HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

SOICT

School of Information and Communication Technology

ONE LOVE. ONE FUTURE.

TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

IT3180 – Introduction to Software Engineering

4 – Software Development Processes

ONE LOVE. ONE FUTURE.

Sequence of Processes

Lecture 3 introduced several process steps:

- Requirements
- User Interface Design
- System Design
- Program development (design and coding)
- Acceptance and release

Every software project will include these basic steps, in some shape or form, but:

- The steps may be **formal** or **informal**
- The steps may be carried out in **various sequences**

A software development process or methodology is a systematic way of combining these steps to build a software system

Software Development Process

In this lecture, we look at **four categories** of software development processes:

- **Waterfall**
  - Complete each process step before beginning the next
- **Iterative refinement**
  - Go quickly through all the steps to create a rough system, then repeat them to improve the system
- **Spiral**
  - A variant of iterative refinement in which new and updated components are added to the developing system as they are completed
- **Agile development**:
  - Small increments of software are developed in a sequence of sprints, each of which creates deployable code

Heavyweight and Lightweight Software Development

- Heavyweight Process
  - **Fully complete each step** and **have minimal changes** and **revisions** later
  - Each step is **fully documented** before beginning the next
- Example: waterfall or modified waterfall

Heavyweight and Lightweight Software Development

- Lightweight Process
  - Expectation that **changes will be made** based on experience
  - Minimal intermediate documentation
  - Only the **final system** will be documented
- Example: Agile Software Development

Heavyweight vs. Lightweight Methodologies

Heavyweight	Lightweight
Slower process	Speedy process
Release at the final stage	Released as increments
Client negotiation	Client collaboration
Following a plan	Responding to change

History

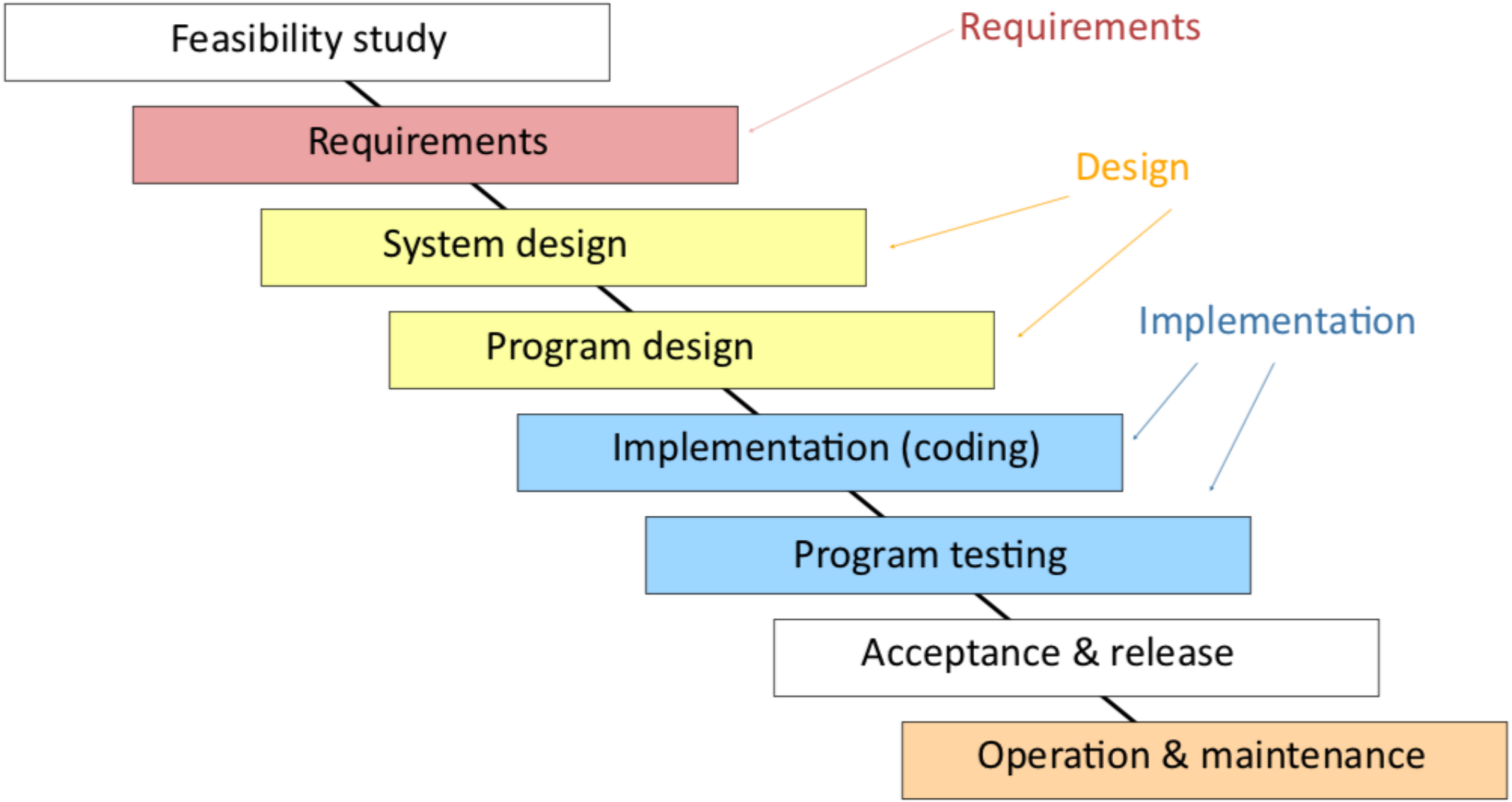
Software engineering became a **discipline**, dated from the early 1970s

At that time:

- Most computer systems were **conversions of systems** that had previously been done manually (billing, airline reservation etc.)
  - The **requirements** were **well** understood
- Many system followed the same architecture, Master File Update
  - The **system design** was **well** understood
- **Coding** was tedious with **none** of the modern languages and tools
- It was important to have a **good program design** before coding

The factors led to the **Waterfall Model** of software development

The Waterfall Model



Discussion of the Waterfall Model

The waterfall model is a **heavyweight process** with **full documentation** of each process step

Avantages:

- Separation of tasks
- Process visibility
- Quality Control at each step
- Cost monitoring at each step

Disadvantages:

- In practice, each stage in the process reveals **new understanding** of the previous stages, which often requires the earlier stages to be **revised**

Discussion of the Waterfall Model

The Waterfall model is not flexible enough!

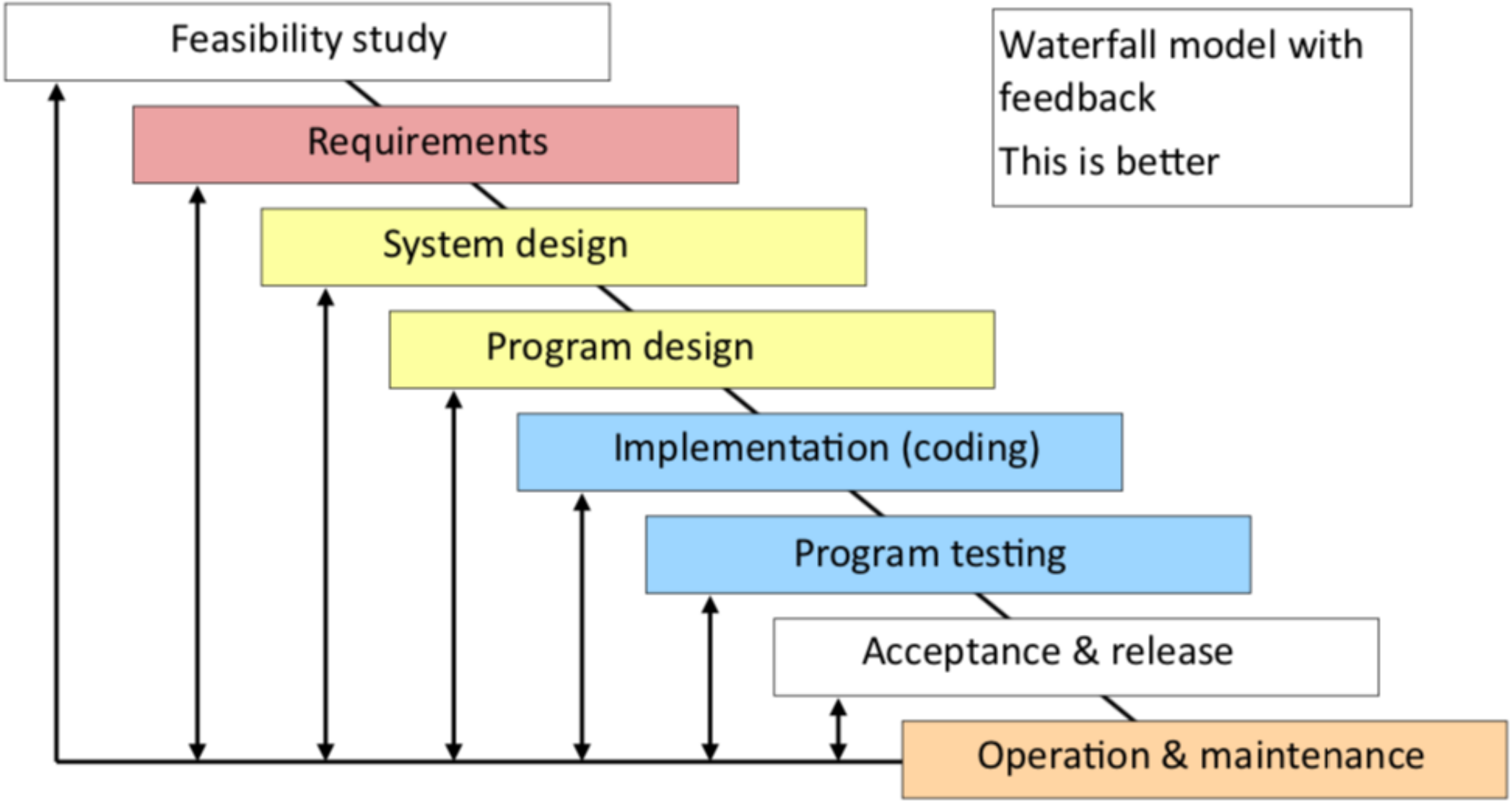
Discussion of the Waterfall Model

- A **pure sequential model** is not **possible**
- The plan must allow for some form of **iteration**

Examples:

- Does the Feasibility Study can create a proposed **budget** and **schedule**?
- Detailed design and implementation reveal gaps in the requirements specification
- What if the client changes requirements or what if the development team decides to change the technology?

Modified Waterfall Model



When to use the Modified Waterfall model

The Modified Waterfall Model works best when the requirements are **well understood** and the design is **straightforward**

For example:

- Converting a manual data processing systems where the requirements were well understood
- New version of a system where the functionality is closely derived from an earlier product
- Portions of a large system where some components have clearly defined requirements and are clearly separated from the rest of the system

Discussion

Consider the following case study:

- A web-based banking application has been developed for two groups of users: personal user and enterprise user of the bank X. Now, the bank desired to develop a mobile application for personal user. They apply the modified waterfall model.
- Give at least 3 reasons why the modified waterfall model is appropriate in this case study.

Iterative Refinement

Concept:

- Requirements are hard to understand until there is an operational system, particularly with user interfaces.
- System and program design may benefit from prototypes

Process:

- Create a **prototype** system early in the development process
- Review the prototype with clients and test it with users, to
  - improve the **understanding of the requirements**
  - **clarify the design**
- Refine the prototype in a series of **iterations**.

