

# Software Engineering

## Software Products and Project Management: Software product management and prototyping

1102SE02

MBA, IM, NTPU (M5010) (Spring 2022)

Wed 2, 3, 4 (9:10-12:00) (B8F40)

Min-Yuh Day, Ph.D,  
Associate Professor

Institute of Information Management, National Taipei University

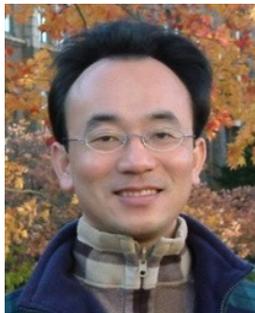
<https://web.ntpu.edu.tw/~myday>



<https://meet.google.com/ish-gzmy-pmo>



aws  
educate | Cloud  
Ambassador  
2020 Cohort



aws  
academy  
Accredited  
Educator  
aws  
certified  
Cloud  
Practitioner  
aws  
certified  
Solutions  
Architect  
Associate

# Syllabus

| Week | Date       | Subject/Topics  |
|------|------------|---|
| 1    | 2022/02/23 | Introduction to Software Engineering  |
| 2    | 2022/03/02 | <b>Software Products and Project Management:<br/>Software product management and prototyping</b>    |
| 3    | 2022/03/09 | Agile Software Engineering:<br>Agile methods, Scrum, and Extreme Programming                        |
| 4    | 2022/03/16 | Features, Scenarios, and Stories  |
| 5    | 2022/03/23 | Case Study on Software Engineering I  |
| 6    | 2022/03/30 | Software Architecture: Architectural design,<br>System decomposition, and Distribution architecture |

# Syllabus

| Week | Date       | Subject/Topics  |
|------|------------|---|
| 7    | 2022/04/06 | Make-up holiday (No Classes)  |
| 8    | 2022/04/13 | Midterm Project Report  |
| 9    | 2022/04/20 | Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service |
| 10   | 2022/04/27 | Cloud Computing and Cloud Software Architecture   |
| 11   | 2022/05/04 | Microservices Architecture, RESTful services, Service deployment                                    |
| 12   | 2022/05/11 | Industry Practices of Software Engineering  |

# Syllabus

**Week Date Subject/Topics**

**13 2022/05/18 Case Study on Software Engineering II**

**14 2022/05/25 Security and Privacy; Reliable Programming;  
Testing: Test-driven development, and Code reviews;  
DevOps and Code Management: DevOps automation**

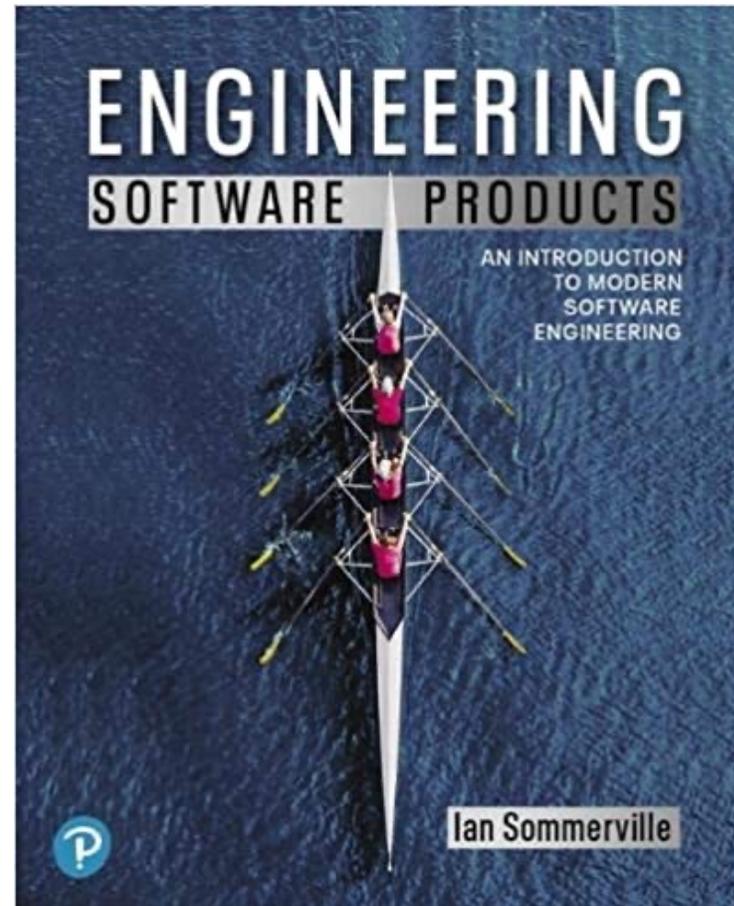
**15 2022/06/01 Final Project Report I**

**16 2022/06/08 Final Project Report II**

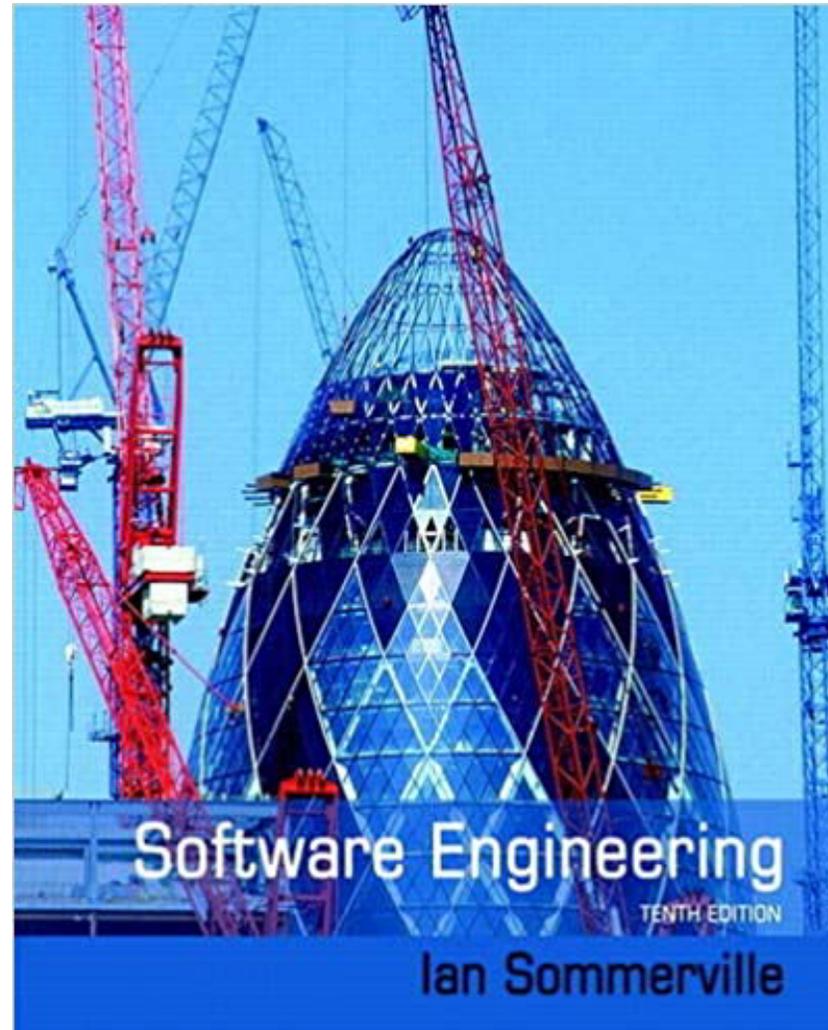
17 2022/06/15 Self-learning

18 2022/06/22 Self-learning

Ian Sommerville (2019),  
**Engineering Software Products:**  
An Introduction to Modern Software Engineering,  
Pearson.

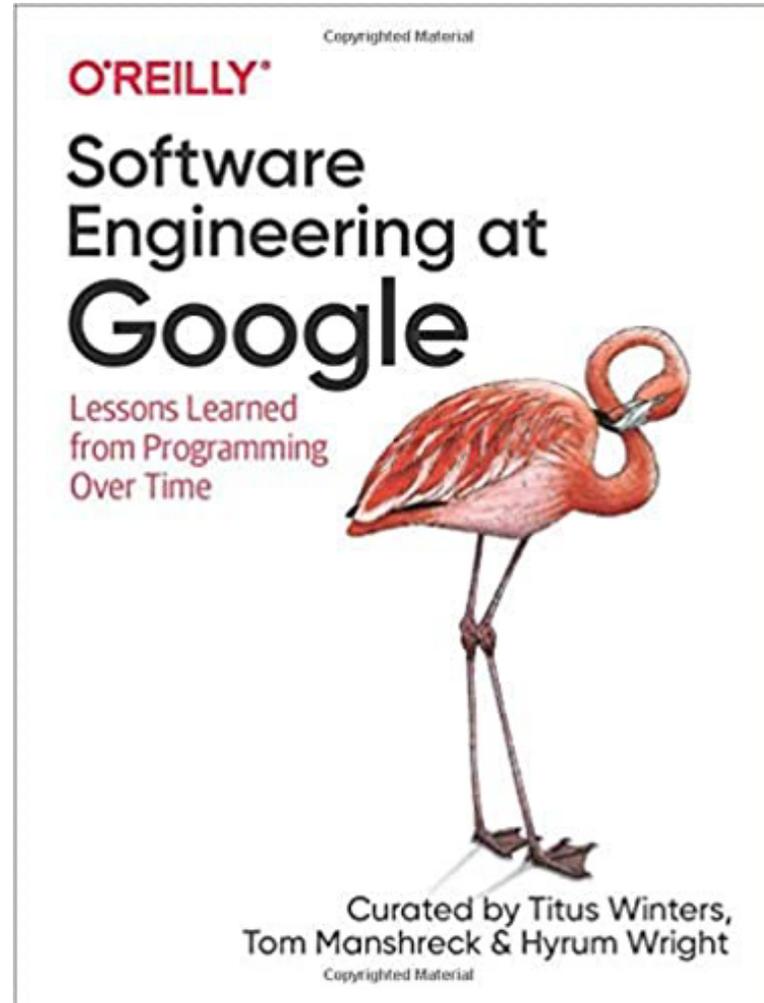


Ian Sommerville (2015),  
**Software Engineering,**  
10<sup>th</sup> Edition, Pearson.



Titus Winters, Tom Manshreck, and Hyrum Wright (2020),

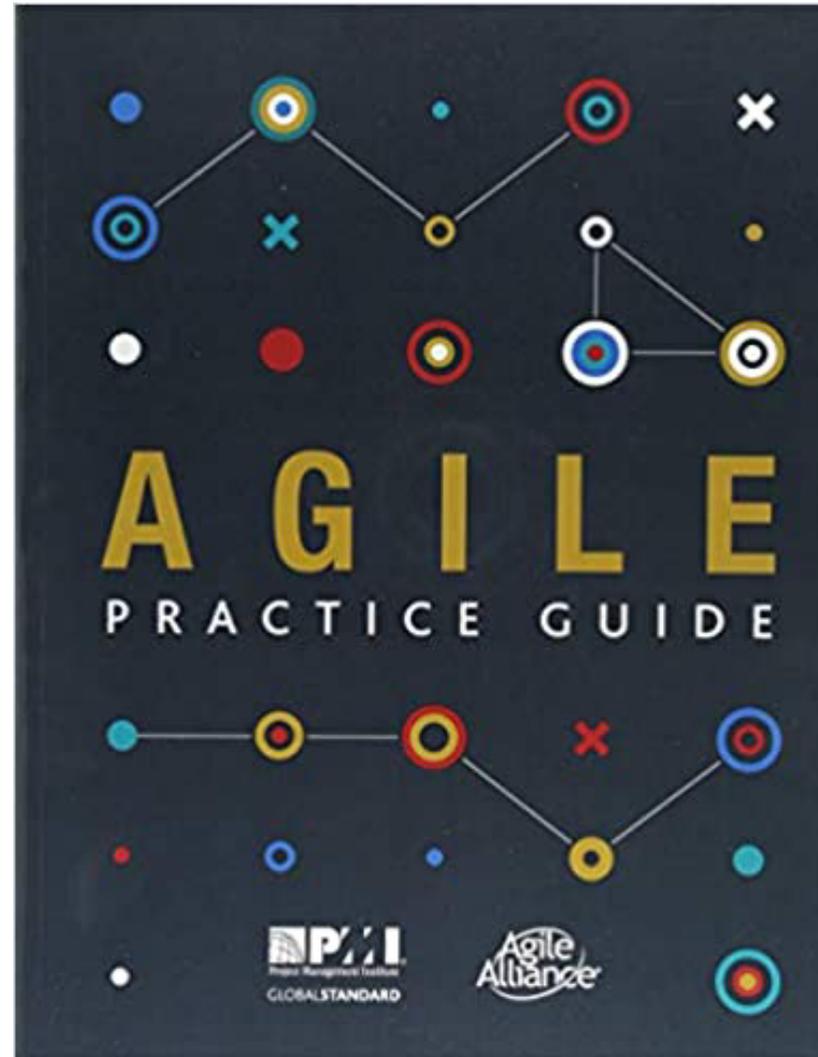
**Software Engineering at Google:**  
**Lessons Learned from Programming Over Time,**  
O'Reilly Media.



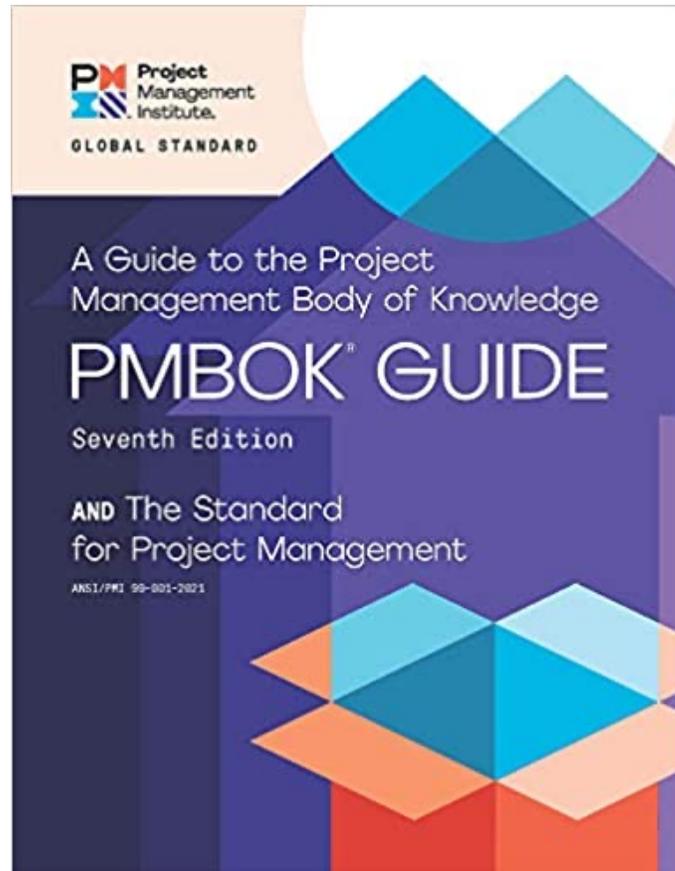
Project Management Institute (2017),

# Agile Practice Guide

PMI

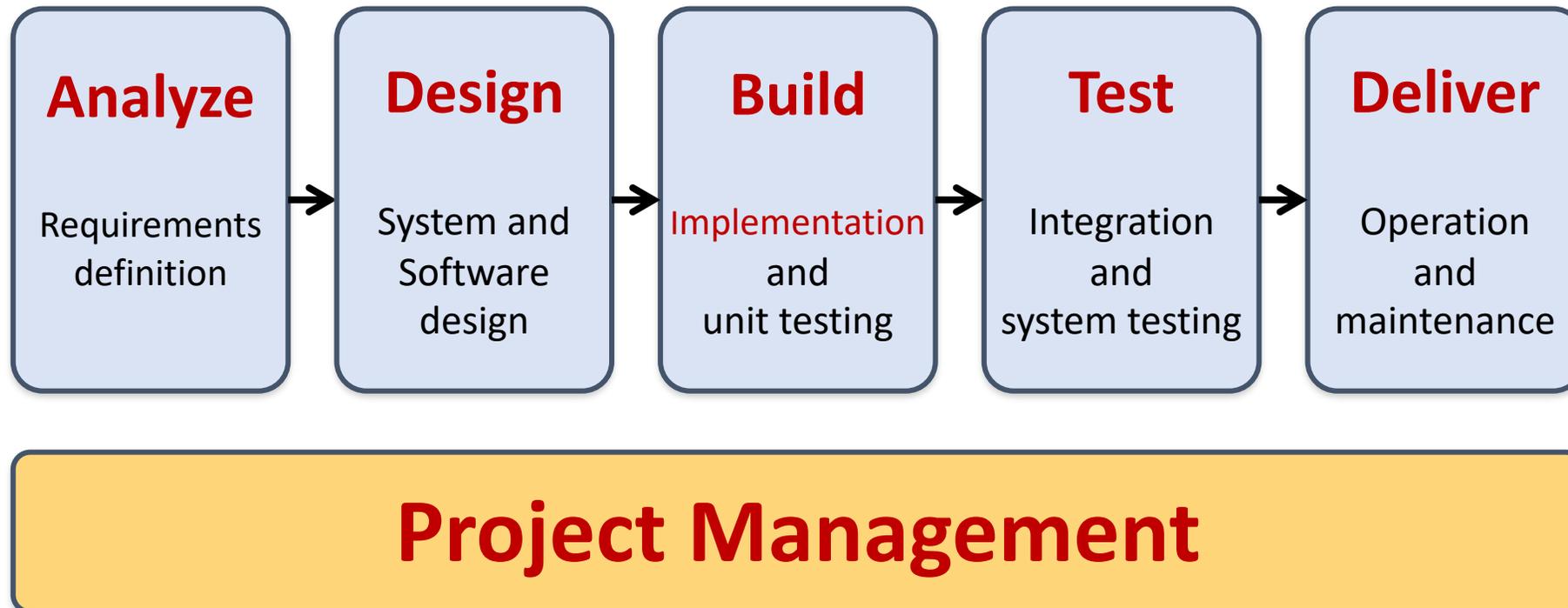


Project Management Institute (2021),  
**A Guide to the  
Project Management Body of Knowledge  
(PMBOK Guide) –  
Seventh Edition and The Standard for Project Management**



# Software Engineering

# Software Engineering and Project Management

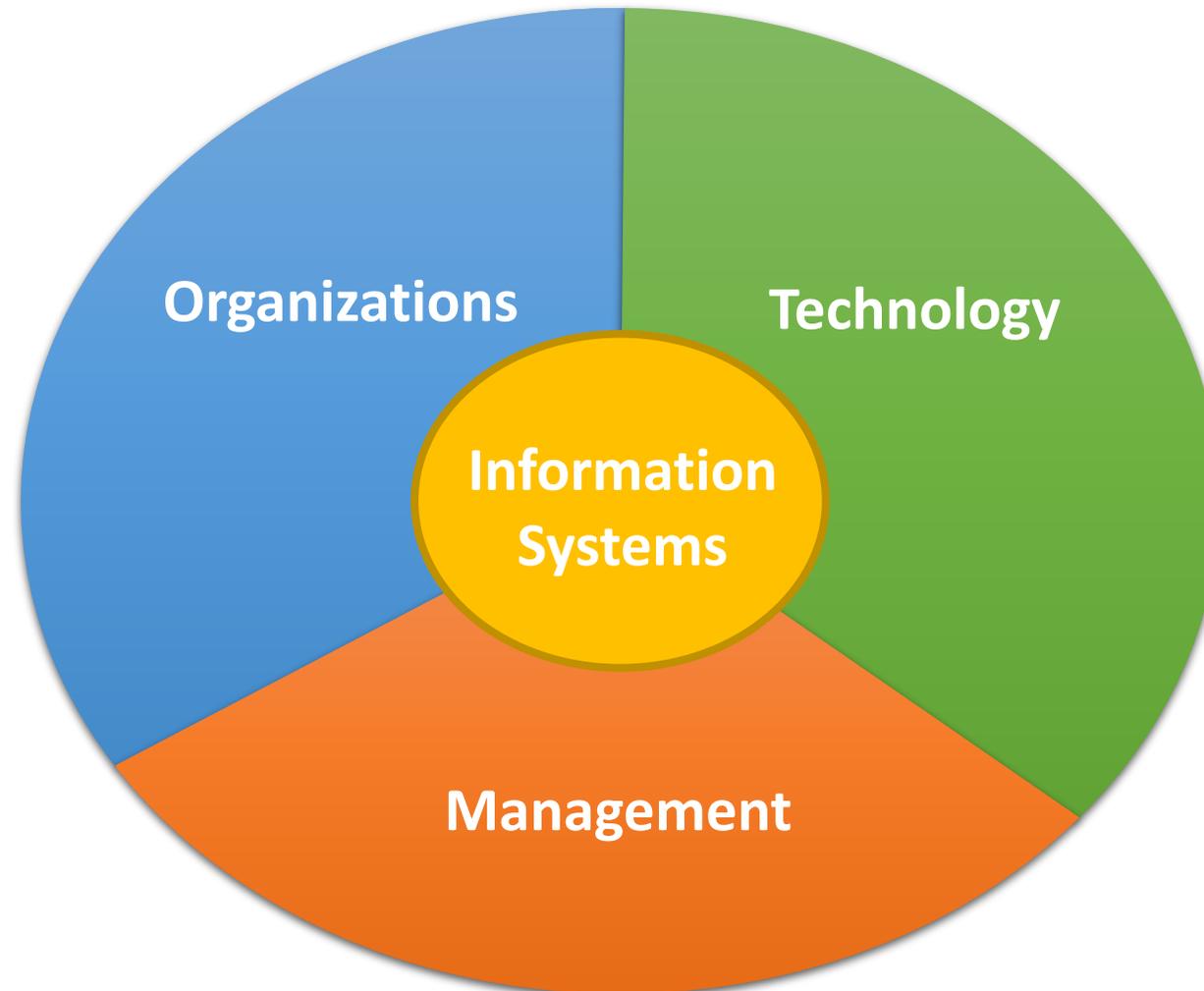


**Information Management**

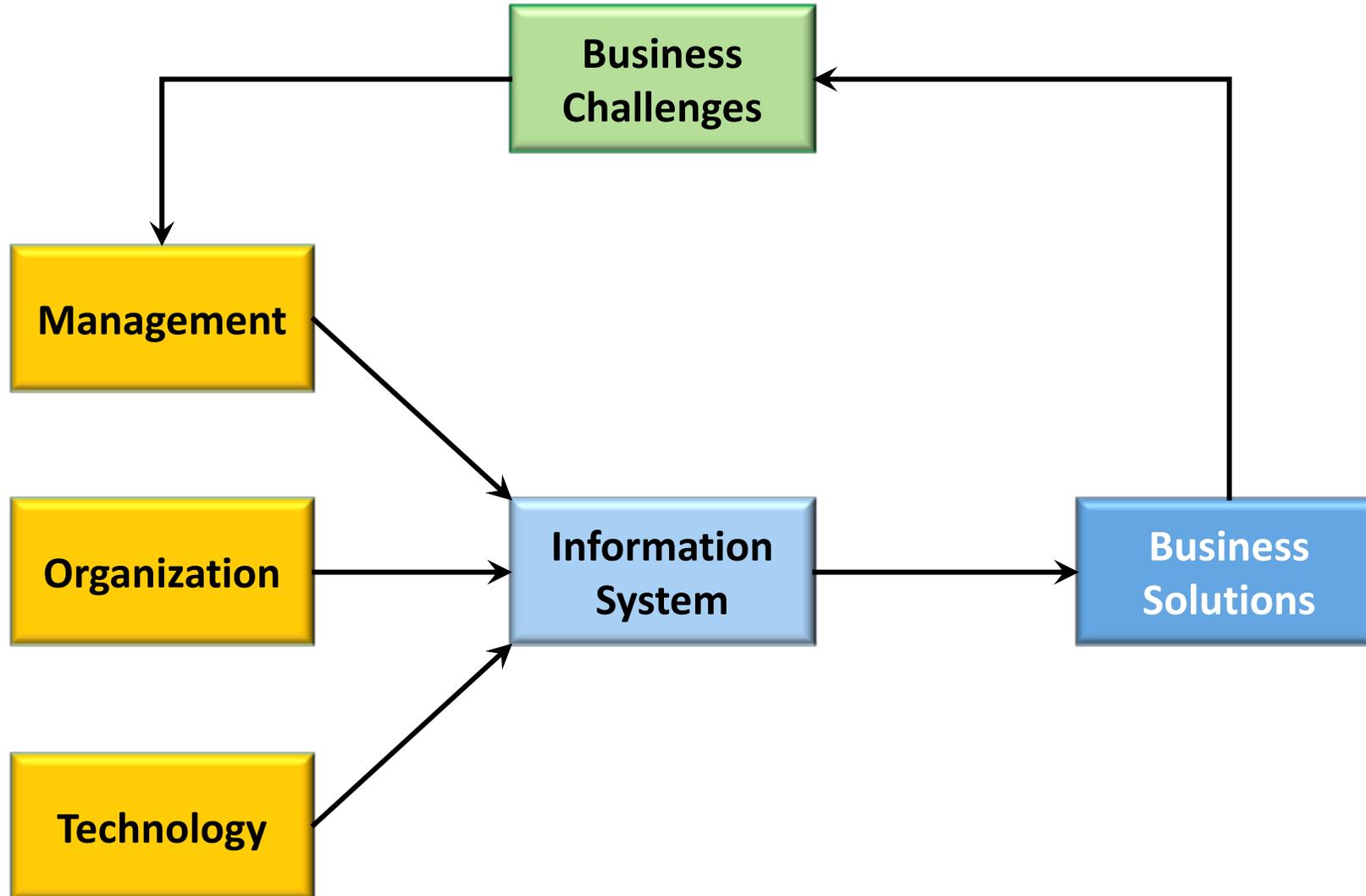
**Management  
Information Systems (MIS)**

