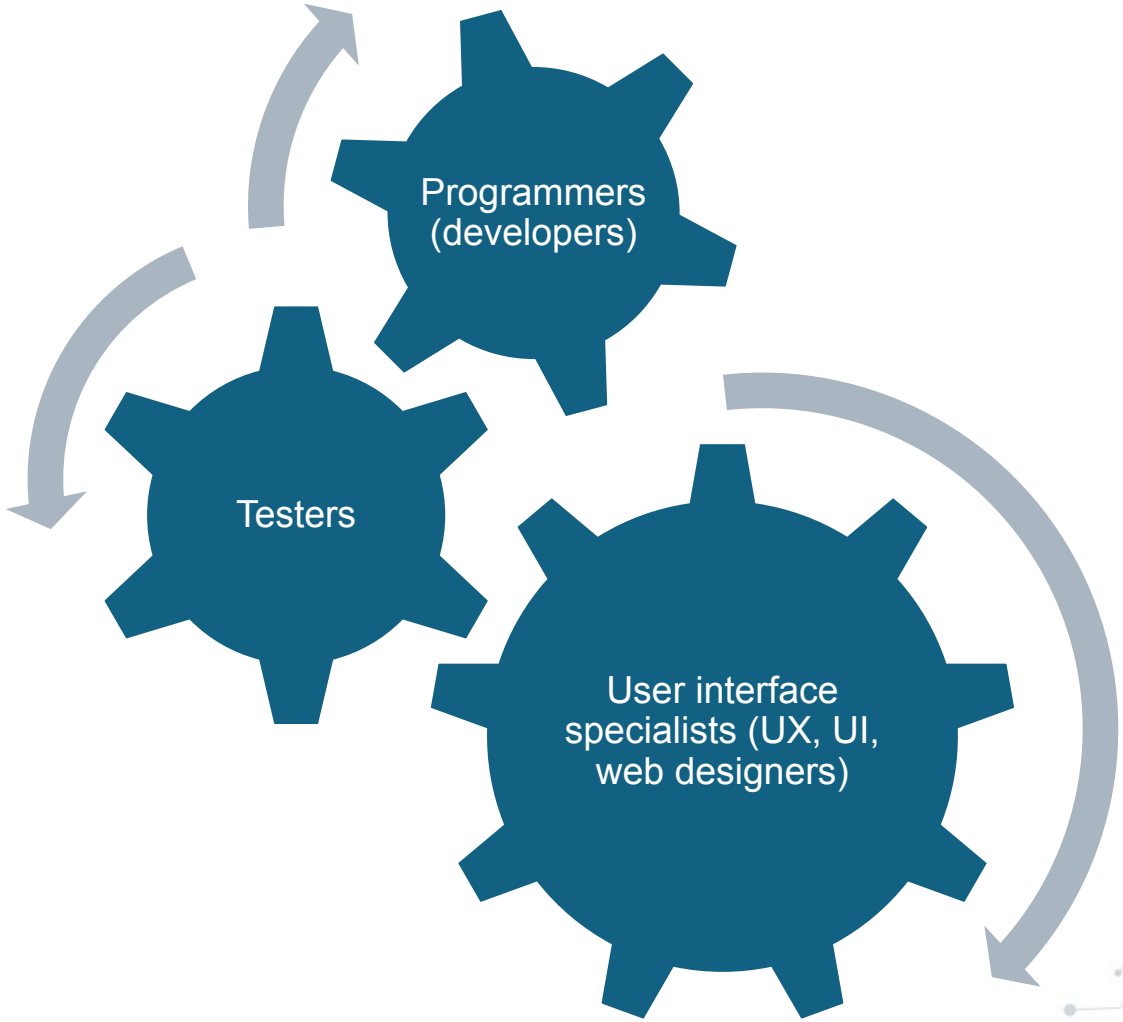


Recap: Software Engineering

Software Development Team

Teamwork:

Collaboration among developers, testers, UX/UI designers, product managers, and project managers ensures successful software development.