Iterative refinement: an example (1)

Problem: **Add graphics package to a programming environment**

Requirements:

- The client was unsure of several important requirements
- E.g., syntax for how to manage coordinates across different objects

Process

- Build a prototype version with a preprocessor and preliminary run-time package
- Have several iterations of development
- The final iteration is the complete version of package

Iterative refinement: an example (2)

Problem: Add graphics package to a programming environment

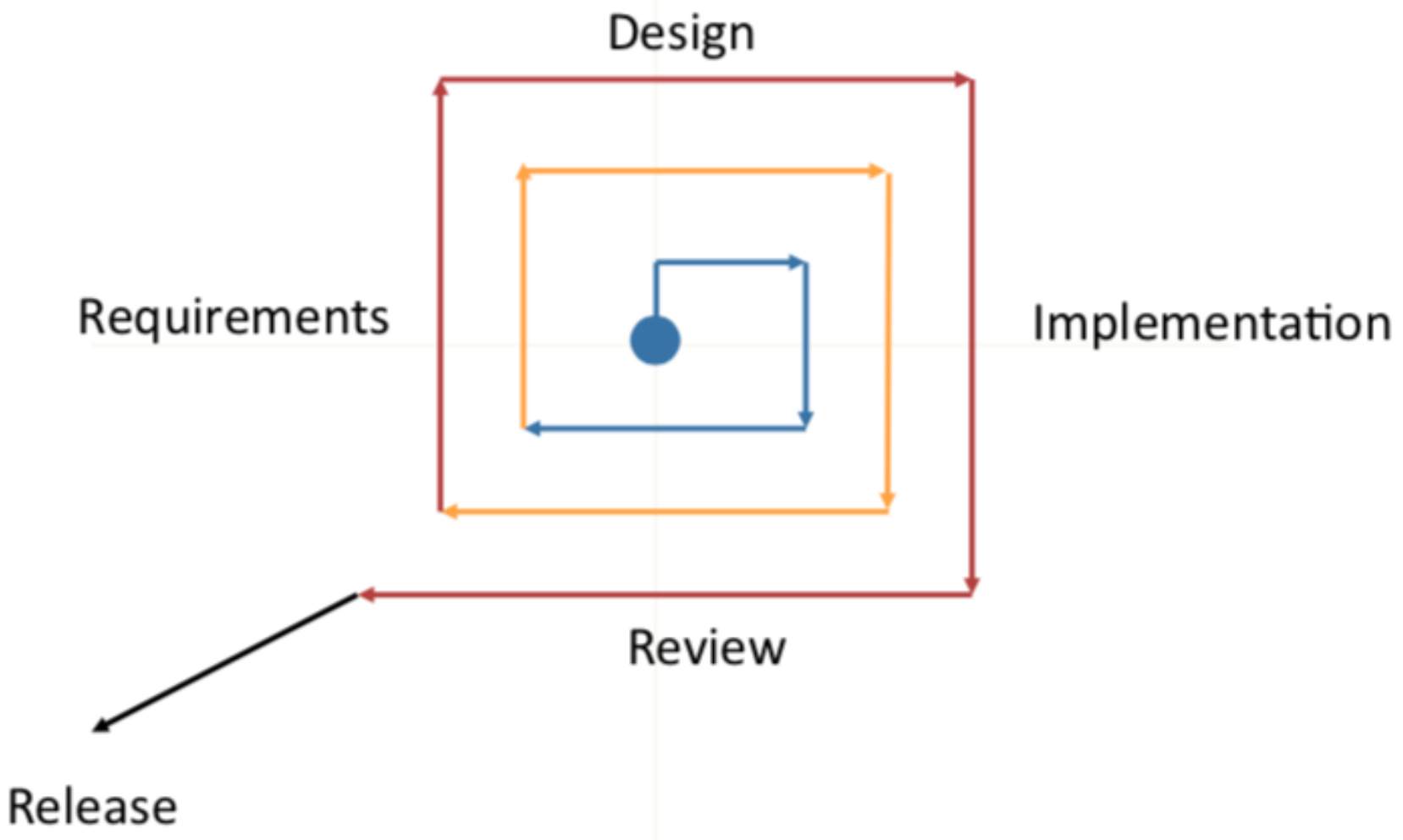
Requirements:

- The client was unsure of several important requirements
- E.g., syntax for how to manage coordinates across different objects

Process

- For each iteration:
  - Test the system with users
  - Make modifications
  - Repeat until users are pleased with function

Iterative Refinement - Schema



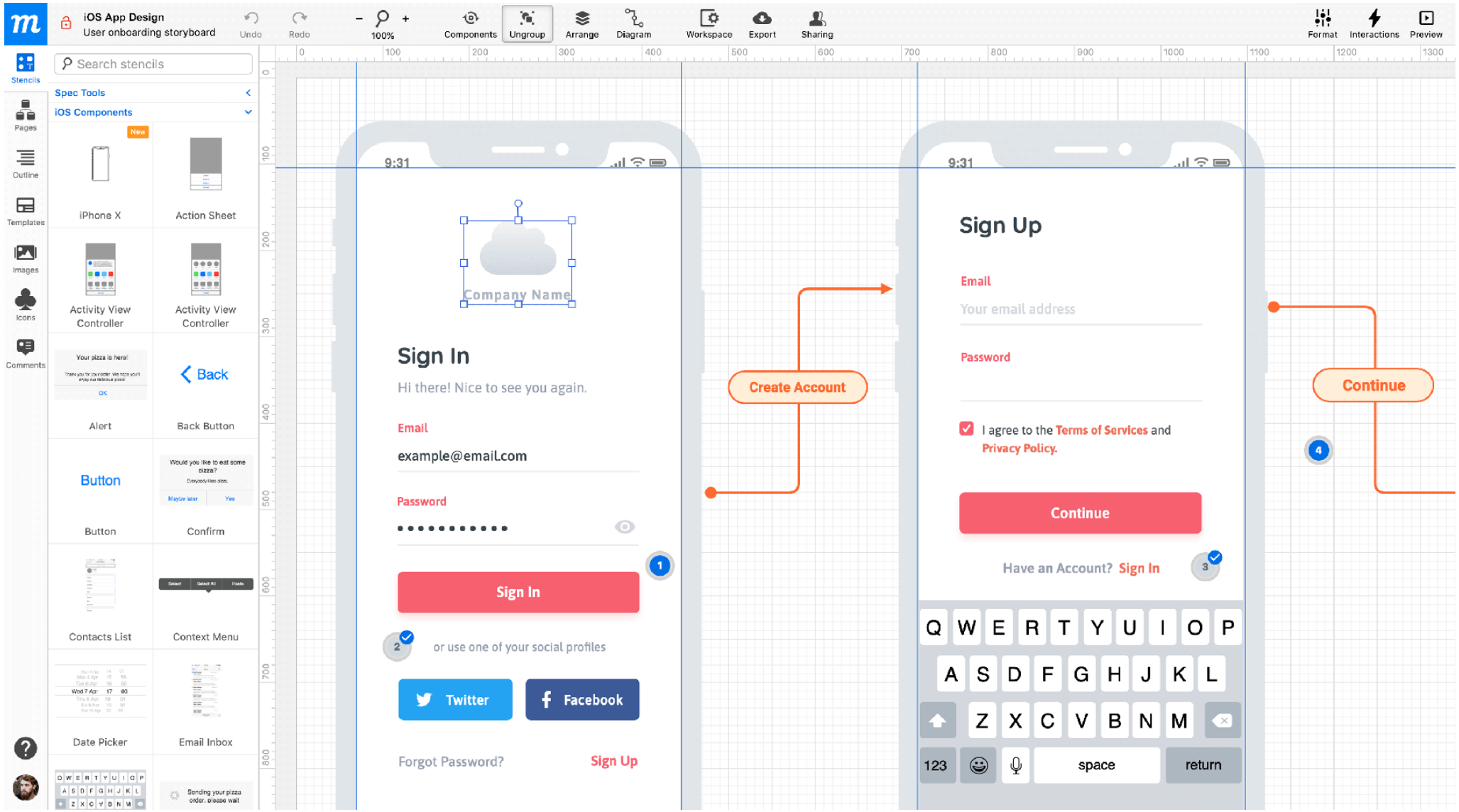
Discussion of Iterative Refinement

This is a medium weight process with documentation created during the process

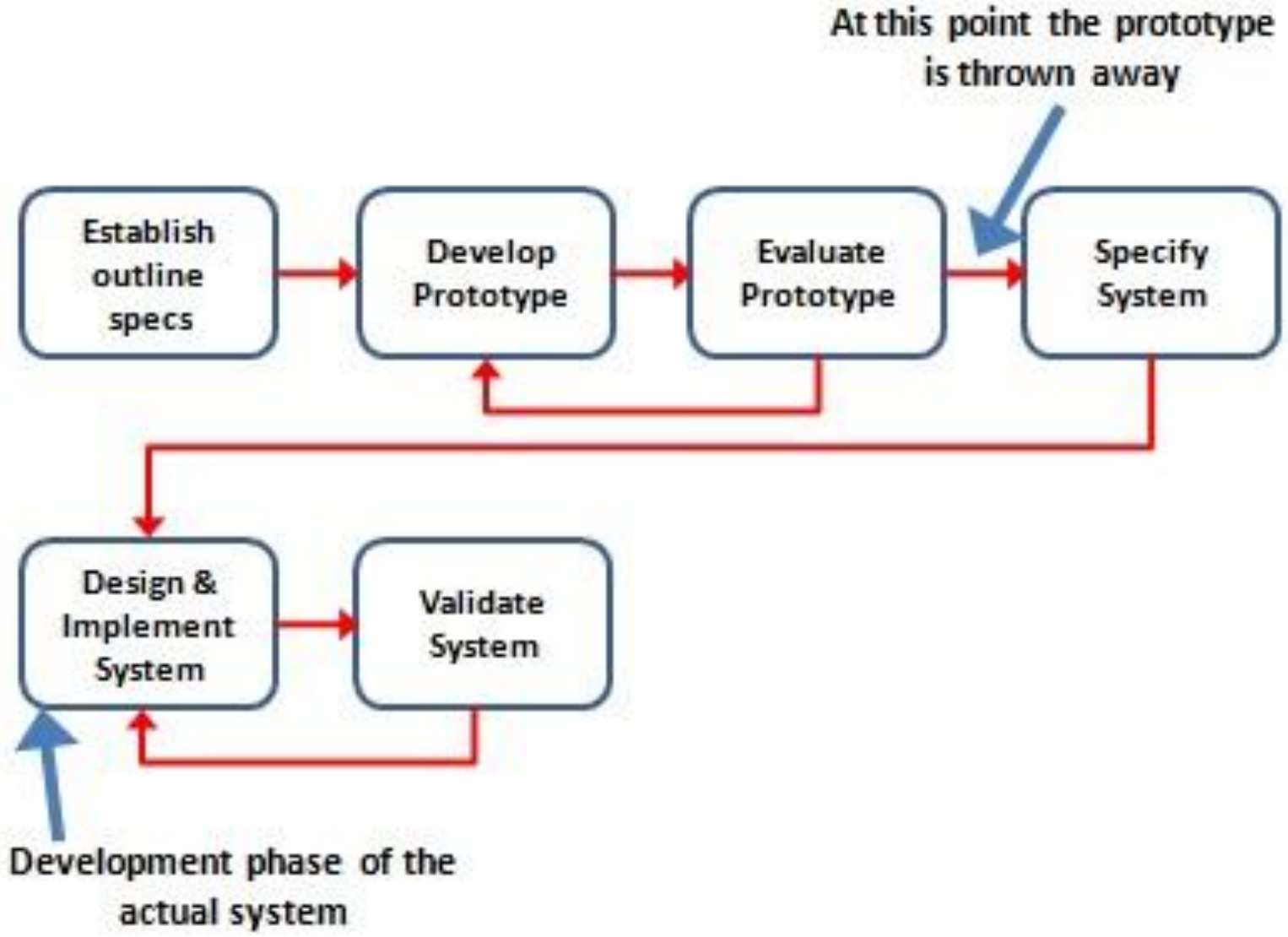
Iterative refinement uses various techniques that enable the client to review the planned system early during development:

- User interface mock-ups
- Throw-away software components
- Rapid prototyping
- Successive refinement

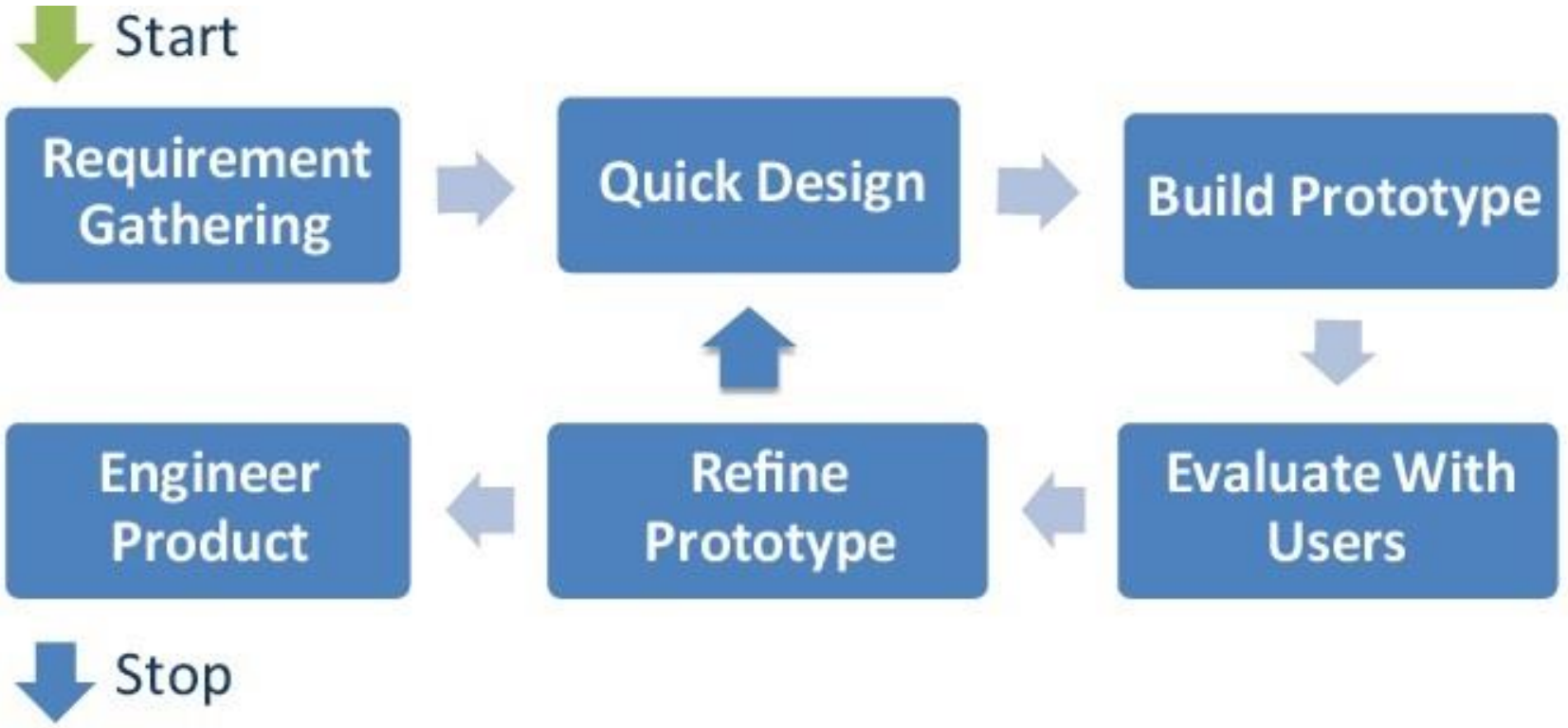
User interface mock-up



Throw-away Prototyping



Evolutionary Prototyping

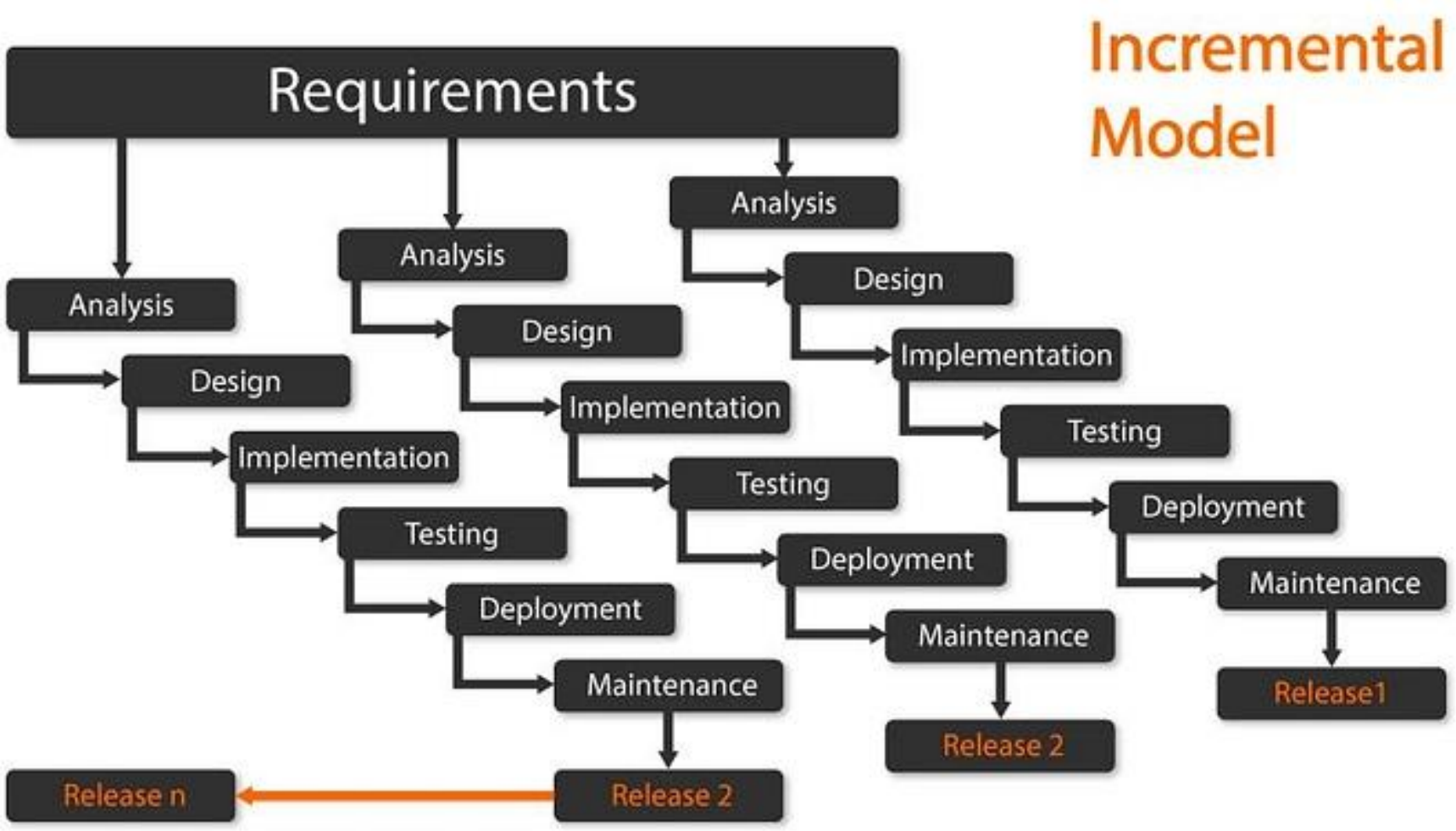


Prototyping Model

Get something working as quickly as possible, for client and user evaluation

...but do not release it

Incremental Development



Incremental Development - Example

Ecommerce website

- Search, Product Information, Shopping Basket, Checkout, Favourites, Customer Review

1 increment = 1 release

Incremental Development - Example

3 incrementals:

- 1st:

1st release

Incremental Development - Example

3 incrementals:

- 2nd:

2nd release

Incremental Development - Example

3 incrementals:

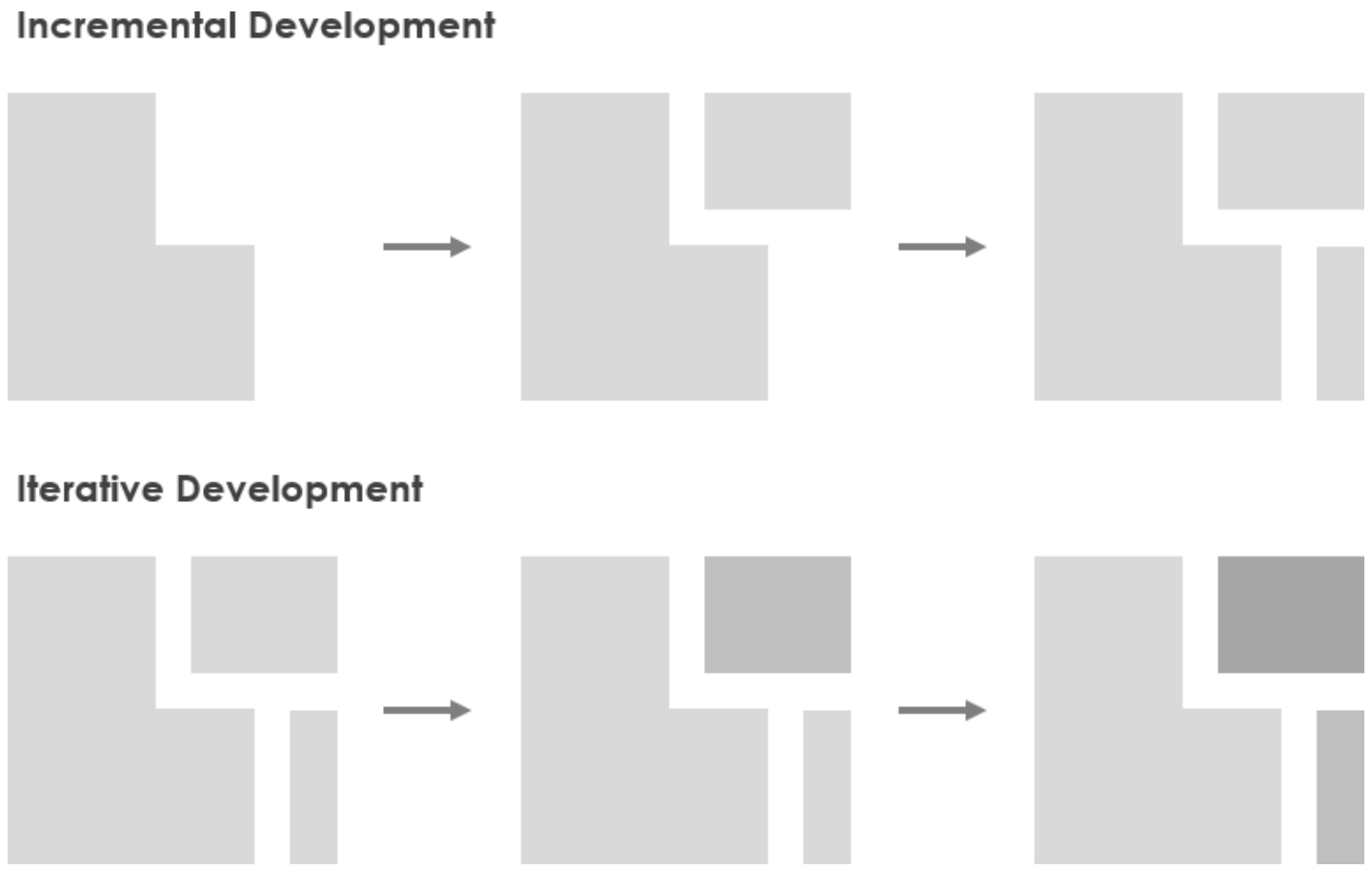
- 3rd:

3rd release

Incremental Development - Discussion

Is incremental model also an iterative one?