**Information Systems**

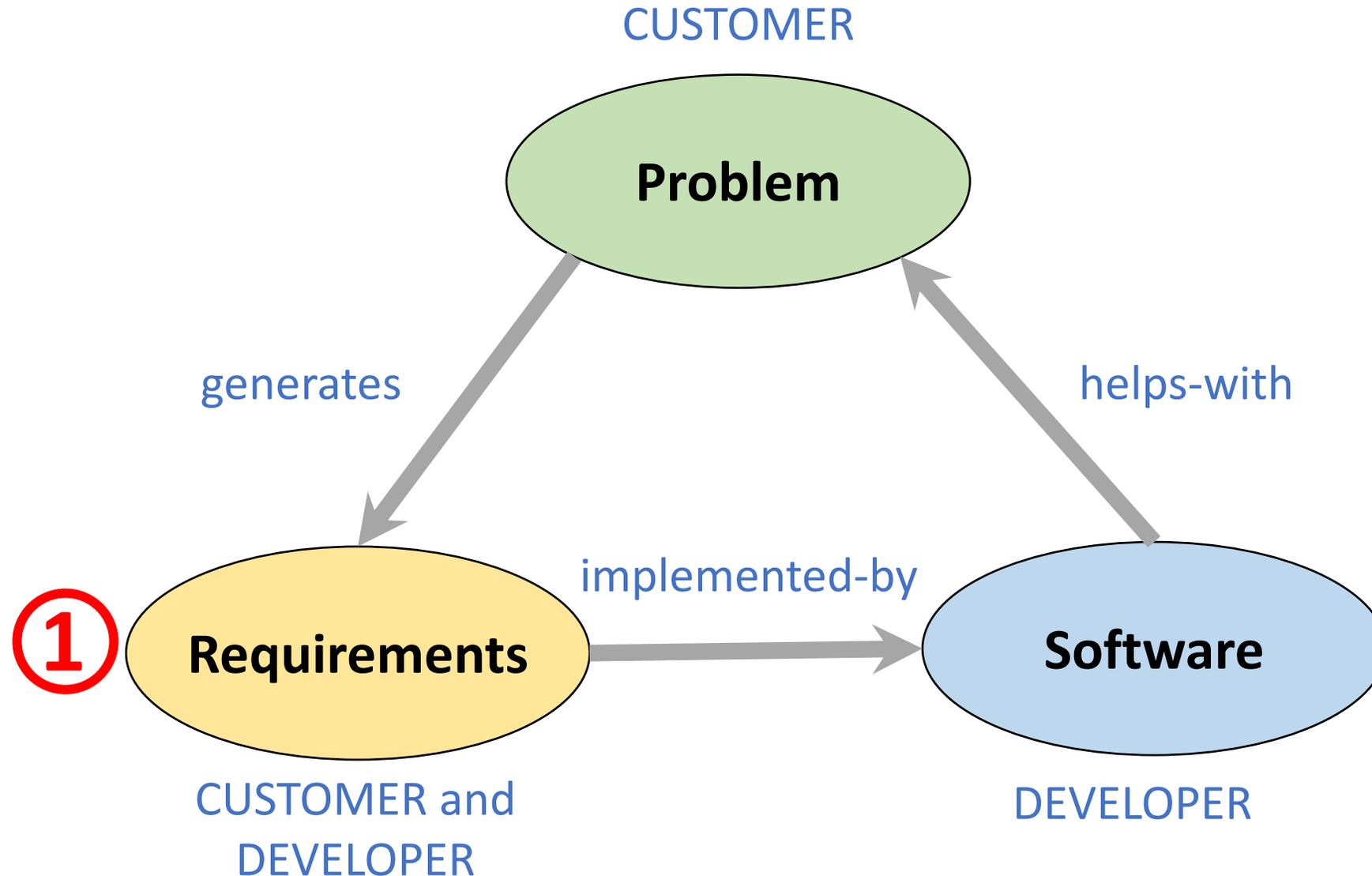
# Information Management (MIS) Information Systems



# Fundamental MIS Concepts



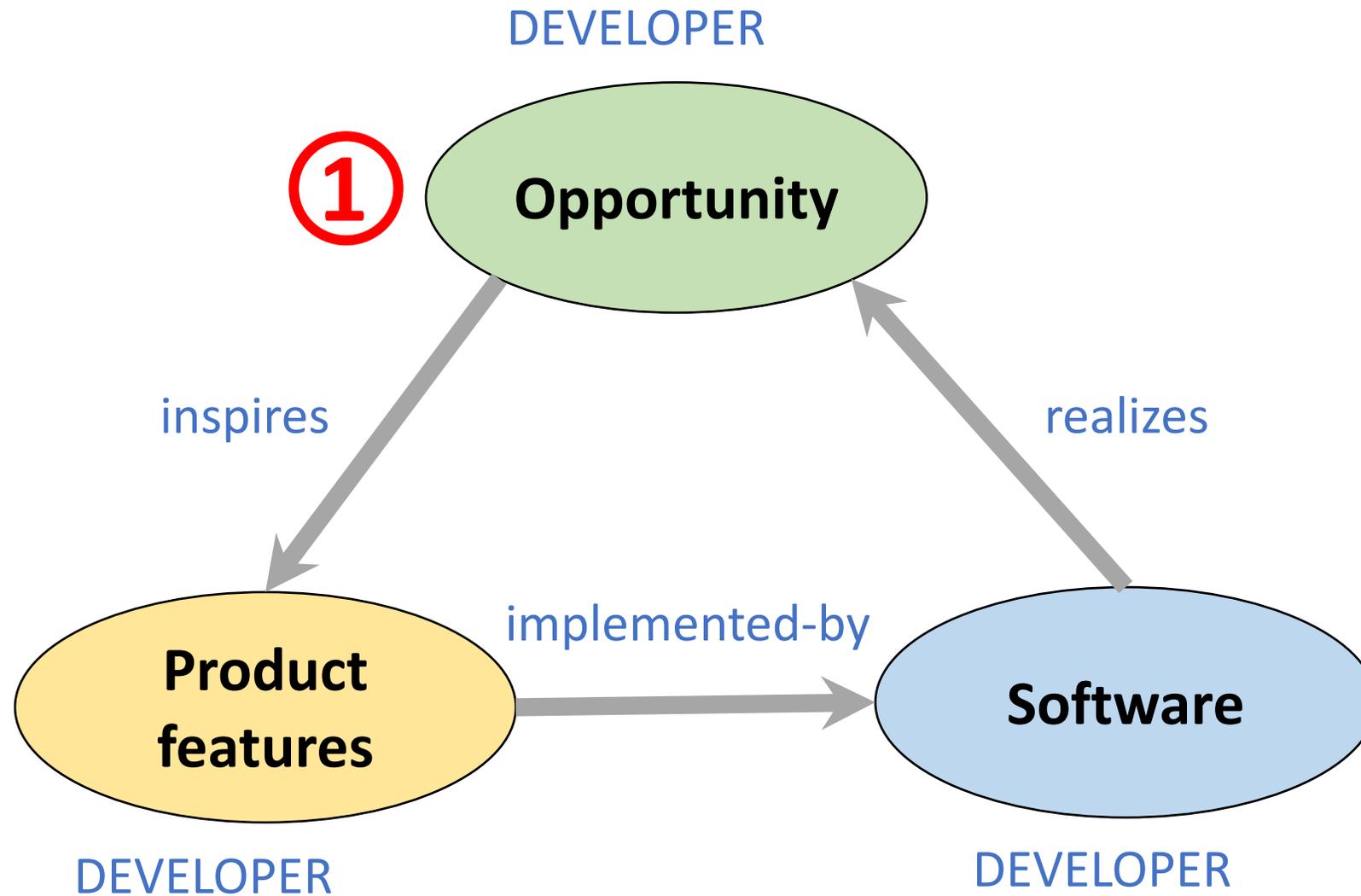
# Project-based software engineering



# Project-based software engineering

- The starting point for the software development is a set of ‘**software requirements**’ that are owned by an external client and which set out what they want a software system to do to support their business processes.
- The software is developed by a software company (the contractor) who **design and implement a system** that delivers functionality to meet the requirements.
- The customer may change the requirements at any time in response to business changes (they usually do). The contractor must change the software to reflect these requirements changes.
- Custom software usually has a long-lifetime (10 years or more) and it must be supported over that lifetime.

# Product software engineering

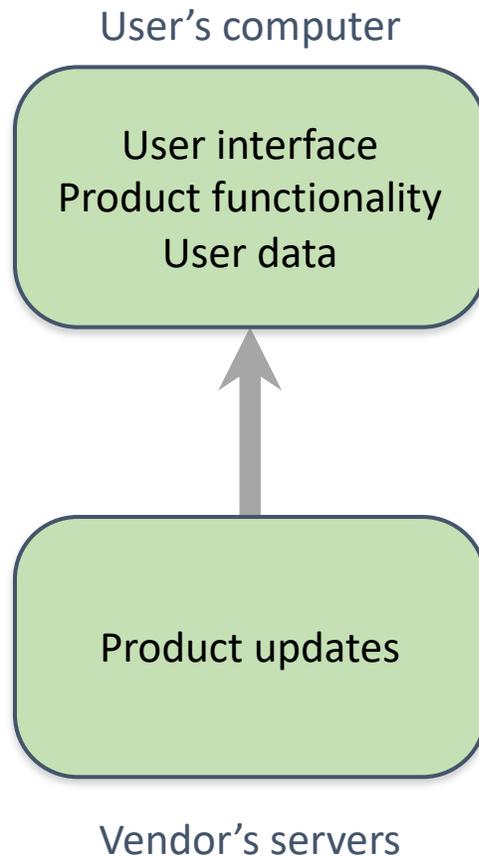


# Product software engineering

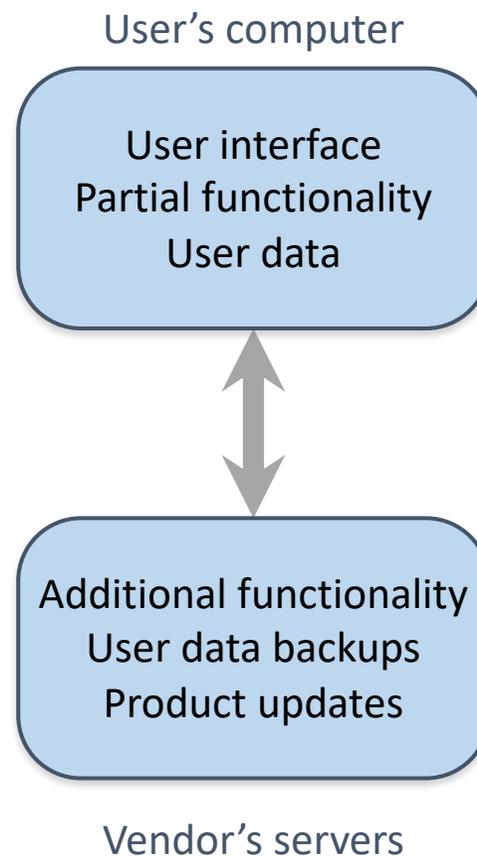
- The starting point for product development is a **business opportunity** that is identified by individuals or a company.  
They develop a software product to take advantage of this opportunity and sell this to customers.
- The company who identified the opportunity **design and implement a set of software features** that realize the opportunity and that will be useful to customers.
- The software development company are responsible for deciding on the development timescale, what features to include and when the product should change.
- Rapid delivery of software products is essential to capture the market for that type of product.

# Software execution models

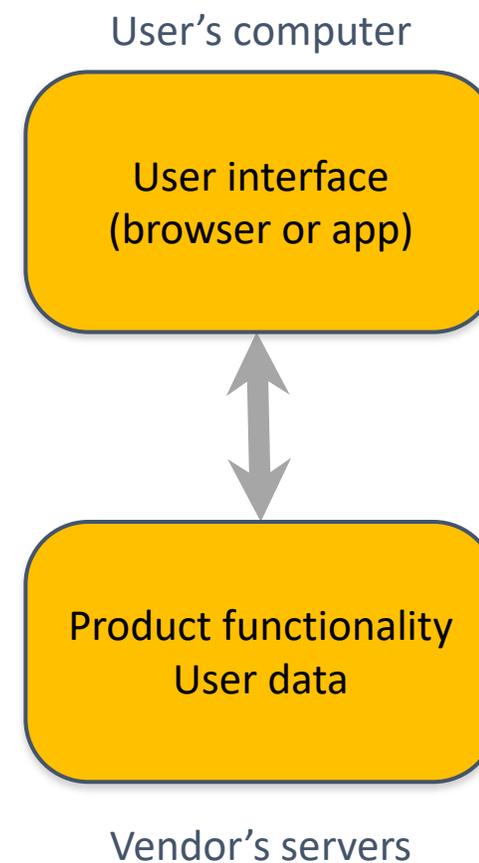
## Stand-alone execution



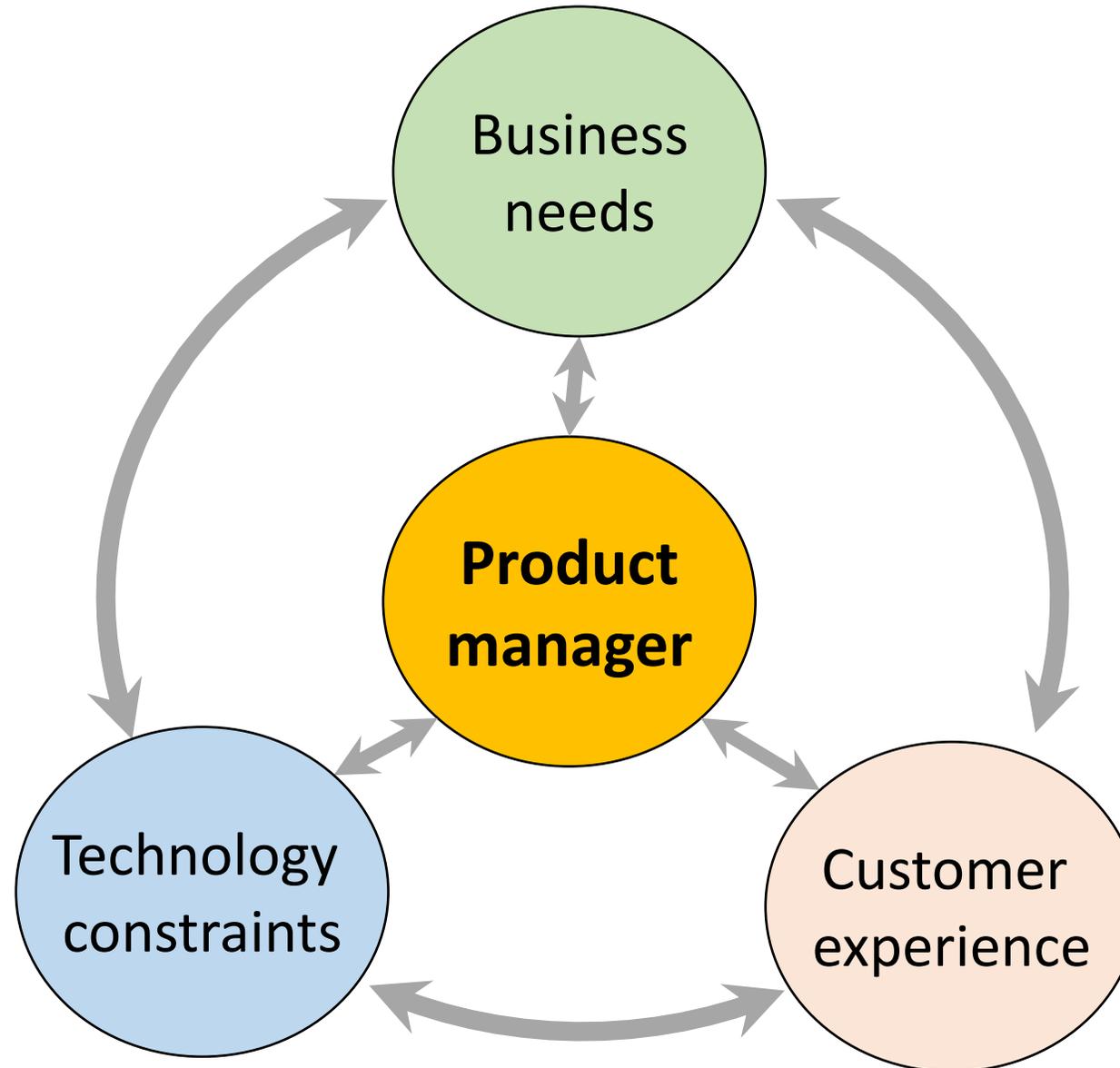
## Hybrid execution



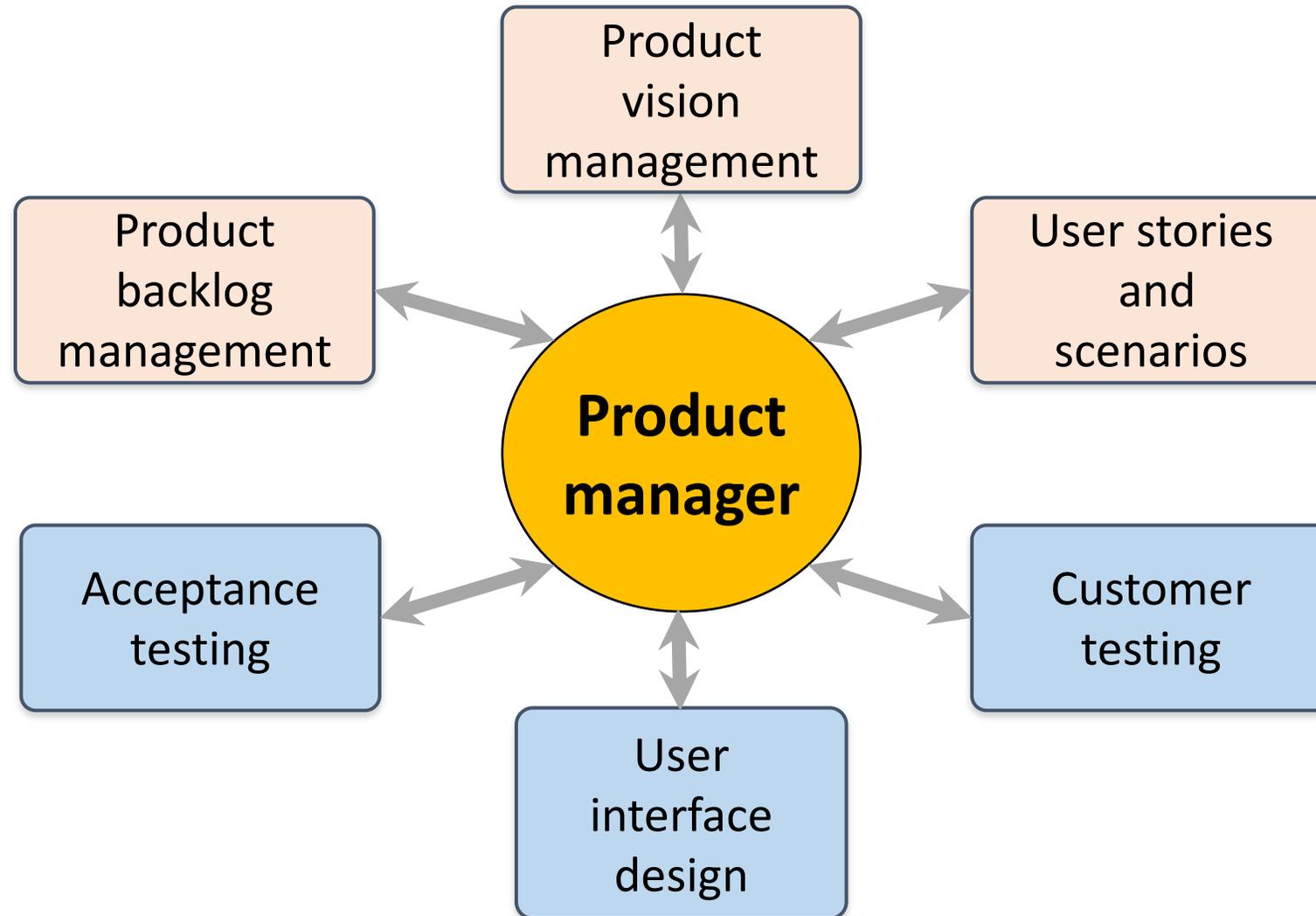
## Software as a service



# Product management concerns

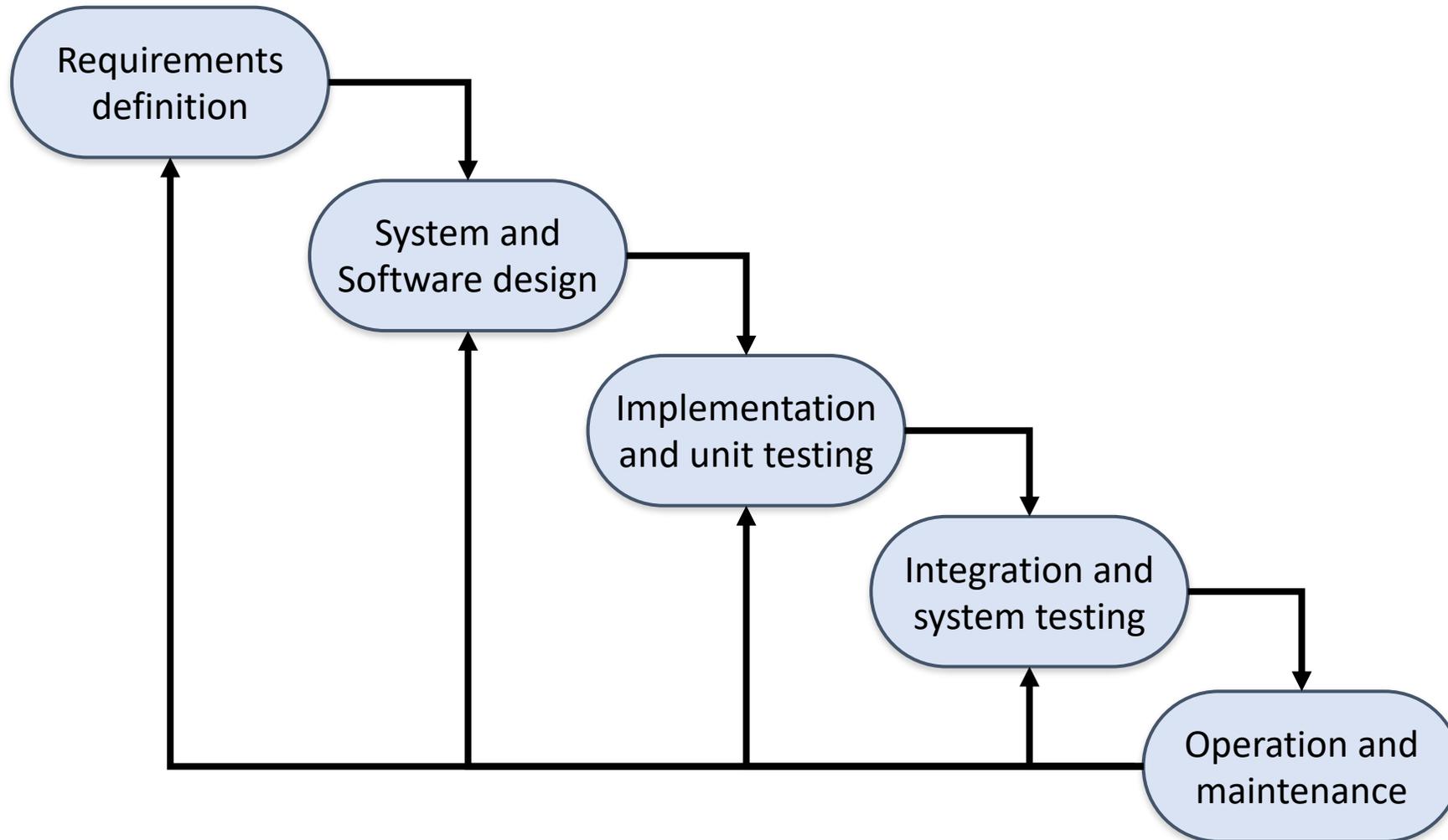


# Technical interactions of product managers



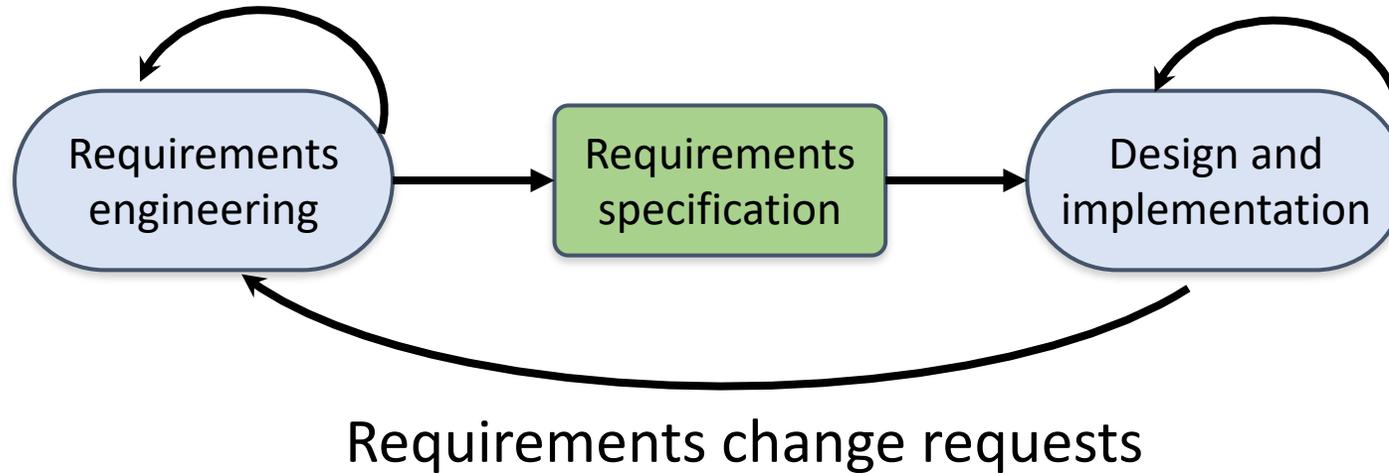
# Software Development Life Cycle (SDLC)

