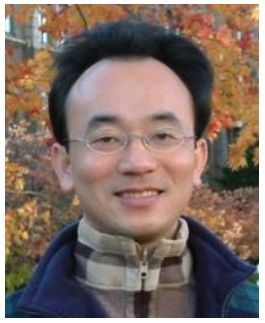
Software Engineering

Features, Scenarios, and Stories

1132SE04
MBA, IM, NTPU (M5010) (Spring 2025)
Wed 2, 3, 4 (9:10-12:00) (B3F17)

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



Min-Yuh Day, Ph.D,
Professor

Institute of Information Management, National Taipei University

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

2025-03-19

Syllabus

Week	Date	Subject/Topics
1	2025/02/19	Introduction to Software Engineering
2	2025/02/26	Software Products and Project Management: Software product management and prototyping with Generative AI
3	2025/03/05	Agile Software Engineering: Agile methods, Scrum, and Extreme Programming
4	2025/03/12	Case Study on Software Engineering I
5	2025/03/19	Features, Scenarios, and Stories
6	2025/03/26	Software Architecture: Architectural design, System decomposition, and Distribution architecture

Syllabus

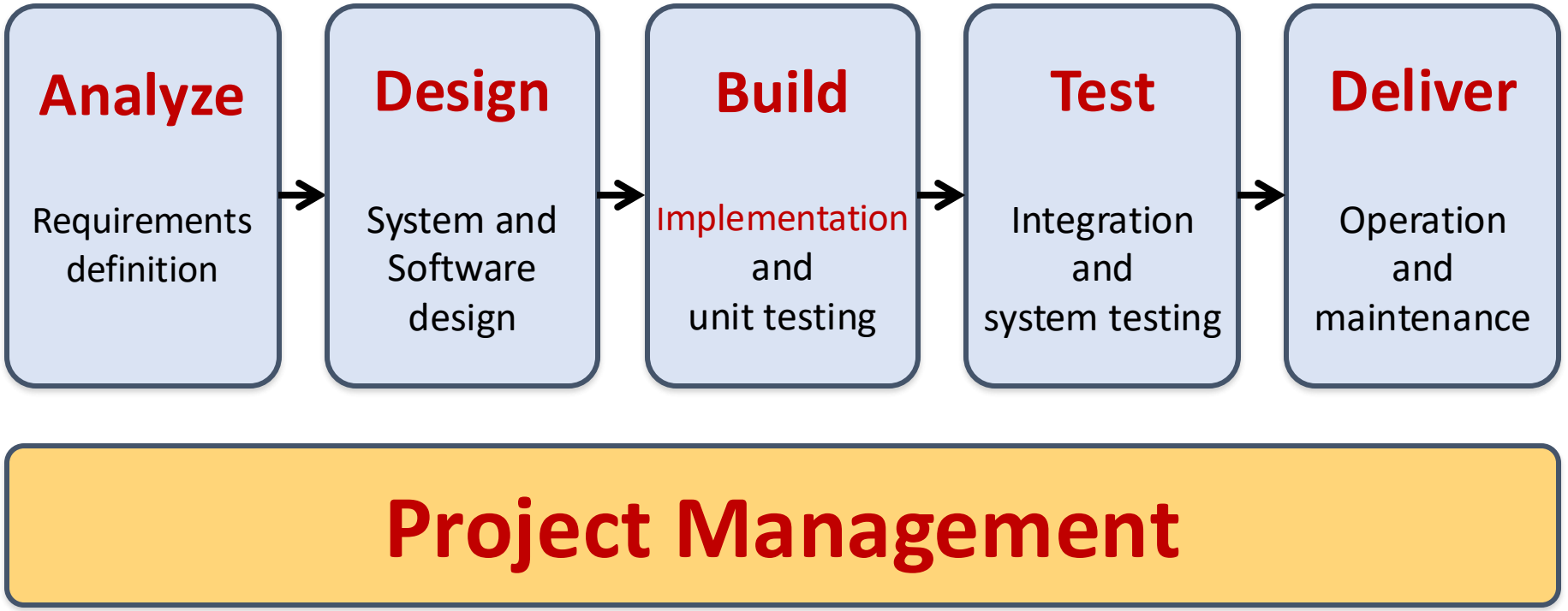
Week	Date	Subject/Topics
7	2025/04/02	Make-up holiday for NTPU Sports Day (No Classes)
8	2025/04/09	Midterm Project Report
9	2025/04/16	Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service
10	2025/04/23	Cloud Computing and Cloud Software Architecture
11	2025/04/30	Case Study on Software Engineering II
12	2025/05/07	Microservices Architecture, RESTful services, Service deployment

Syllabus

Week	Date	Subject/Topics
13	2025/05/14	Industry Practices of Software Engineering
14	2025/05/21	Security and Privacy; Reliable Programming; Testing: Functional testing, Test automation, Test-driven development, and Code reviews; DevOps and Code Management: Code management and DevOps automation
15	2025/05/28	Final Project Report I
16	2025/06/04	Final Project Report II

Features, Scenarios, and Stories

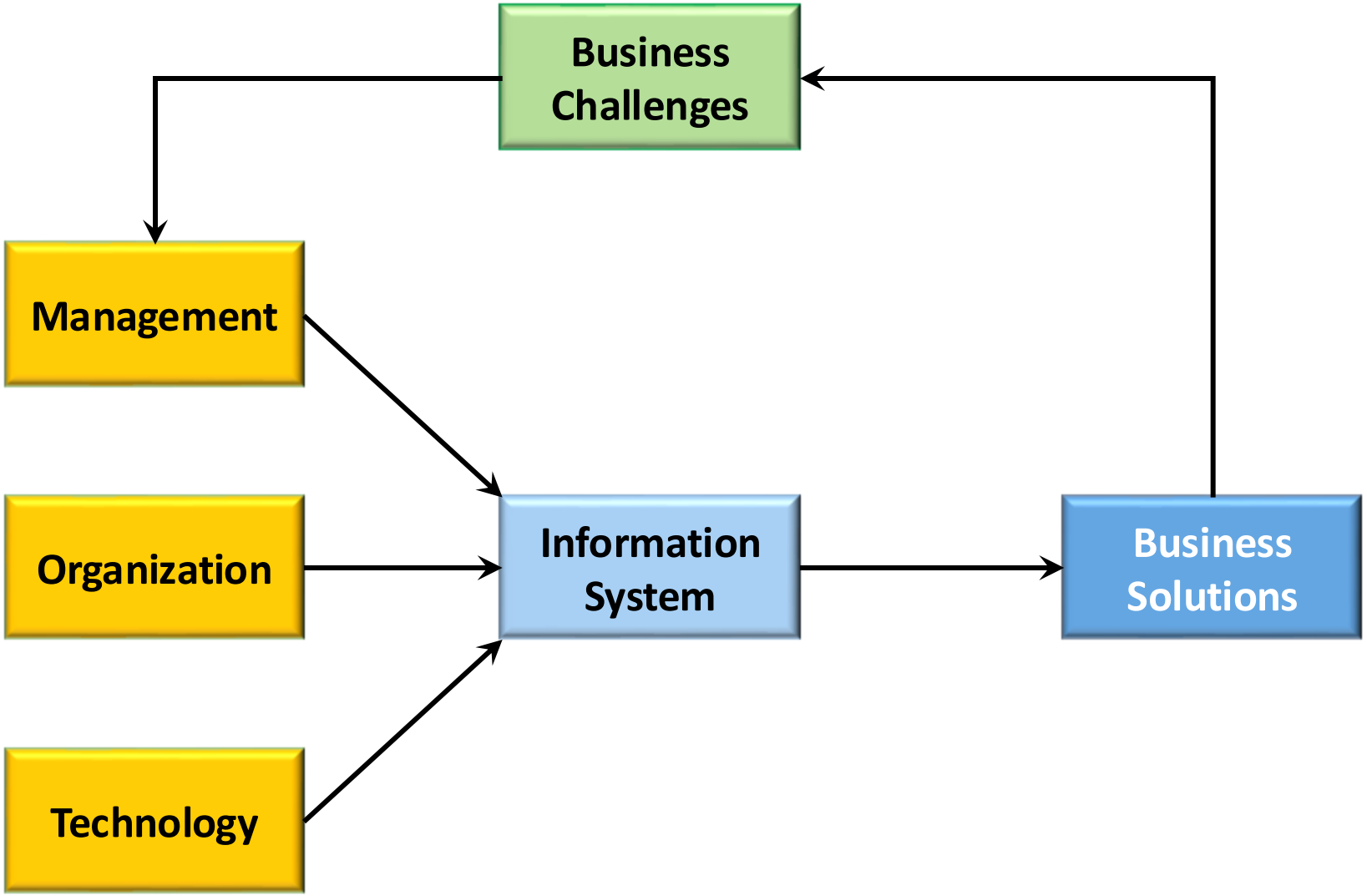
Software Engineering
and
Project Management



Information Management (MIS)
Information Systems

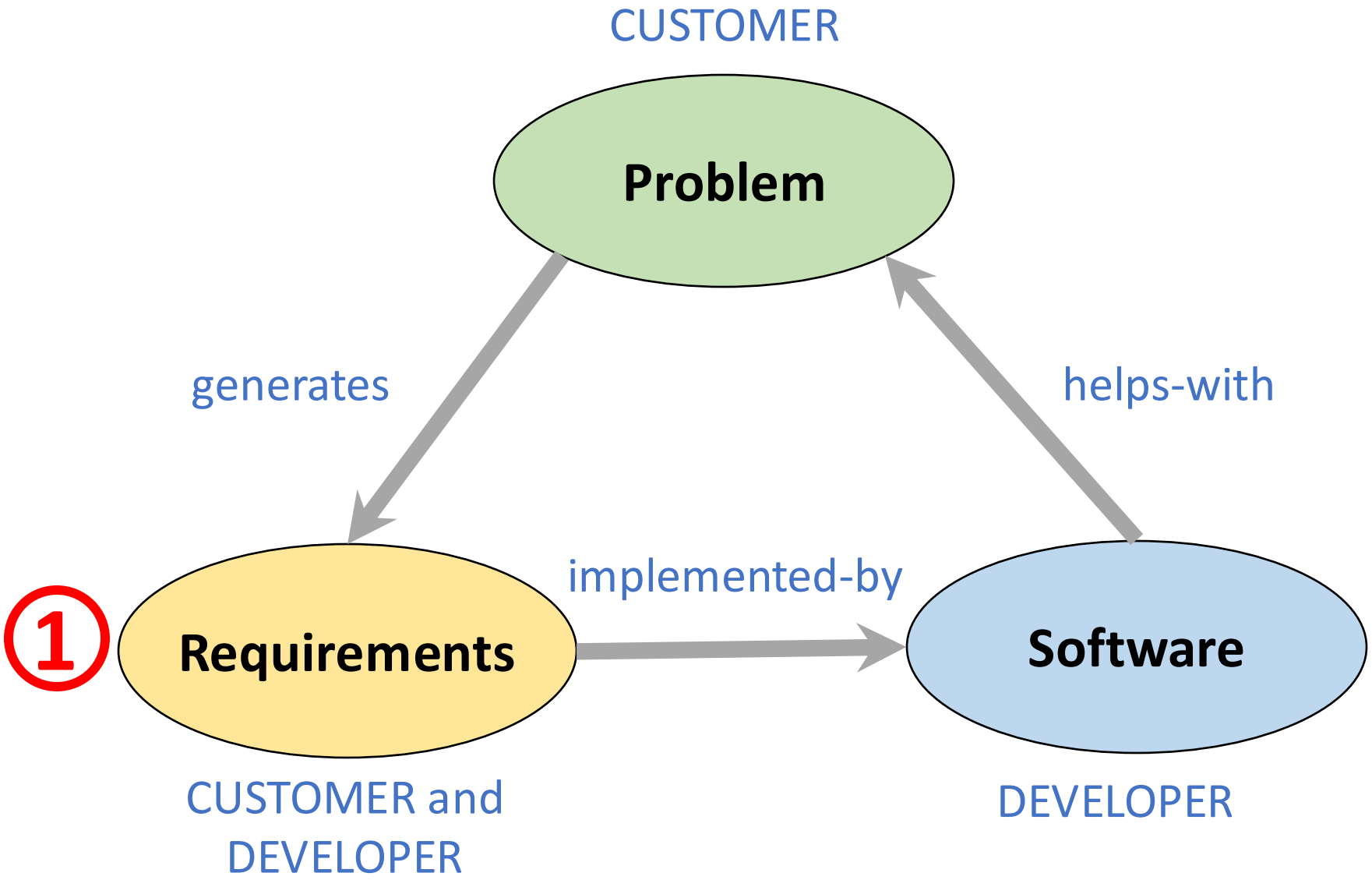
Source: Kenneth C. Laudon & Jane P. Laudon (2014), Management Information Systems: Managing the Digital Firm, Thirteenth Edition, Pearson.

Fundamental MIS Concepts



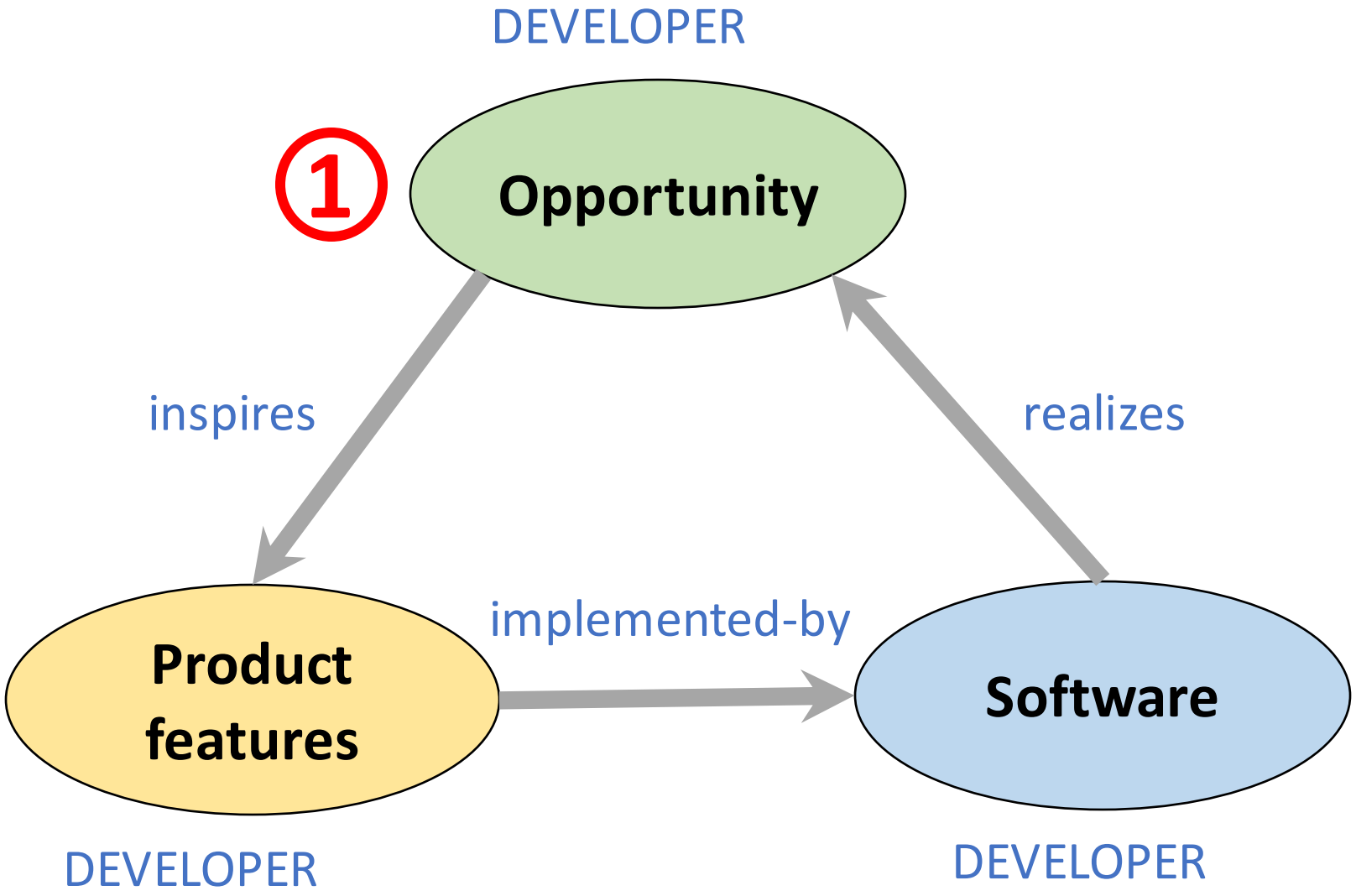
Source: Kenneth C. Laudon & Jane P. Laudon (2014), Management Information Systems: Managing the Digital Firm, Thirteenth Edition, Pearson.