Recap: Software Engineering

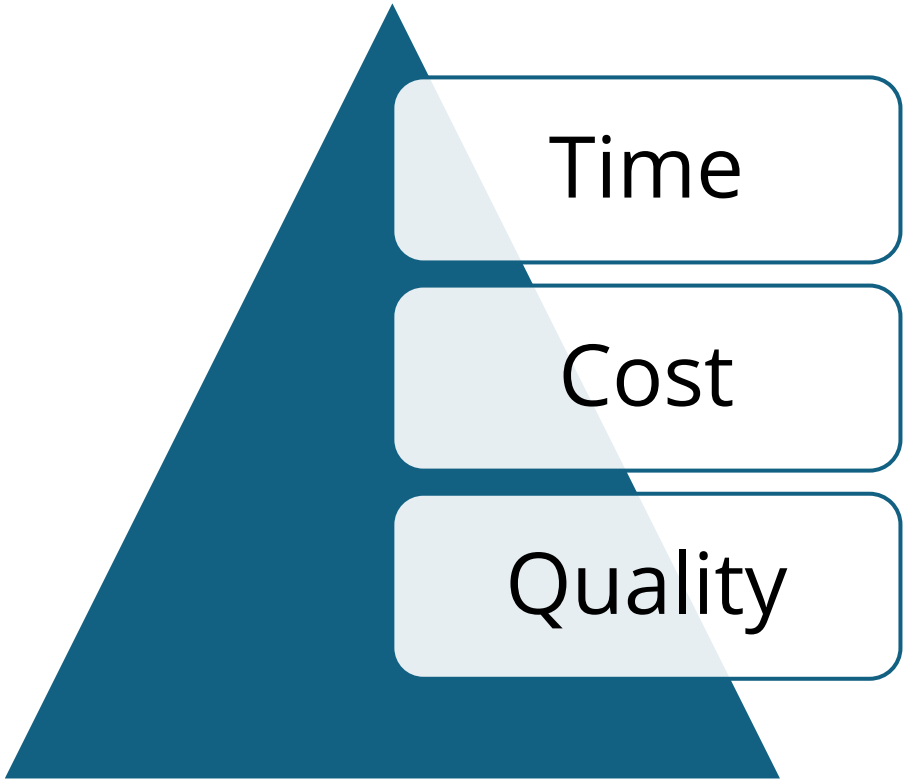
Engineering

- Software development is an engineering activity, not just a hobby.
- Engineering optimizes resource utilization and process efficiency while serving humanity's needs.
- Engineering applies scientific methods to ensure reproducibility and quality in software development.
- Engineers prioritize compliance with functional and nonfunctional requirements, including accessibility and data protection.
- Compliance with legal and ethical standards, such as data protection regulations, is essential.

Recap: Software Engineering

Project Management Trilemma

- Project managers often prioritize two out of three factors: time, cost, and quality, sacrificing the third.
- As an engineer or quality control specialist, there's an opportunity to influence and improve all three factors simultaneously.
- Software development often requires balancing these factors to meet project objectives effectively.
- Underestimation of project complexities can lead to challenges in meeting time, cost, and quality goals.



Recap: Software Engineering

Development Process

- The development process serves as a pipeline, transforming user requirements (input) into working software (output).
- It involves stages like requirement formalization, specification creation, software design, implementation, testing, and continuous maintenance, guided by engineering principles.
- Different projects require different processes, there is no one size fits all process.
- Different stakeholders view the software in a different way. As a developer, we have to talk to all stakeholders

Recap: Software Engineering

Development Process

- Formalized requirements and technical writing help ensure clarity and specificity throughout the development process.
- Documentation serves various stakeholders and plays a critical role in ensuring clarity, understanding, and consistency throughout the development lifecycle.
- Deliverables are e.g. Source code, Tests, Documentation, Design, Requirements