## The waterfall model

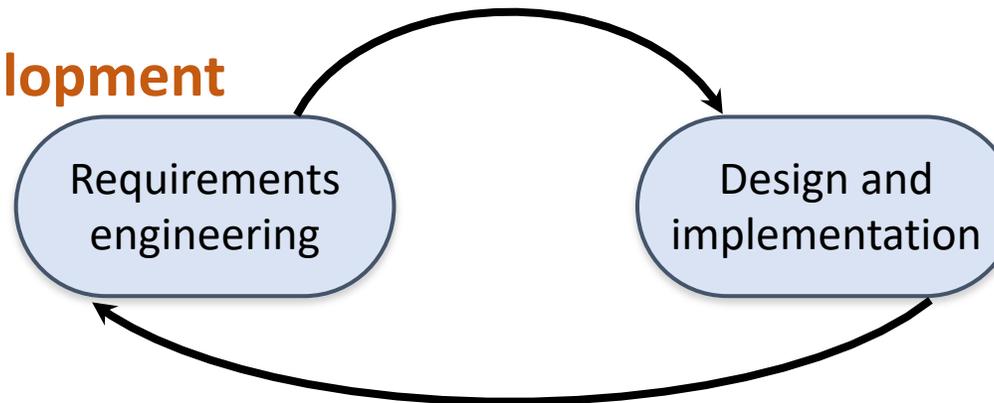


# Plan-based and Agile development

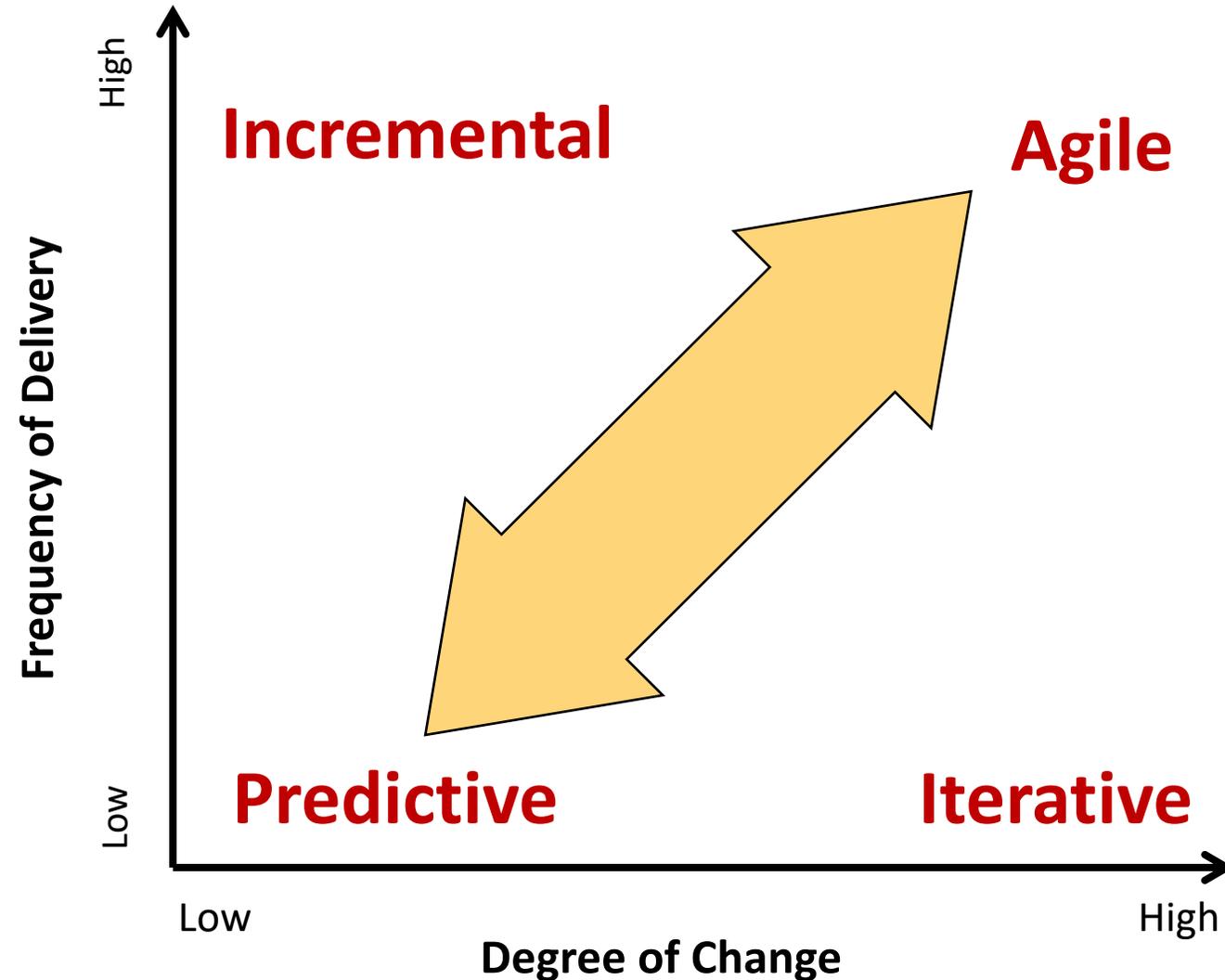
## Plan-based development



## Agile development



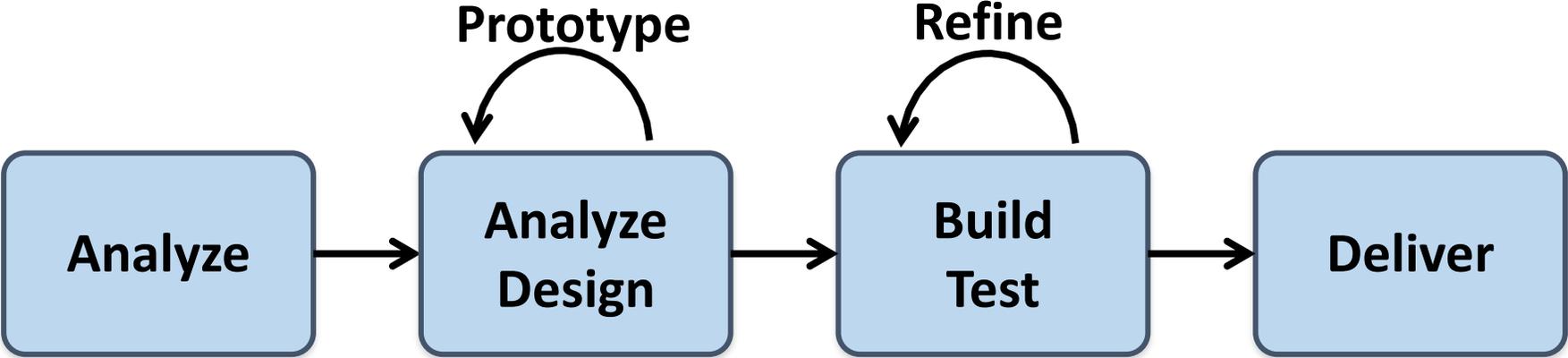
# The Continuum of Life Cycles



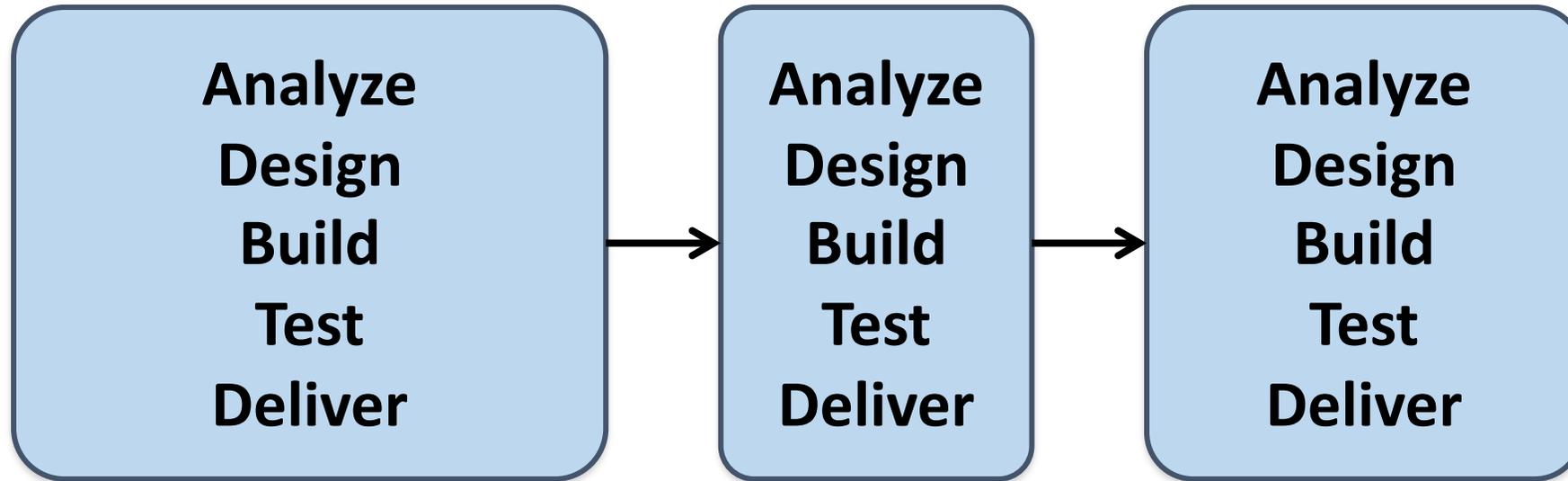
# Predictive Life Cycle



# Iterative Life Cycle

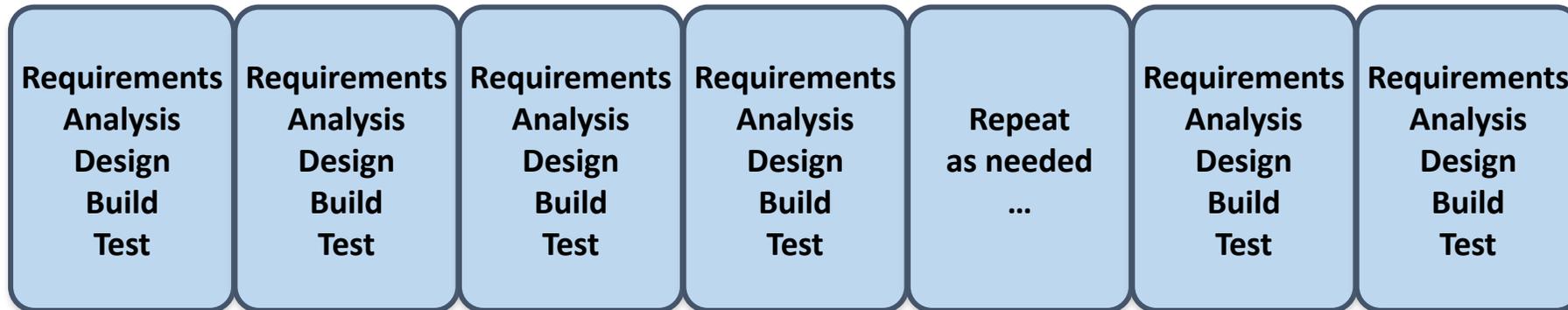


# A Life Cycle of Varying-Sized Increments

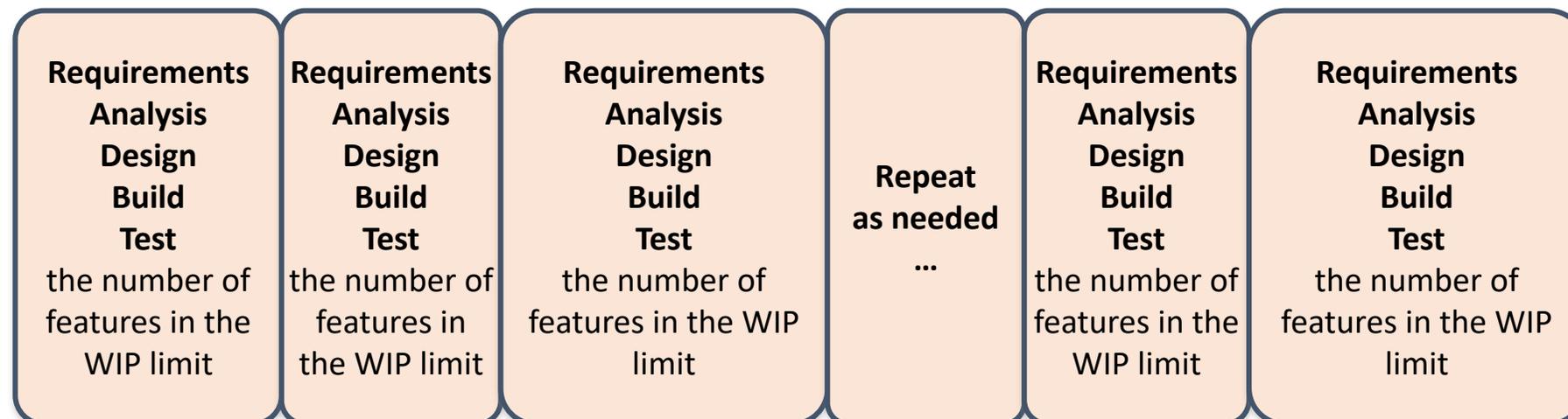


# Iteration-Based and Flow-Based Agile Life Cycles

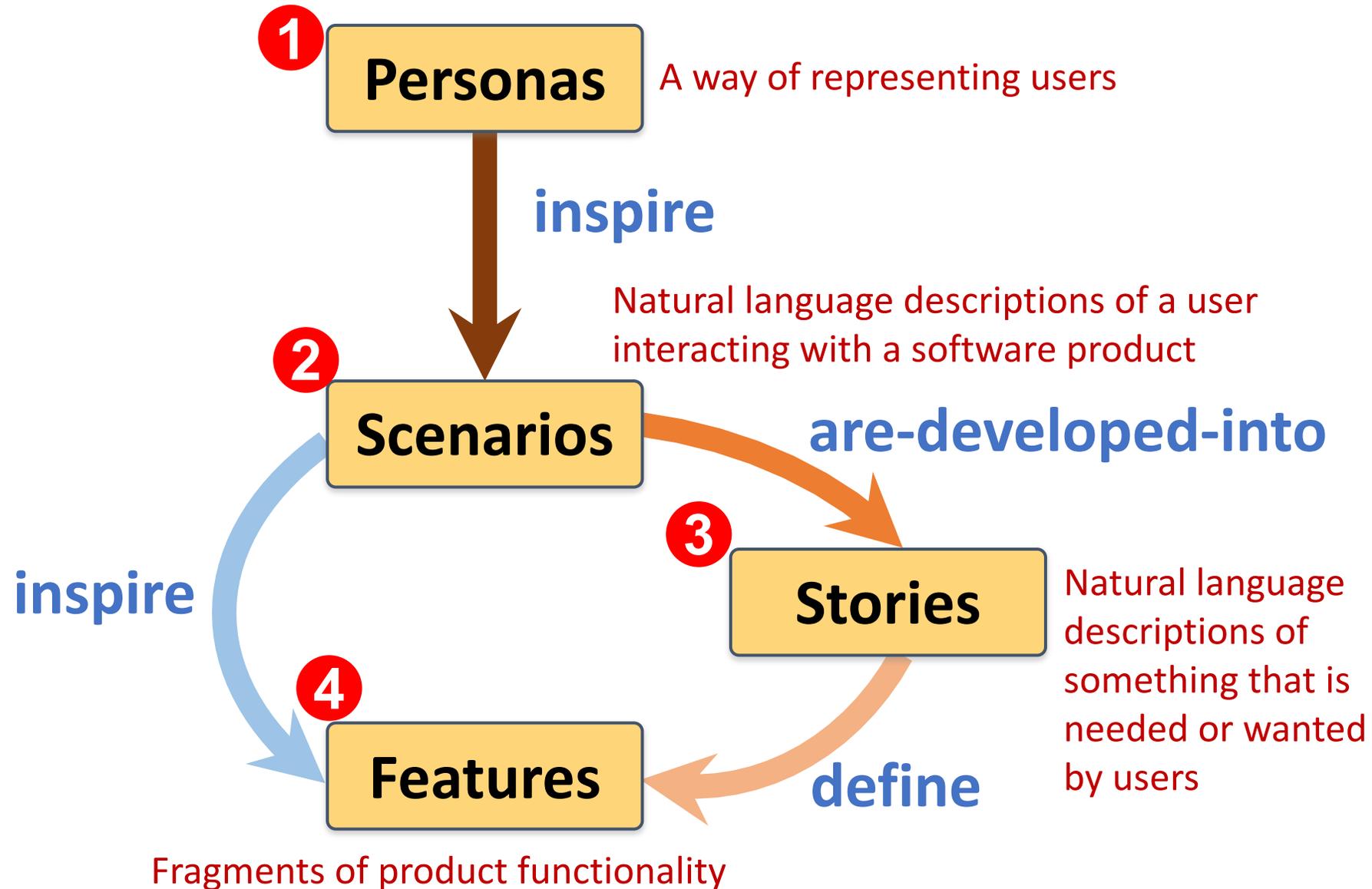
## Iteration-Based Agile



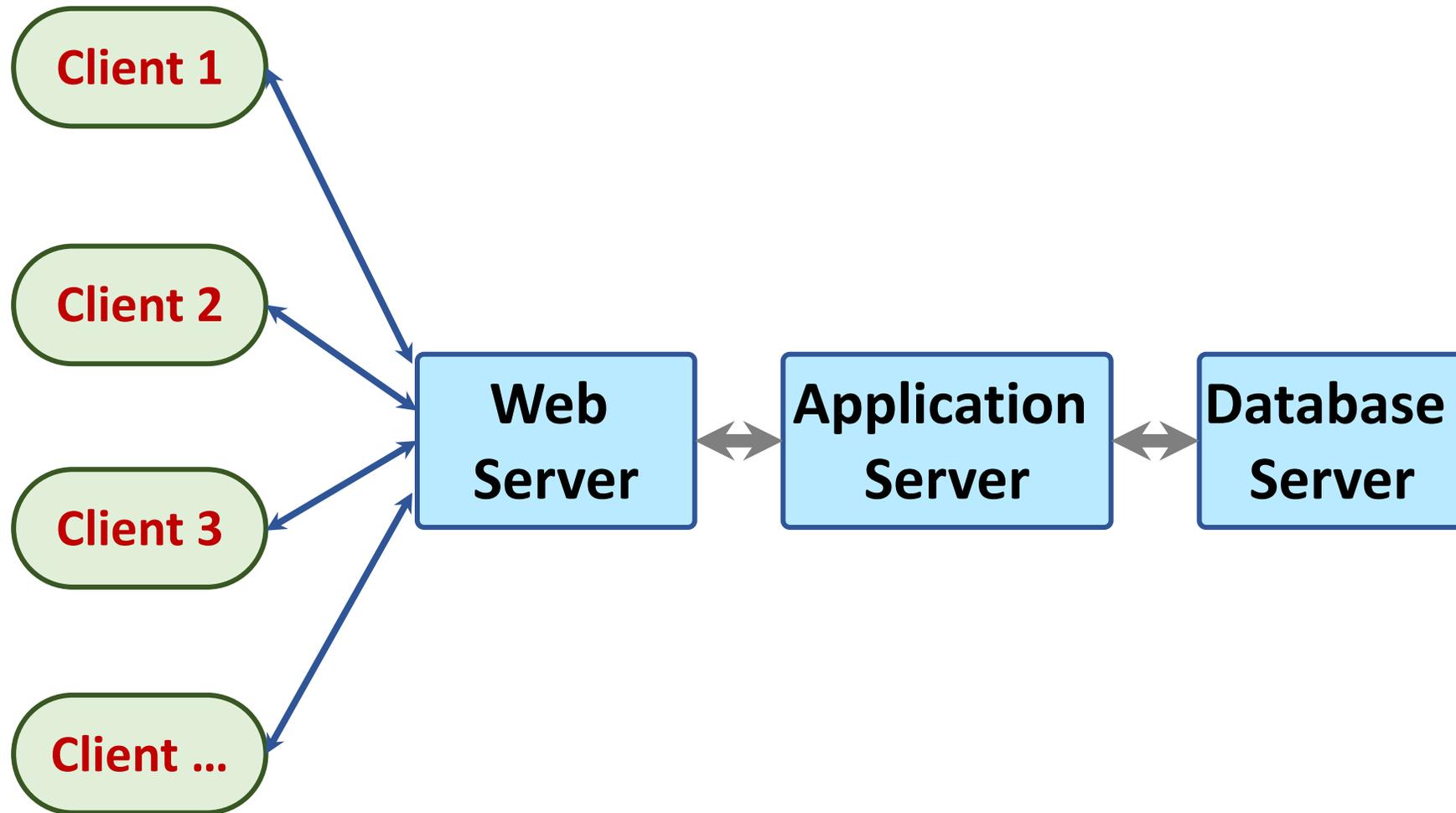
## Flow-Based Agile



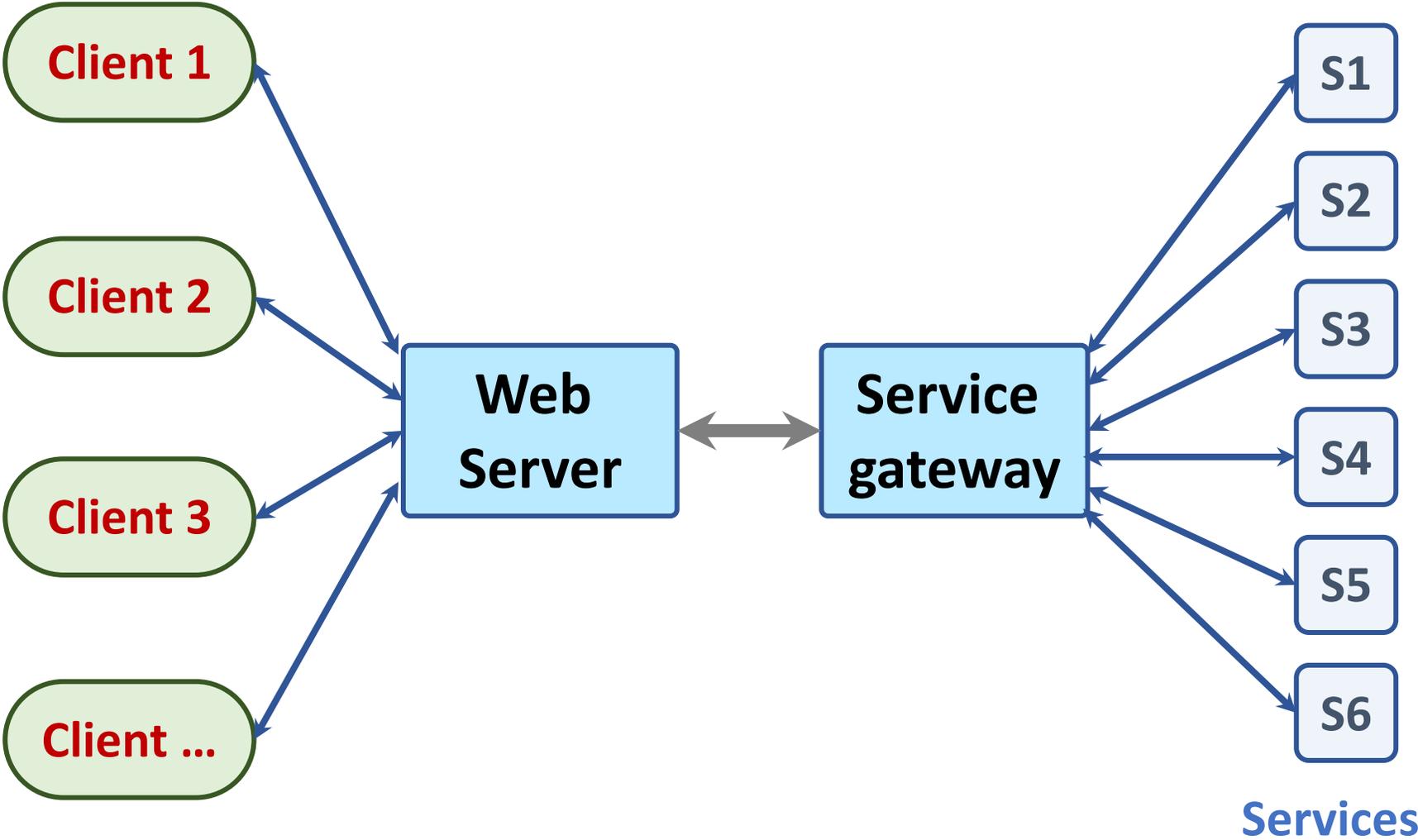
# From personas to features



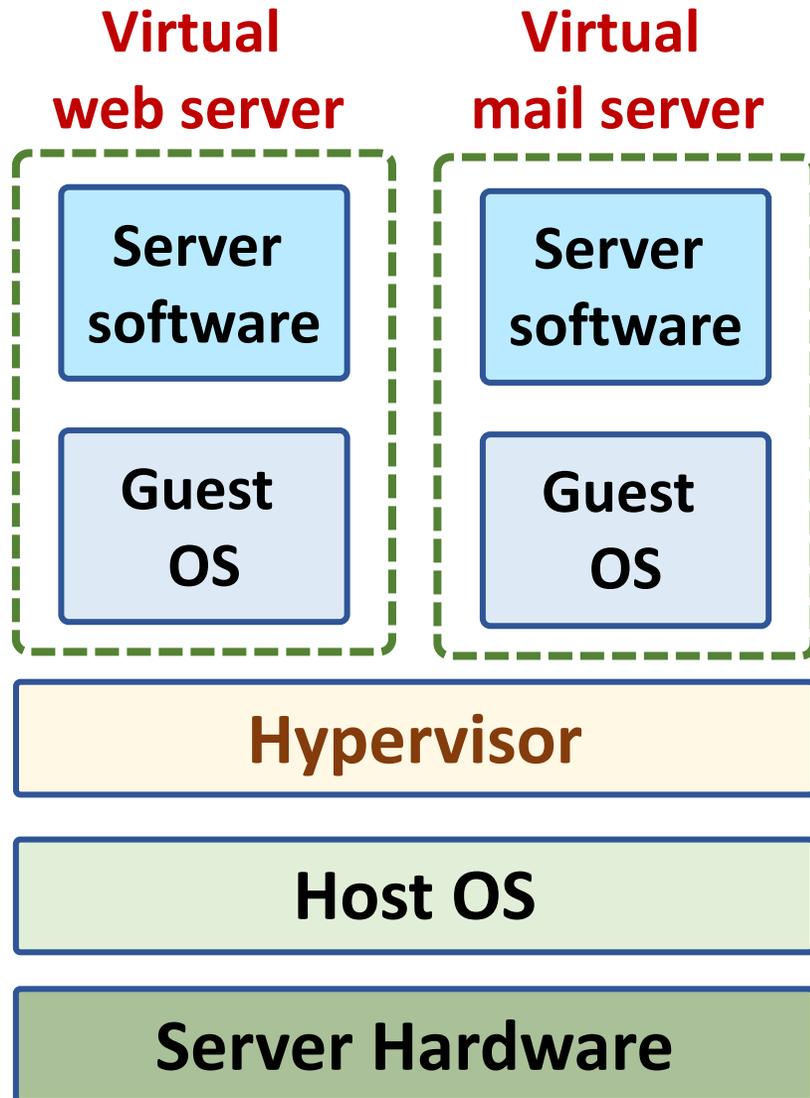
# Multi-tier client-server architecture



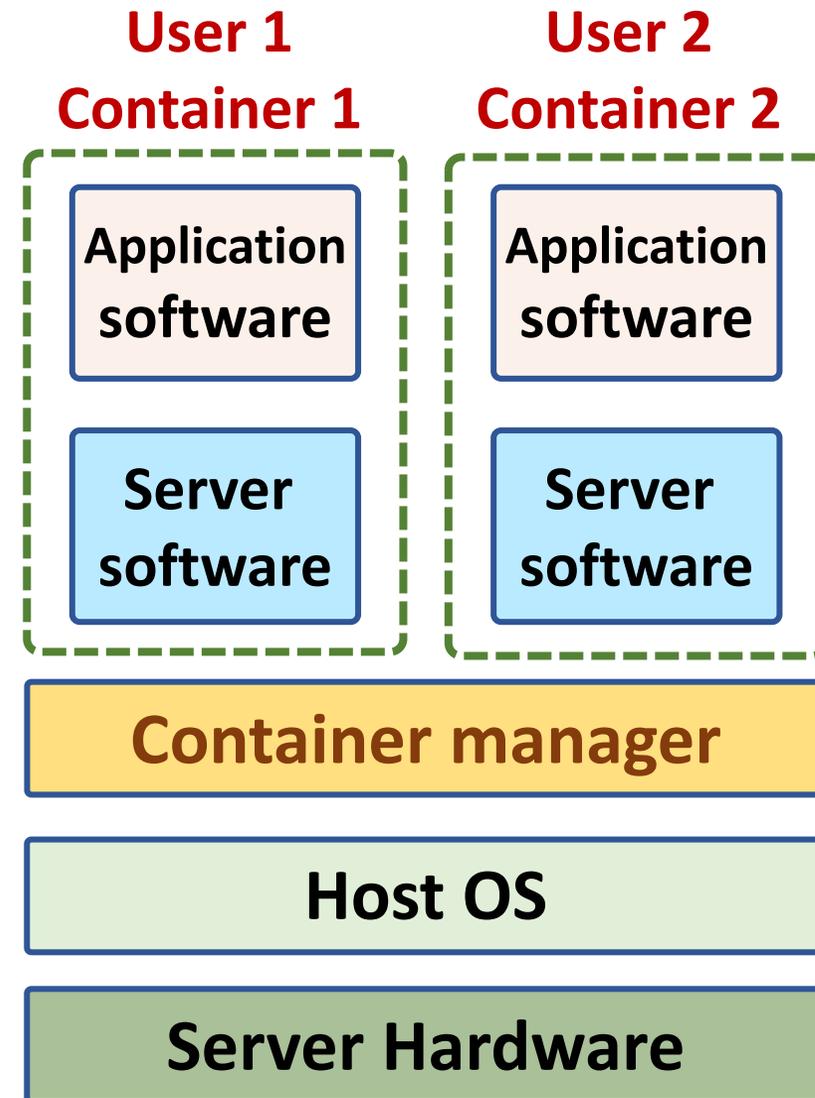
# Service-oriented Architecture



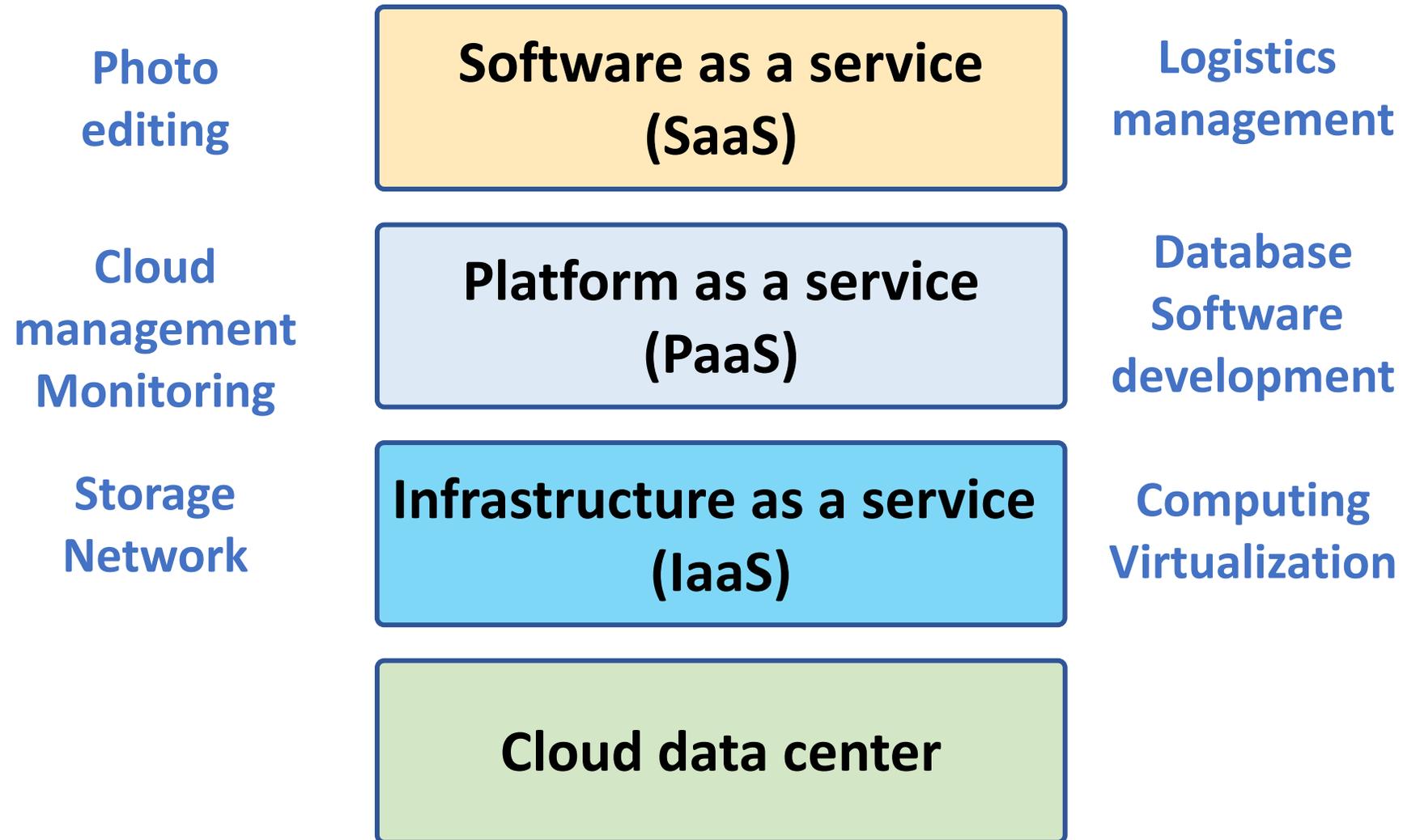
# VM



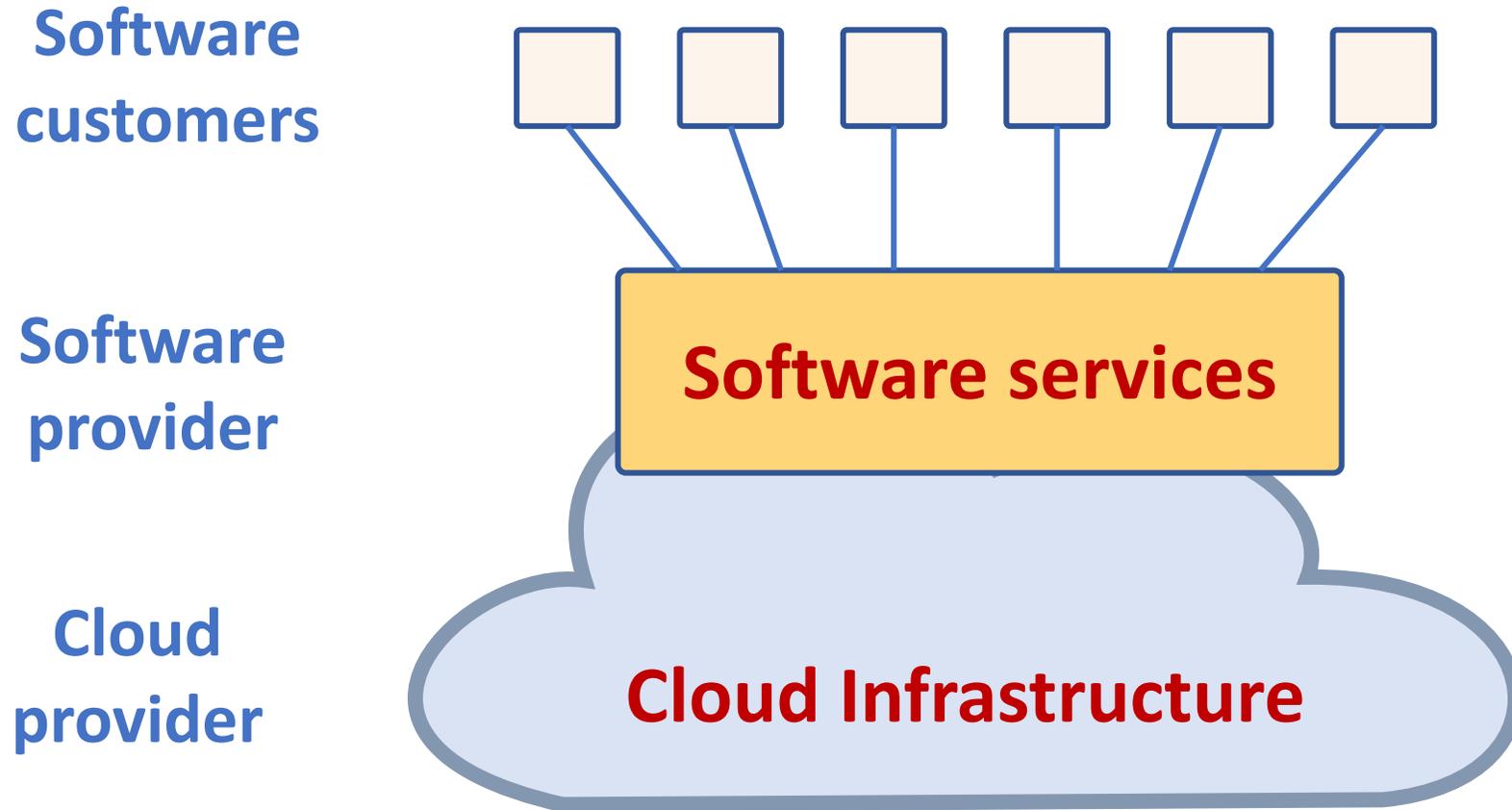
# Container



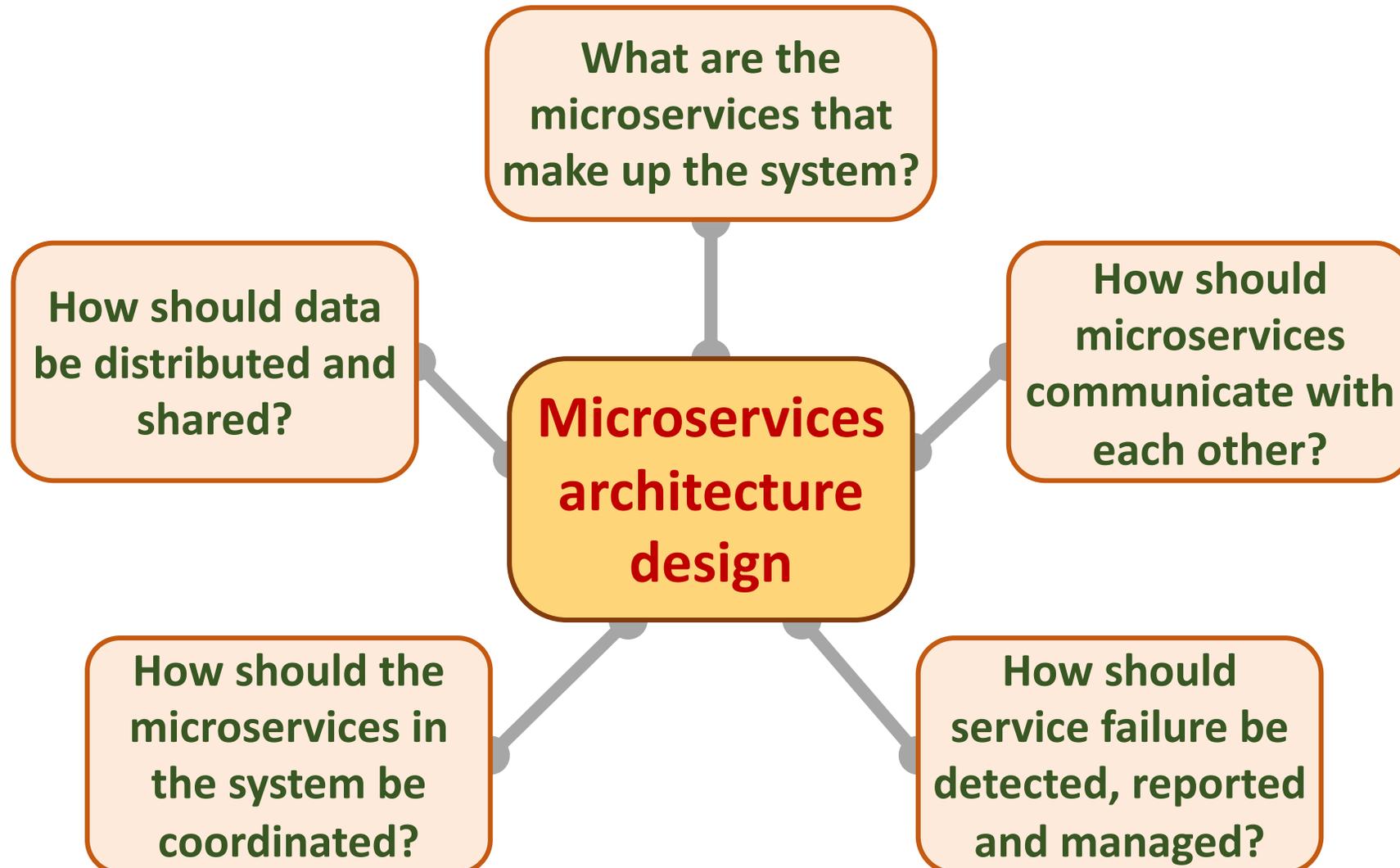
# Everything as a service



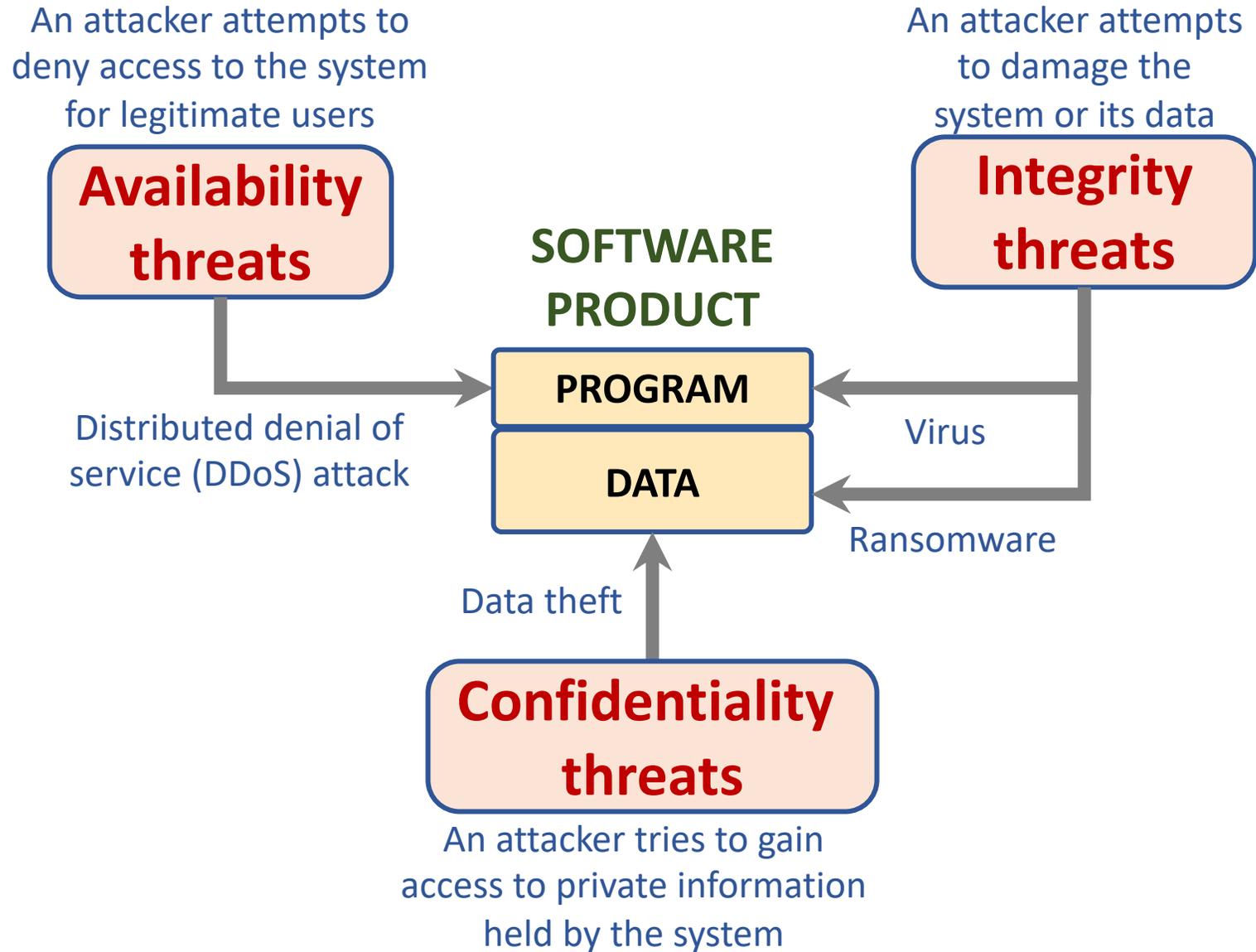
# Software as a service



# Microservices architecture – key design questions



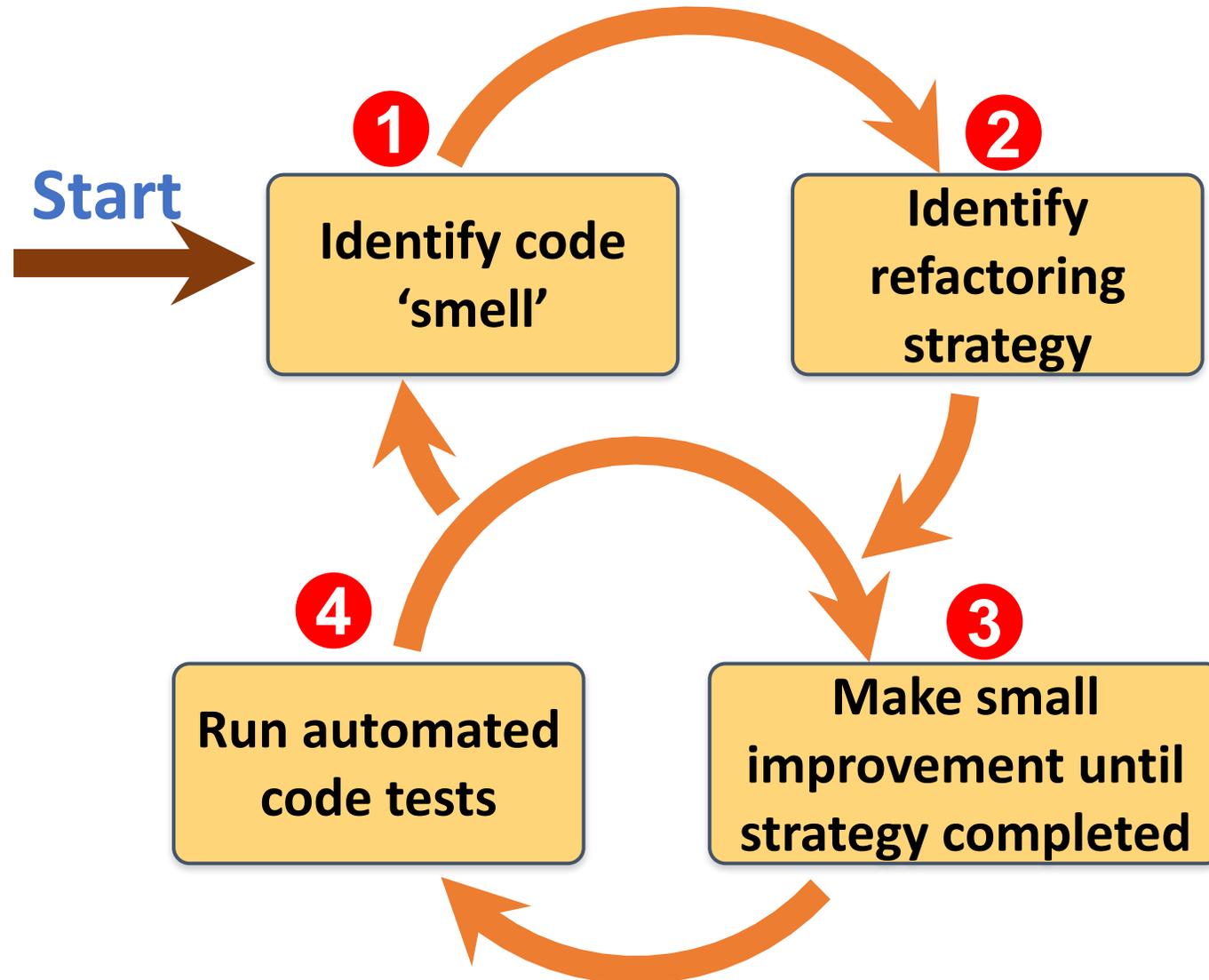
# Types of security threat



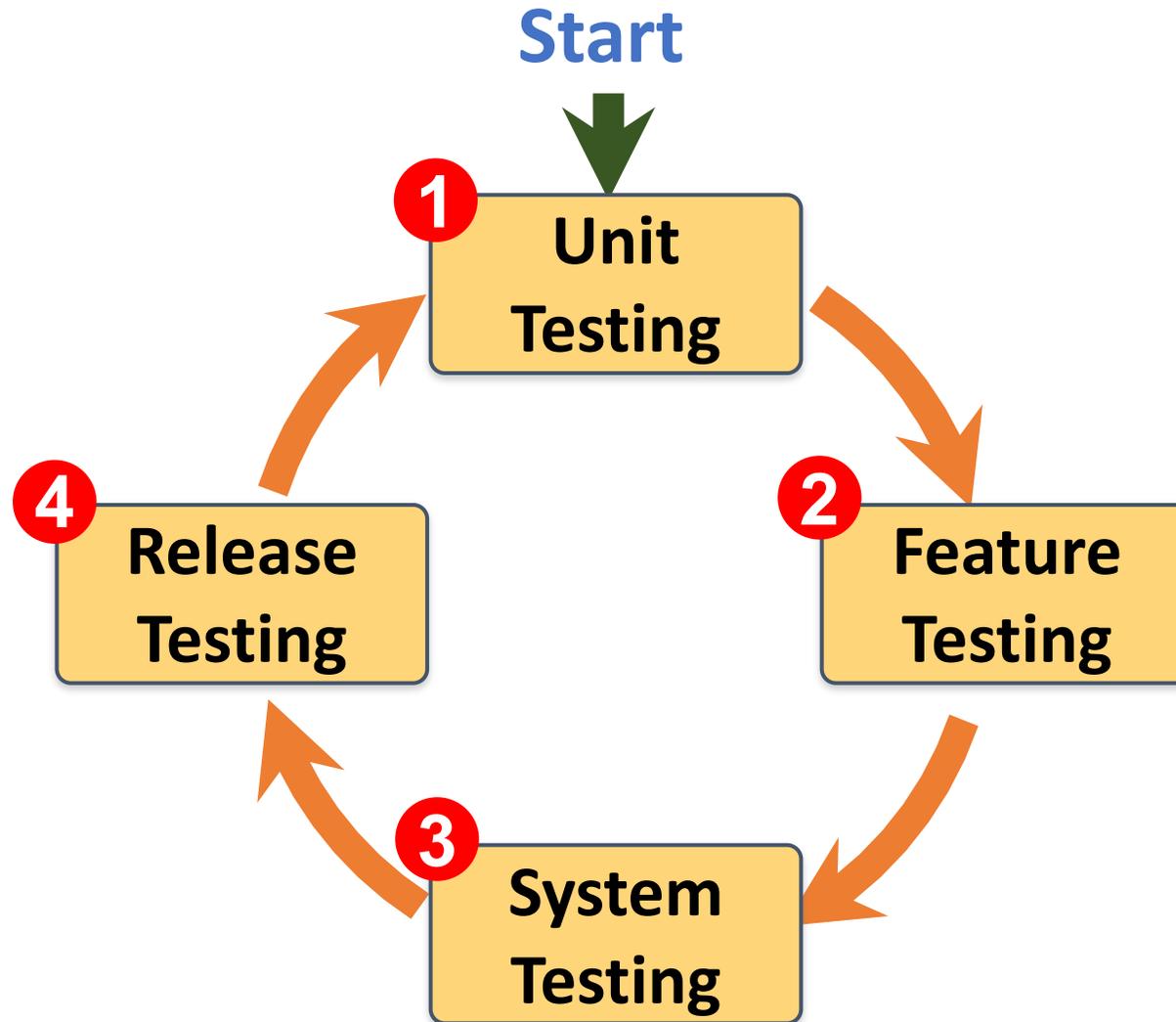
# Software product quality attributes



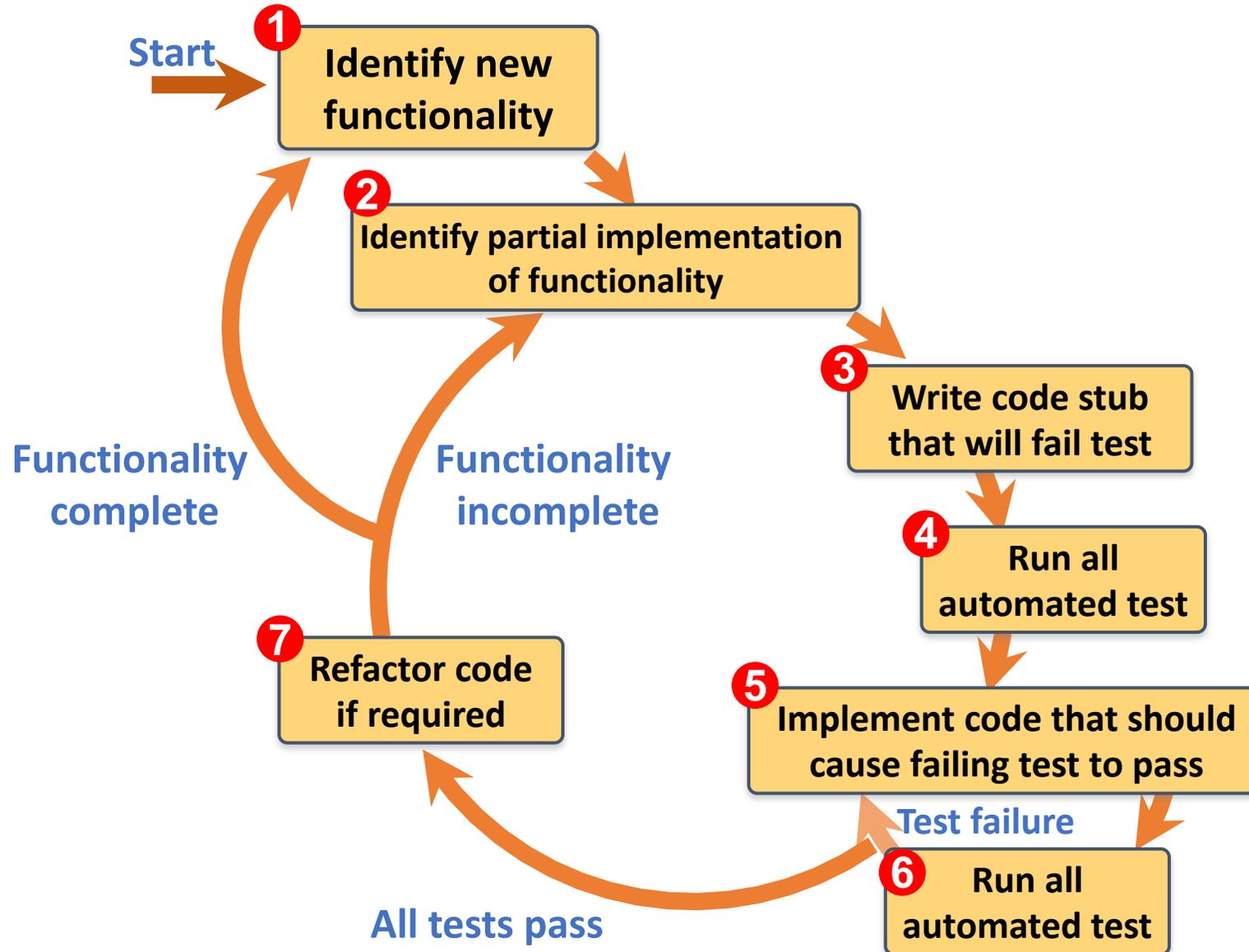
# A refactoring process



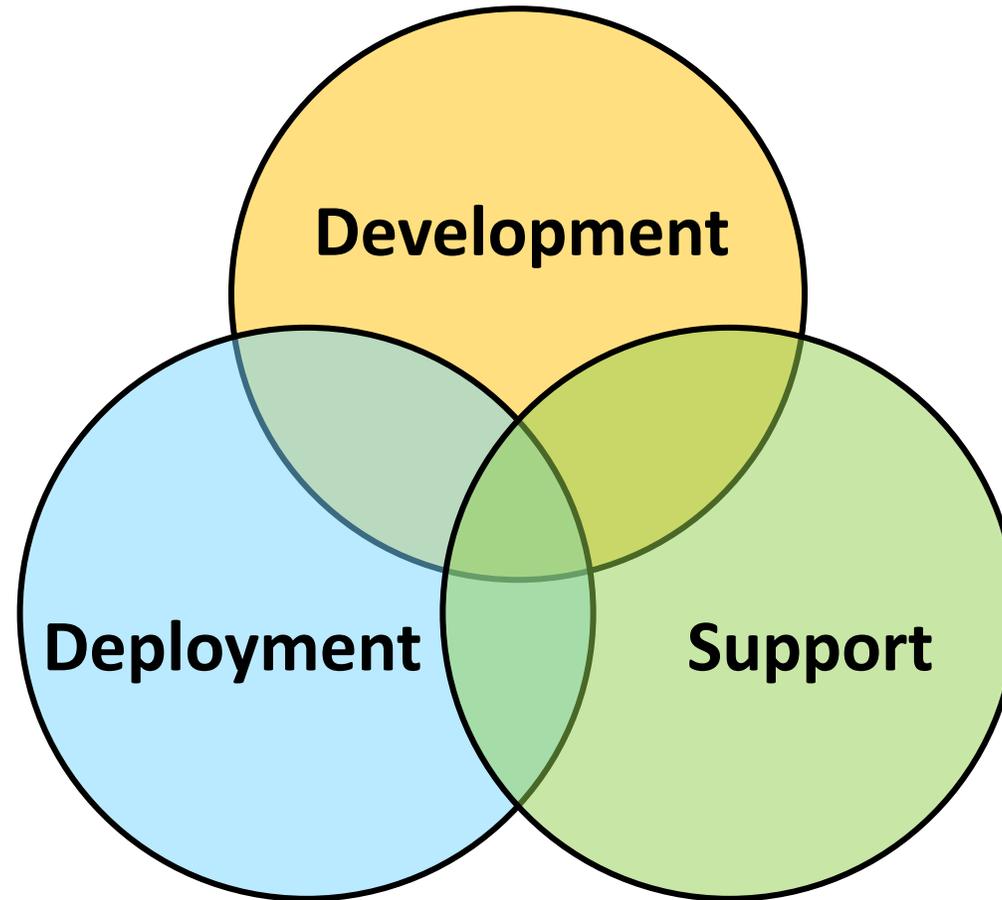
# Functional testing



# Test-driven development (TDD)

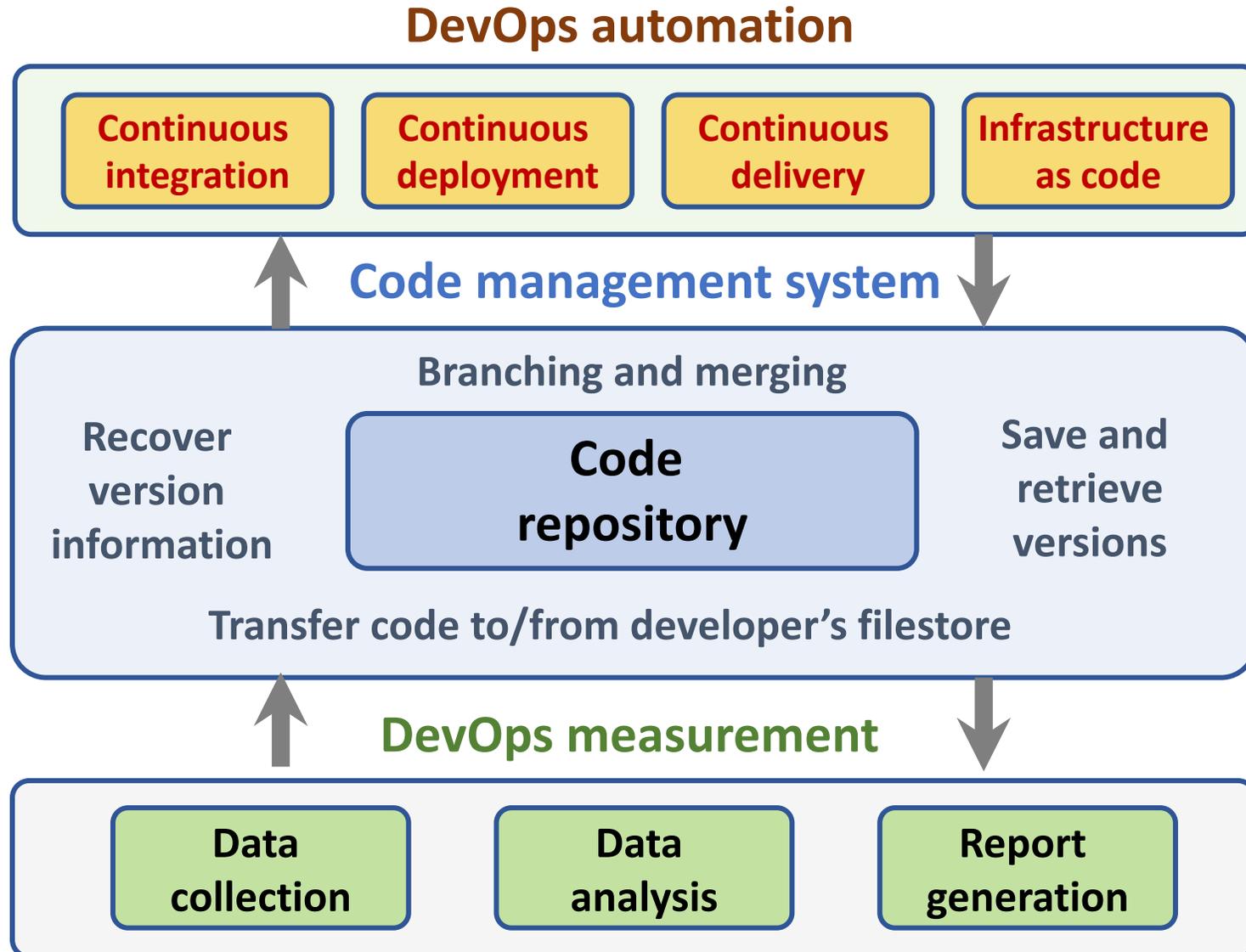


# DevOps



## Multi-skilled DevOps team

# Code management and DevOps



# Marketing

**Marketing**  
“**Meeting**  
**needs**  
**profitably**”

# Marketing

**“Marketing is an organizational function and a set of processes for creating, communicating, and delivering value to customers and for managing customer relationships in ways that benefit the organization and its stakeholders.”**

# Marketing Management

# Marketing Management

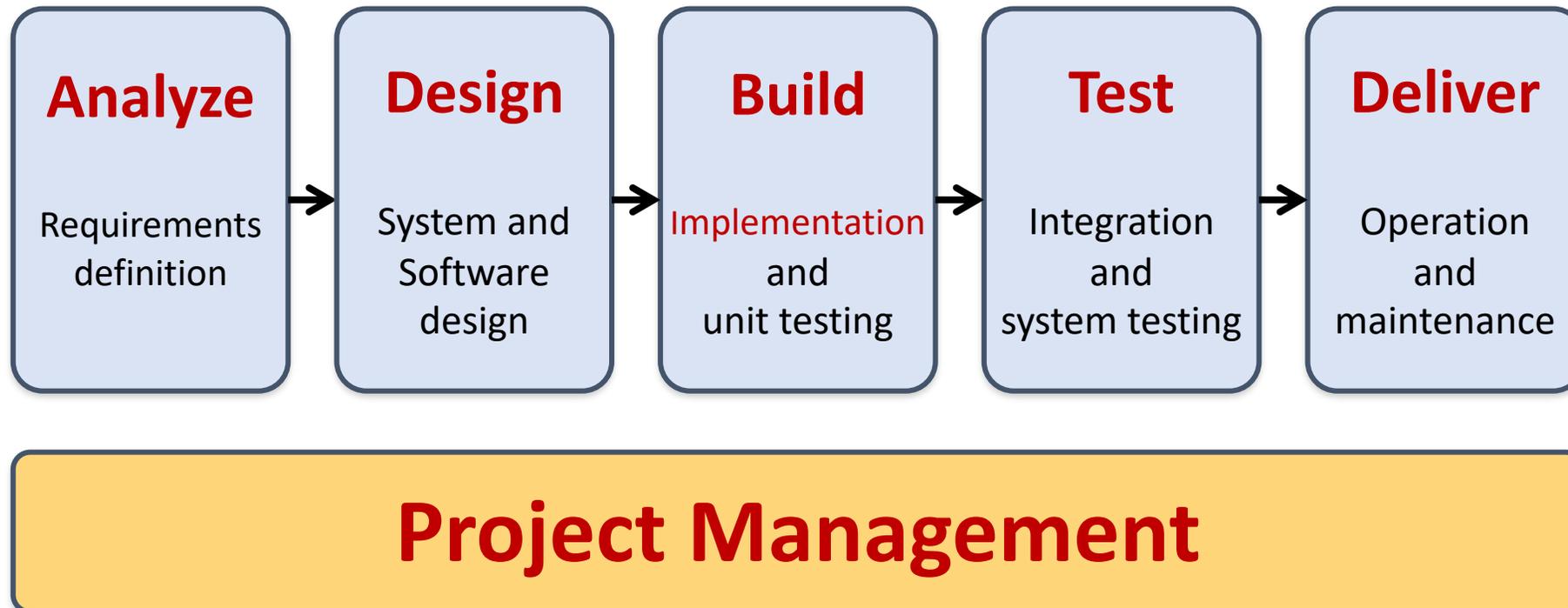