Project-based software engineering



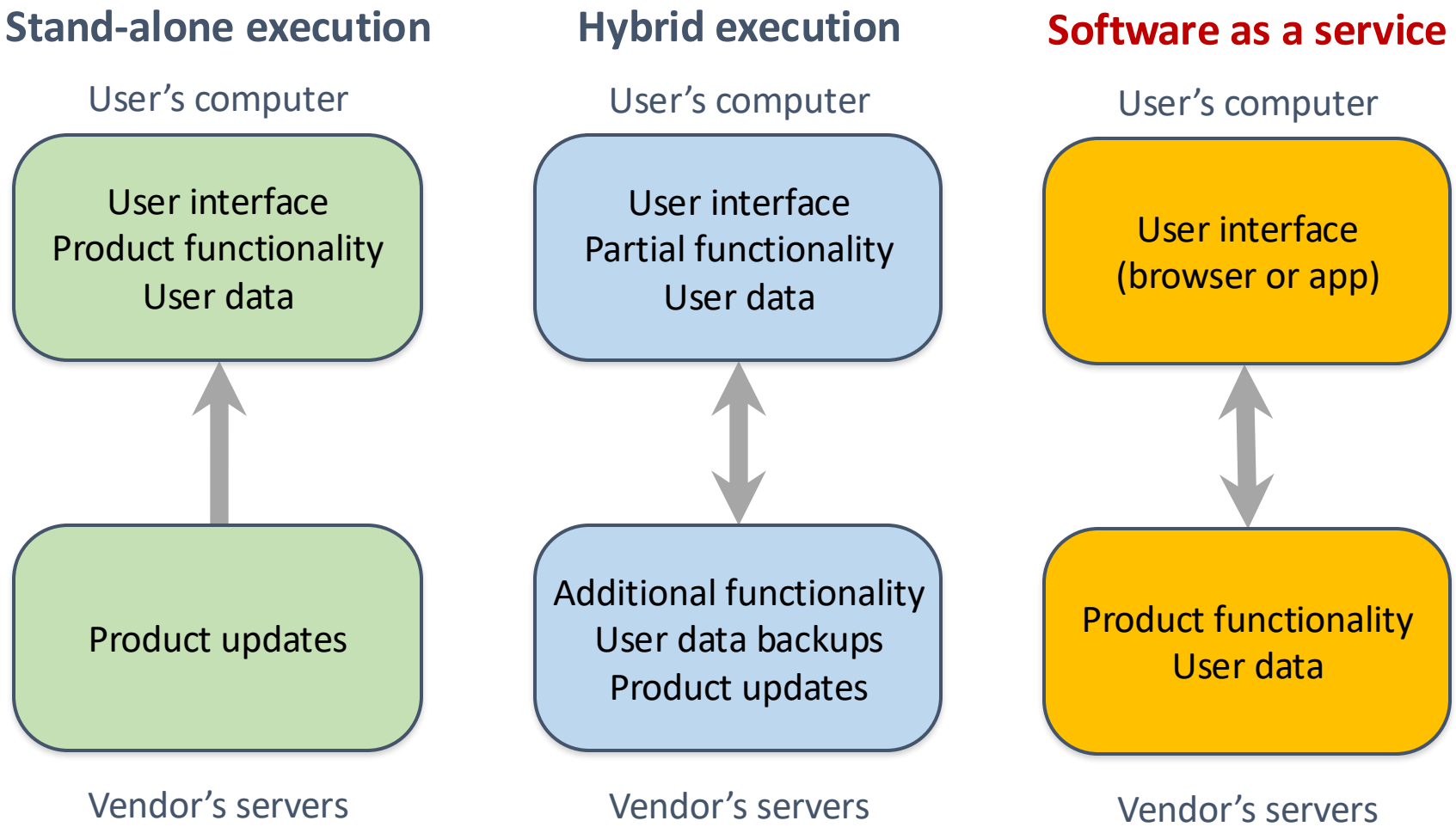
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Product software engineering



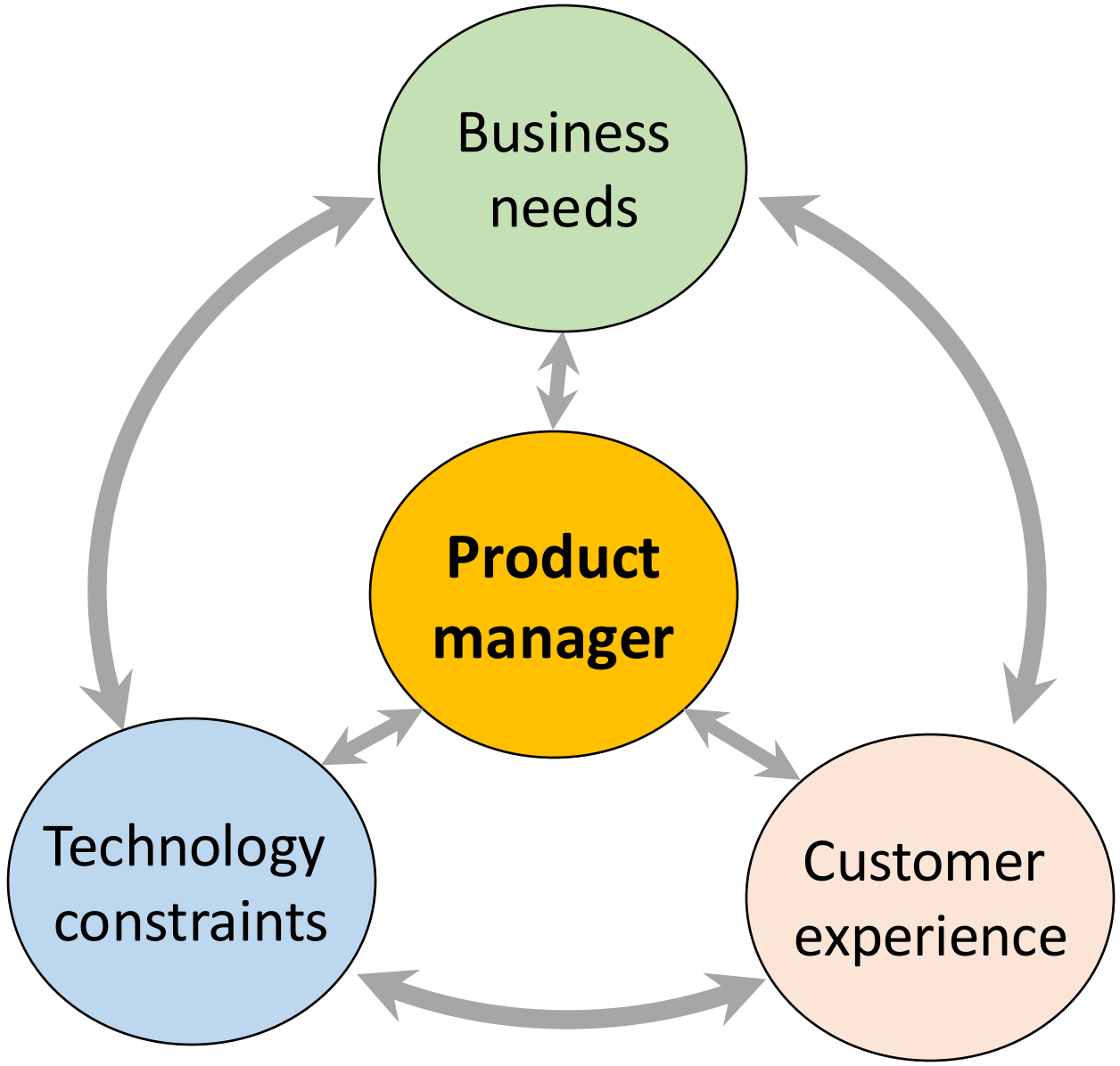
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software execution models



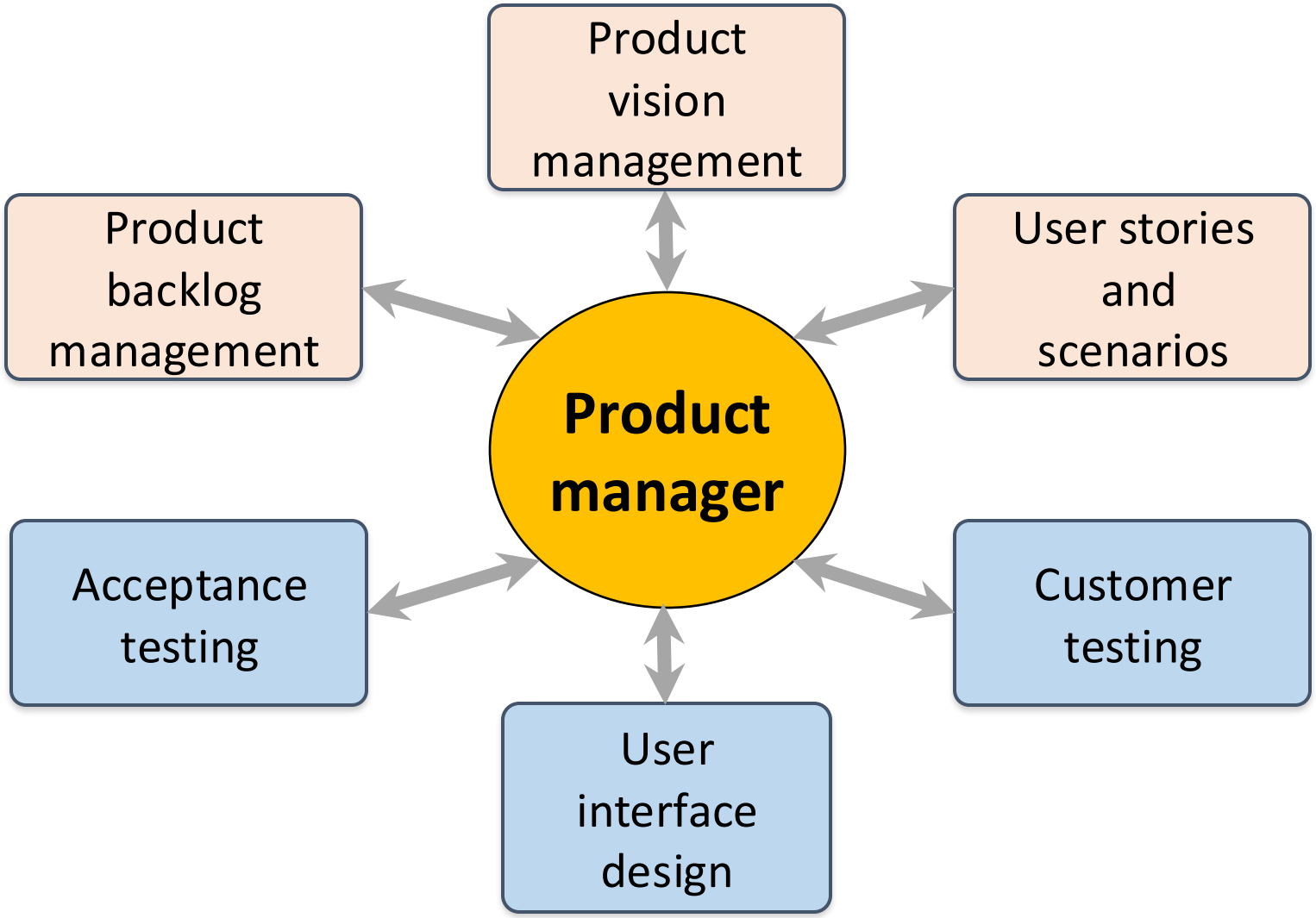
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Product management concerns