Incremental vs. Iterative



Can you see the difference?

Spiral Development

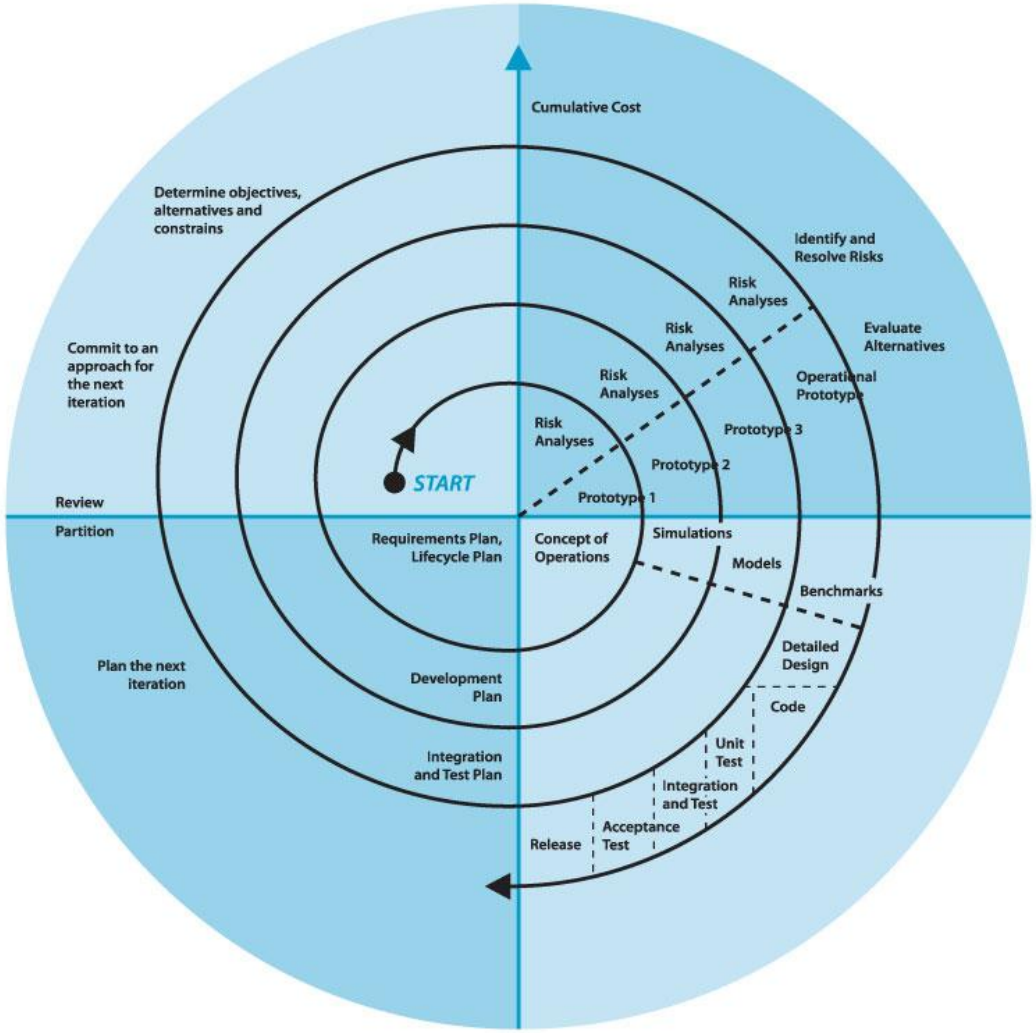
- Iterative and incremental model with more emphasis placed on risk analysis
- 4 main phases: **Planning** – **Design** – **Construct** – **Evaluation**
- An iteration = A spiral

Spiral Development - Schema

- **1st quadrant**: Requirements are gathered and analyzed at the start of every spiral

- **2nd quadrant**: All proposed solutions for identified problems are evaluated. Risks are identified for the selected solution, then are resolved using the best possible strategy.

Prototype is built for the best possible solution



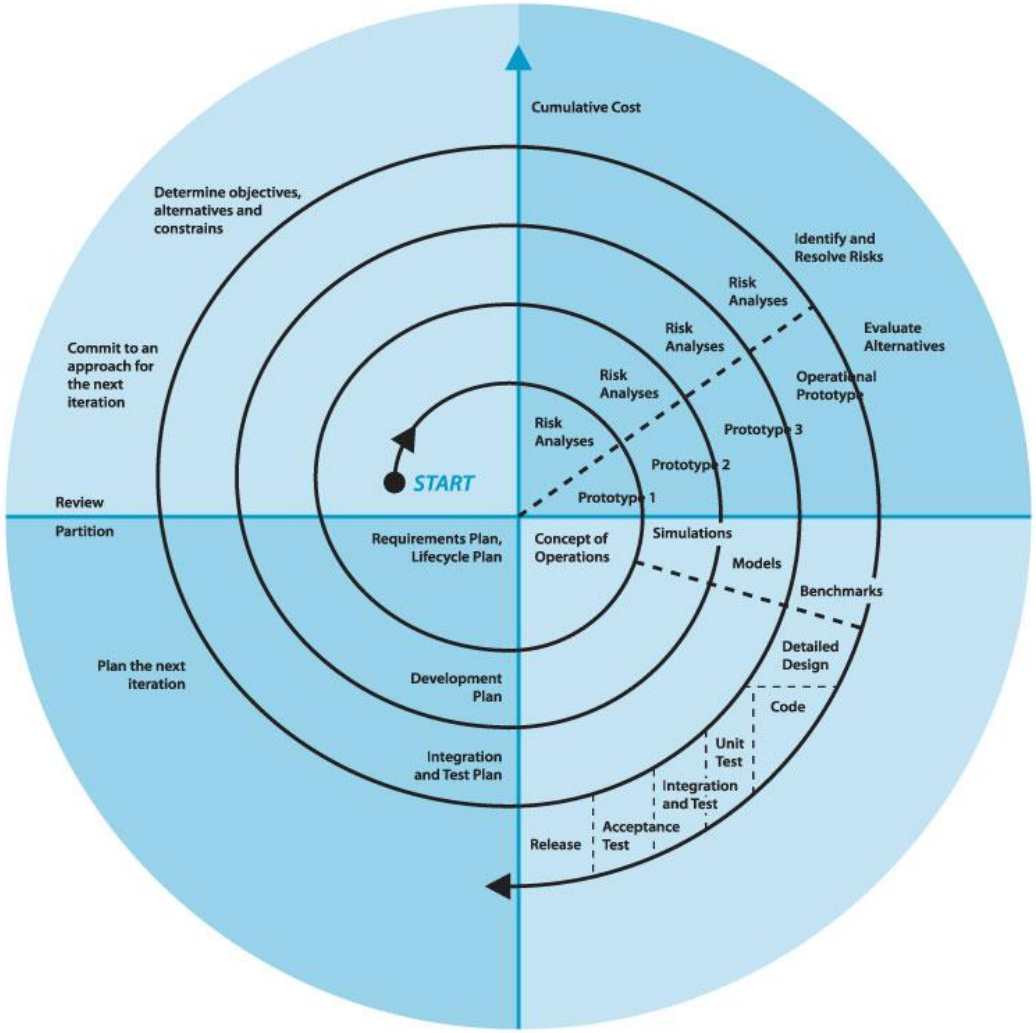
Spiral Development - Schema

- **3rd quadrant**: identified features are developed and verified through testing

Next version of the system is available

- **4th quadrant:** customers evaluate the so far developed version

Planning for the next phase then is started



Spiral Development – Risk Handling

Risk: any **situation** that might affect the successful completion of the project

- Spiral model emphasizes risk handling by developing a **prototype**
- After some iterations, most of risks are **studied** and **resolved**
- For example: What is the risk involved in accessing data from a remote database system?
  - What if the data access might be too slow?
  - Risk solution: Building a prototype of the data access subsystem

Spiral model viewed as a Meta-model

The spiral model subsumes all the other models

- Single loop represents the **waterfall model**
- **Prototyping** approach is used to first draft the solution before embarking on the actual product
- **Iterations** along the spiral model can be considered as the **evolutionary** levels through which the complete system is built
- At the end of each spiral, the result will typically be incorporated with the large base system

Agile Methodology

Spring 2000, in Oregon, US, big question:

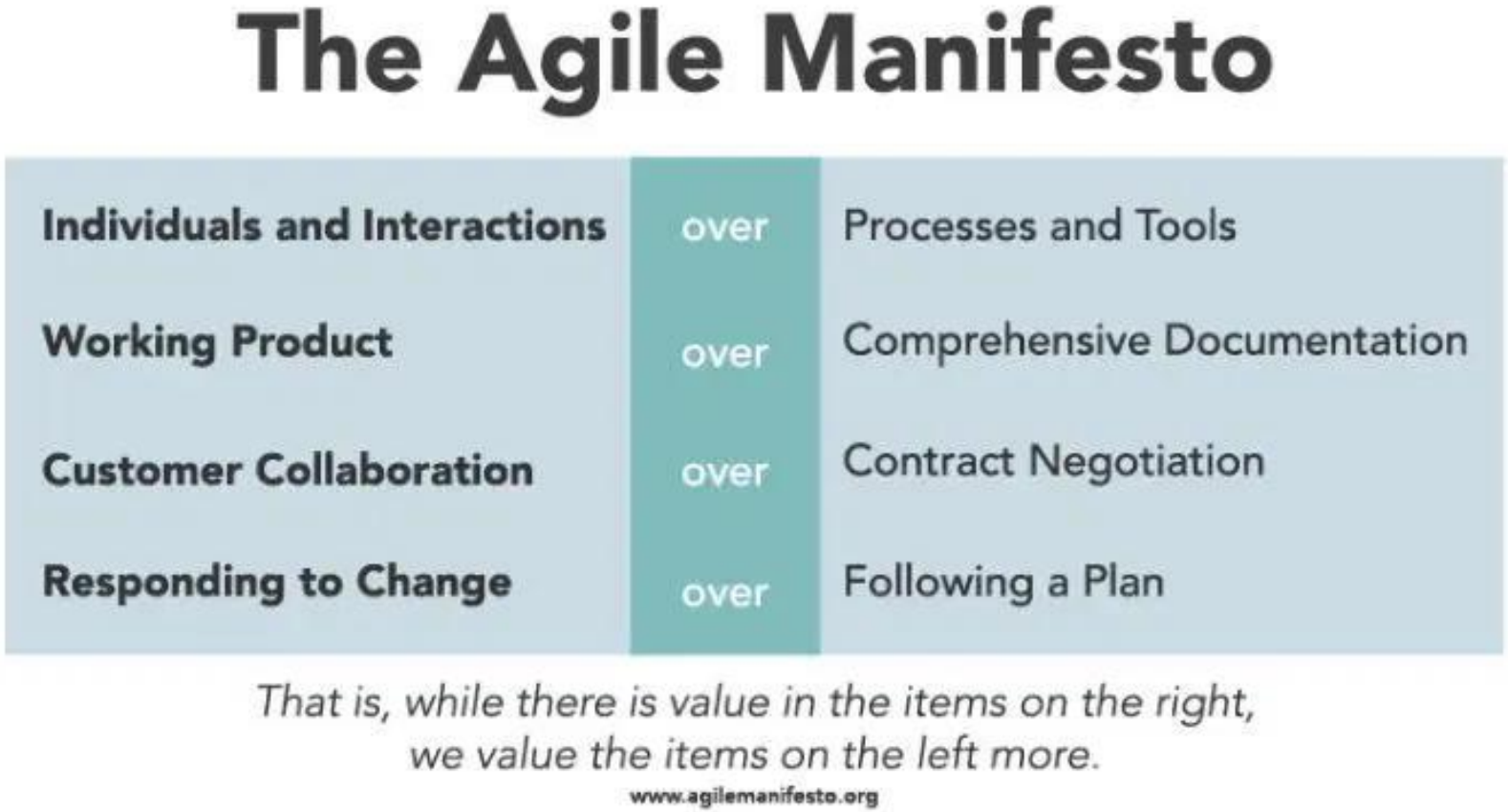
How could speed up development times in order to bring new software to market faster?

Critical milestone in the history of Agile with 3 key ideas:

- Speed to market
- Rapid feedback
- Continuous improvement

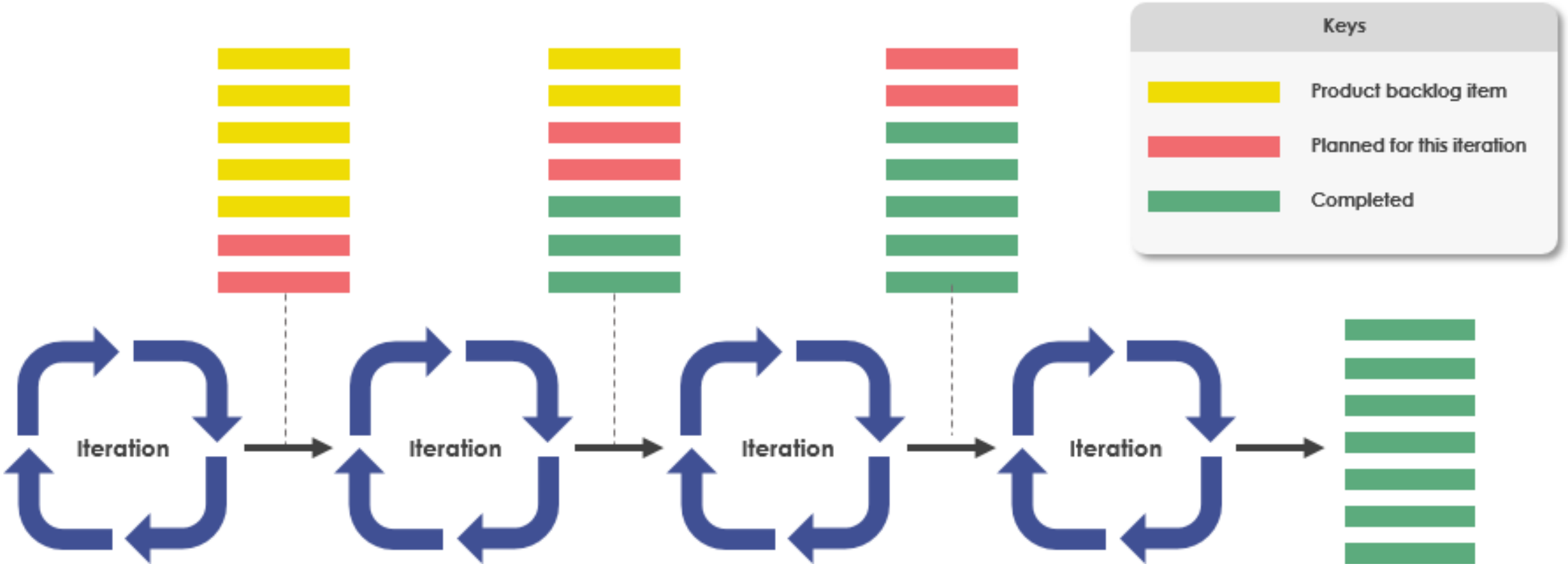
Agile Methodology - History

- Oregon 2001, Manifesto for Agile Software Development:

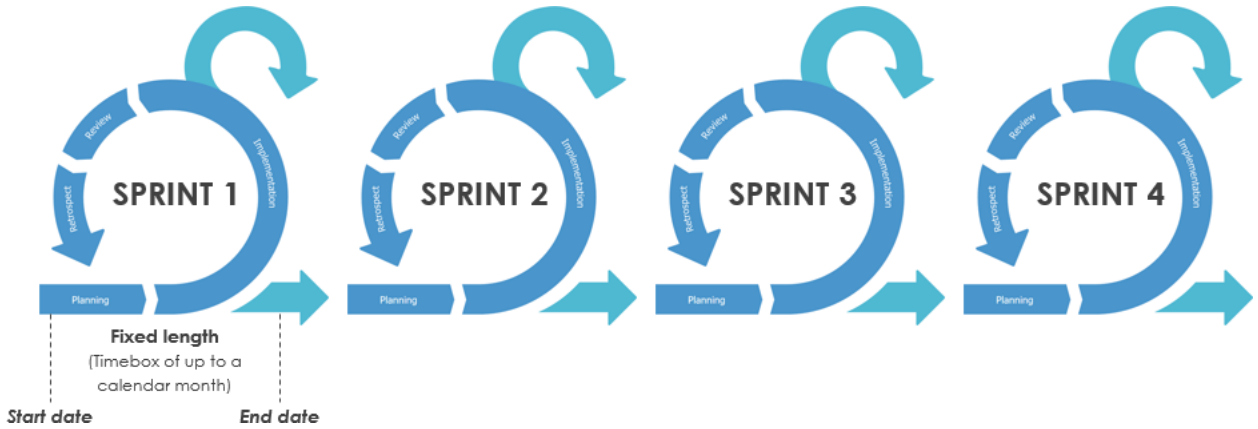


12 principals

Agile Approach: Iterative and Incremental



Agile Approach: Iterative and Incremental



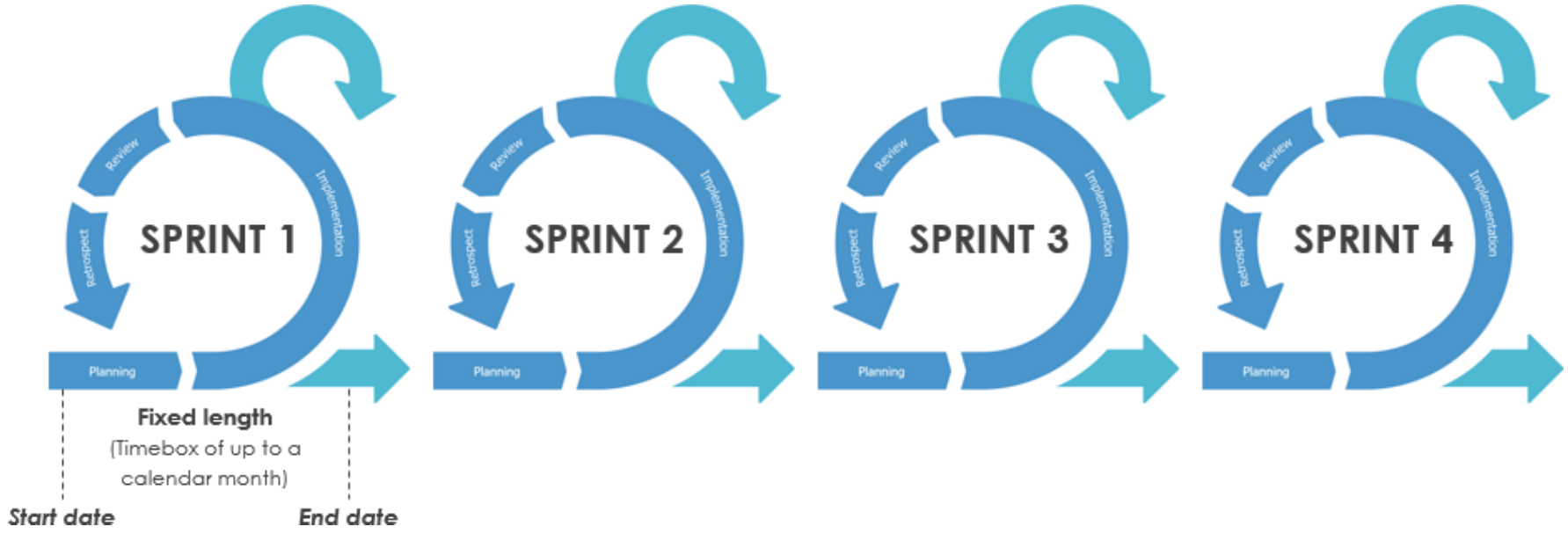
- **Single sprint**: requirements, design, coding and testing
- Each sprint ends with **fully** tested code, **ready** to put into production

- A large project is divided into small **increments** called **sprints**
- The development is carried out by **small** teams of 4 to 9 people
- The schedule is divided into **fixed time boxes**, perhaps 2 to 4 weeks
- Each sprint is a time box during which the team completes **part** of a software project