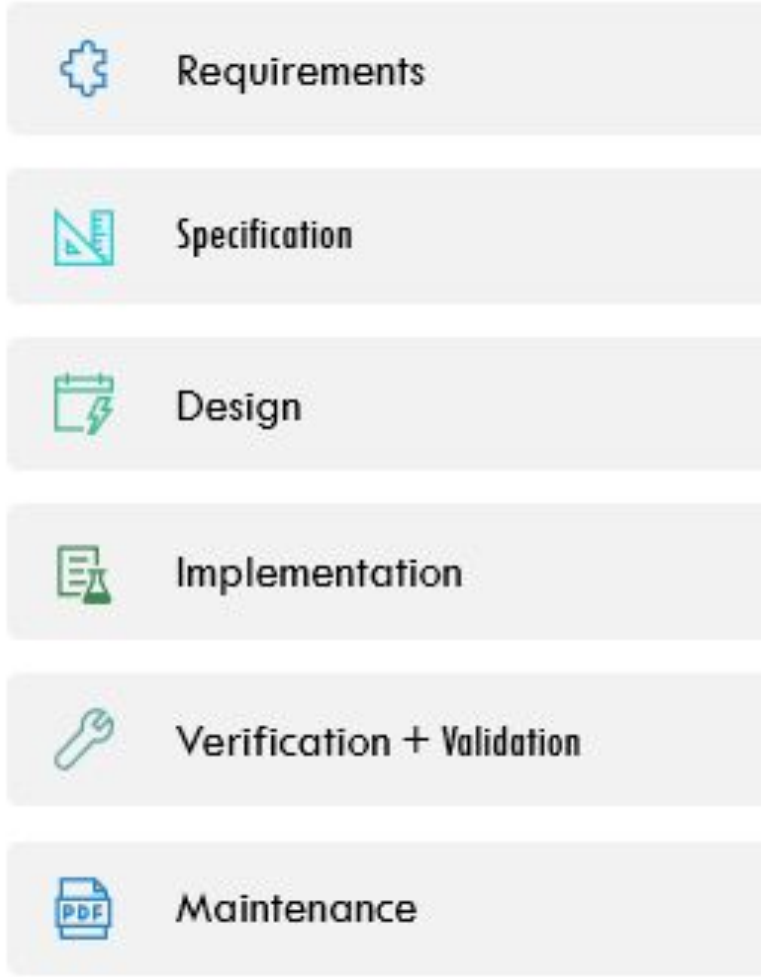
Knowledge Checks

Software Engineering

- What is the project management trilemma, and what are its components?
- Discuss the potential consequences of sacrificing one aspect of the project trilemma over the others.
- How can engineers influence all three components of the trilemma simultaneously?
- Why is it important for development processes to be adaptable?
- Describe the significance of documentation in the software development process.
- What is continuous maintenance in software development?
- How does effective communication among team members contribute to project success?

Recap: Software Development Lifecycle

Summary



Recap: Software Development Lifecycle

Requirements

- Requirement analysis: discover and sufficiently describe the problem

- Functional & Non-functional Requirements

Requirement Formalization

- Make sure requirements can be verified in a quantifiable way
- Specify exact response times, availability etc.
- Quantified requirements defend the interest of both parties
- Technical writing
- Be open to feedback from later phases

Recap: Software Development Lifecycle

Specification

- Describe an abstract solution that satisfies the requirements
- Formalize what data are present in the system, how processes transform data in the system and how processes are to be synchronized in the system (concurrency, asynchronous calls, timing)
- Data Flow Diagrams
- Entity Relationship Model
- Temporal Modelling – State diagrams

Recap: Software Development Lifecycle

Design & Implementation

- Create a formal plan that can be executed to create the solution
- Types of Design: Architectural & Detailed
- Object Oriented Design Principles: Encapsulation, Abstraction, Inheritance, Polymorphism
- Unified Modeling Language (UML)
- Design Patterns: Creational, Structural, Behavioral

Recap: Software Development Lifecycle

Testing & Maintenance

- Verification and Validation: check if the implemented solution meets the requirements
- Types of Testing: Acceptance testing, Integration testing, Unit testing, Performance testing, Smoke testing
- Maintenance: patches, security vulnerabilities, new feature requests
- Maintenance Types: Corrective, Adaptive, Perfective, Preventive

Knowledge Checks

Software Development Lifecycle

- What are the key phases of the Software Development Lifecycle (SDLC)?
- What role does Maintenance play in the SDLC?
- Which phase of the SDLC involves creating the overall structure and architecture of the software?
- Which phase of the SDLC involves writing code based on the design specifications?

Topics covered week 1 / session 2

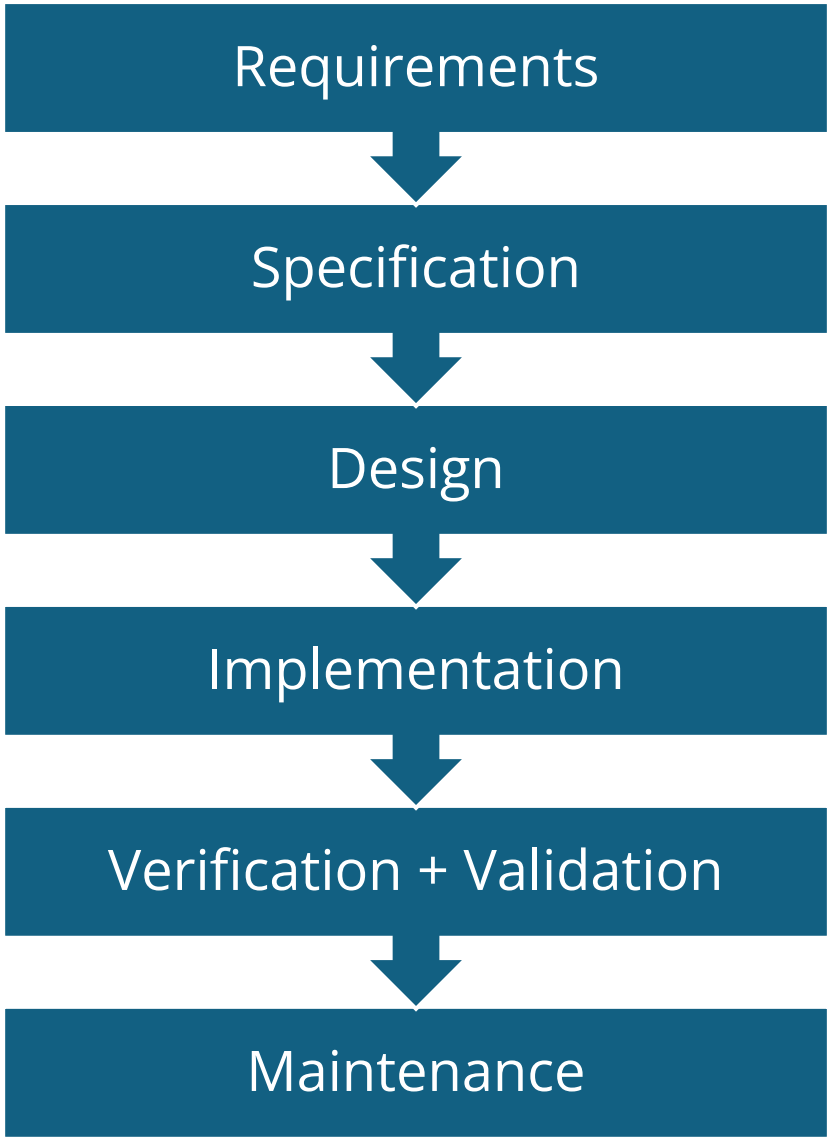
- Module 3: Lifecycle Models and Processes
  - Lesson 1: Lifecycle Models
  - Lesson 2: Agile and Scrum