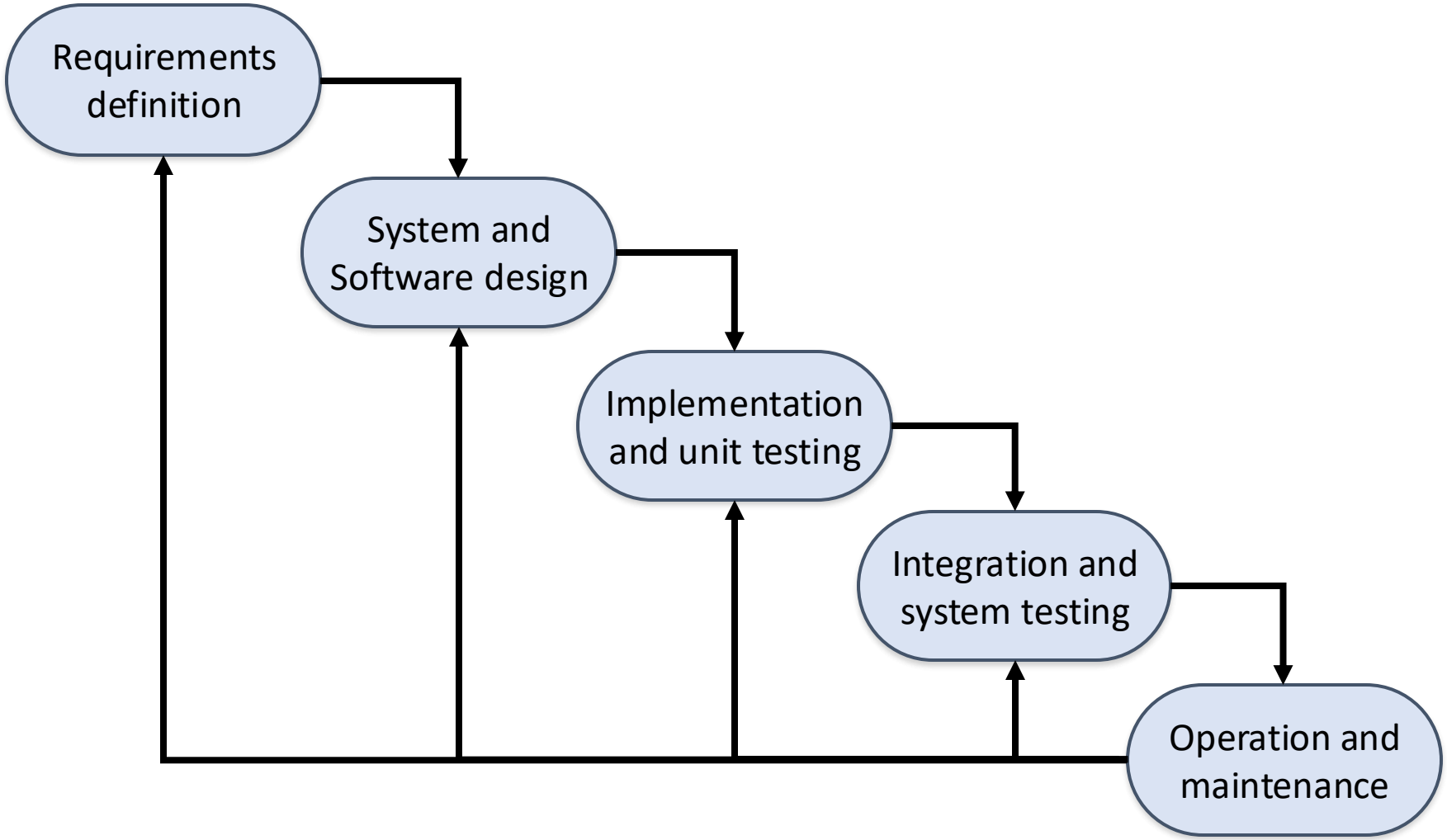
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Technical interactions of product managers



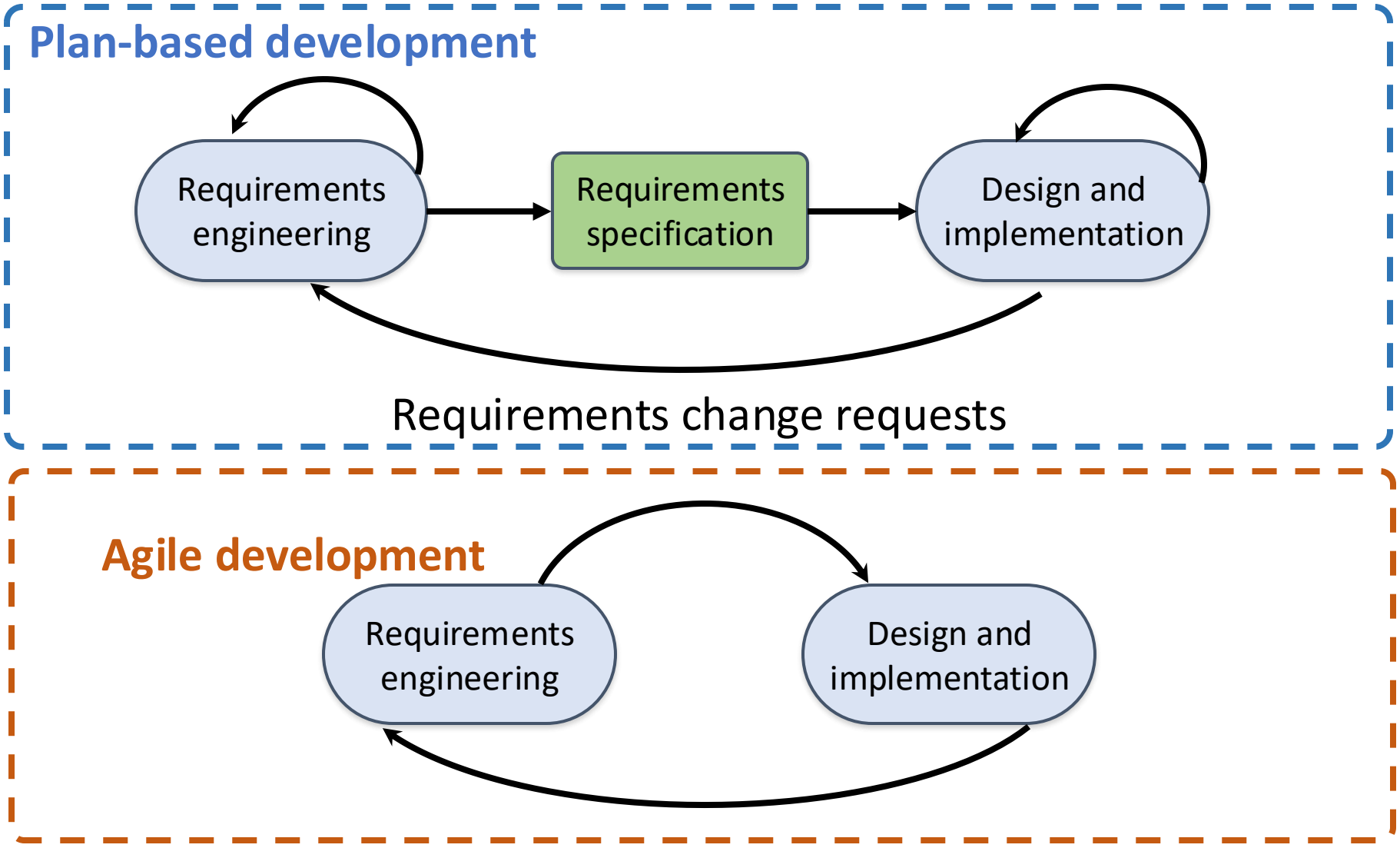
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software Development Life Cycle (SDLC)
The waterfall model



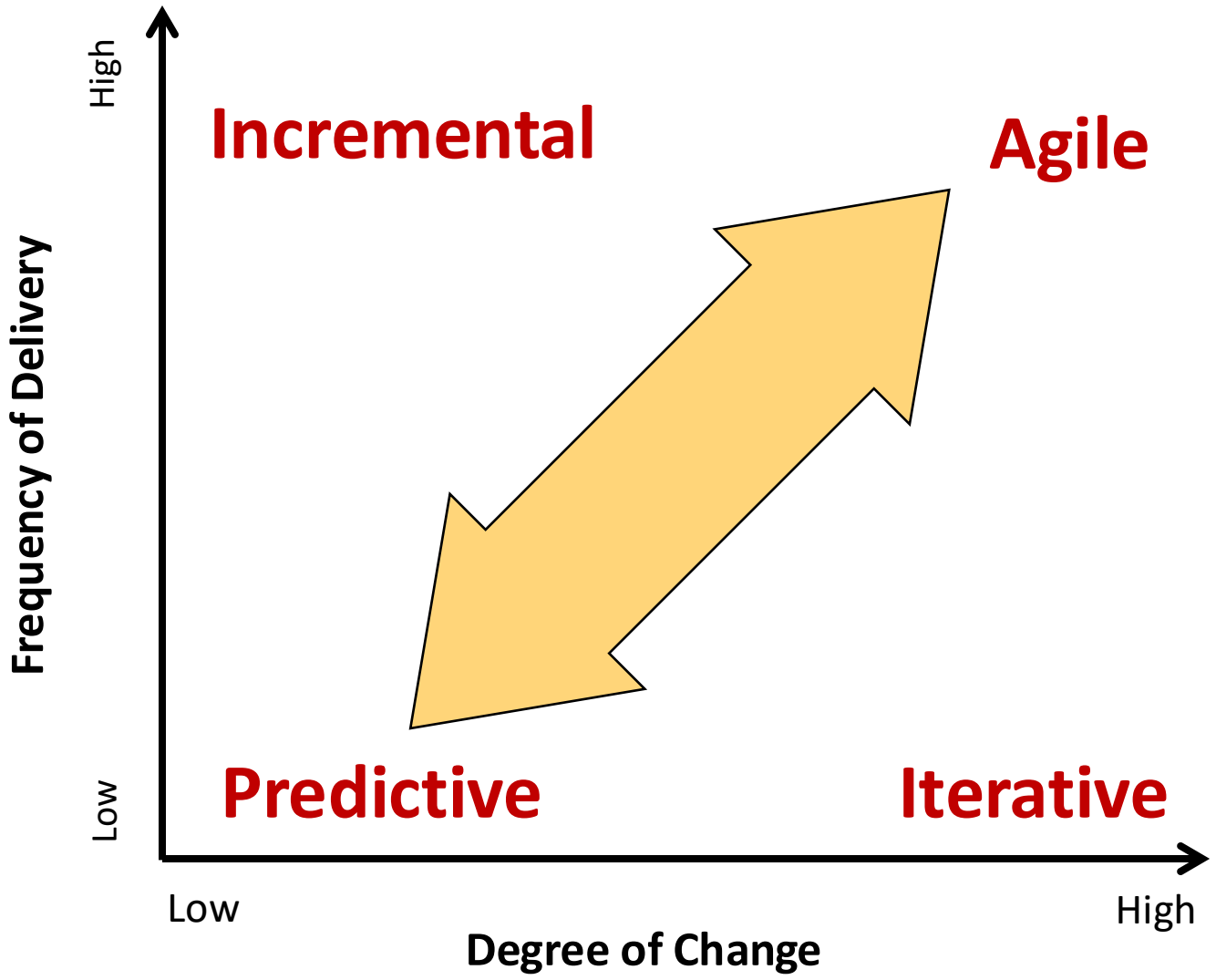
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Plan-based and Agile development



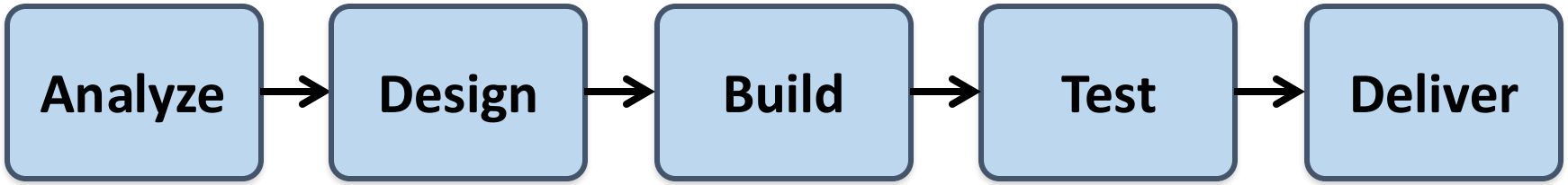
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

The Continuum of Life Cycles



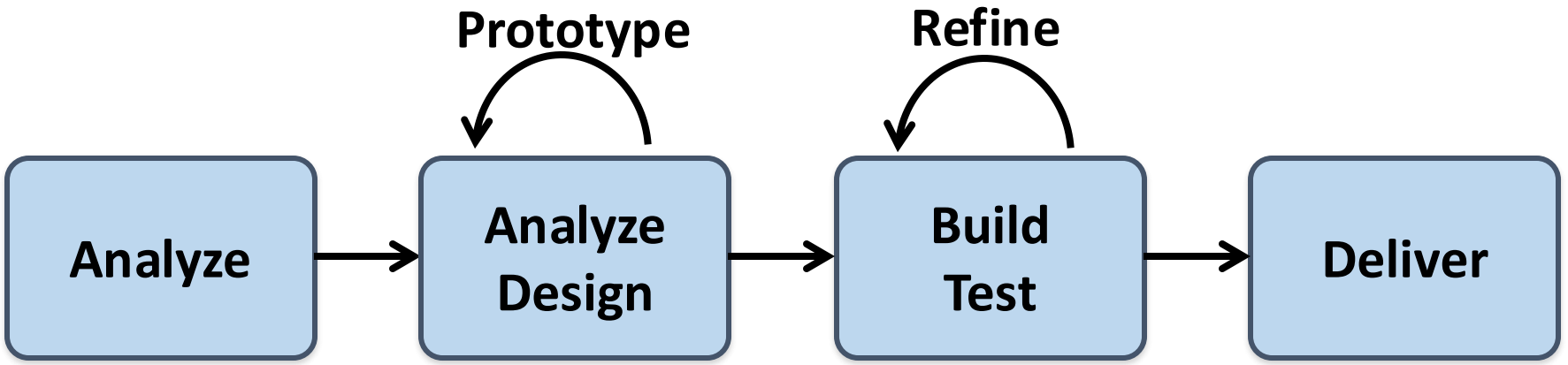
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Predictive Life Cycle



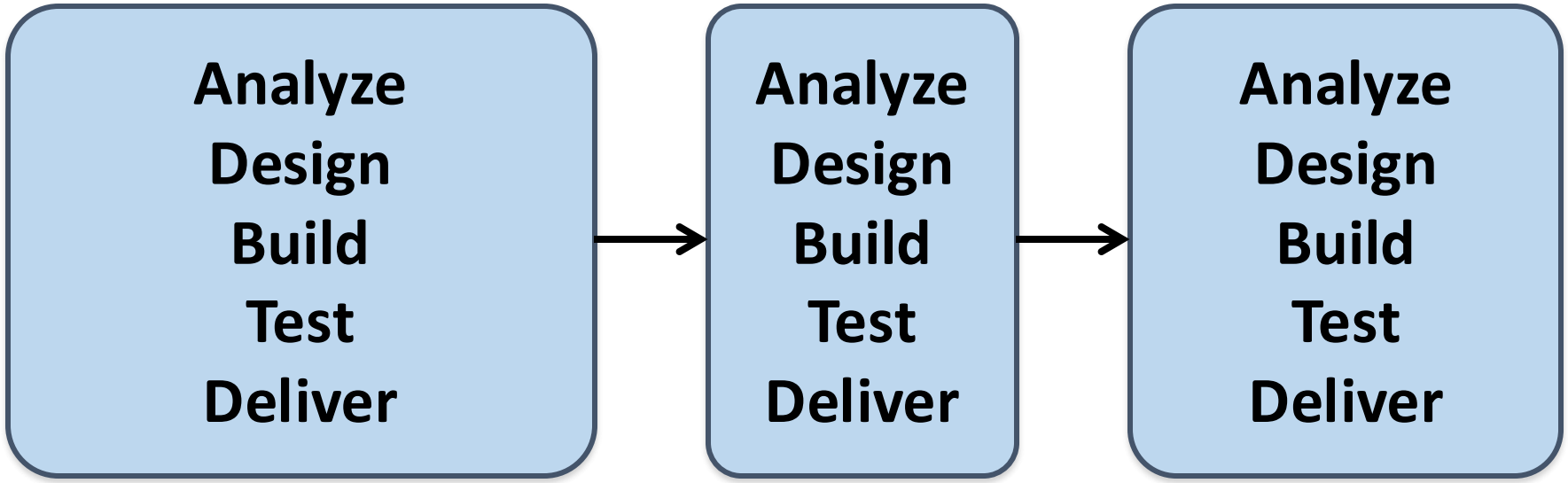
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Iterative Life Cycle



Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

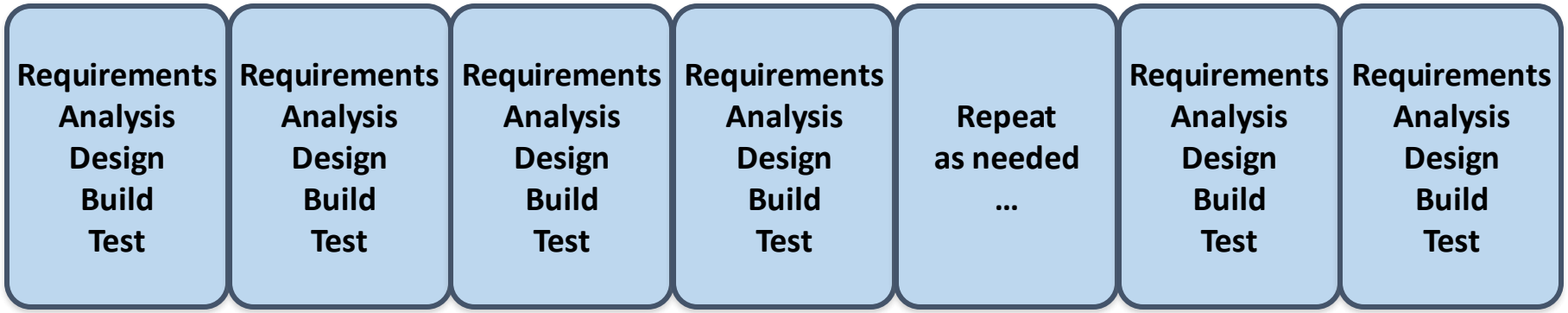
A Life Cycle of Varying-Sized Increments



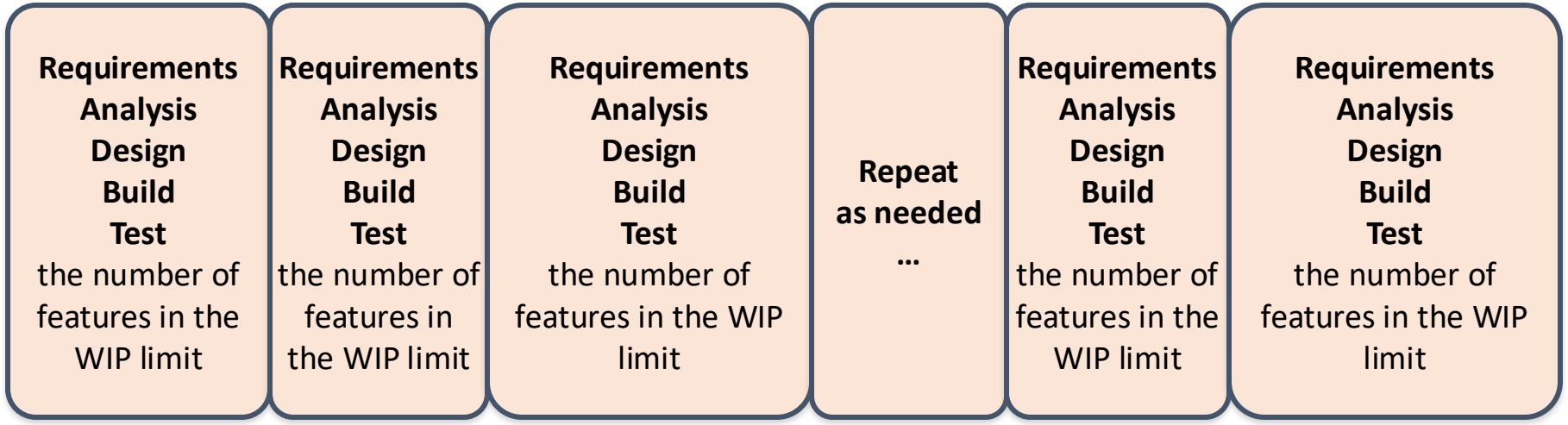
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Iteration-Based and Flow-Based Agile Life Cycles

Iteration-Based Agile

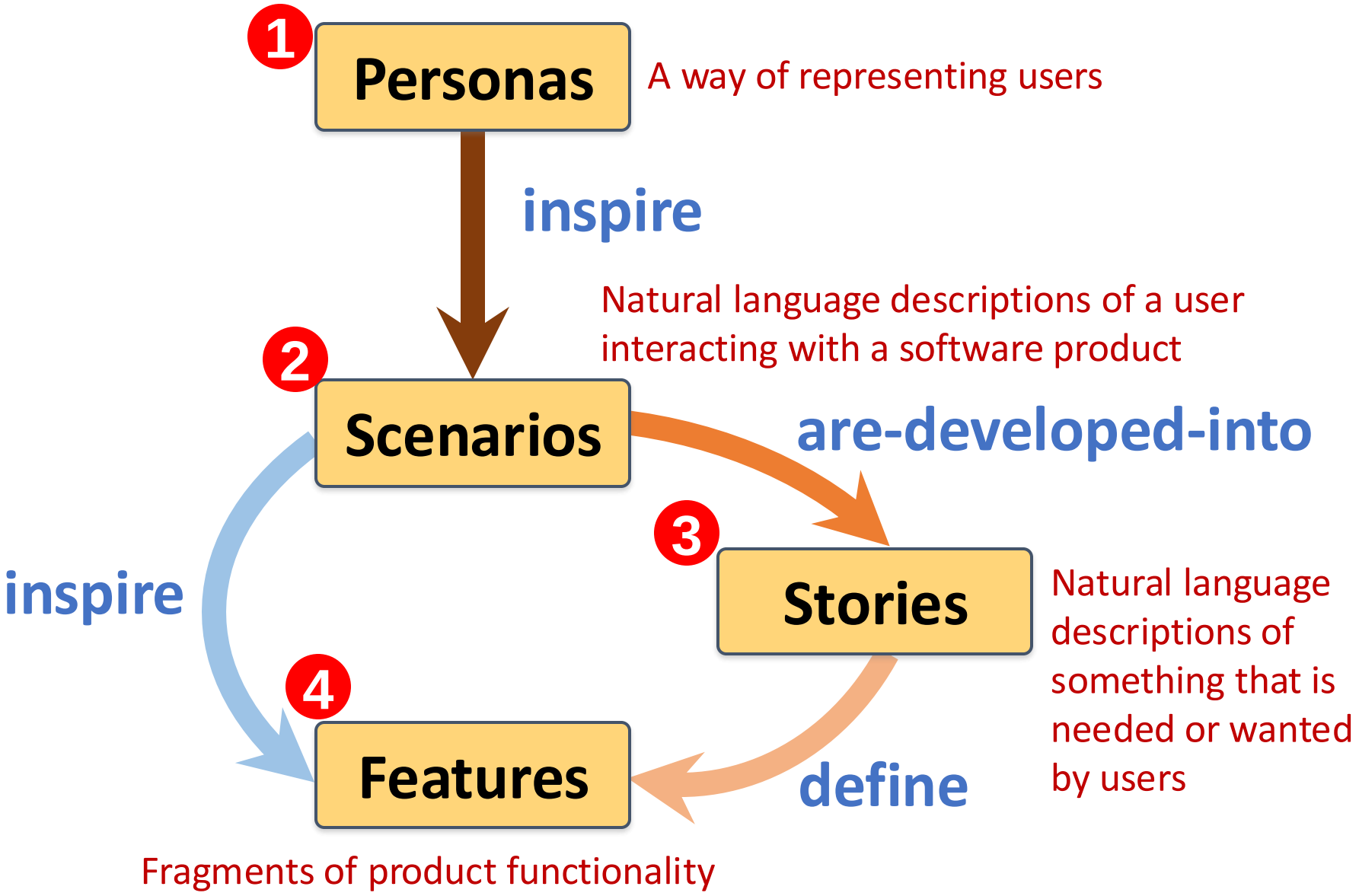


Flow-Based Agile



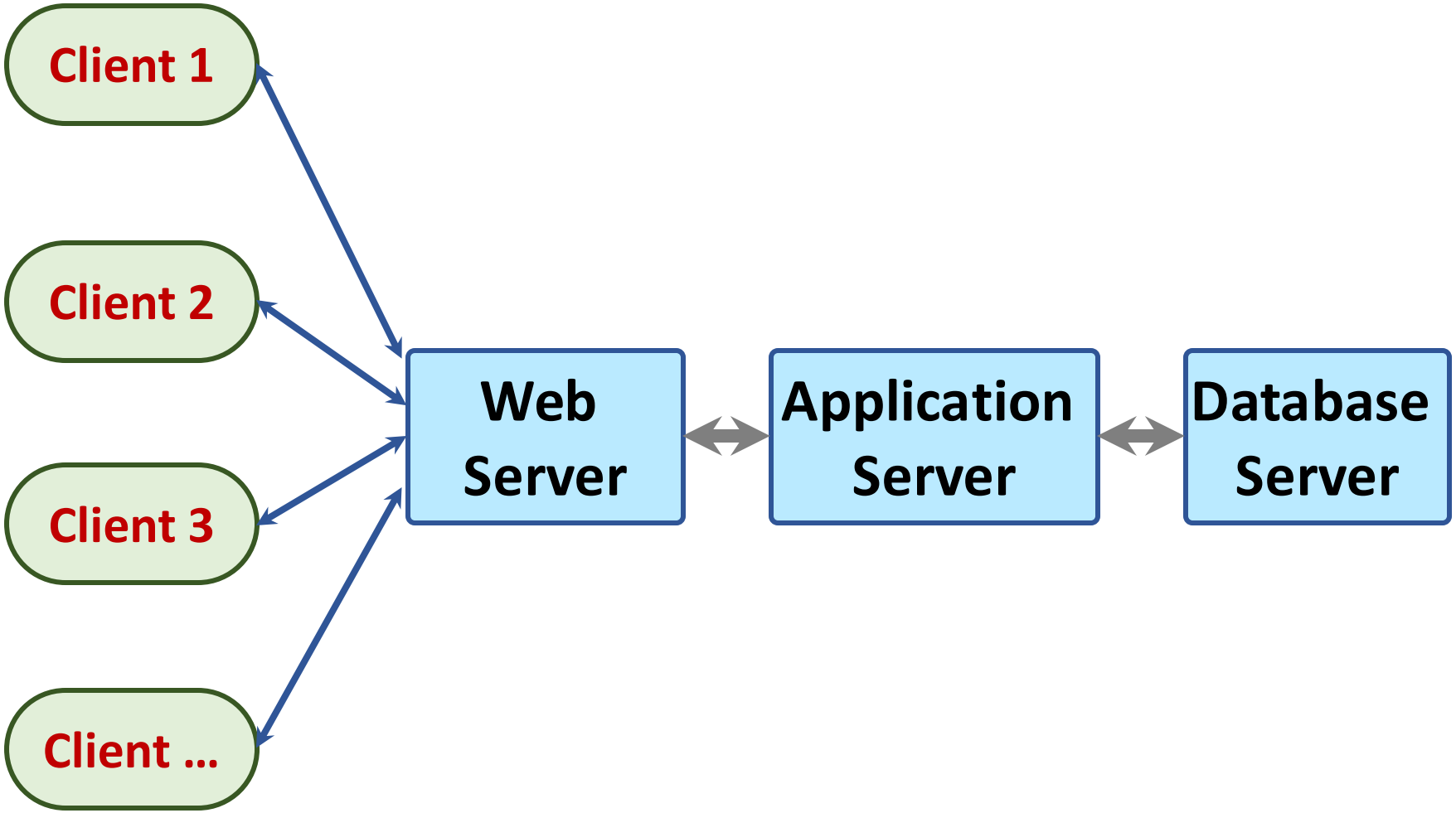
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

From personas to features



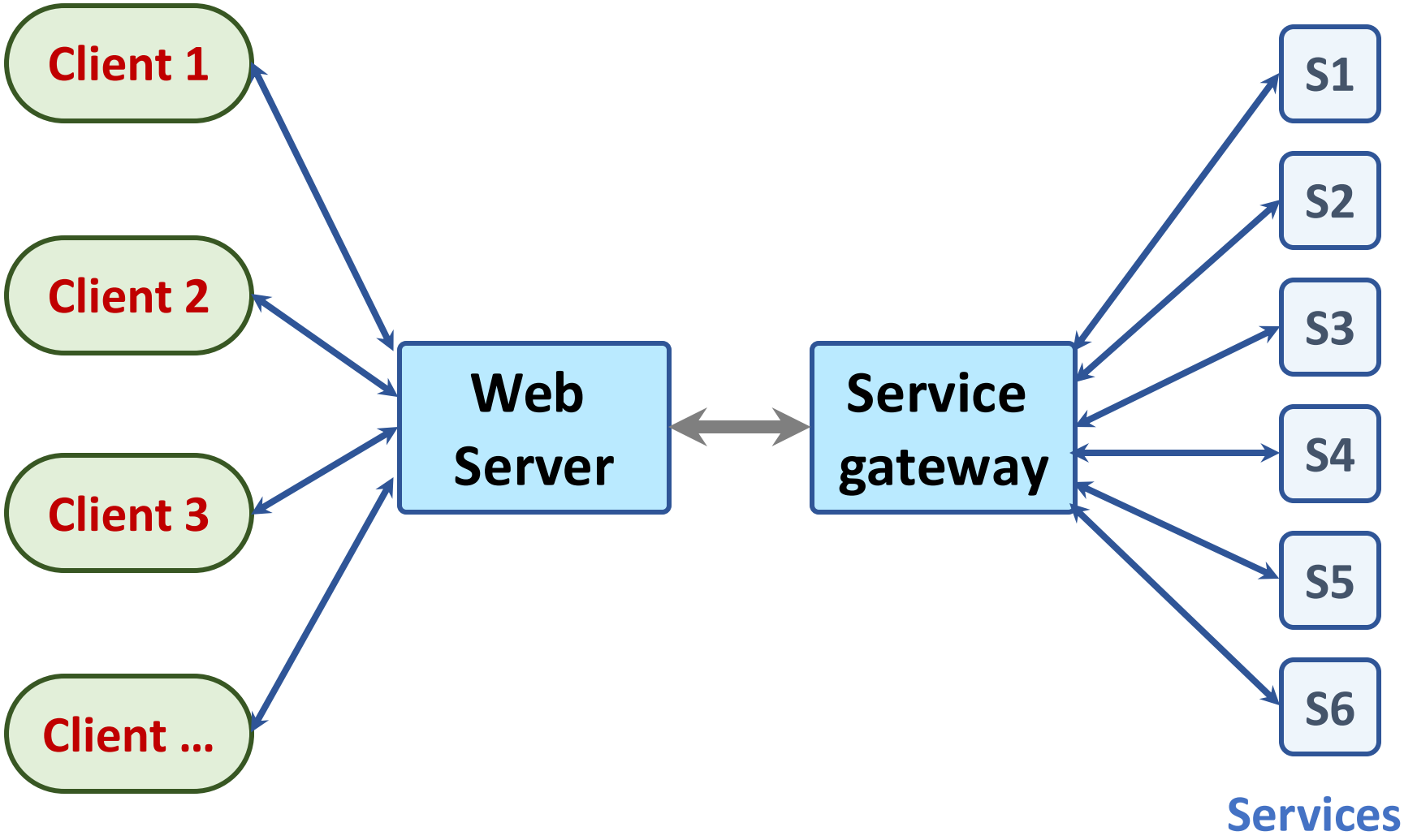
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Multi-tier client-server architecture