**“Marketing management is the  
art and science  
of choosing target markets  
and getting, keeping, and growing  
customers through  
creating, delivering, and communicating  
superior customer value.”**

# Marketing Management

- 1 Understanding Marketing Management
- 2 Capturing Marketing Insights
- 3 Connecting with Customers
- 4 Building Strong Brands
- 5 Creating Value
- 6 Delivering Value
- 7 Communicating Value
- 8 Conducting Marketing Responsibly for Long-term Success

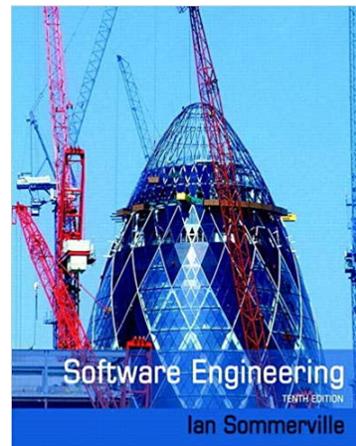
# Software Engineering

# Software Engineering and Project Management



# Software Engineering

- **Software engineering** is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.



# What is software?

- **Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.**

# What are the attributes of good software?

- **Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.**

# What is software engineering?

- **Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance.**

# What are the fundamental software engineering activities?

- Software **specification**, software **development**, software **validation** and software **evolution**.

# What is the difference between software engineering and computer science?

- **Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.**

# What are the best software engineering techniques and methods?

- **While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system.**
- **For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed.**
- **There are no methods and techniques that are good for everything.**

# What are the costs of software engineering?

- **Roughly 60% of software costs are development costs, 40% are testing costs.**
- **For custom software, evolution costs often exceed development costs.**

**Information Management**

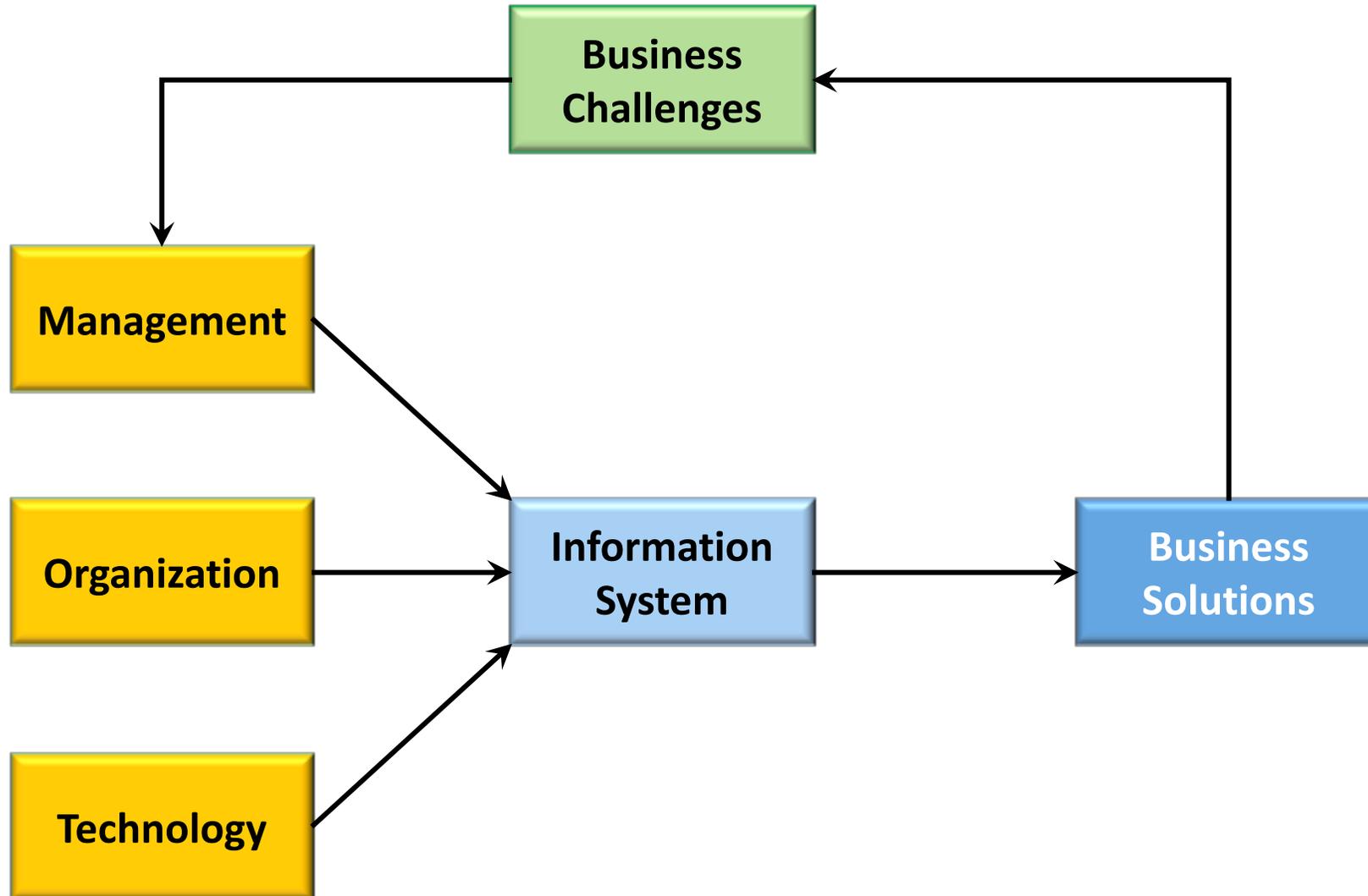
**Management  
Information Systems (MIS)**