- Module 4: The Project Team
  - Lesson 1: Product/ Project Manager
  - Lesson 2: UX designer, Engineer/Architects

Recap: Lifecycle Models and Processes

Linear Model

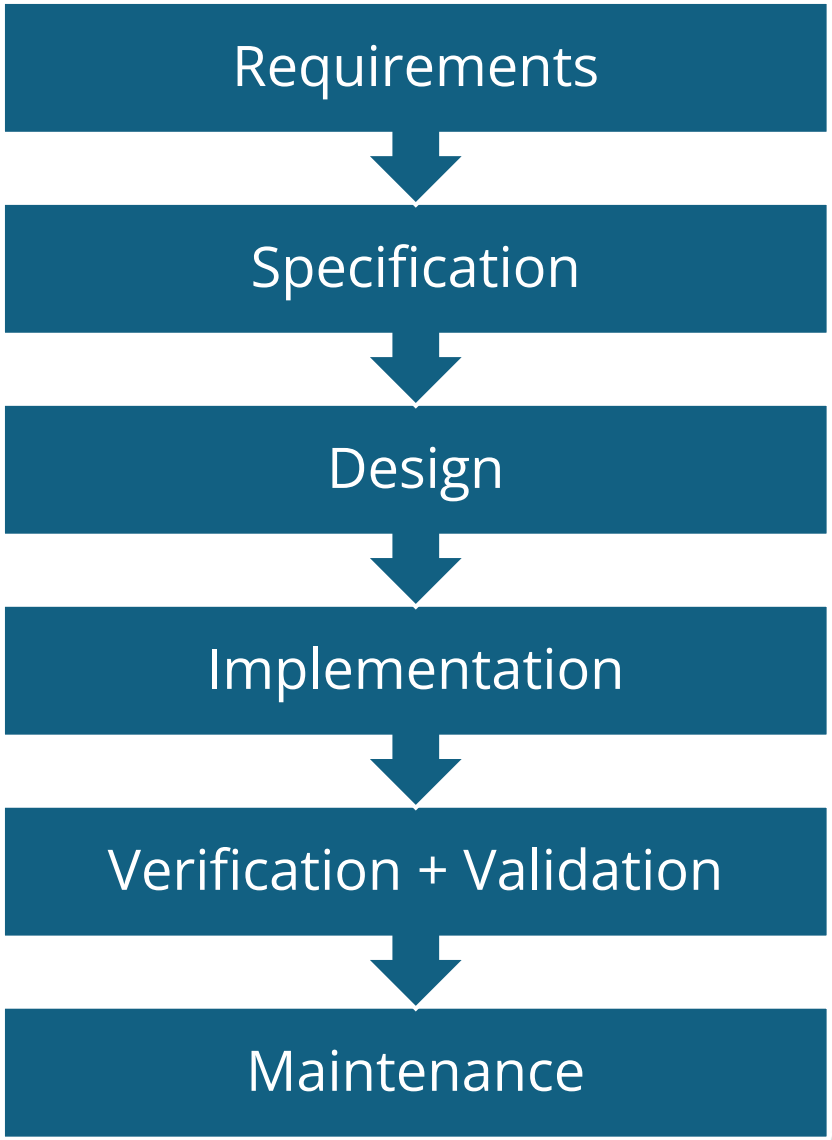
- Not very adaptable to changes
- Emphasis on documentation
- Delivered to the client only at the end of the full process
- The later we discover an error the more expensive it becomes