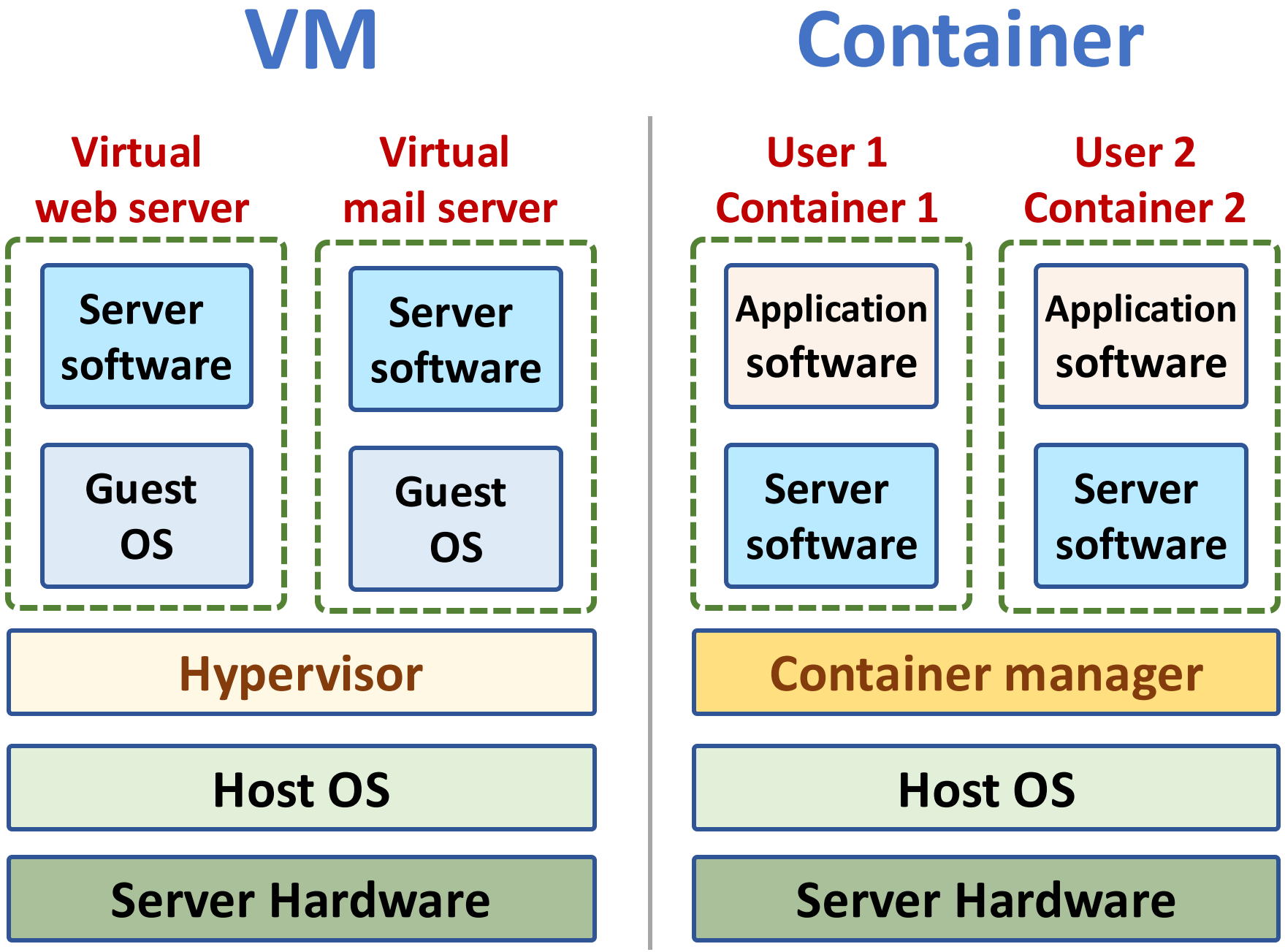


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Service-oriented Architecture

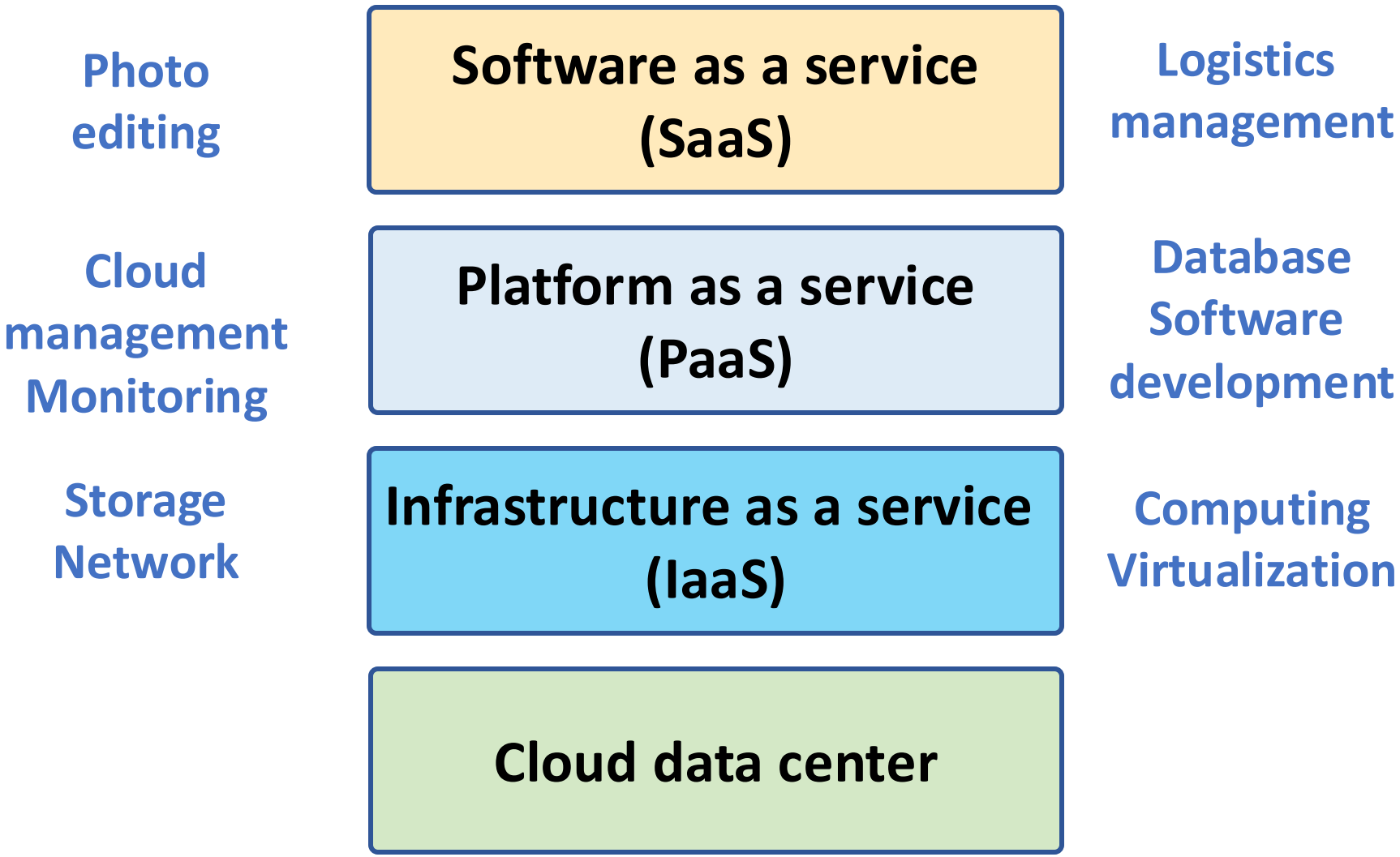


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.



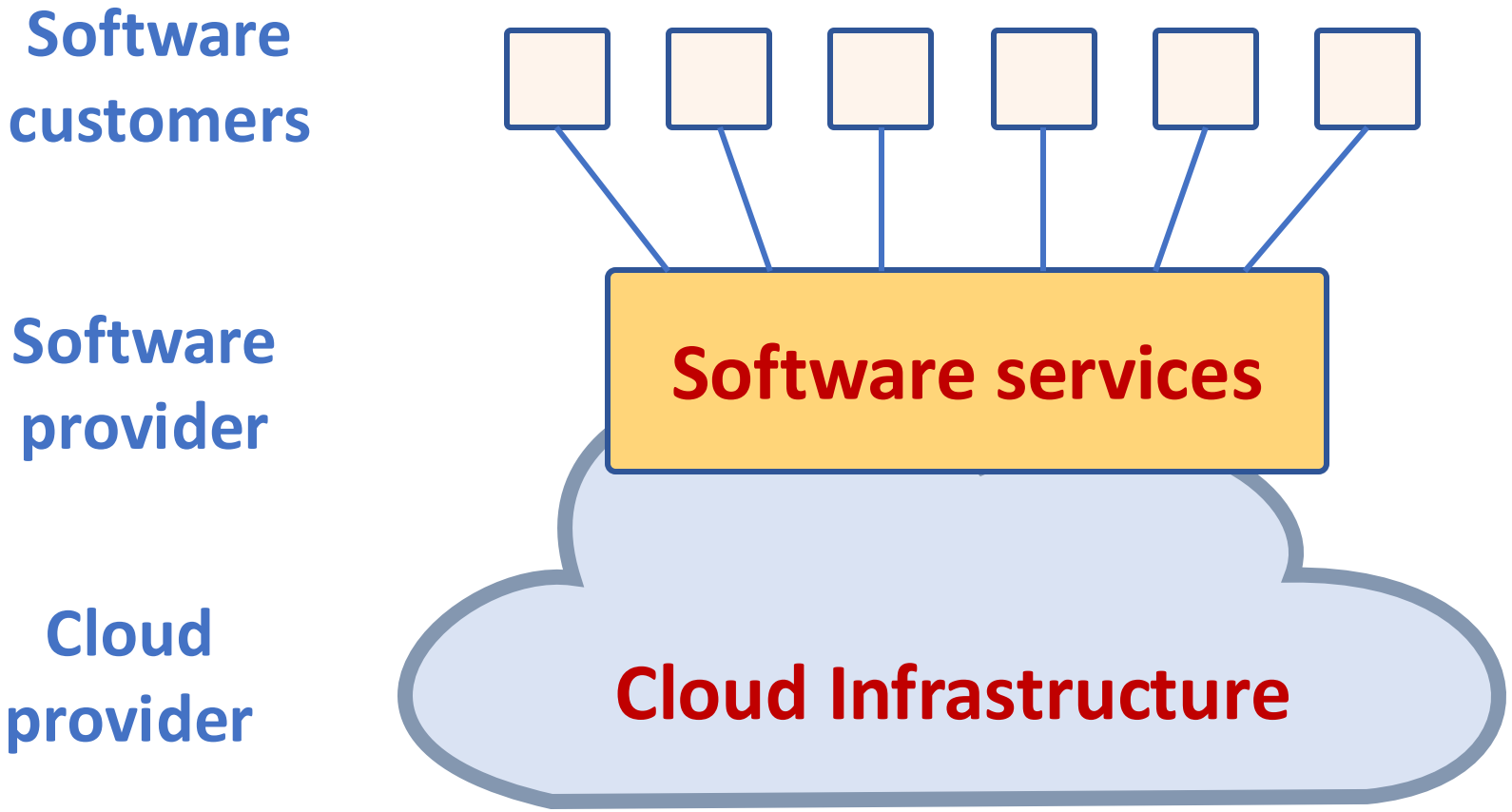
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Everything as a service