**Information Systems**

# Information Management (MIS) Information Systems



# Fundamental MIS Concepts



# Software products

- **Software products** are **generic software systems** that provide functionality that is useful to a range of customers.
- **Software products:**
  - Large-scale business systems (e.g. MS Excel)
  - Personal products (e.g. Evernote)
  - Simple mobile phone apps and games (e.g. Suduko).

# Software product engineering

- **Software product** engineering methods and techniques have evolved from software engineering techniques that support the development of one-off, **custom software systems**.

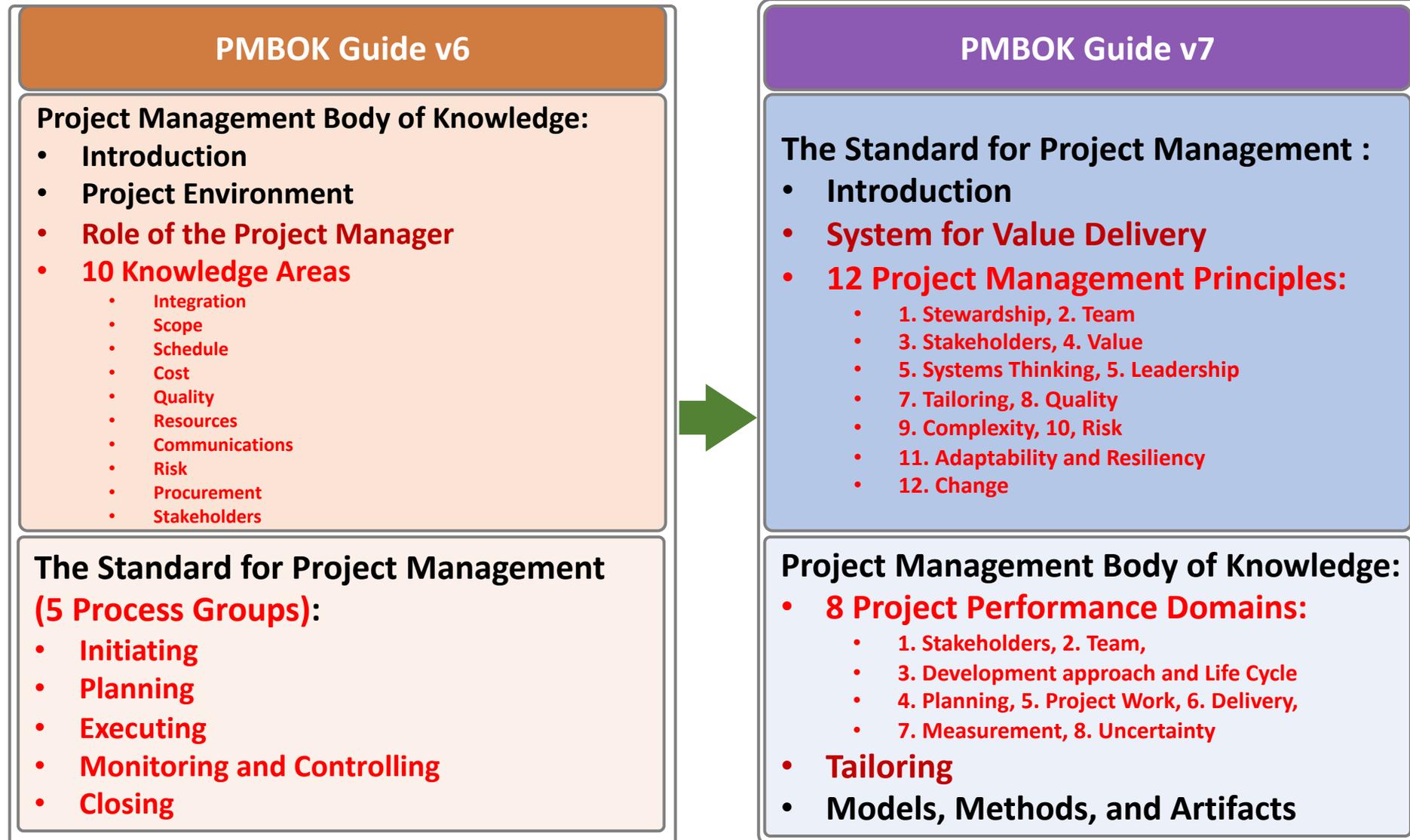
# Software projects

- **Custom software systems** are still important for large businesses, government and public bodies.
- They are developed in dedicated software projects.

# Project

- A **project** is a temporary endeavor undertaken to create a unique product, service, or result.

# Project Management Body of Knowledge (PMBOK Guide) PMBOK v6 vs. PMBOK v7



# Project Management Knowledge Areas

(PMBOK v6)

1. Project **Integration** Management
2. Project **Scope** Management
3. Project **Schedule** Management
4. Project **Cost** Management
5. Project **Quality** Management
6. Project **Resource** Management
7. Project **Communications** Management
8. Project **Risk** Management
9. Project **Procurement** Management
10. Project **Stakeholder** Management

# Project Management Process Groups

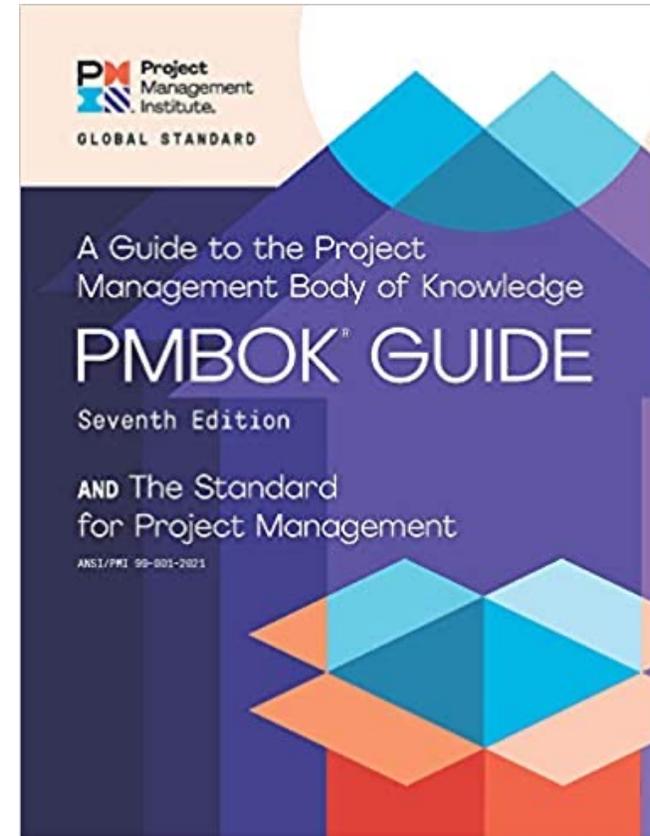
(PMBOK v6)

1. **Initiating Process Group**
2. **Planning Process Group**
3. **Executing Process Group**
4. **Monitoring and Controlling Process Group**
5. **Closing Process Group**

# Project Management 12 Principles

## (PMBOK v7)

1. Stewardship
2. Team
3. Stakeholders
4. Value
5. Systems Thinking
6. Leadership
7. Tailoring
8. Quality
9. Complexity
10. Risk
11. Adaptability and Resiliency
12. Change

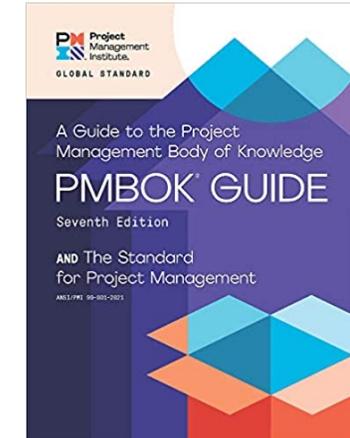


# Project Management

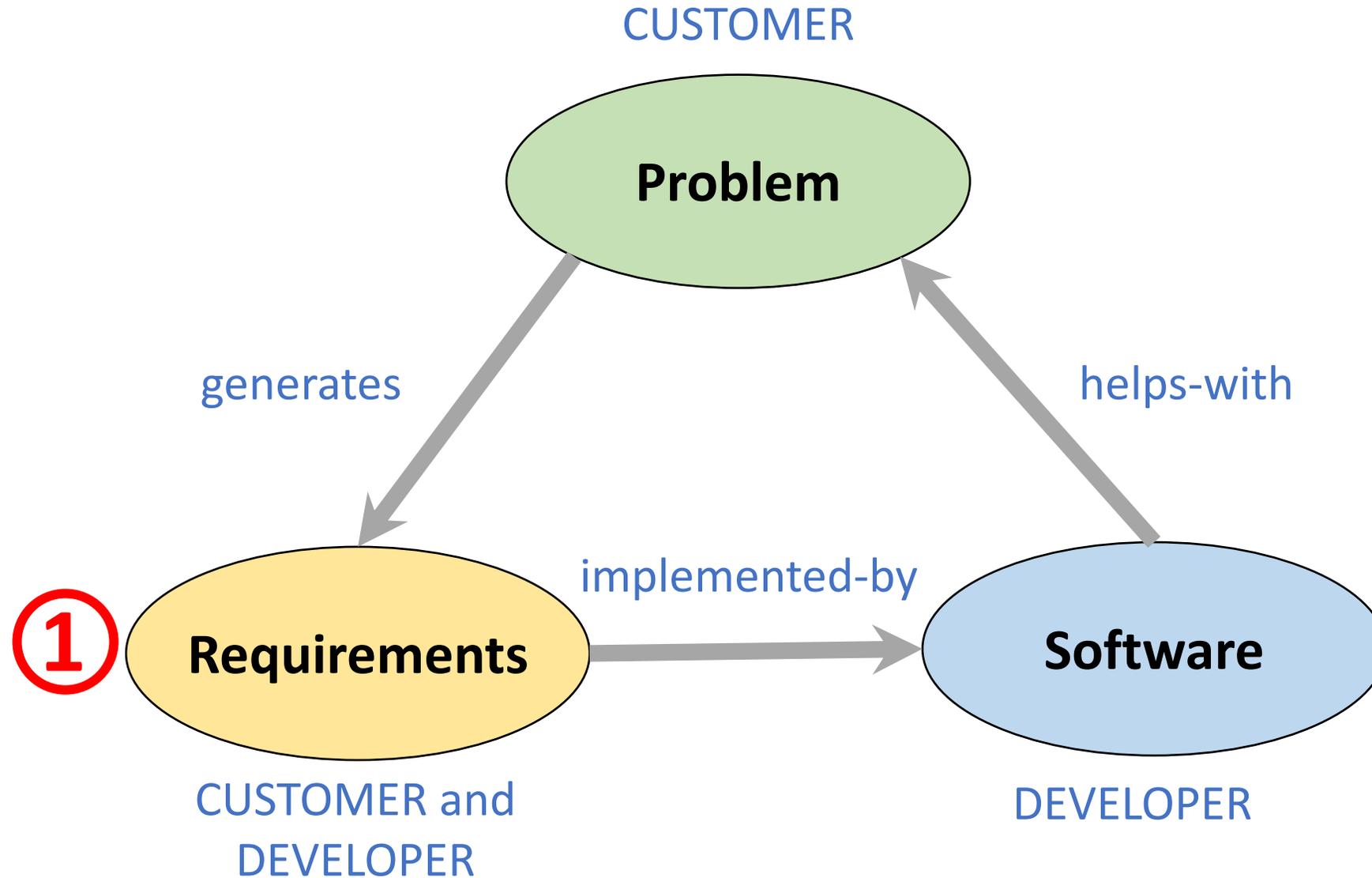
## 8 Project Performance Domains

(PMBOK v7)