Recap: Lifecycle Models and Processes

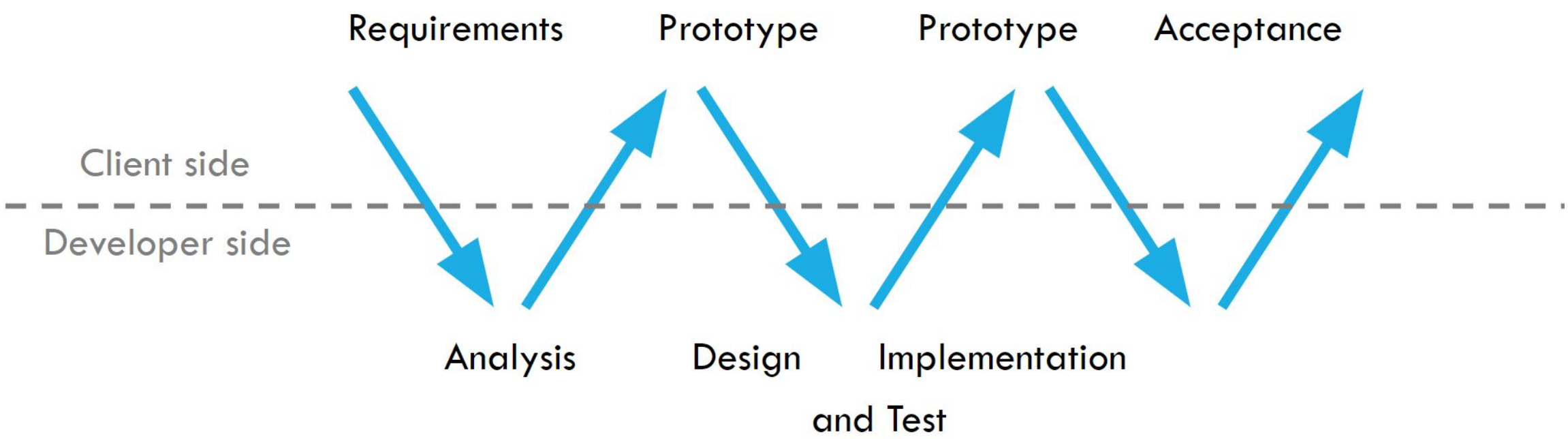
Waterfall Model

- Waterfall model: linear model + feedback loops
- At the end of each cycle, we have acceptance criteria. If we fail, we move back a step
- Validation may link all the way back to specification when errors in specification are discovered
- During the maintenance phase, new requirements may be discovered in the form of feature updates