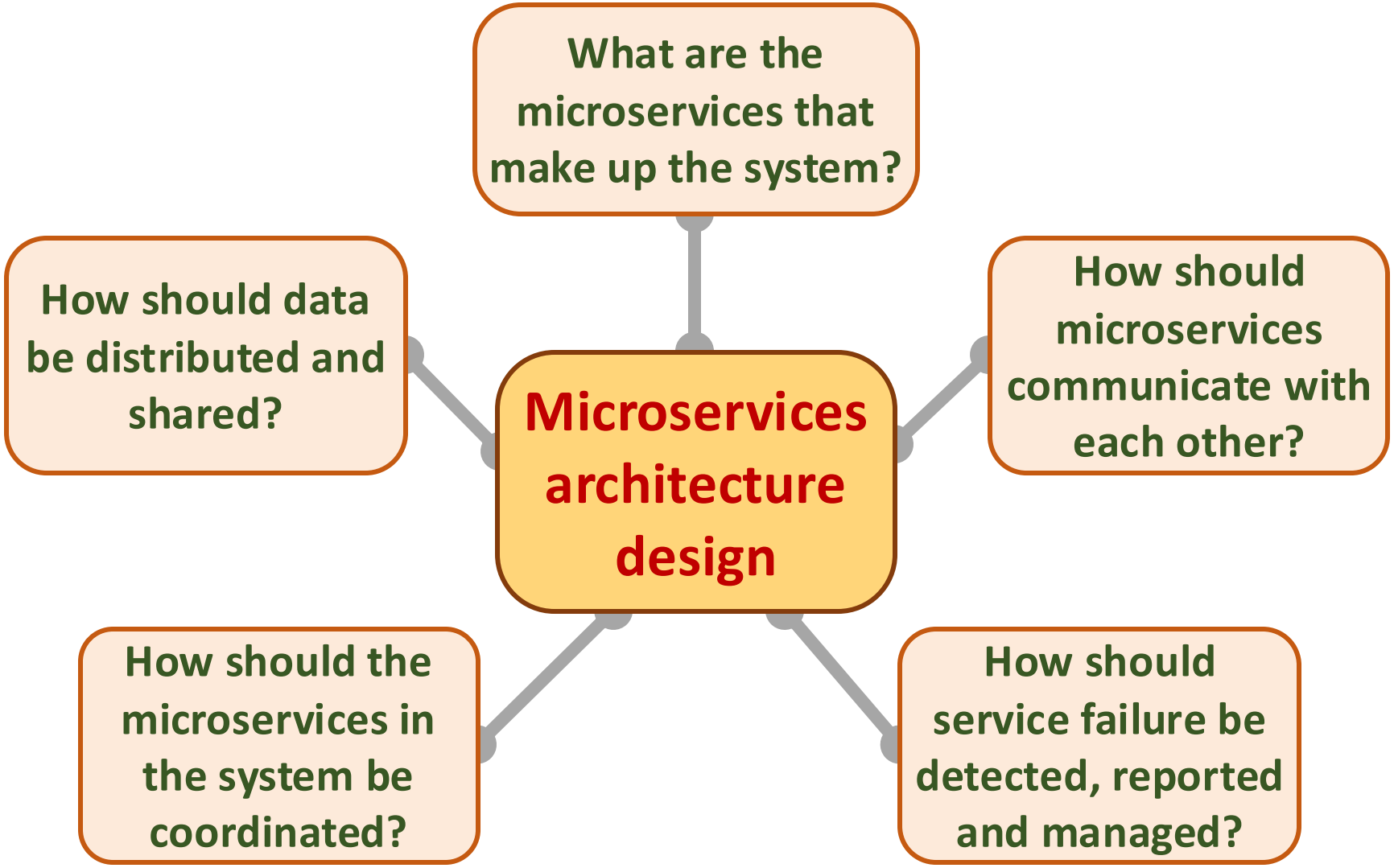
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software as a service



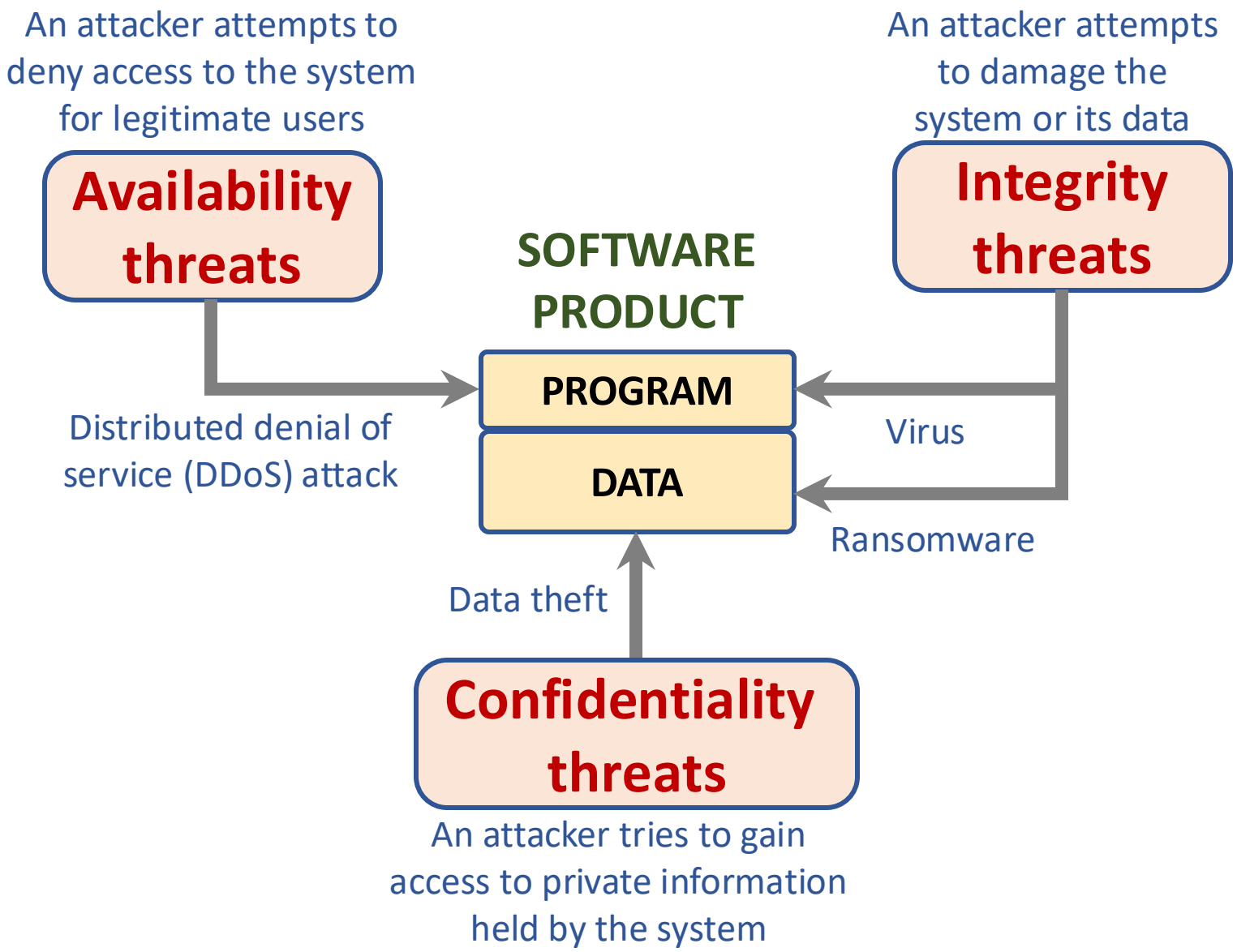
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Microservices architecture – key design questions



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

Types of security threat

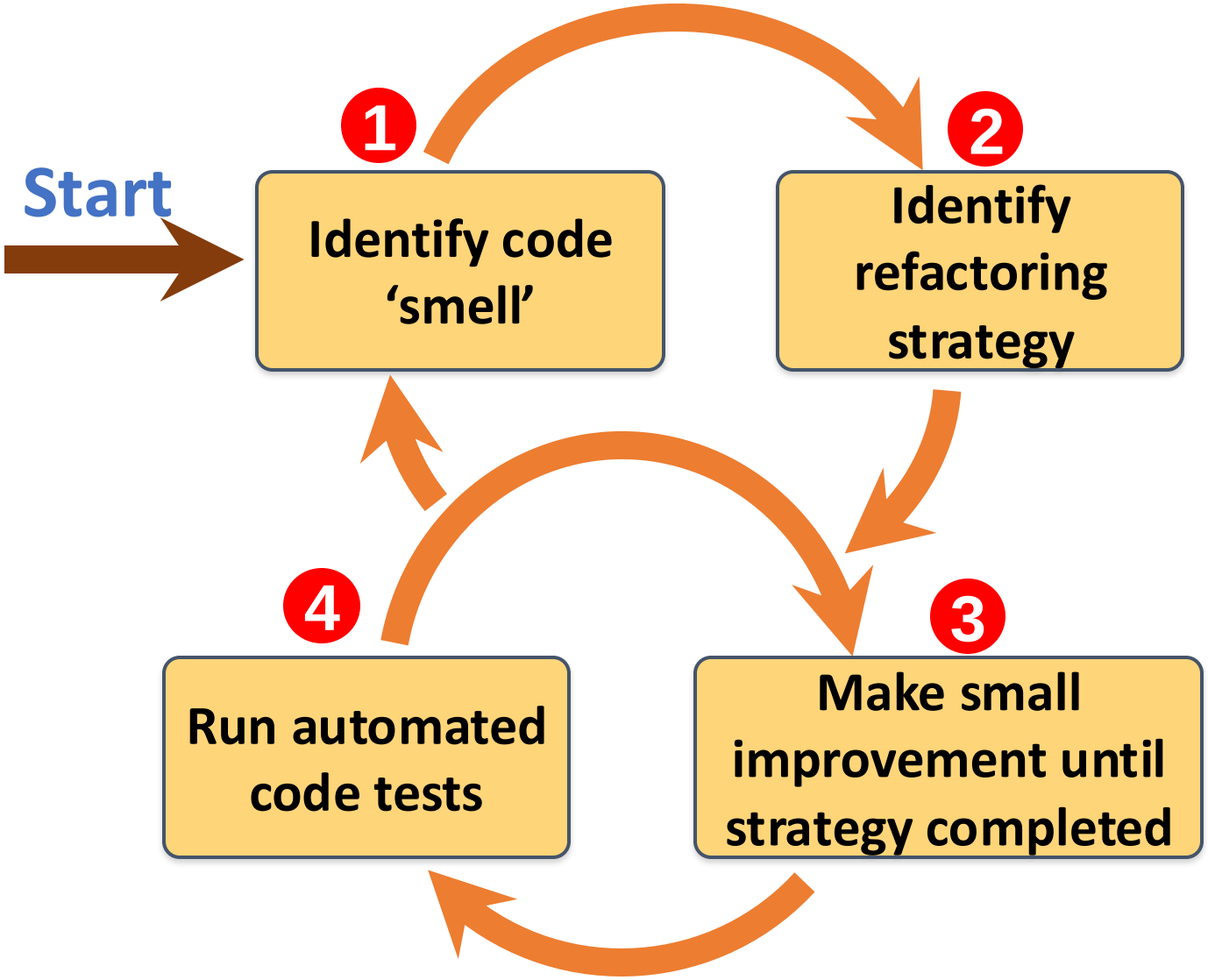


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software product quality attributes

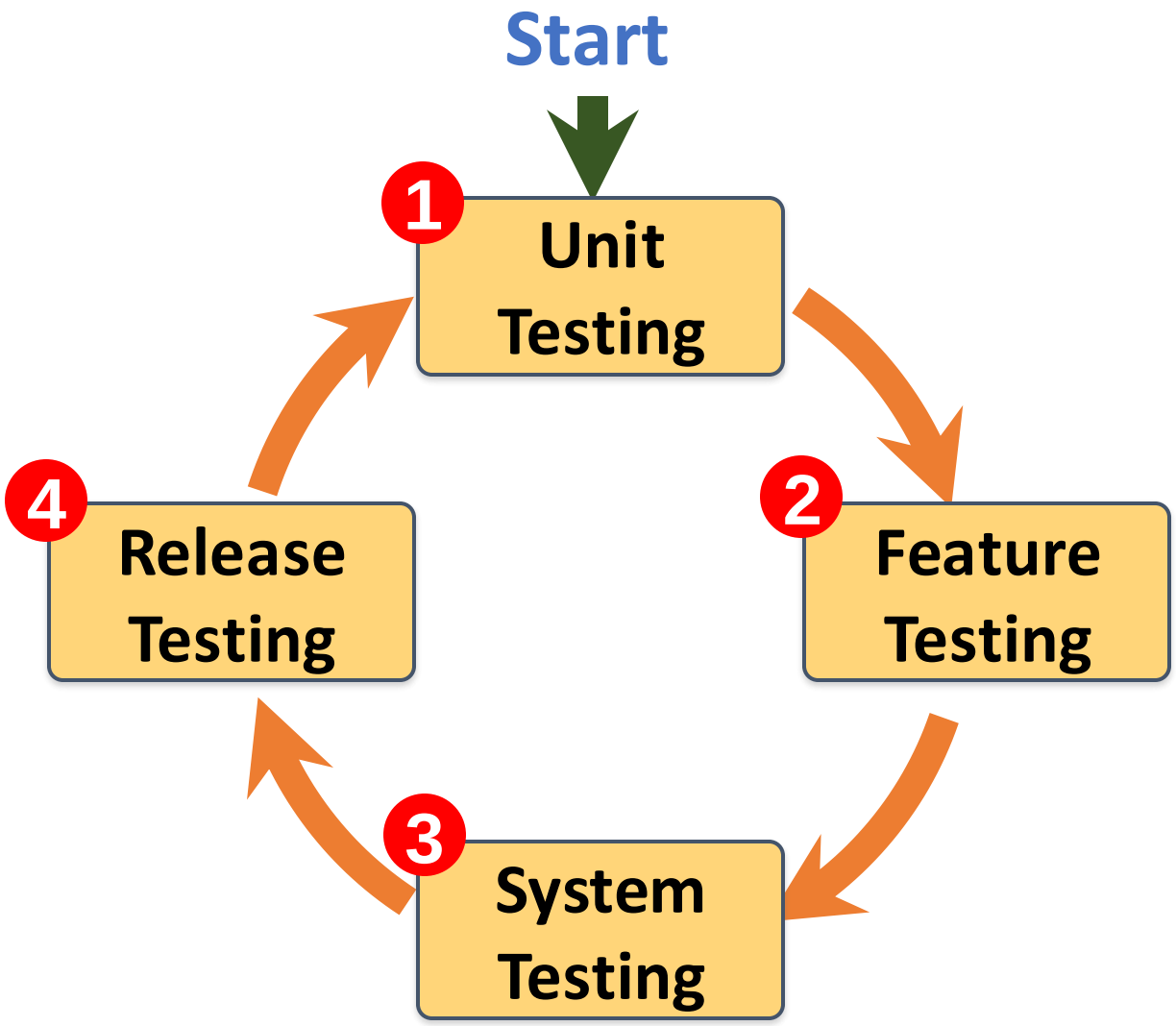
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