1. Stakeholders
2. Team
3. Development Approach and Life Cycle
4. Planning
5. Project Work
6. Delivery
7. Measurement
8. Uncertainty



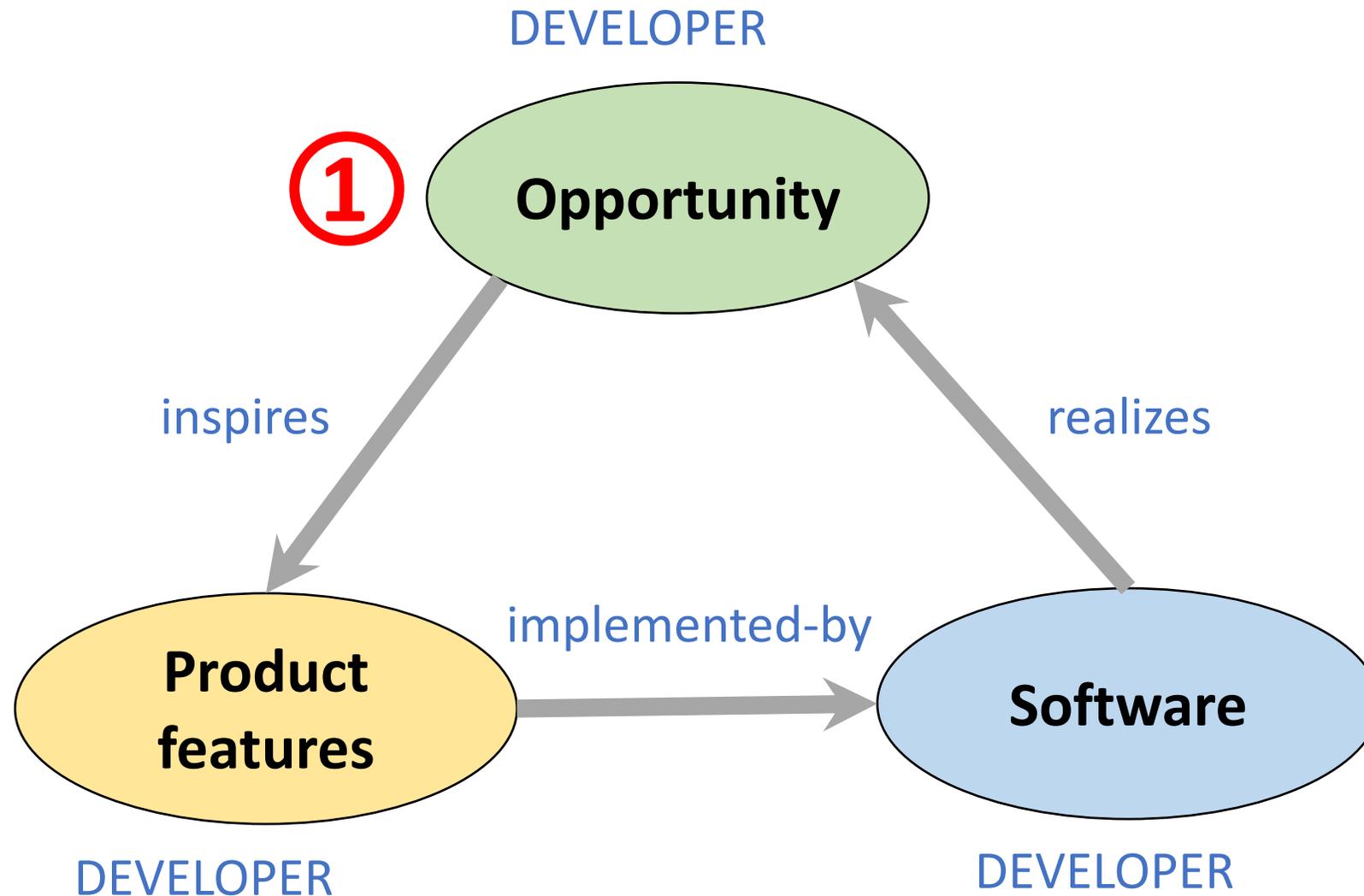
# Project-based software engineering



# Project-based software engineering

- The starting point for the software development is a set of **'software requirements'** that are owned by an external client and which set out what they want a software system to do to support their business processes.
- The software is developed by a software company (the contractor) who **design and implement a system** that delivers functionality to meet the requirements.
- The customer may change the requirements at any time in response to business changes (they usually do). The contractor must change the software to reflect these requirements changes.
- Custom software usually has a long-lifetime (10 years or more) and it must be supported over that lifetime.

# Product software engineering



# Product software engineering

- The starting point for product development is a **business opportunity** that is identified by individuals or a company.  
They develop a software product to take advantage of this opportunity and sell this to customers.
- The company who identified the opportunity **design and implement a set of software features** that realize the opportunity and that will be useful to customers.
- The software development company are responsible for deciding on the development timescale, what features to include and when the product should change.
- Rapid delivery of software products is essential to capture the market for that type of product.

# Software product line

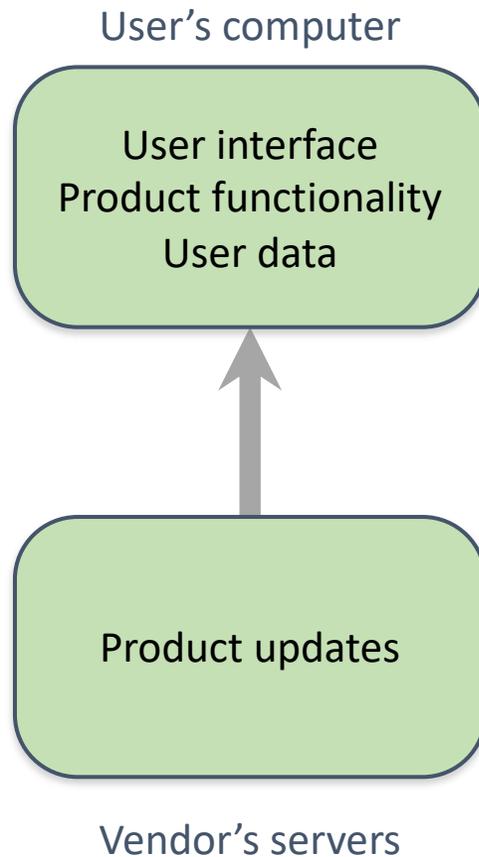
- **A set of software products that share a common core.**
- **Each member of the product line includes customer-specific adaptations and additions.**
- **Software product lines may be used to implement a custom system for a customer with specific needs that can't be met by a generic product.**

# Platform

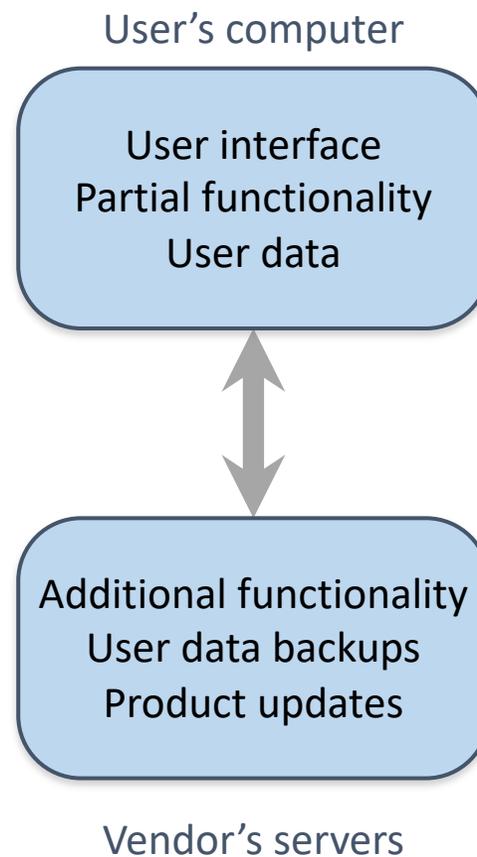
- **A software (or software + hardware) product that includes functionality so that new applications can be built on it.**
- **An example of a platform that you probably use is Facebook.**
- **It provides an extensive set of product functionality but also provides support for creating ‘Facebook apps’.**
- **These add new features that may be used by a business or a Facebook interest group.**

# Software execution models

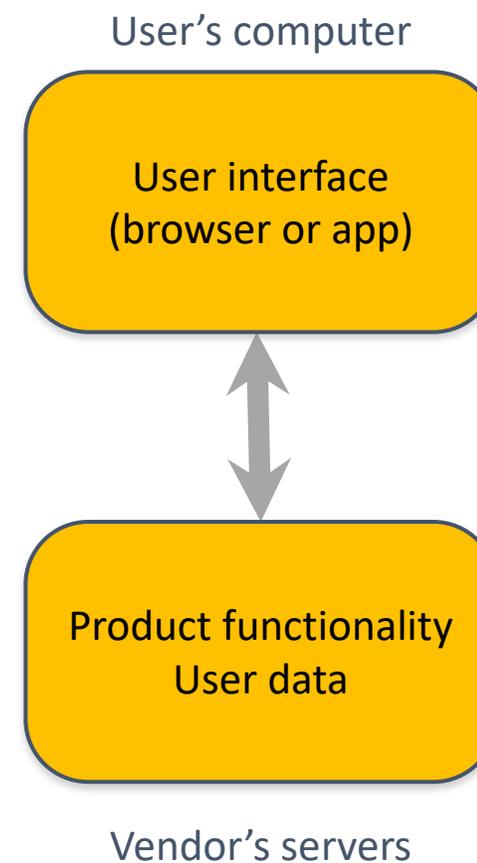
## Stand-alone execution



## Hybrid execution



## Software as a service



# Software execution models

- **Stand-alone**
  - The software executes entirely on the customer's computers.
- **Hybrid**
  - Part of the software's functionality is implemented on the customer's computer but some features are implemented on the product developer's servers.
- **Software service**
  - All of the product's features are implemented on the developer's servers and the customer accesses these through a browser or a mobile app.

# Comparable software development

- The key feature of product development is that there is no external customer that generates requirements and pays for the software.
- **Student projects**
  - Individuals or student groups develop software as part of their course. Given an assignment, they decide what features to include in the software.
- **Research software**
  - Researchers develop software to help them answer questions that are relevant to their research.
- **Internal tool development**

# The product vision

- The starting point for software product development is a **'product vision'**.
- **Product visions** are simple statements that define the **essence of the product** to be developed.
- The product vision should answer three fundamental questions:
  - **What** is the product to be developed?
  - **Who** are the target customers and users?
  - **Why** should customers buy this product?

# Moore's vision template

- **FOR** (target customer)
- **WHO** (statement of the need or opportunity)
- **The (PRODUCT NAME) is a** (product category)
- **THAT** (key benefit, compelling reason to buy)
- **UNLIKE** (primary competitive alternative)
- **OUR PRODUCT** (statement of primary differentiation)

# Vision template example

- **“FOR** a mid-sized company's marketing and sales departments  
**WHO** need basic CRM functionality,  
**THE** CRM-Innovator **is** a Web-based service  
**THAT** provides sales tracking, lead generation, and sales representative support features that improve customer relationships at critical touch points.  
**UNLIKE** other services or package software products,  
**OUR** product provides very capable services at a moderate cost.”

# Information sources for developing a product vision

- **Domain experience**
- **Product experience**
- **Customer experience**
- **Prototyping and playing around**

# Information sources for developing a product vision

- **Domain experience**

- The product developers may work in a particular area (say marketing and sales) and understand the software support that they need.
- They may be frustrated by the deficiencies in the software they use and see opportunities for an improved system.

# Information sources for developing a product vision

- **Product experience**

- Users of existing software (such as word processing software) may see simpler and better ways of providing comparable functionality and propose a new system that implements this.
- New products can take advantage of recent technological developments such as speech interfaces.

# Information sources for developing a product vision

- **Customer experience**

- The software developers may have extensive discussions with prospective customers of the product to understand the problems that they face, constraints, such as interoperability, that limit their flexibility to buy new software, and the critical attributes of the software that they need.

# Information sources for developing a product vision

- **Prototyping and playing around**

- Developers may have an idea for software but need to develop a better understanding of that idea and what might be involved in developing it into a product.
- They may develop a prototype system as an experiment and ‘play around’ with ideas and variations using that prototype system as a platform.

# A vision statement for the iLearn system

- FOR teachers and educators WHO need a way to help students use web-based learning resources and applications, THE iLearn system is an open learning environment THAT allows the set of resources used by classes and students to be easily configured for these students and classes by teachers themselves. UNLIKE Virtual Learning Environments, such as Moodle, the focus of iLearn is the learning process rather than the administration and management of materials, assessments and coursework. OUR product enables teachers to create subject and age-specific environments for their students using any web-based resources, such as videos, simulations and written materials that are appropriate.
- Schools and universities are the target customers for the iLearn system as it will significantly improve the learning experience of students at relatively low cost. It will collect and process learner analytics that will reduce the costs of progress tracking and reporting.

# The Essence of Strategic Marketing (STP)

**S**egmentation

**T**argeting

**P**ositioning

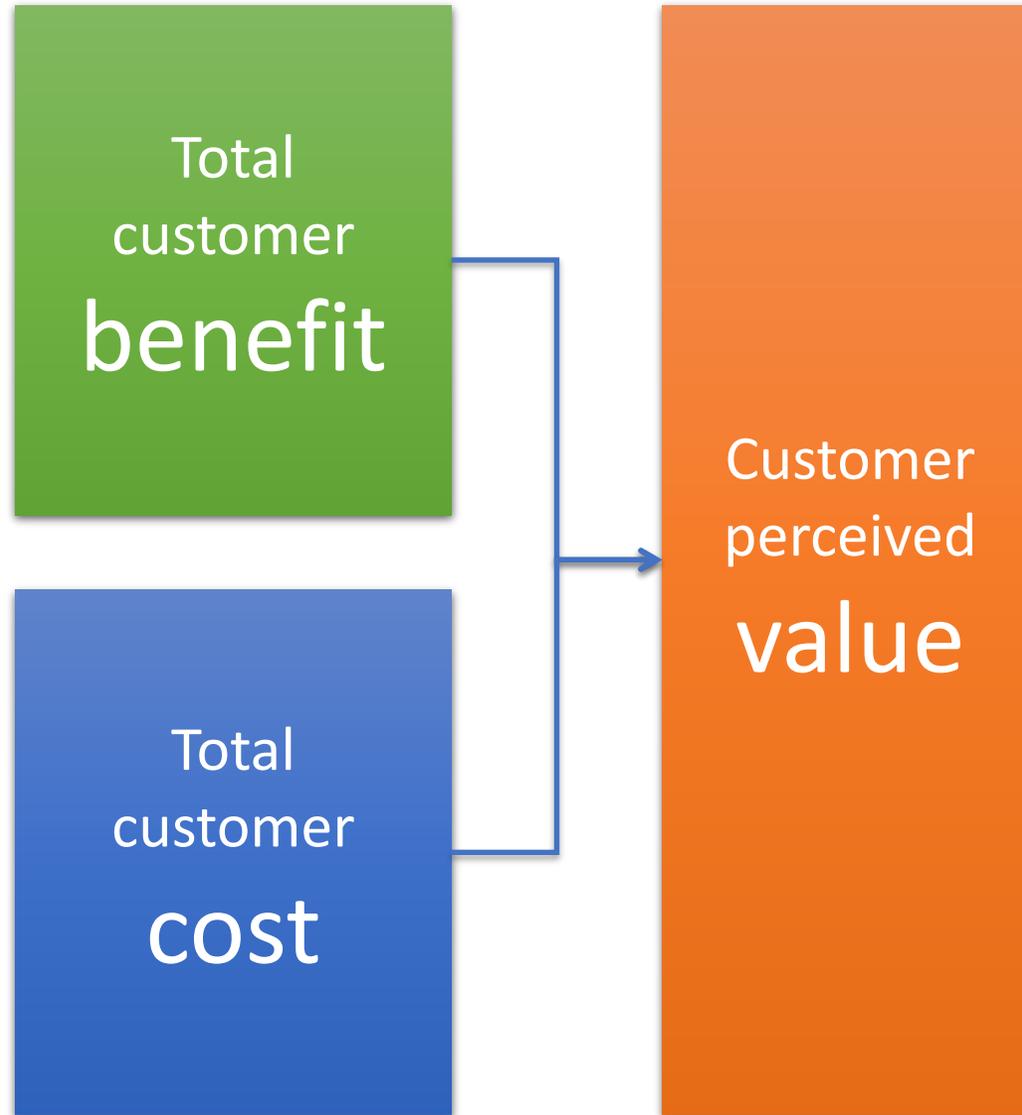
# Customer Value

# Value

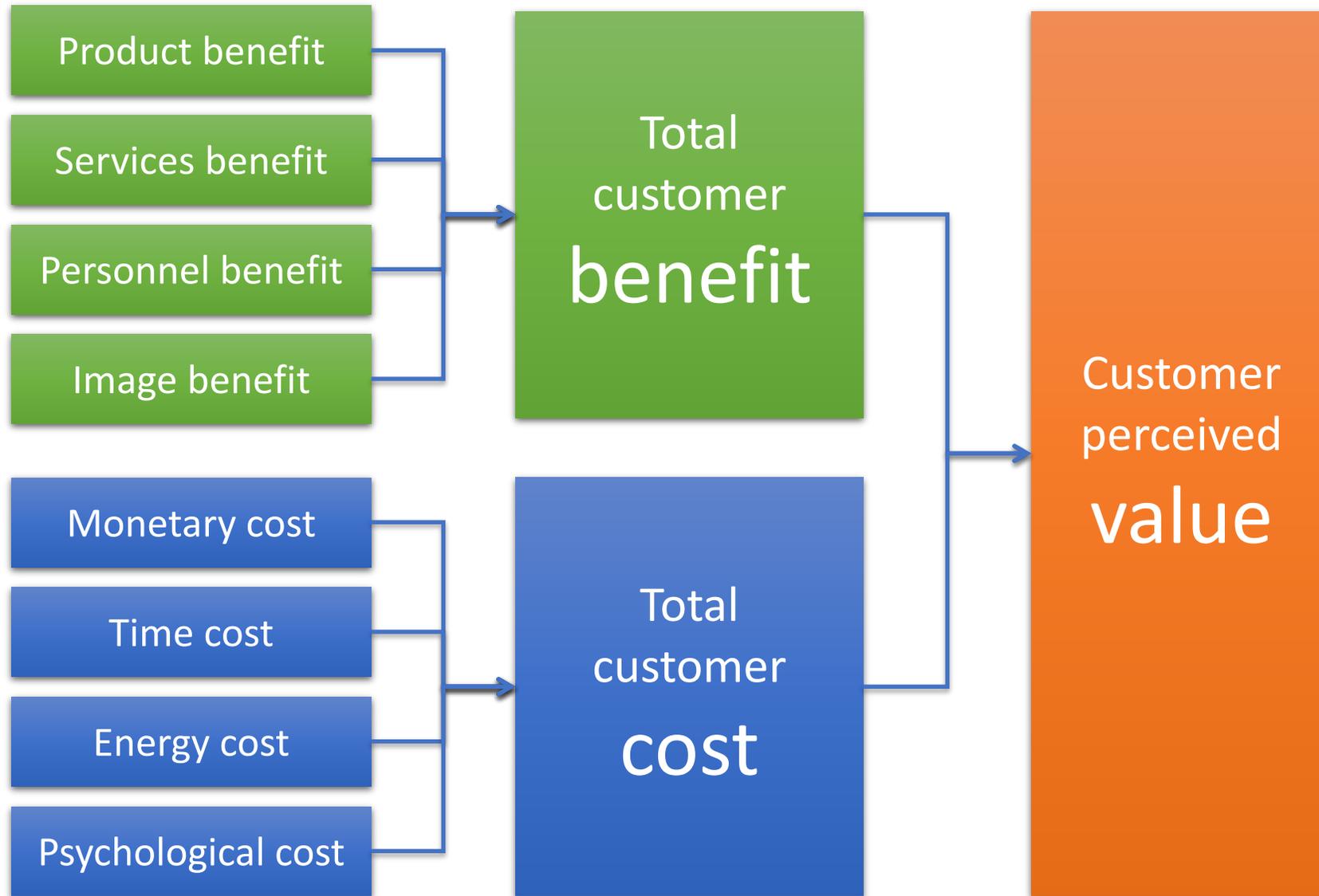
the sum of the  
tangible and  
intangible

benefits and costs

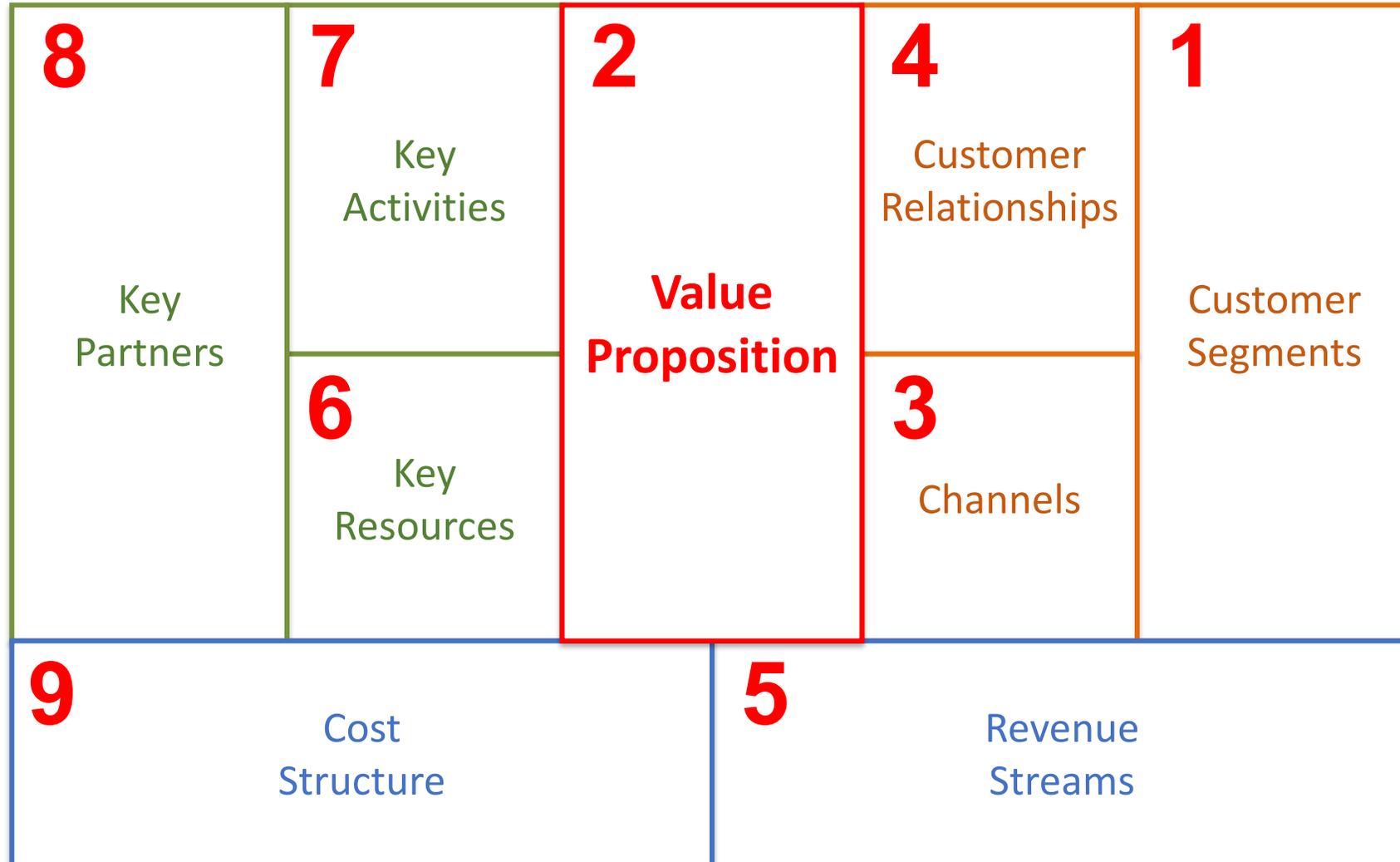
# Value



# Customer Perceived Value



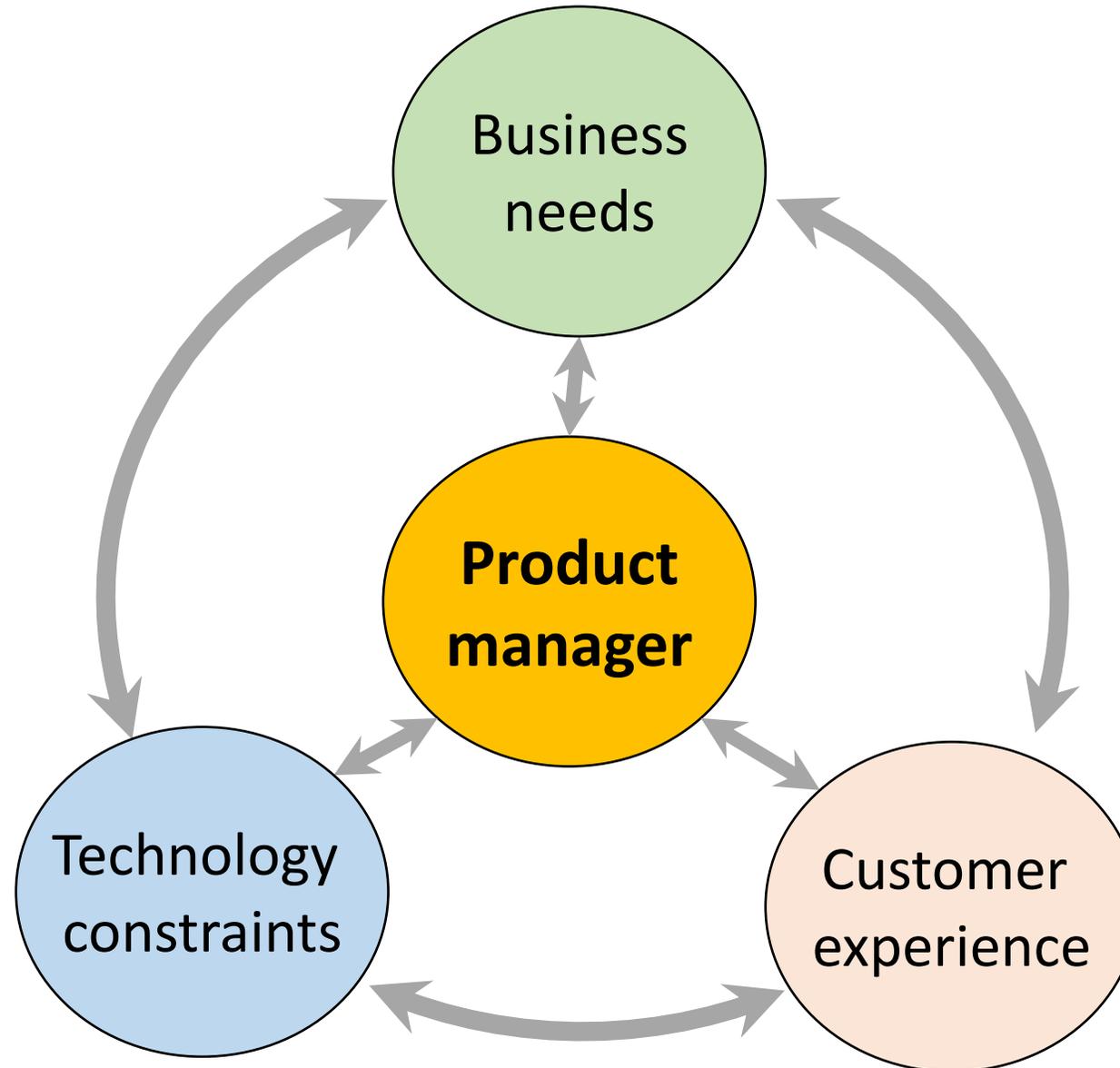
# Business Model



# Software product management

- **Software product management** is a business activity that focuses on the software products developed and sold by the business.
- **Product managers (PMs)** take overall responsibility for the product and are involved in planning, development and product marketing.
- Product managers are the interface between the organization, its customers and the software development team. They are involved at all stages of a product's lifetime from initial conception through to withdrawal of the product from the market.
- Product managers must look outward to customers and potential customers rather than focus on the software being developed

# Product management concerns



# Product management concerns

- **Business needs**

- PMs have to ensure that the software being developed meets the business goals of the software development company.