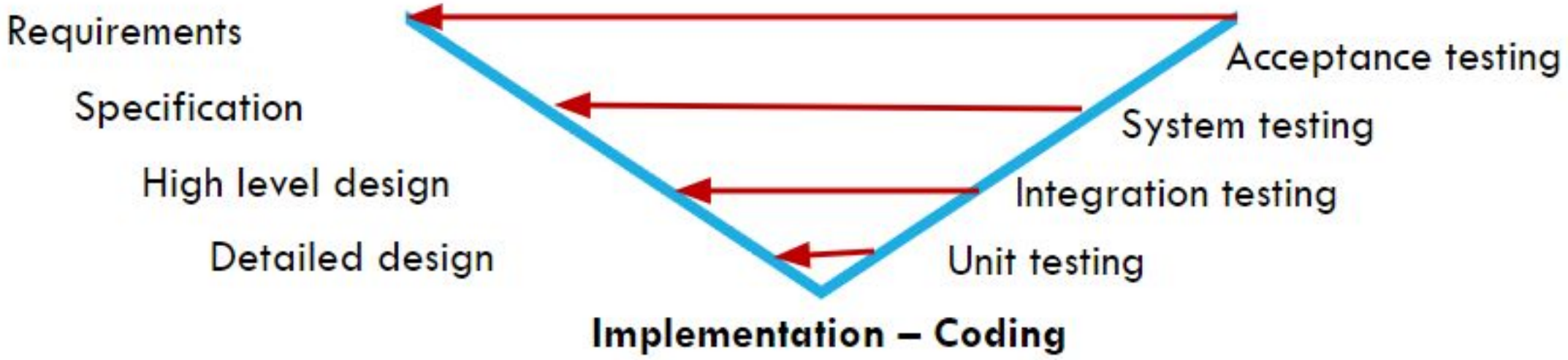
Recap: Lifecycle Models and Processes

Sawtooth Model



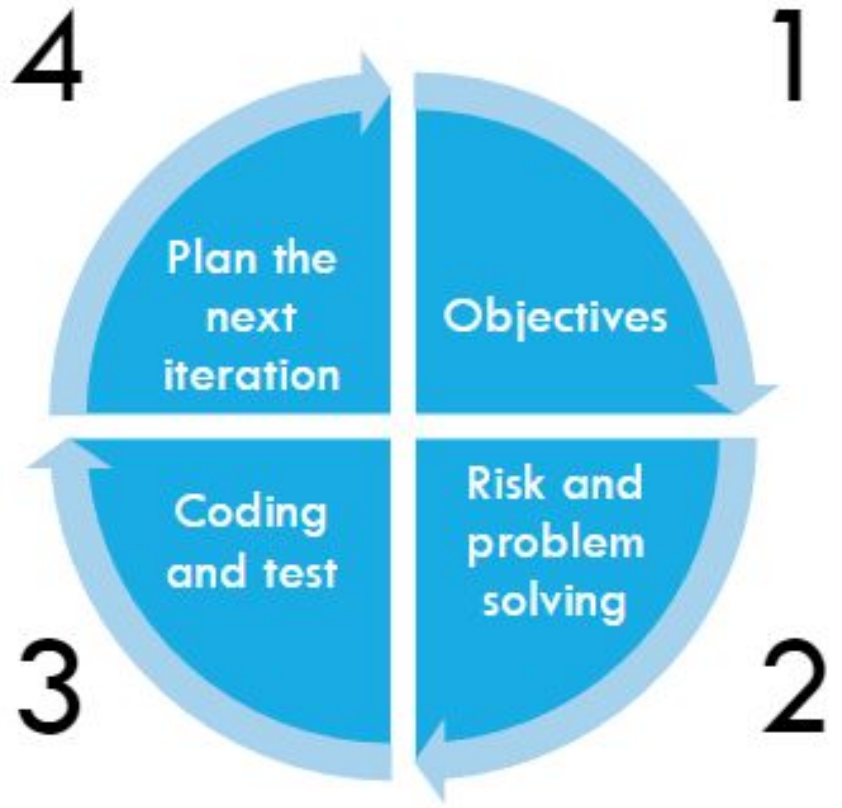
Recap: Lifecycle Models and Processes

V-Model



Recap: Lifecycle Models and Processes

Spiral Model



Recap: Lifecycle Models and Processes

Agile and Scrum

- Agile prioritizes satisfying the customer through early and continuous delivery of valuable software, welcoming changing requirements for the customer's competitive advantage.

- Agile promotes face-to-face communication within teams and focusing on working software as the primary measure of progress, all while encouraging continuous improvement through reflection and adjustment.

- Scrum utilizes sprint cycles (1-4 weeks) where a team, consisting of a Scrum Master, Product Owner, and members, collaboratively work on product backlog items to produce a working prototype, holding regular sprint meetings for planning, review, and retrospective to facilitate adaptation and progress.

Knowledge Checks

Lifecycle Models and Processes

- In the Waterfall model, which phase comes first in the sequential flow?
- What is a key feature of the Sawtooth model?
- The Spiral model is characterized by iterative development cycles, each cycle involving phases of planning, risk analysis, engineering, and evaluation. True or False?
- Which lifecycle model emphasizes the verification and validation of each phase before proceeding to the next, forming a V-shaped structure?

Recap: The Project Team

Product & Project Manager

- Project Team members: Product Manager, Project Manager, UX designer, Engineers/ Architects

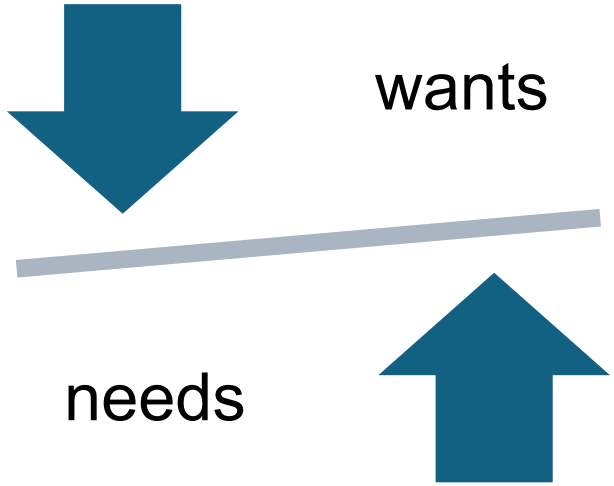
- Role of Product Manager:
  - Ensuring product quality, Managing development process, Interacting with clients, Dealing with the development team
  - Solving a customer problem by leading the team and the project

- Role of Project Managers: Allocate resources, Track progress, Manage risk, Execute the process

Recap: The Project Team

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



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)

Recap: The Project Team

Project Planning and Resource Management

- Planning: Scheduling, Estimations, Potential risks
- Tasks and Resources: Money, Time, Know-how, Human resources, Technology
- Resource Planning: PERT, Gantt Chart

Recap: The Project Team

Role of UX designer, Engineer, Architect

Role of UX Designers

Ensure usability by designing interfaces that are effective, efficient, and satisfying for users, incorporating affordances, signifiers, and feedback mechanisms to enhance user experience.

Role of QA Engineers

Accountable for the quality of the deliverables and its automation.