A refactoring process



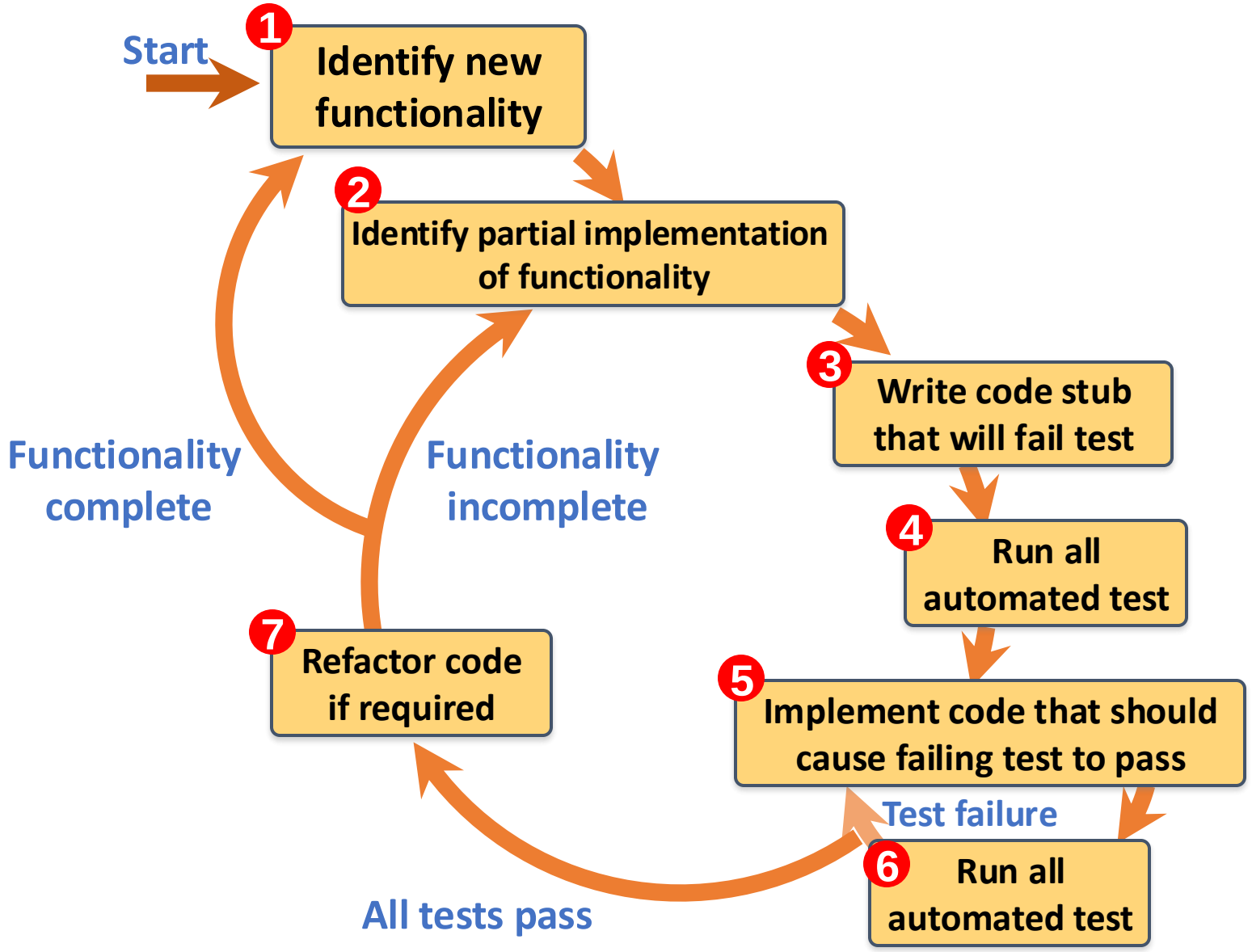
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Functional testing



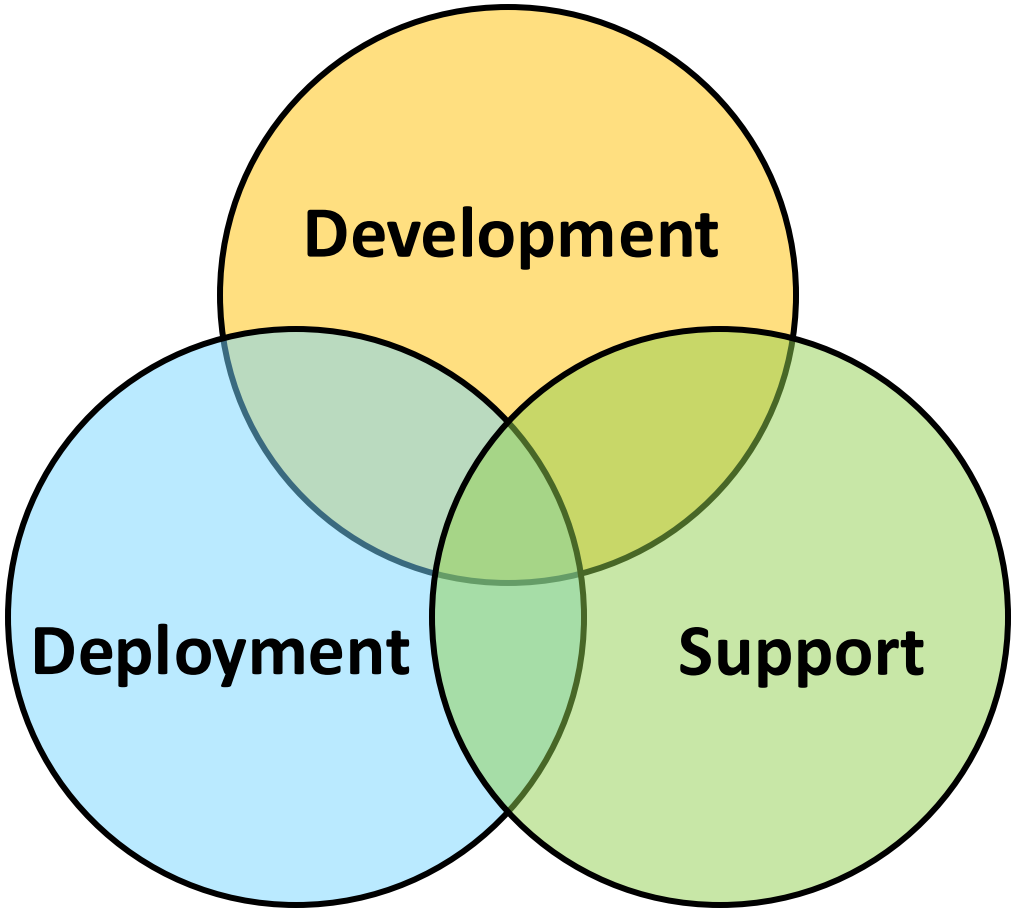
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Test-driven development (TDD)



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

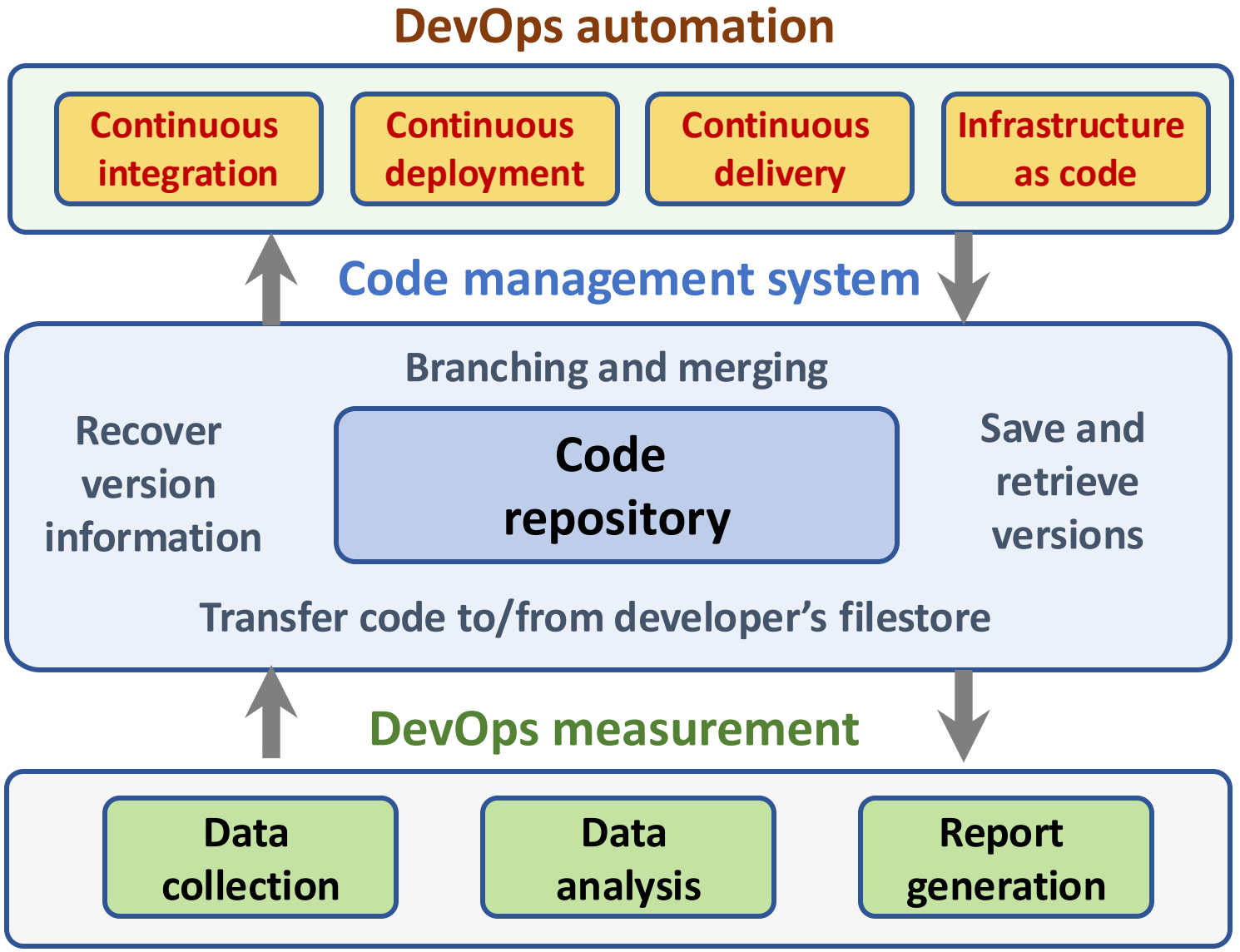
DevOps



Multi-skilled DevOps team

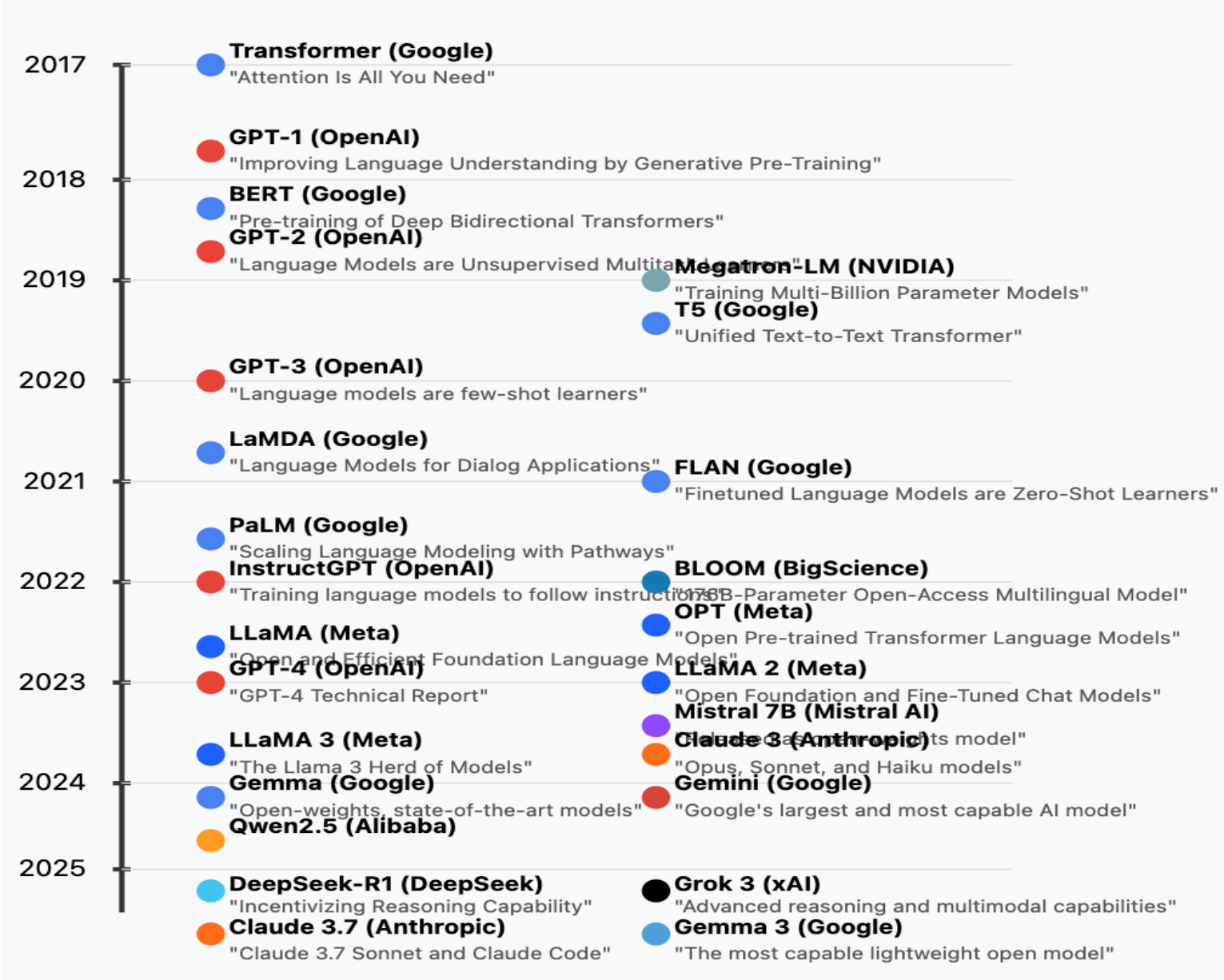
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Code management and DevOps



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

Generative AI LLMs (2017-2025)



Key Organizations

- Google
- OpenAI
- Meta
- Mistral AI
- Alibaba
- xAI
- Anthropic
- NVIDIA
- BigScience

Key Milestones

2017: Transformer architecture

2018: First-gen GPT, BERT

2020: GPT-3 (175B parameters)