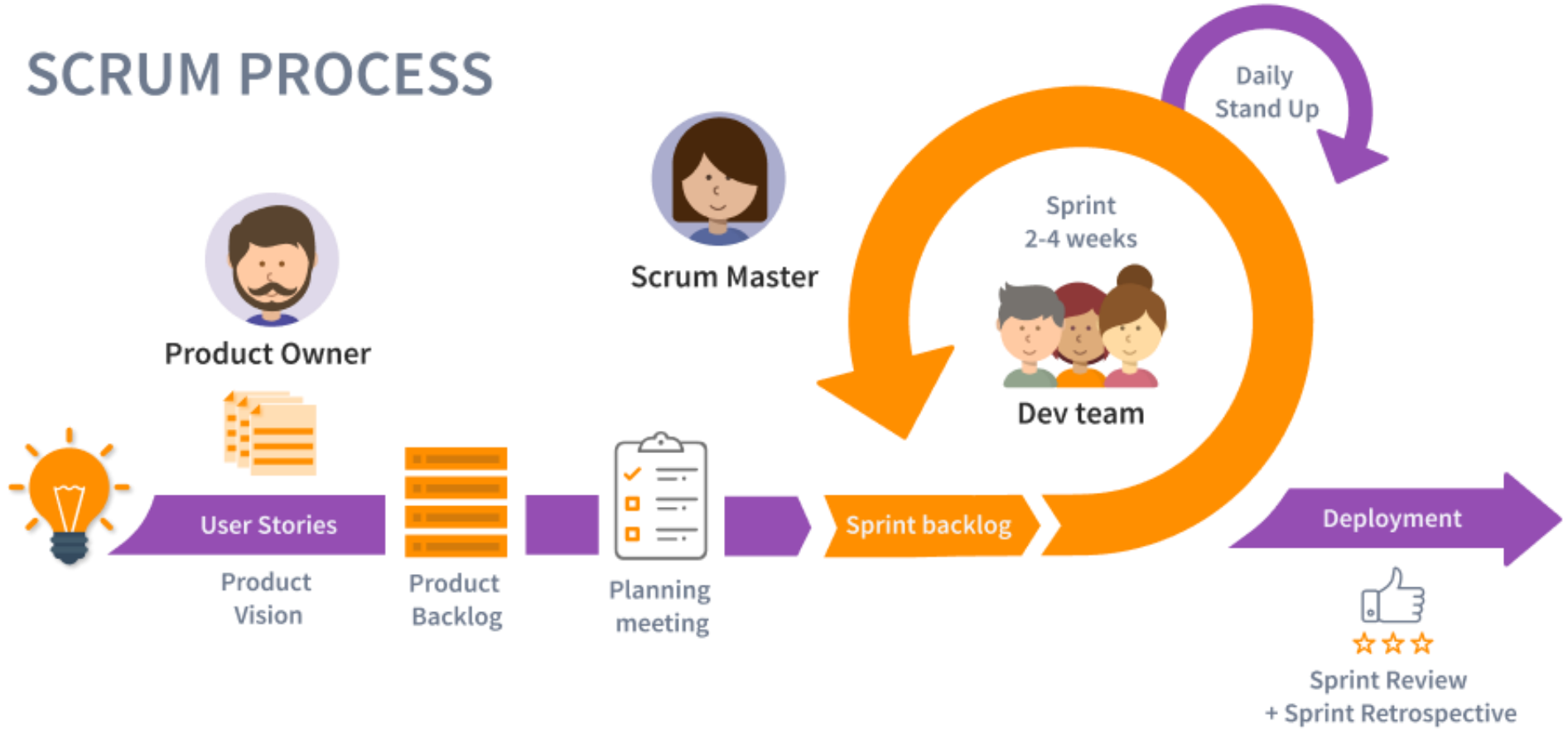
Agile Development - Sprint

After each sprint the code may be:

- **Released** (original agile method)
- Combined with code from other sprints for **subsequent release**
- **Incorporated** into a larger **code base** (spiral development)



Scrum – The most widely used agile method for software development



Scrum roles

Scrum Master

- Understand the theory, practices, rules and values of Scrum
- Help others improve interactions to maximize the value by the Scrum team

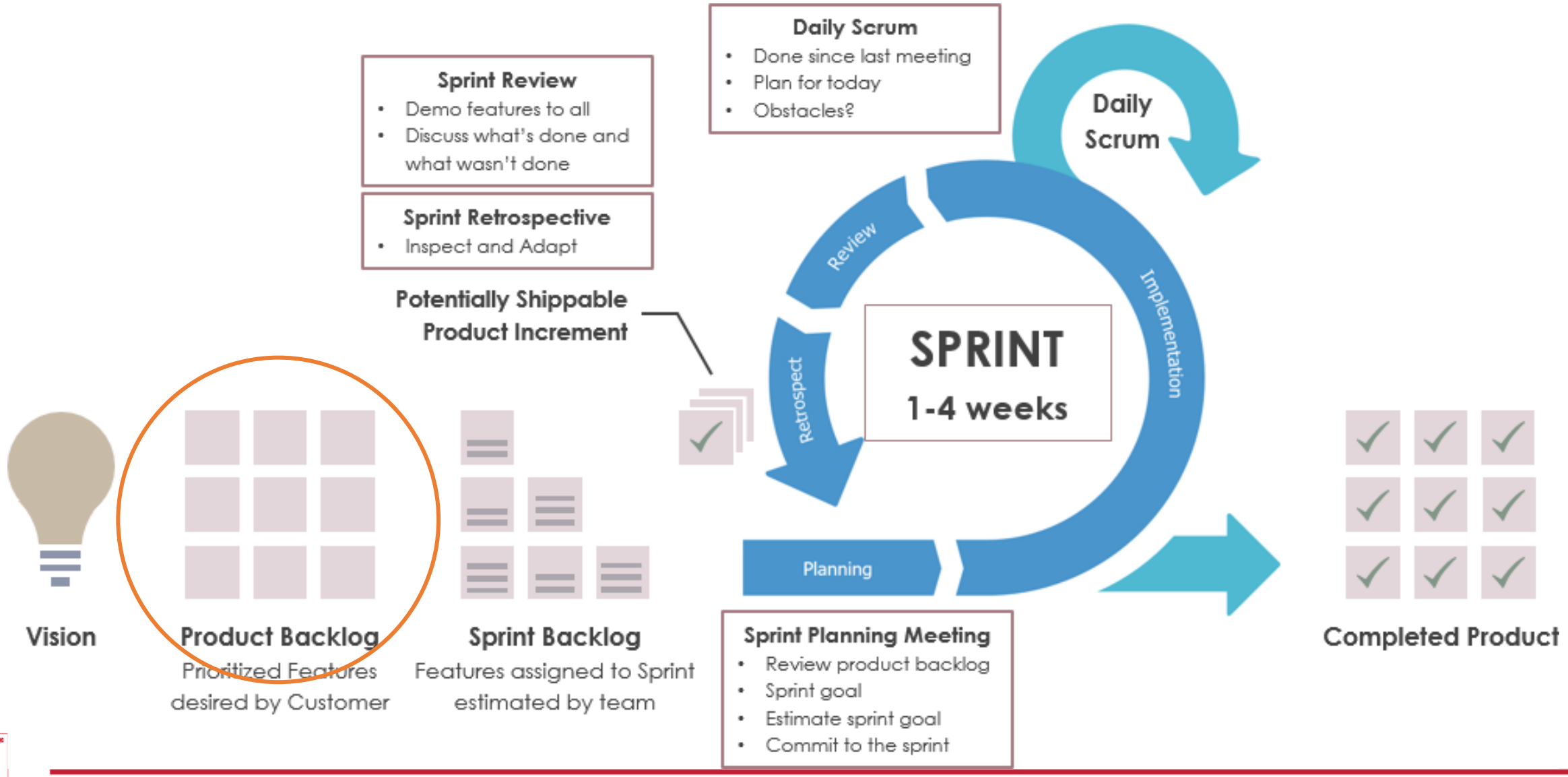
Product Owner

- The bridge between the business part and the technical part of the project
- Responsible for writing user stories and for keeping the Product backlog up to date

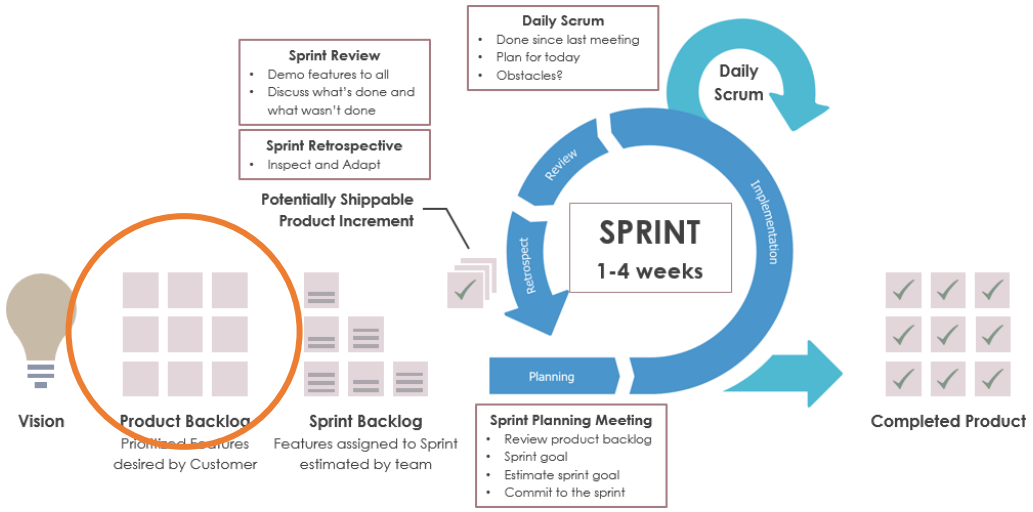
Development Team

- Transform the expressed needs into usable functionalities
- Developers, software architects, functional analyist, graphic designers

Product Backlog



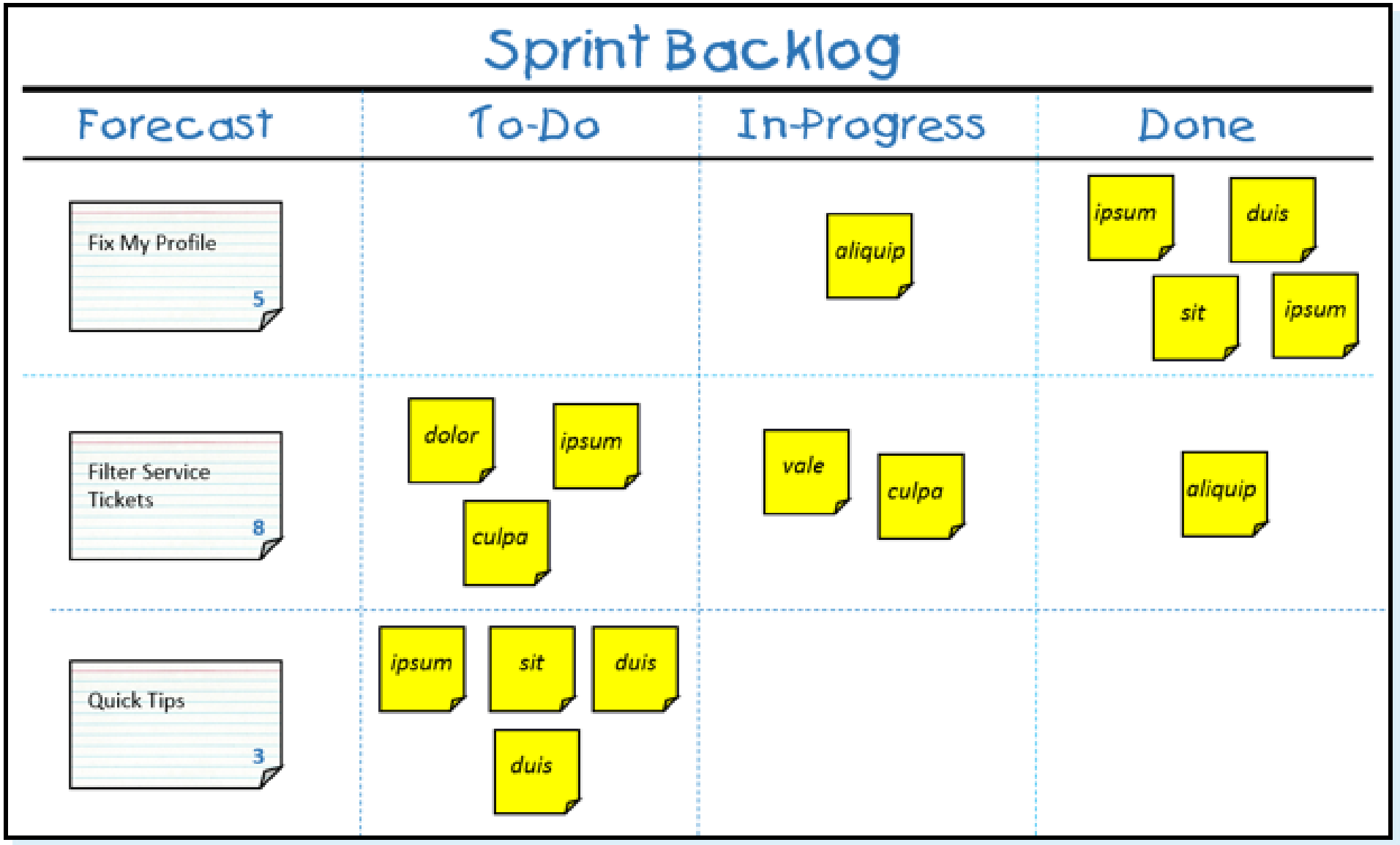
Product Backlog



- Contains all the userstories which will be turned into tasks so that the scrum team can select and plan in upcoming sprints