Role of Architects

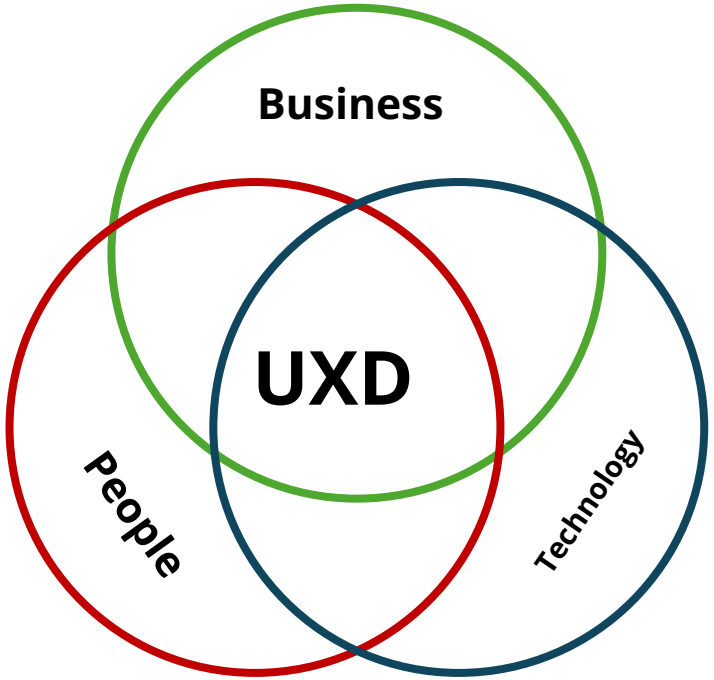
Eliminates expensive mistakes during early phases of the lifecycle by planning properly.

Recap: The Project Team

User Experience Design

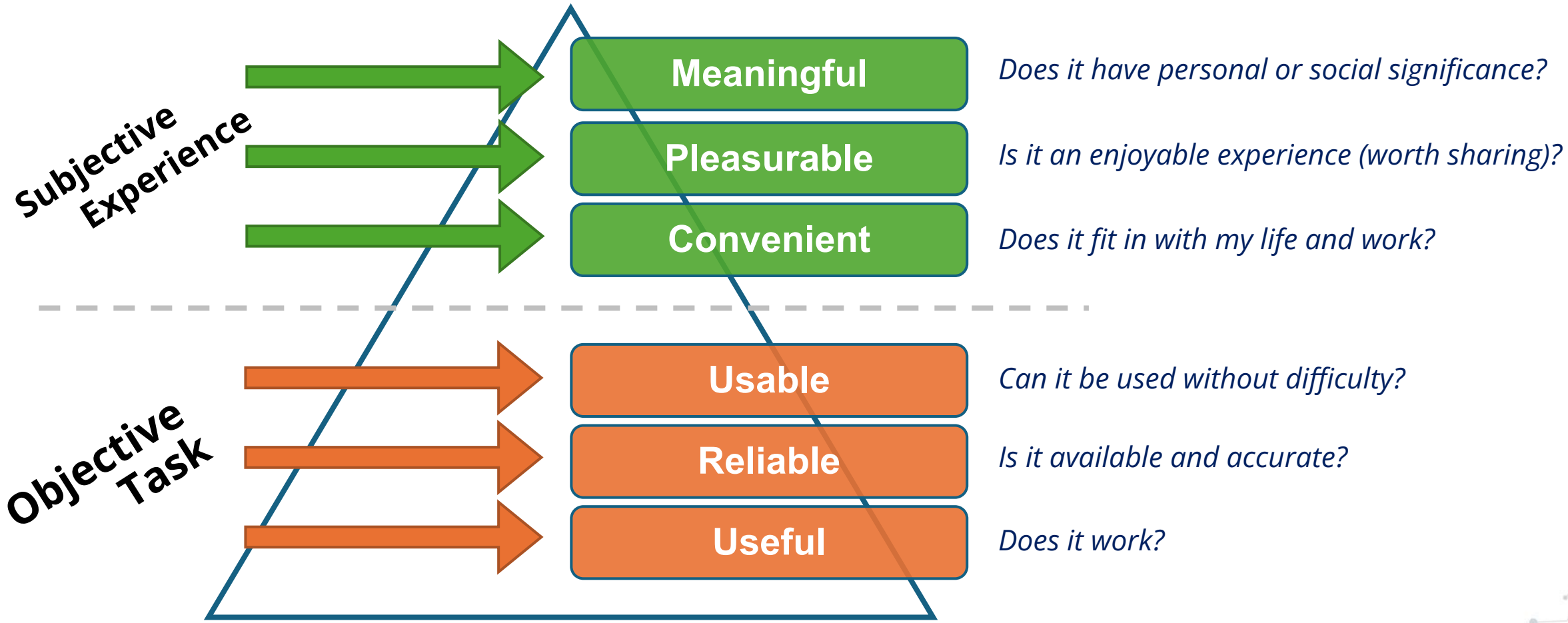
- Interfaces that are usable and useful
  - Usable: efficiency and satisfaction during usage
  - Useful: The user can complete a task