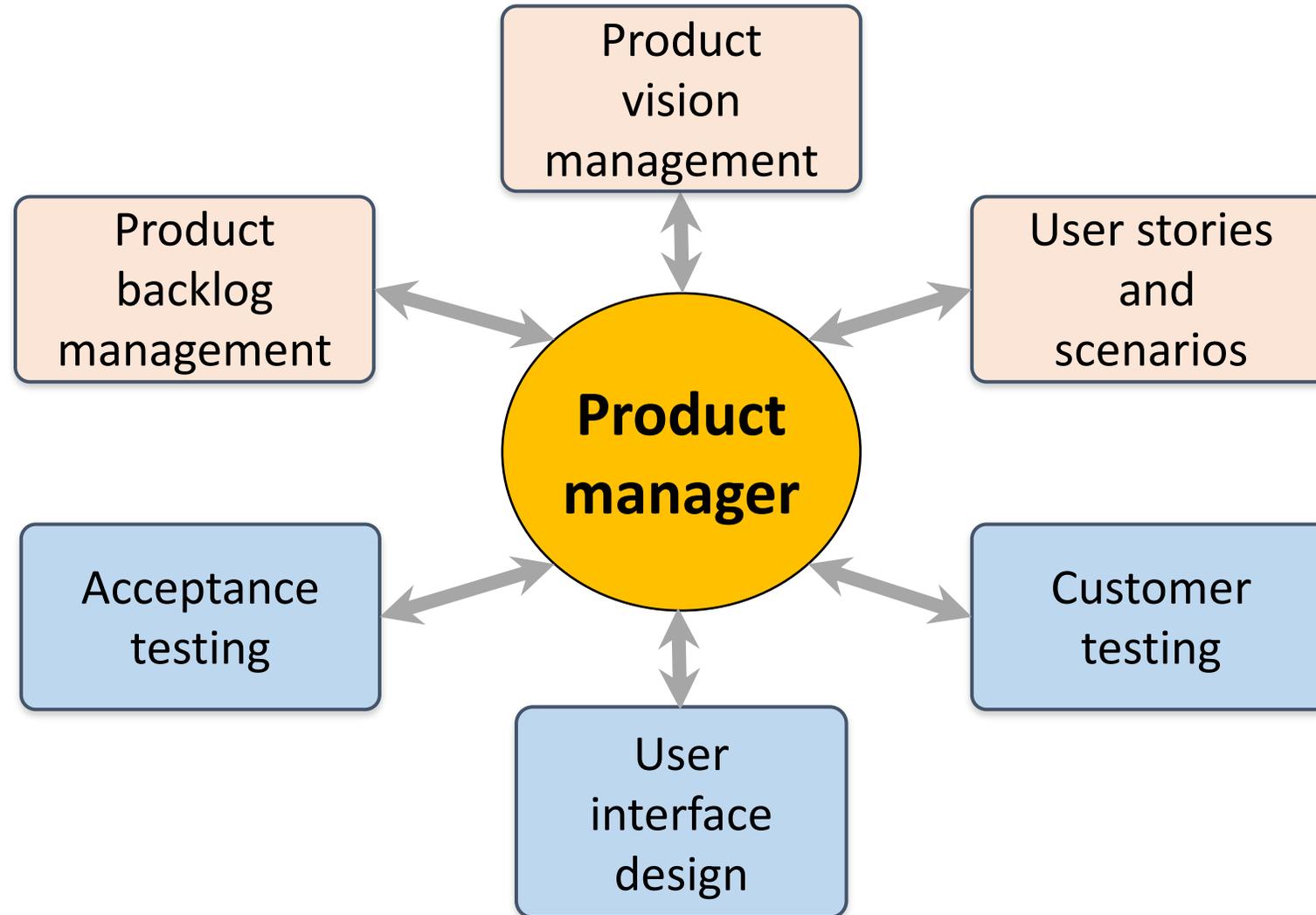
- **Technology constraints**

- PMs must make developers aware of technology issues that are important to customers.

- **Customer experience**

- PMs should be in regular contact with customers and potential customers to understand what they are looking for in a product, the types of users and their backgrounds and the ways that the product may be used.

# Technical interactions of product managers



# Technical interactions of product managers

- **Product vision management**
  - **The product manager may be responsible for helping with the development of the product vision.**
  - **The should always be responsible for managing the vision, which involves assessing and evaluating proposed changes against the product vision.**
  - **They should ensure that there is no ‘vision drift’**

# Technical interactions of product managers

- **Product roadmap** development
  - A product roadmap is a plan for the development, release and marketing of the software.
  - The PM should lead roadmap development and should be the ultimate authority in deciding if changes to the roadmap should be made.

# Technical interactions of product managers

- **User story and scenario** development
  - User stories and scenarios are used to refine a product vision and identify product features.
  - Based on his or her knowledge of customers, the PM should lead the development of stories and scenarios.

# Technical interactions of product managers

- **Product backlog** creation and management
  - The product backlog is a prioritized 'to-do' list of what has to be developed.
  - PMs should be involved in creating and refining the backlog and deciding on the priority of product features to be developed.

# Technical interactions of product managers

- **Acceptance testing**

- **Acceptance testing is the process of verifying that a software release meets the goals set out in the product roadmap and that the product is efficient and reliable.**
- **The PM should be involved in developing tests of the product features that reflect how customers use the product.**

# Technical interactions of product managers

- **Customer testing**

- **Customer testing involves taking a release of a product to customers and getting feedback on the product's features, usability and business.**
- **PMs are involved in selecting customers to be involved in the customer testing process and working with them during that process.**

# Technical interactions of product managers

- **User interface design**

- **Product managers should understand user limitations and act as surrogate users in their interactions with the development team.**
- **They should evaluate user interface features as they are developed to check that these features are not unnecessarily complex or force users to work in an unnatural way.**

# Product prototyping

- **Product prototyping is the process of developing an early version of a product to test your ideas and to convince yourself and company funders that your product has real market potential.**

# Product prototyping

- **You may be able to write an inspiring product vision, but your potential users can only really relate to your product when they see a working version of your software.**
- **They can point out what they like and don't like about it and make suggestions for new features.**
- **A prototype may be also used to help identify fundamental software components or services and to test technology.**

# Product prototyping

- **Building a prototype should be the first thing that you do when developing a software product.**
- **Your aim should be to have a working version of your software that can be used to demonstrate its key features.**
- **You should always plan to throw-away the prototype after development and to re-implement the software, taking account of issues such as security and reliability.**

# Two-stage prototyping

## 1. Feasibility demonstration

- You create an executable system that demonstrates the new ideas in your product.
- The aims at this stage are to see if your ideas actually work and to show funders and/or company management the original product features that are better than those in competing products.

## 2. Customer demonstration

# Two-stage prototyping

## 1. Feasibility demonstration

## 2. Customer demonstration

- You take an existing prototype created to demonstrate feasibility and extend this with your ideas for specific customer features and how these can be realized.
- Before you develop this type of prototype, you need to do some user studies and have a clearer idea of your potential users and scenarios of use.

# Software process models

- **The waterfall model**

- This takes the fundamental process activities of **specification, development, validation, and evolution** and represents them as separate process phases such as **requirements specification, software design, implementation, and testing**.

- **Incremental development**

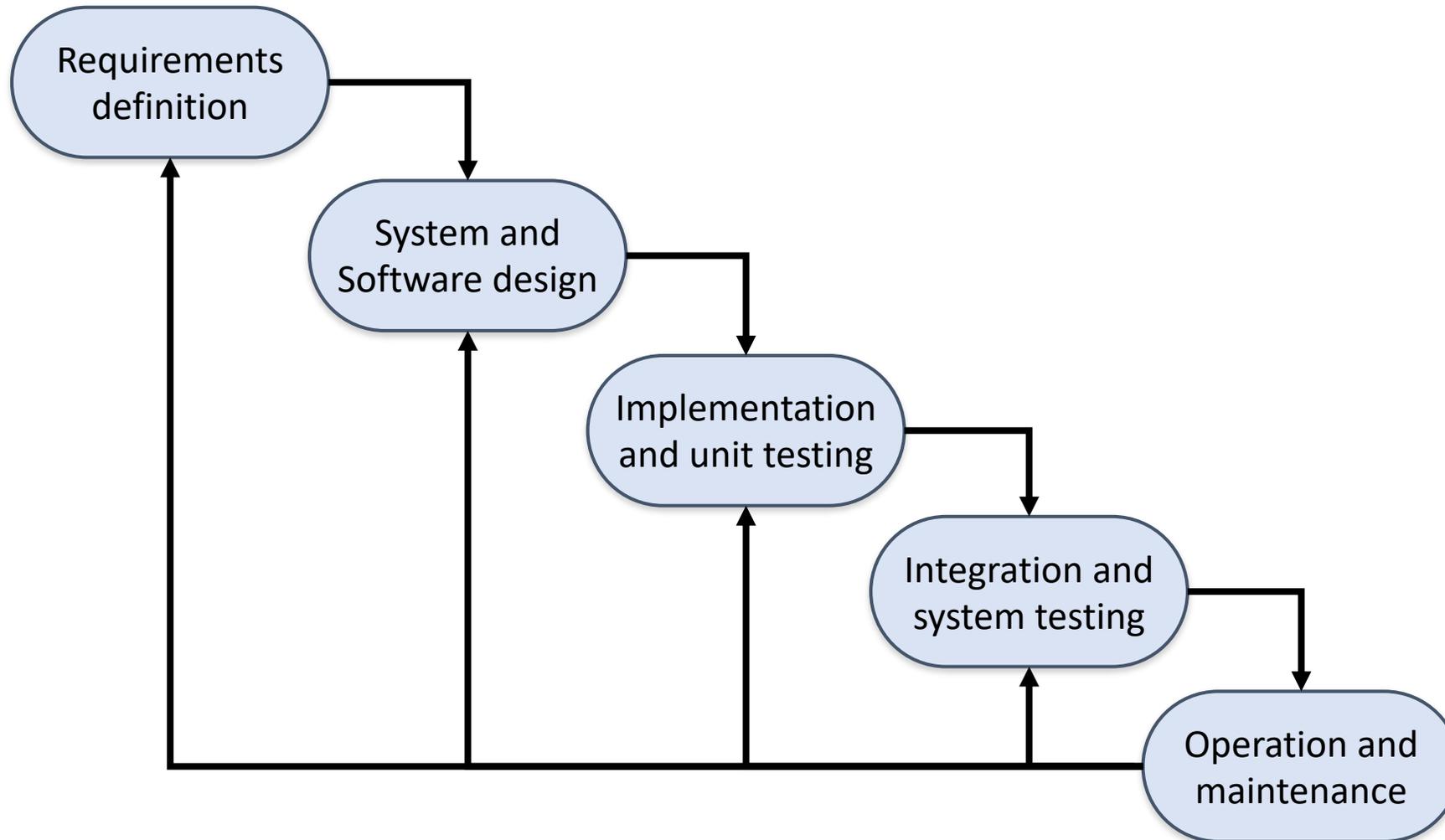
- This approach interleaves the activities of **specification, development, and validation**. The system is developed as a series of **versions (increments)**, with each version adding functionality to the previous version.

- **Integration and configuration**

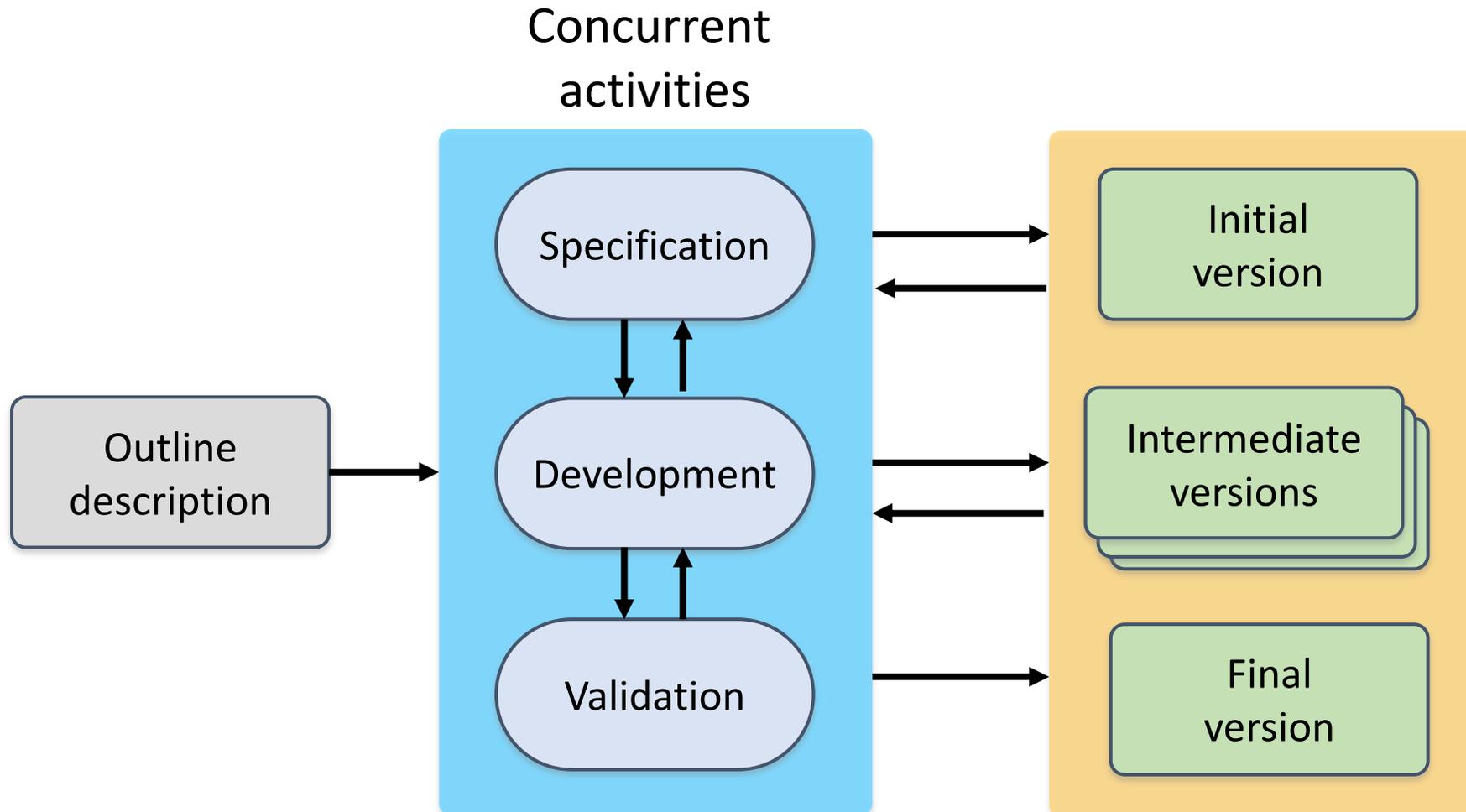
- This approach relies on the availability of **reusable components** or systems. The system development process focuses on configuring these components for use in a new setting and integrating them into a system.