- PO is in charge of managing and keeping the product backlog up to date

Sprint Backlog



Epics – Userstories – Tasks

Epic

- A **functionality** of the product to be developed
- A multiple **sets** of **userstories** grouped by categories or themes

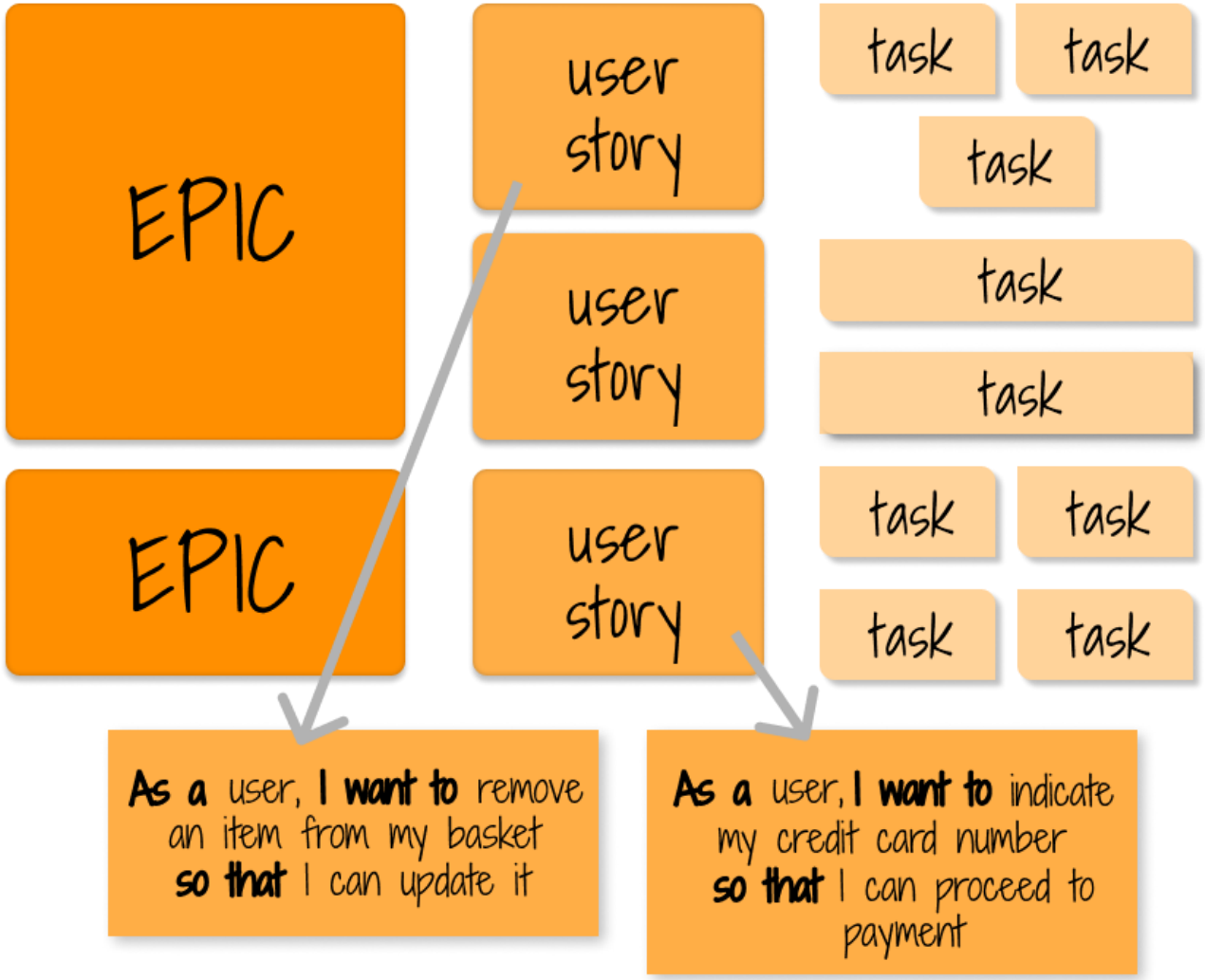
User Story

- Not a task, neither a specification
- A **statement** of a user expectation

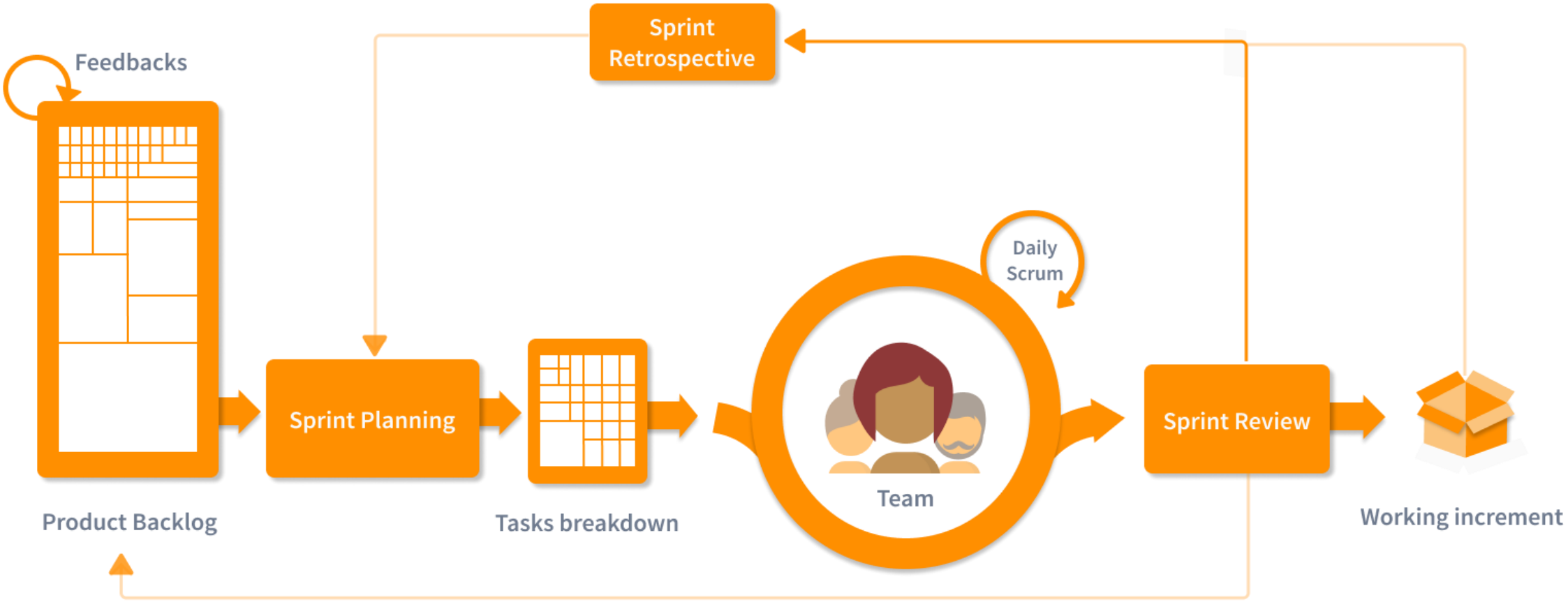
Task

- Technical **activities** that helps respond to user stories
- Tasks should be same sized but may be of different nature: **design**, **development**, **test**, etc.

Epics – Userstories – Tasks



Scrum Meetings



Sprint Planning – Sprint Retrospective – Daily Scrum

Sprint Planning Meeting

- **Goal**: The development team selects the **priority elements** of the Product Backlog to complete in the current sprint

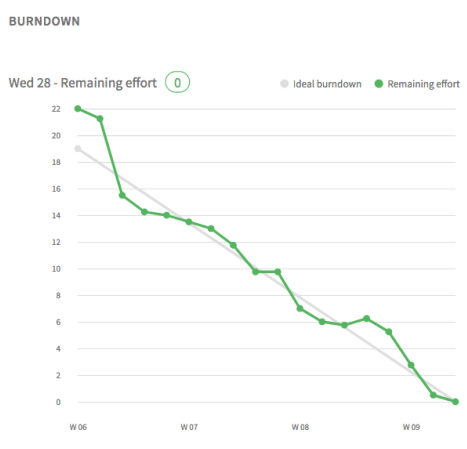
Daily Stand-up Meeting

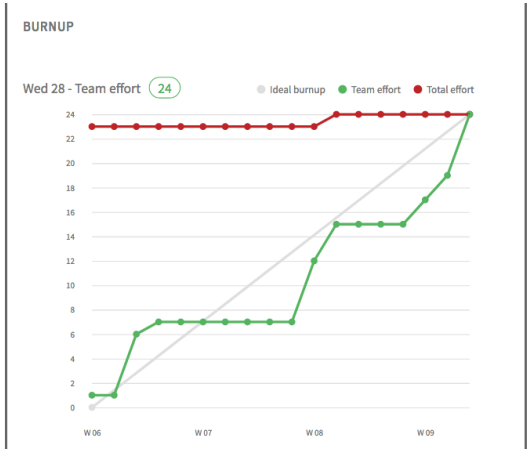
- Daily synchronizing meeting
- **Goal**: to enable team members to gather on a daily basis
  - to **discuss** tasks and work **progress** as well as potential **problems**
  - to overcome possible **blockages**
  - to promote mutual **support**

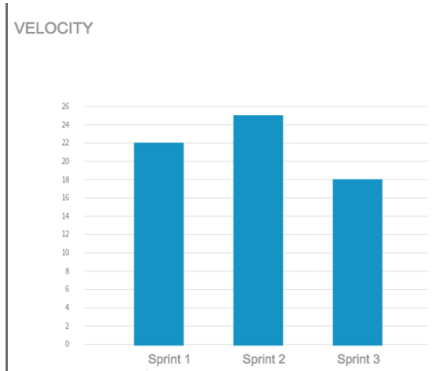
Sprint Planning – Sprint Retrospective – Daily Scrum