- 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



Recap: The Project Team

UX Pyramid



Recap: The Project Team

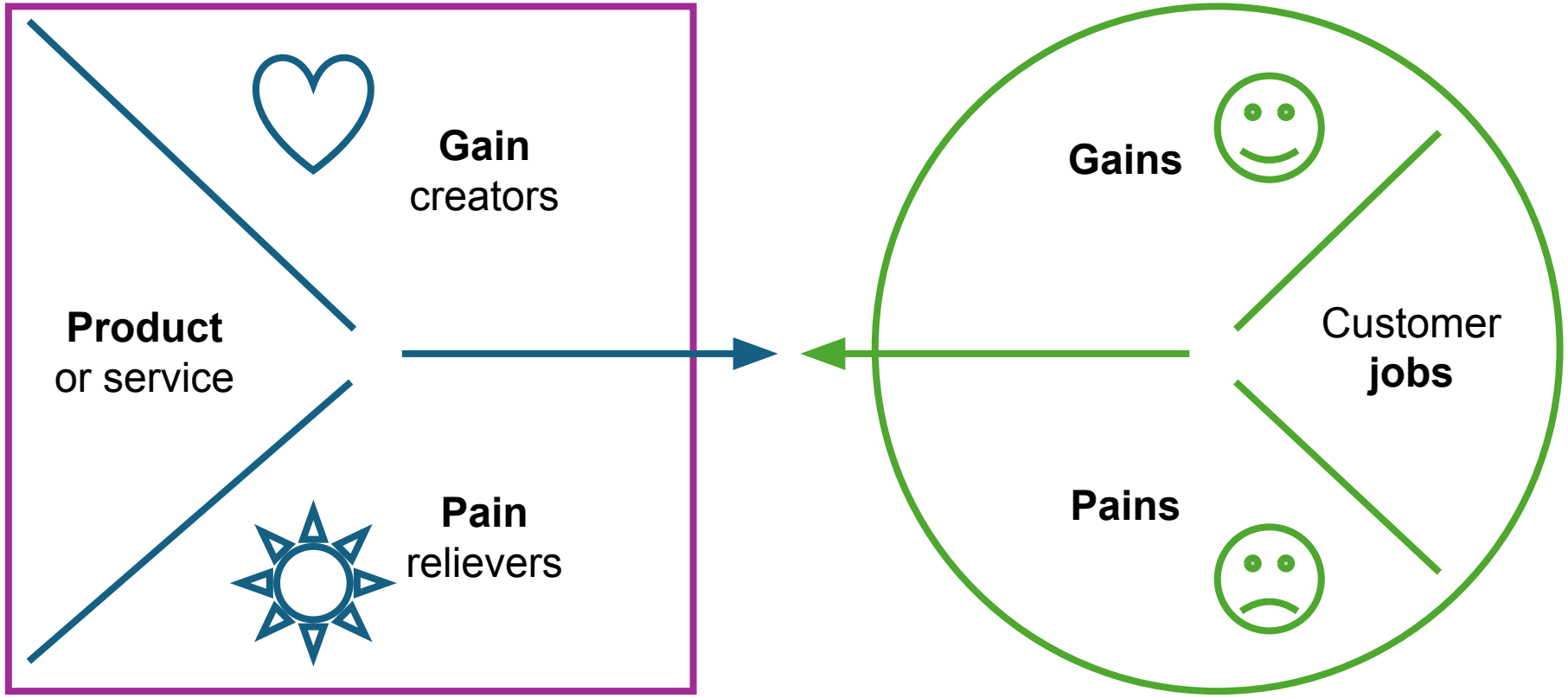
Personas

- "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

Recap: The Project Team

Value Proposition Canvas



- The pain or gain the product addresses
- The customer segment it serves
- How the product compares to competitors

Recap: The Project Team

Understanding design tools

- A **User Story** clearly outlines a specific requirement
- Good user stories are concise, client-centric descriptions of valuable features, independently deliverable, estimable, verifiable, ensuring manageable development efforts.
- **UX Prototypes**: An early model of new design
- **Epic:** is a huge user story that is too large to fit into a sprint
- **Wireframe** is a Basic visual (schematic) representation of the product for specifying requirements
- **Storyboard** is a sequential visual representation of a user-software interaction

Knowledge Checks

The Project Team

- Which team member is responsible for creating wireframes and prototypes to visualize the user experience?
- Who is responsible for ensuring that the software meets quality standards through testing and validation?
- Which tool provides a basic visual representation of product requirements?
- What does a Storyboard represent?
- What is a Wireframe used for?