# Software Development Life Cycle (SDLC)

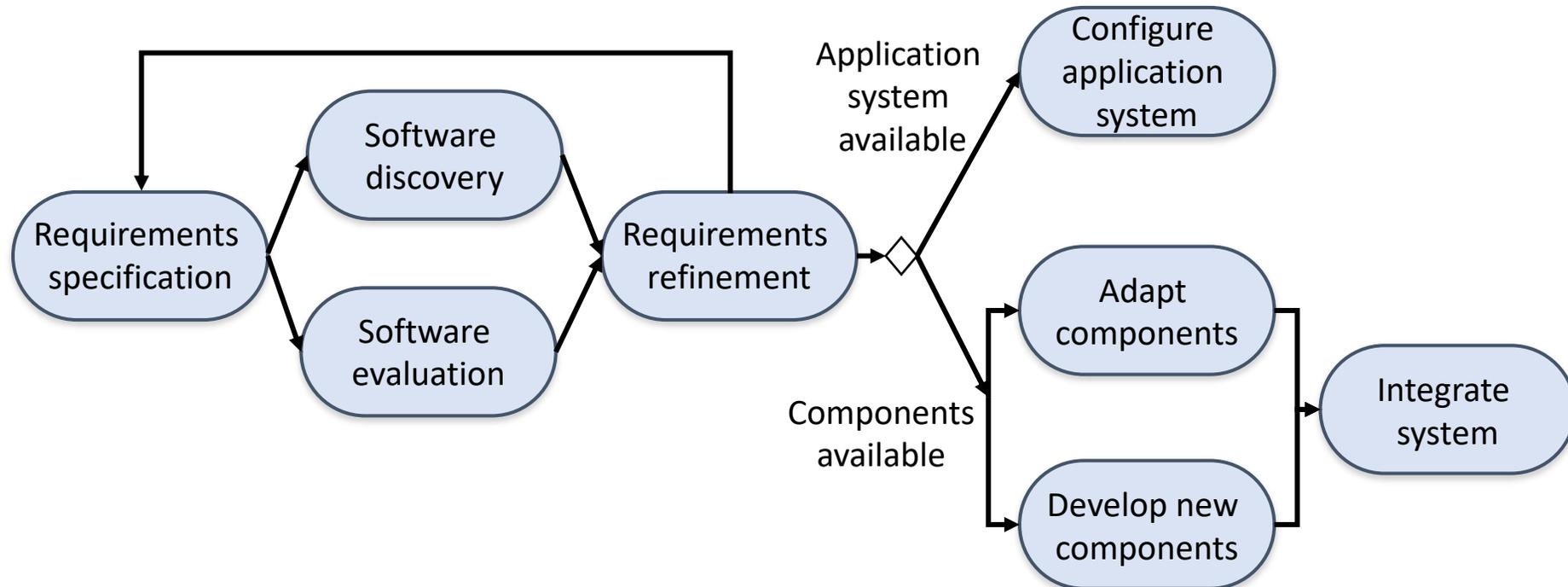
## The waterfall model



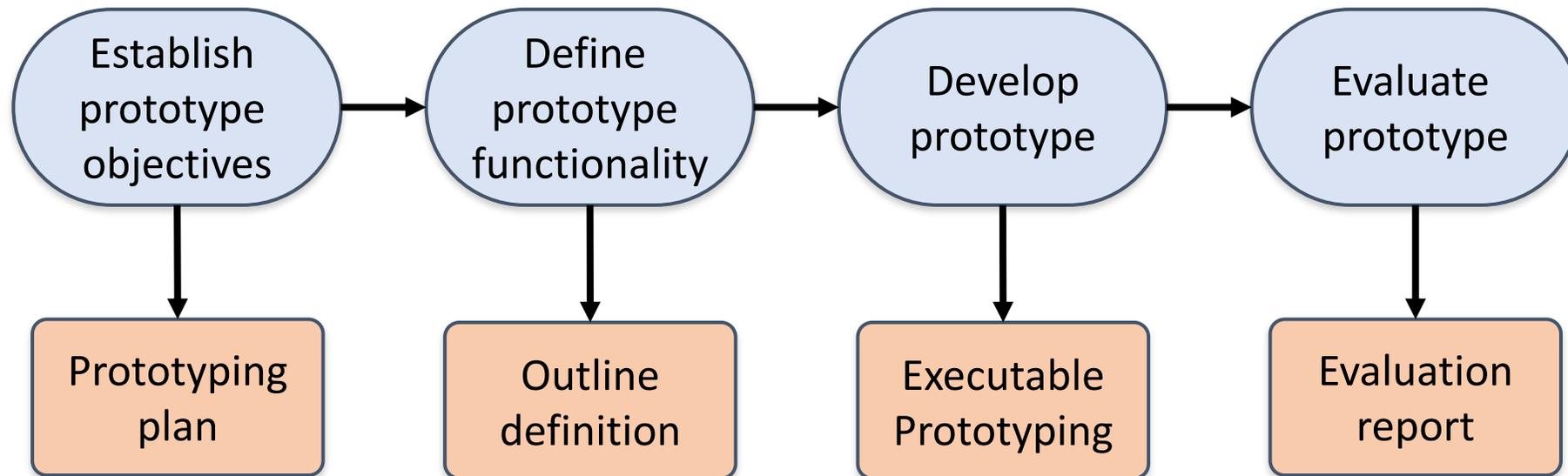
# Incremental development



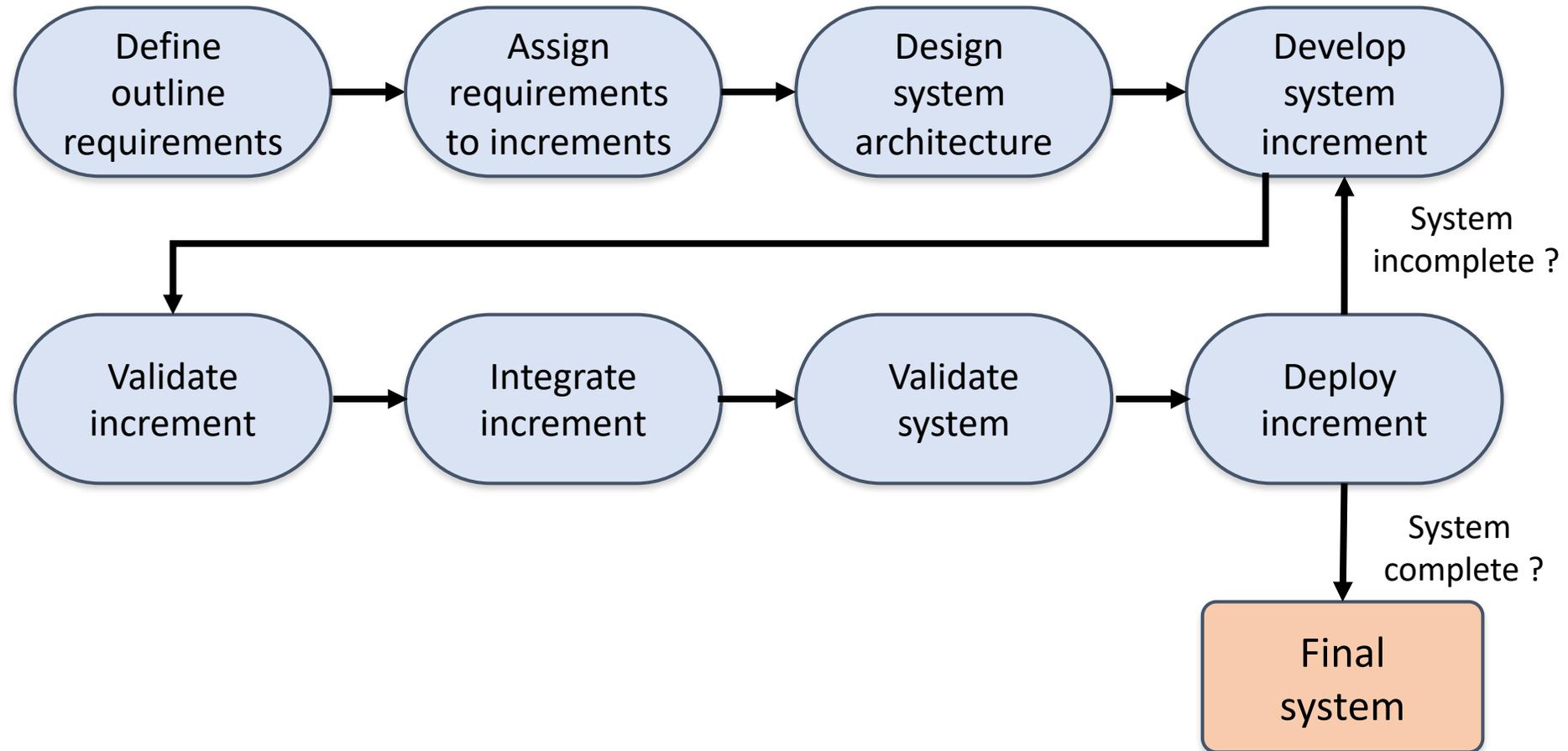
# Reuse-oriented software engineering



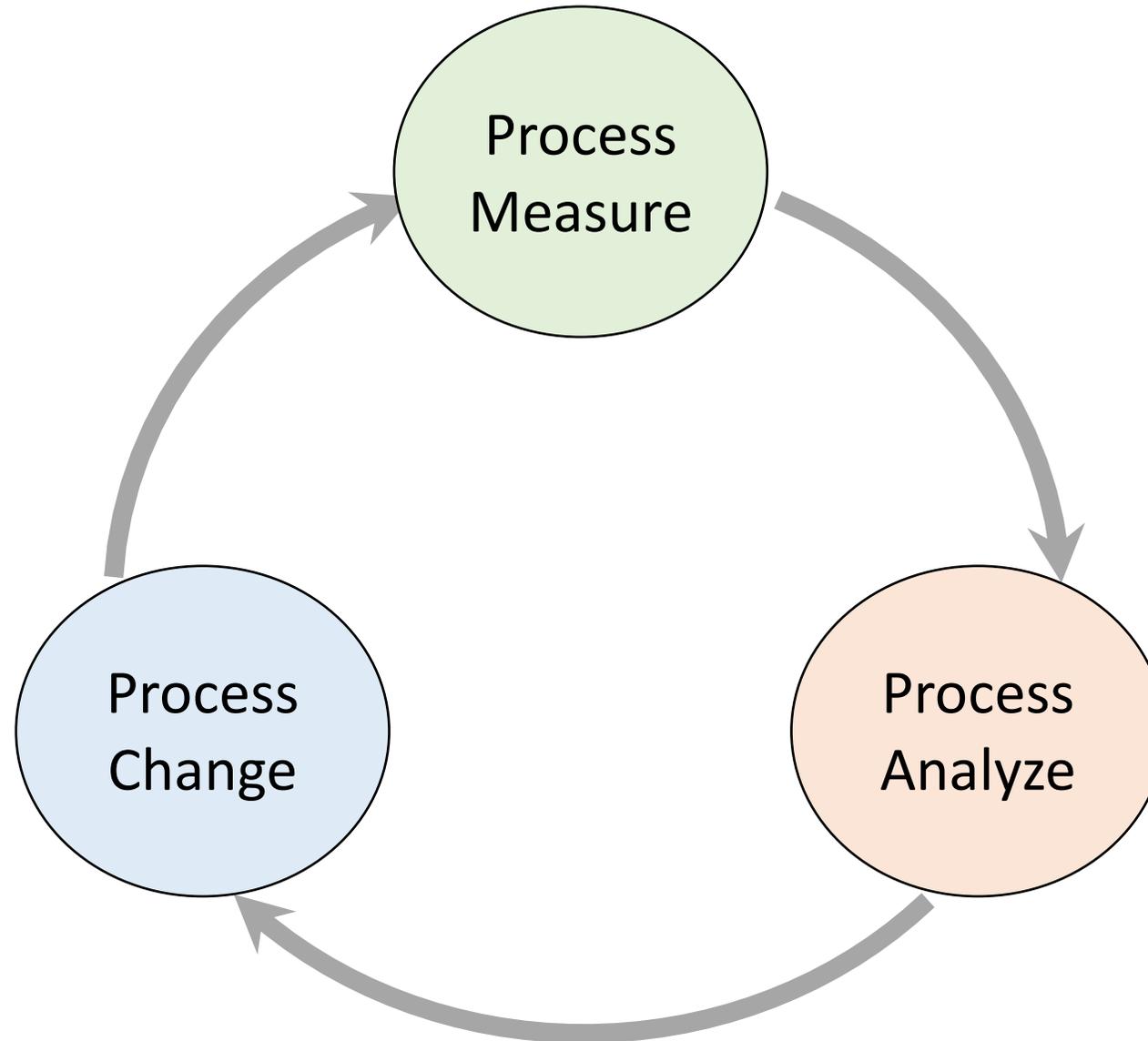
# Prototype development



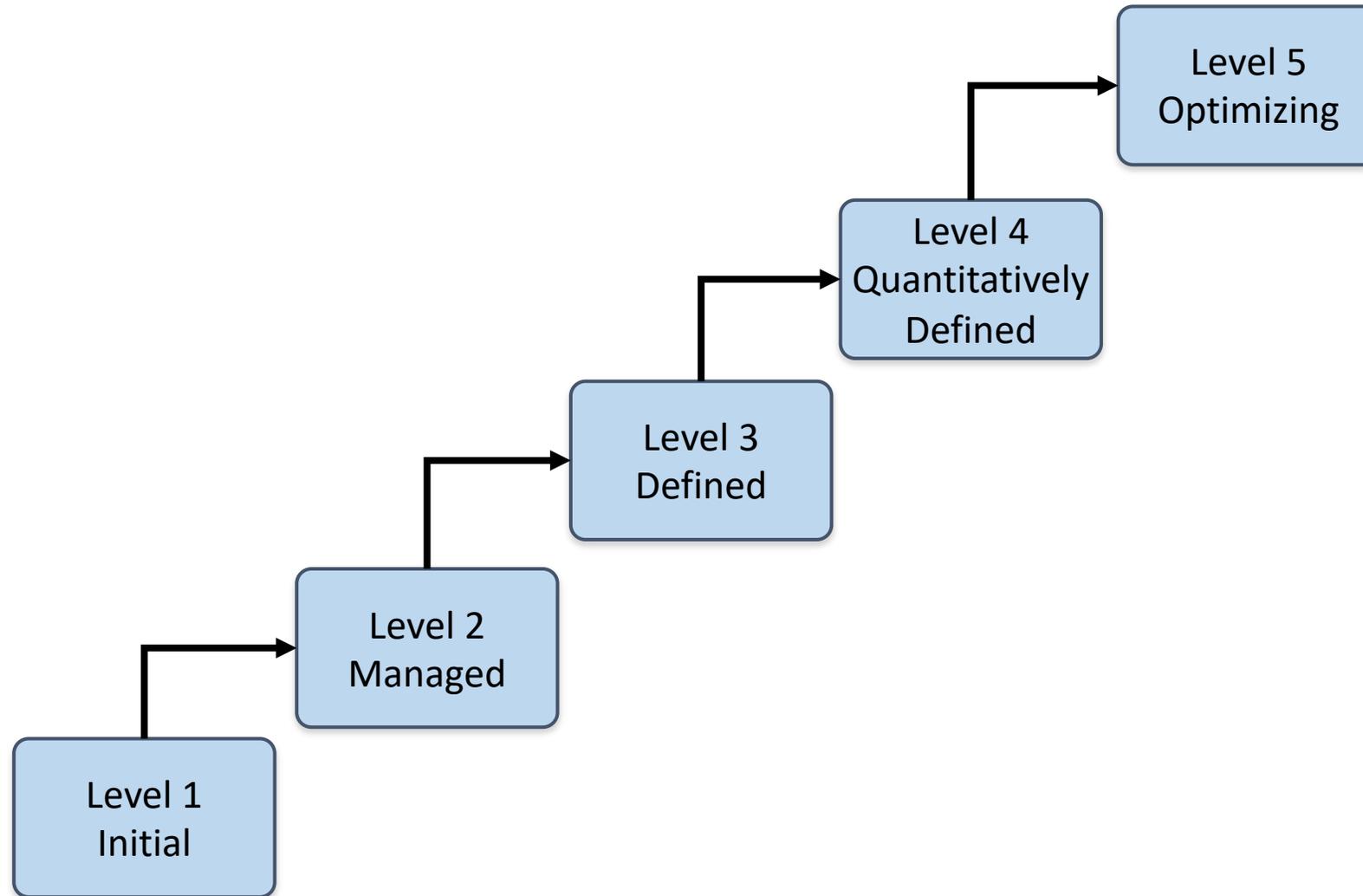
# Incremental delivery



# The process improvement model

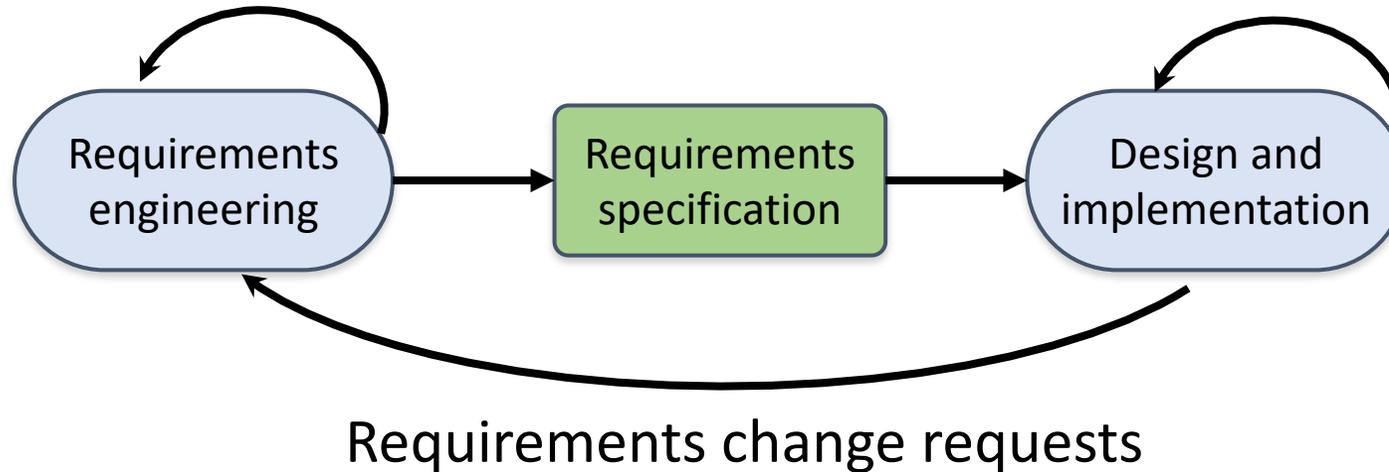


# Capability maturity levels

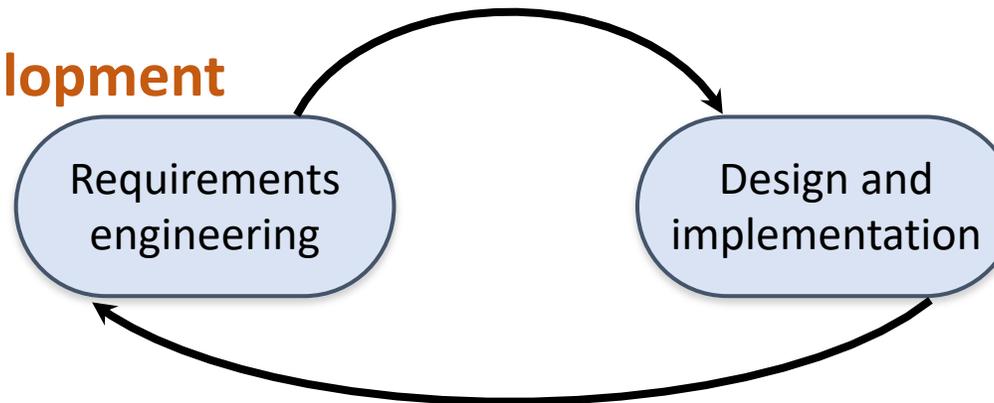


# Plan-based and Agile development

## Plan-based development

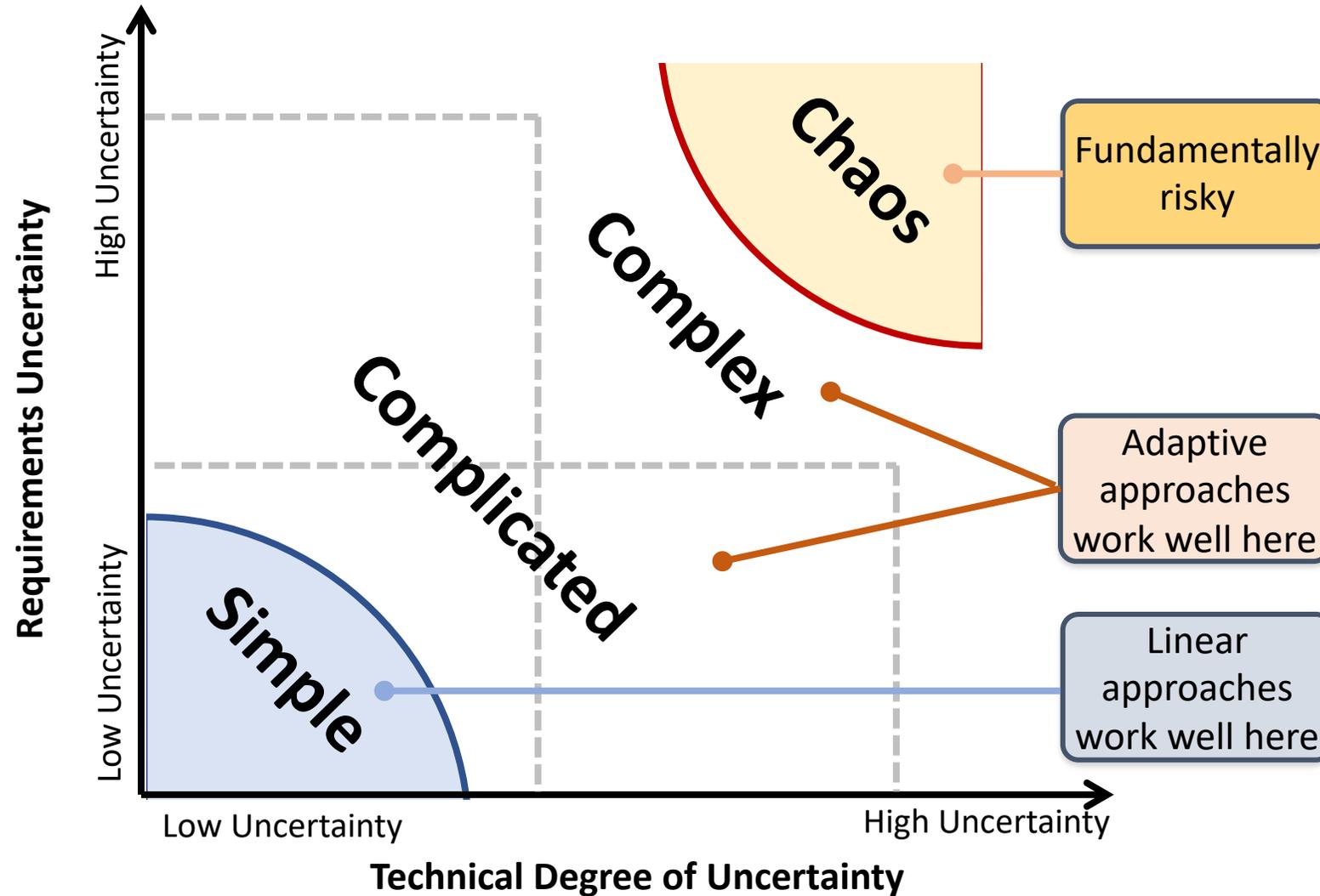


## Agile development



# Uncertainty and Complexity Model

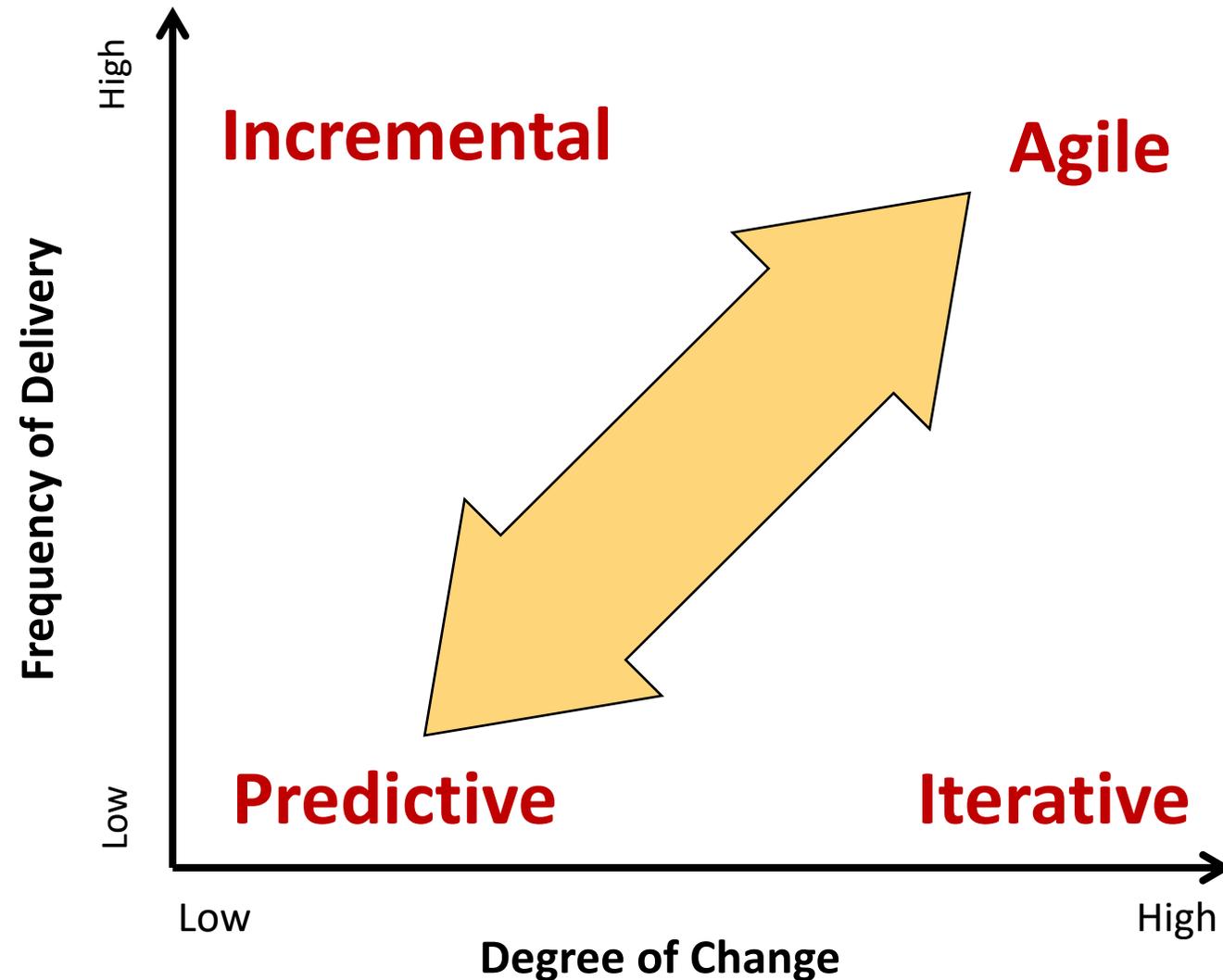
Inspired by the Stacey Complexity Model



# Characteristics of Four Categories of Life Cycles

| Approach           | Requirements | Activities                            | Delivery                    | Goal  |
|--------------------|--------------|---------------------------------------|-----------------------------|---|
| <b>Predictive</b>  | Fixed        | Performed once for the entire project | Single delivery             | Manage cost   |
| <b>Iterative</b>   | Dynamic      | Repeated until correct                | Single delivery             | Correctness of solution                             |
| <b>Incremental</b> | Dynamic      | Performed once for a given increment  | Frequent smaller deliveries | Speed   |
| <b>Agile</b>       | Dynamic      | Repeated until correct                | Frequent smaller deliveries | Customer value via frequent deliveries and feedback |

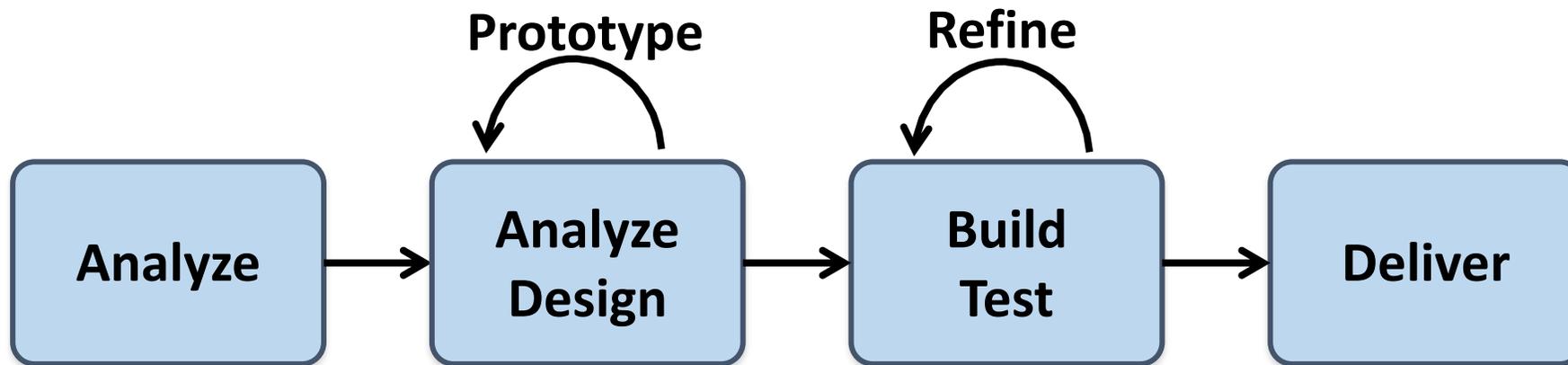
# The Continuum of Life Cycles



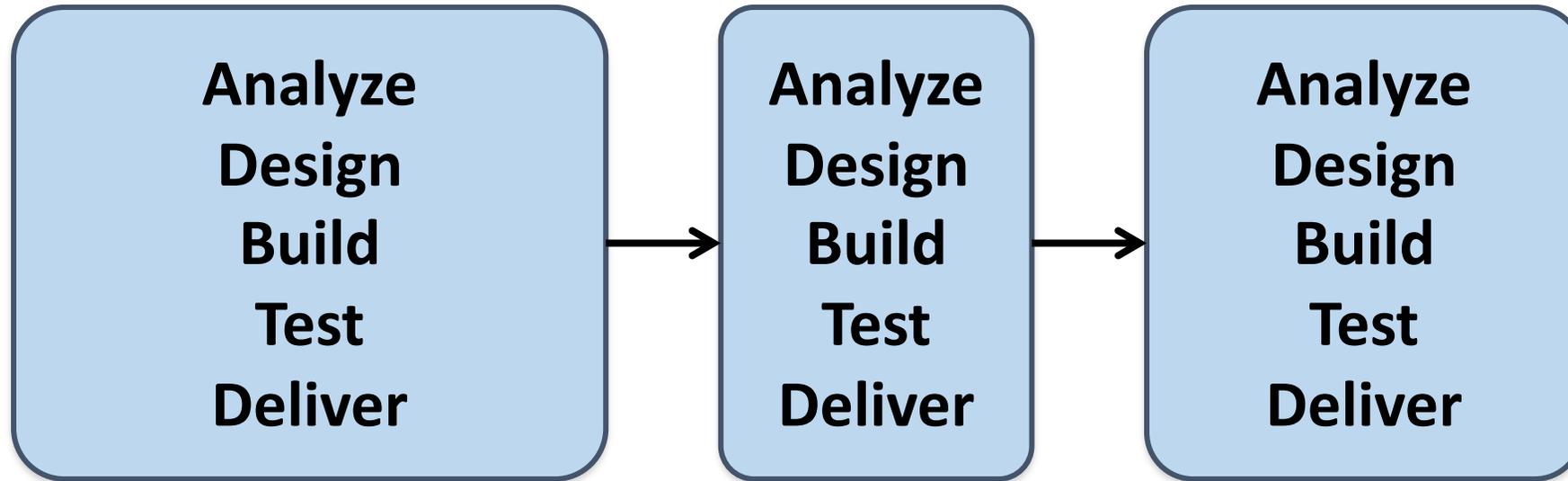
# Predictive Life Cycle



# Iterative Life Cycle

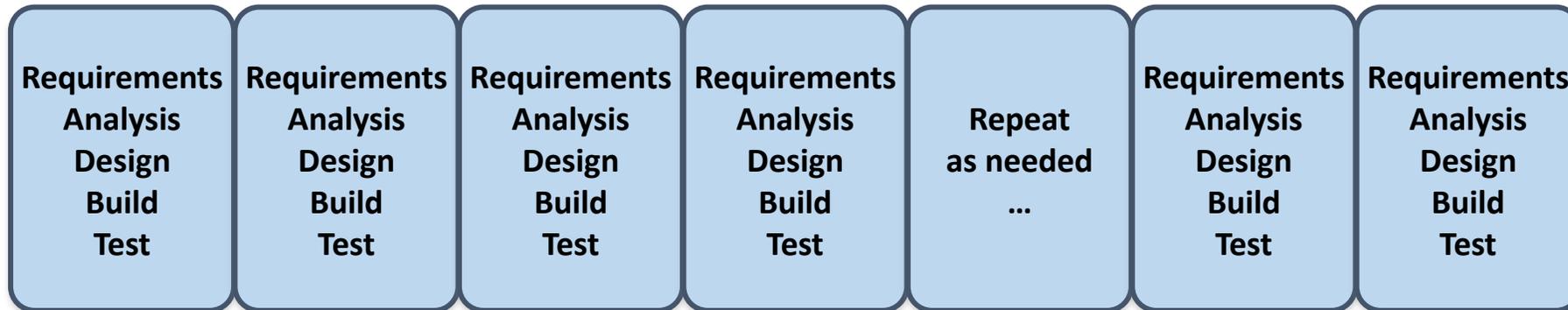


# A Life Cycle of Varying-Sized Increments

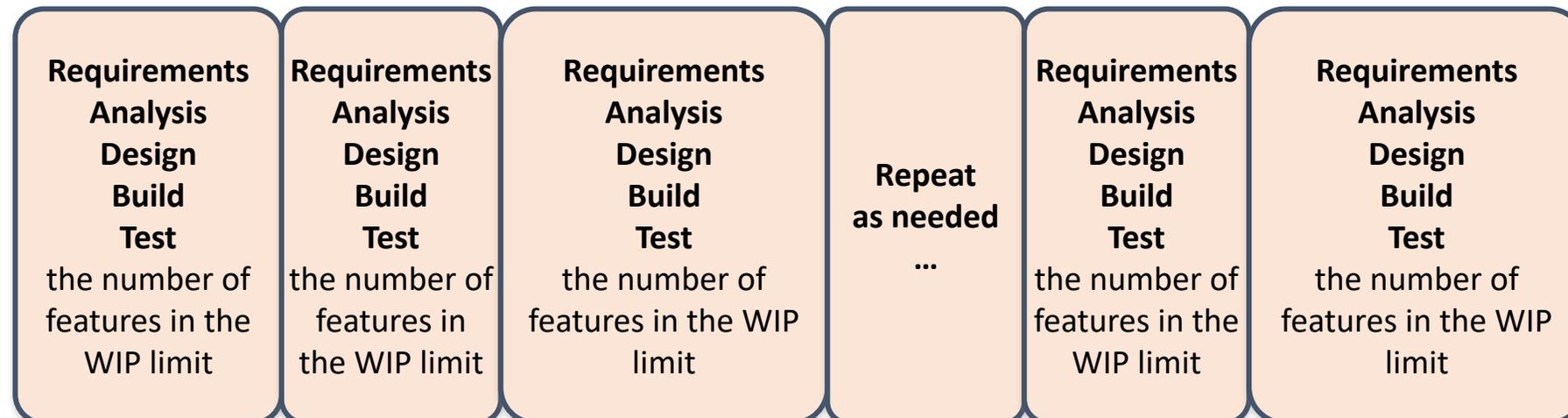


# Iteration-Based and Flow-Based Agile Life Cycles

## Iteration-Based Agile



## Flow-Based Agile



# Summary

- **Software products are software systems that include general functionality that is likely to be useful to a wide range of customers.**
- **In product software engineering, the same company is responsible for deciding on the features that should be part of the product and the implementation of these features.**

# Summary

- **Software products may be delivered as stand-alone systems running on the customer's computers, hybrid systems or service-based systems.**
- **In hybrid systems, some features are implemented locally and others are accessed over the Internet.**
- **All product features are remotely accessed in service-based products.**

# Summary

- **A product vision should succinctly describe what is to be developed, who are the target customers for the product and why they should buy the product that you are developing.**
- **Domain experience, product experience, customer experience and an experimental software prototype may all contribute to the development of the product vision.**

# Summary

- **Key responsibilities of product managers are product vision ownership, product roadmap development, creating user stories and the product backlog, customer and acceptance testing and user interface design.**
- **Product managers work at the interface between the business, the software development team and the product customers.**
- **They facilitate communications between these groups.**

# Summary

- **You should always develop a product prototype to refine your own ideas and to demonstrate the planned product features to potential customers**

# References

- Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.
- Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.
- Titus Winters, Tom Manshreck, and Hyrum Wright (2020), Software Engineering at Google: Lessons Learned from Programming Over Time, O'Reilly Media.
- Project Management Institute (2021), A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Seventh Edition and The Standard for Project Management, PMI.
- Project Management Institute (2017), A Guide to the Project Management Body of Knowledge (PMBOK Guide), Sixth Edition, Project Management Institute.
- Project Management Institute (2017), Agile Practice Guide, Project Management Institute.