Sprint Retrospective Meeting

- Toward continuous improvement
- Take place at the end of the sprint
- **Goal**: discussing and taking a step back from the latest sprint
  - to optimize interactions between individuals,
  - to raise product quality
  - to improve productivity
- Tools:
  - **Burnup chart**
  - **Burndown chart**
  - **Velocity**







Mixed Processes

In practice, many large projects use processes that **mix aspects** of the four types of software process

Example:

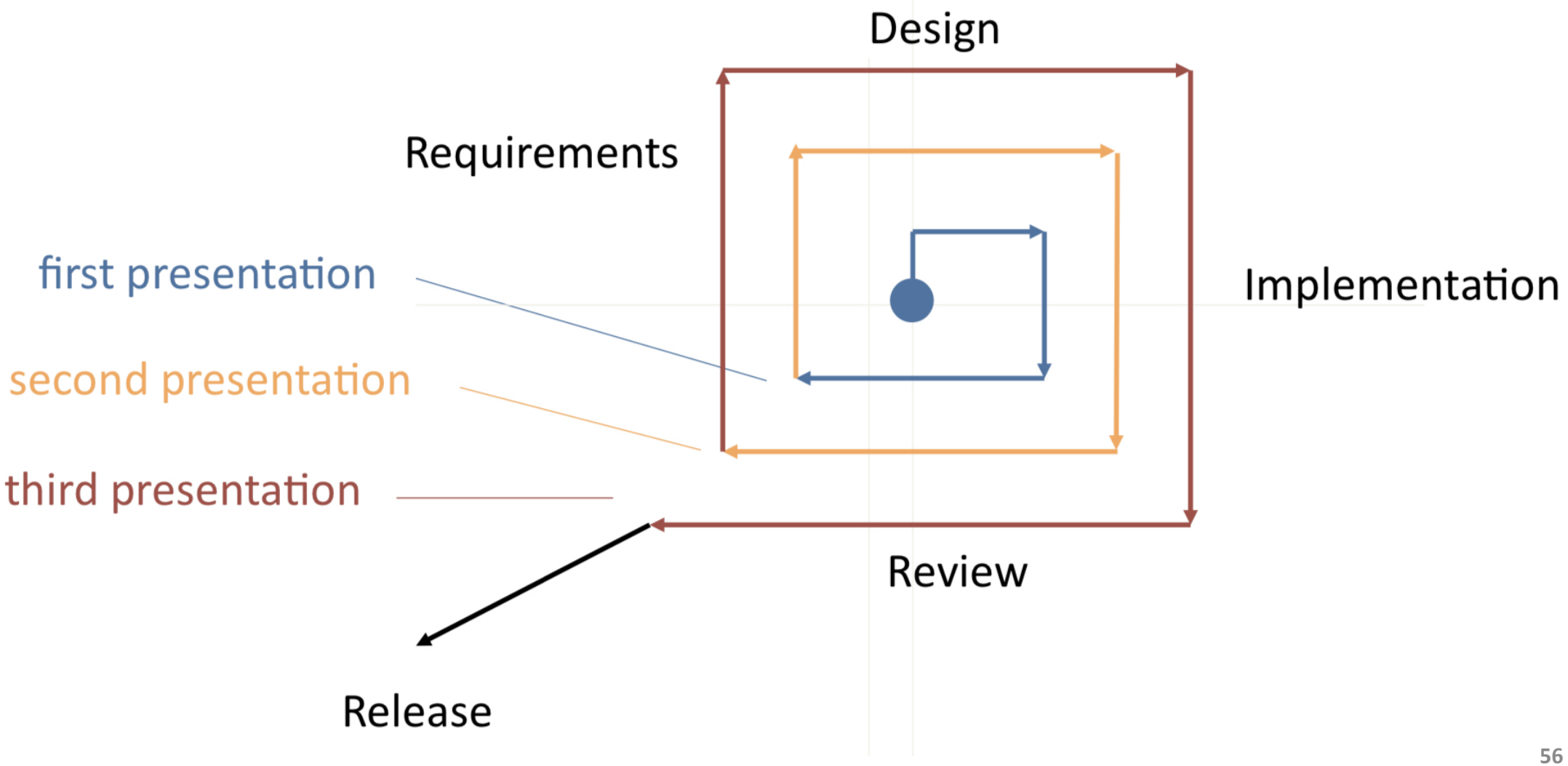
- A project with well-understood requirements might use a modified waterfall approach to specify the requirements and system design, followed by a series of agile sprints
- A project with vague requirements might use iterative refinement to clarify the requirements, followed by a modified waterfall model to build the final version
- With spiral development, new components may be developed as a series of sprints

The most important:

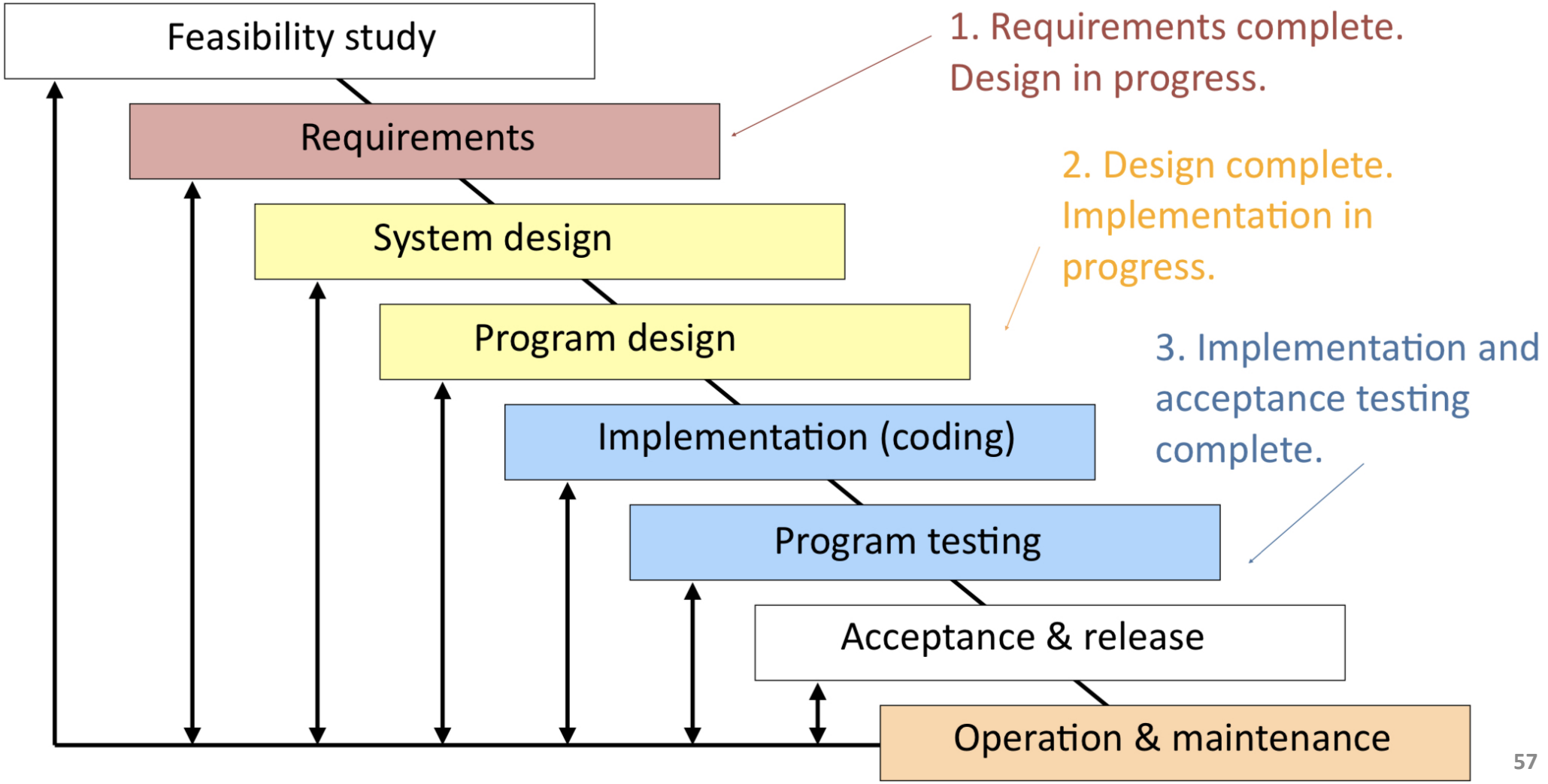
User Interface should be tested with users…

➔ Iterative development, whatever process is used for the rest of the system

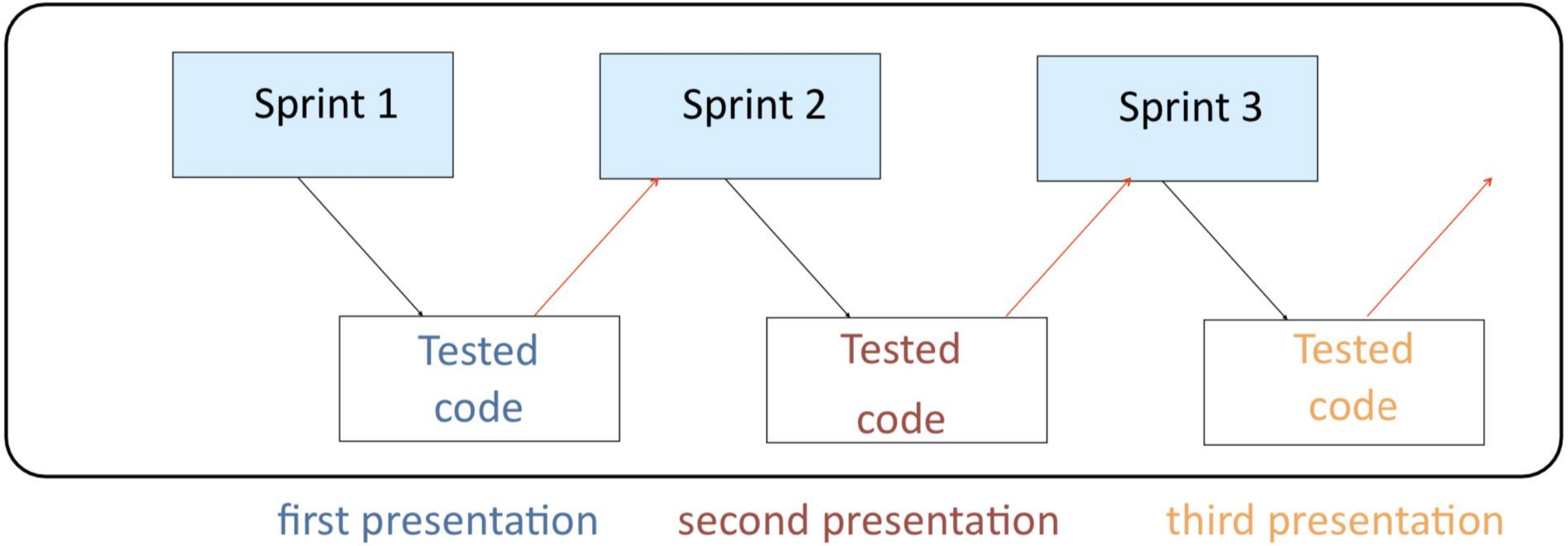
IT3180 Project: Iterative Refinement



IT3180 Project: Modified Waterfall Model



IT3180 Project: Agile Development



Each sprint aim should complete a section of the code.

4. Software development processes

(end of lecture)

ONE LOVE. ONE FUTURE.