2022: Emergent abilities, instruction tuning

2023: GPT-4, multimodal models

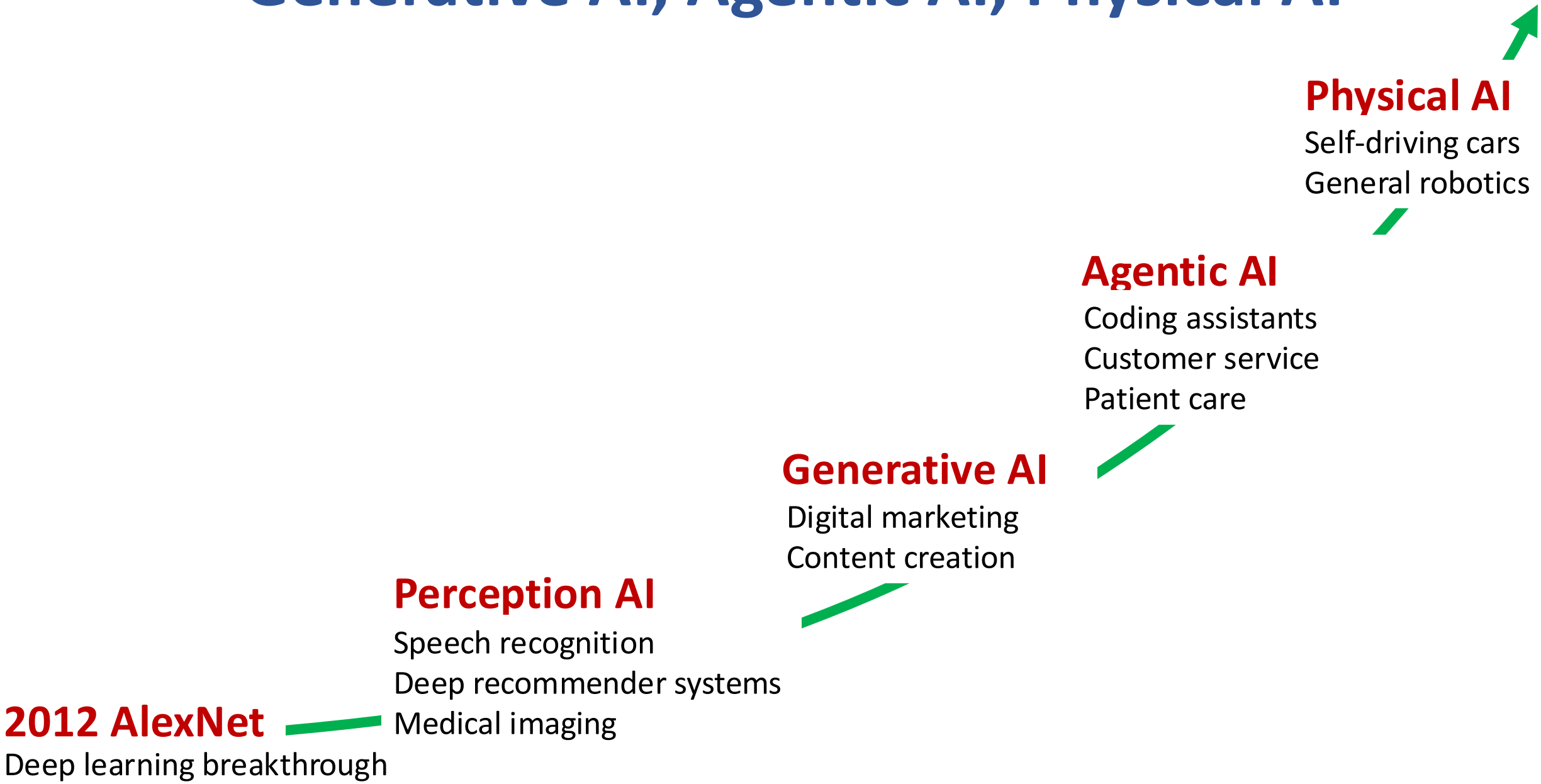
2024: Open-weights race, Mamba2

2025: DeepSeek-R1, Grok 3

Claude 3.7, Gemma 3

Source: https://github.com/Hannibal046/Awesome-LLM

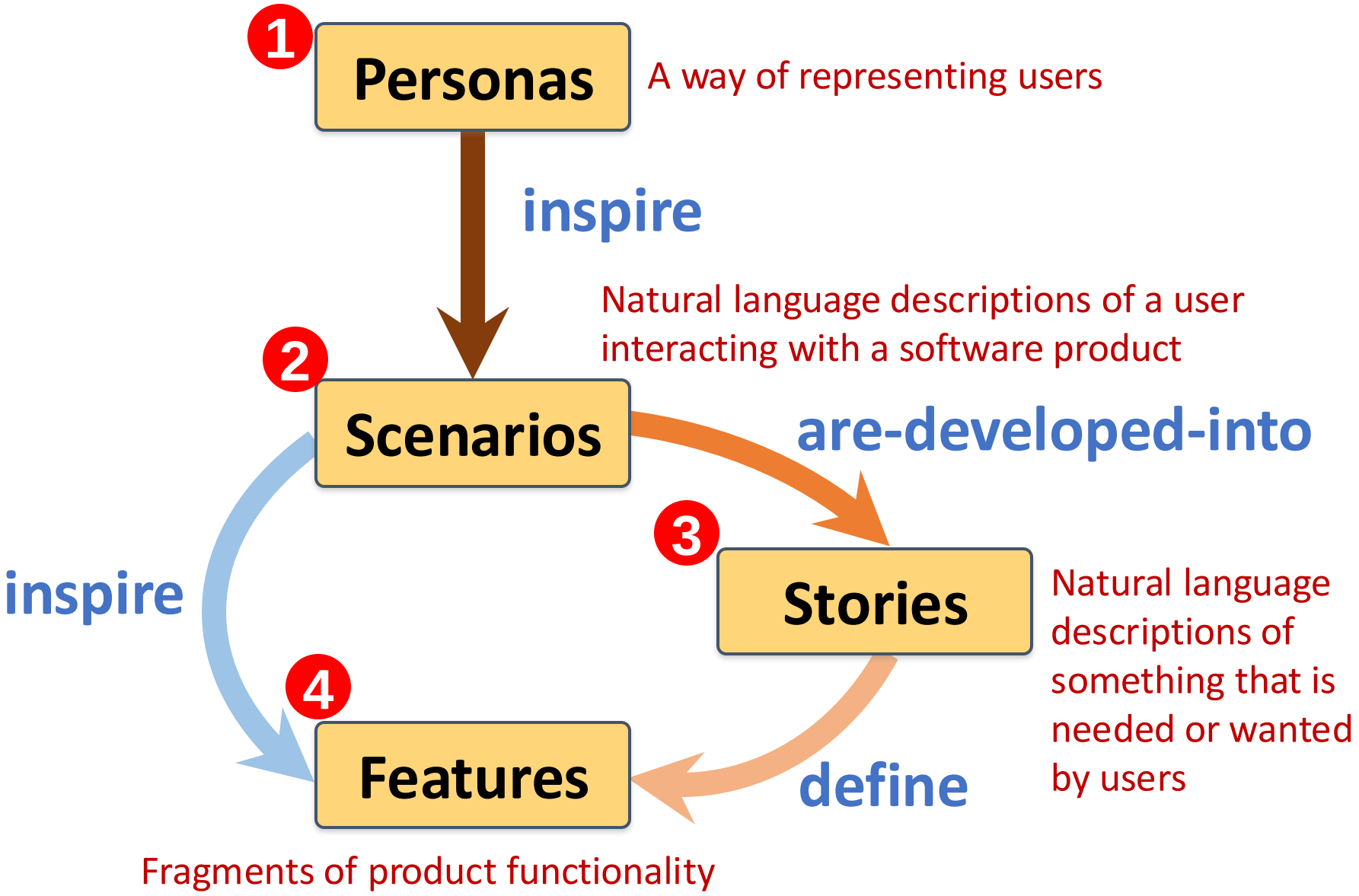
Generative AI, Agentic AI, Physical AI



Source: NVIDIA (2025), GTC March 2025 Keynote with NVIDIA CEO Jensen Huang, https://www.youtube.com/watch?v=_waPvOwL9Z8

Features, Scenarios, and Stories

From personas to features



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

Marketing for Software Products

- Background Introduction
- SWOT
  - Strengths, Weaknesses, Opportunities, Threats
  - PESTLE, Five Forces, Value Chain Analysis
  - TOWS
- STP
  - Segmentation, Targeting, Positioning
- 4P
  - Product, Price, Place, Promotion
- Business Model

The Essence of
Strategic Marketing (STP)

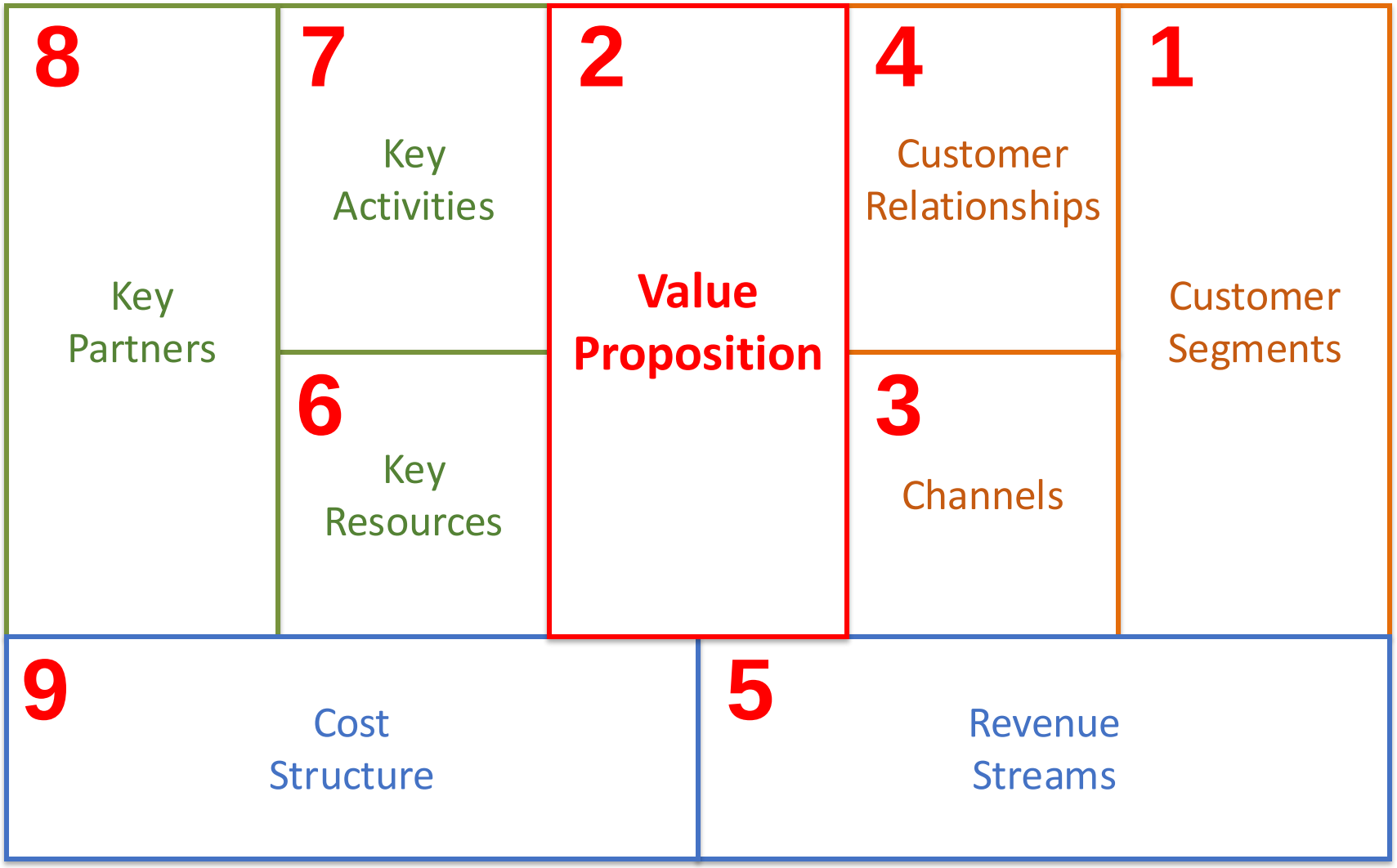
Segmentation

Targeting

Positioning

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Business Model



Source: Alexander Osterwalder & Yves Pigneur, Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers, Wiley, 2010.

Marketing

Marketing "Meeting needs profitably"

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

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."

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

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."

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

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

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Customer Value

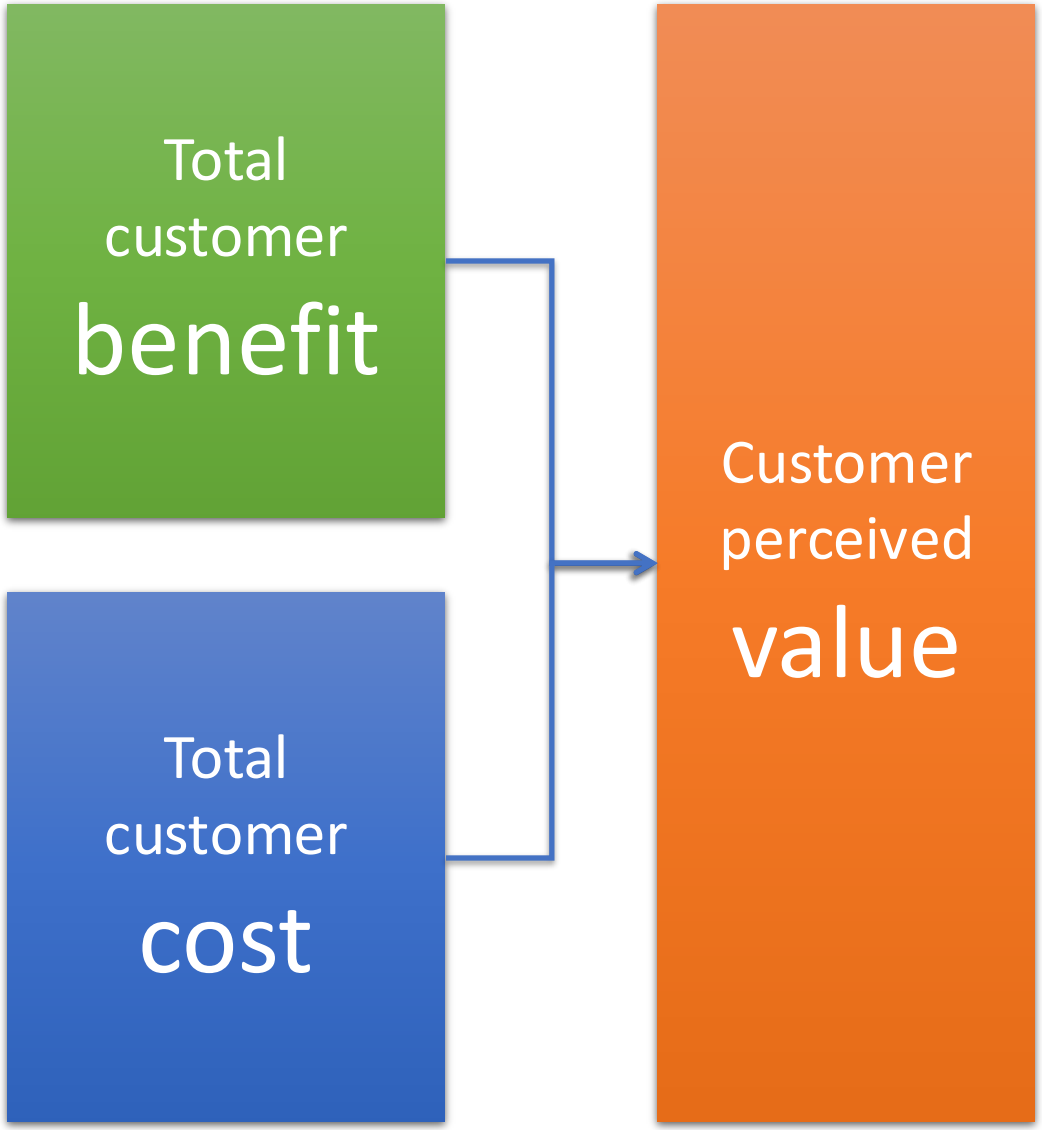
Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Value

the sum of the tangible and intangible benefits and costs

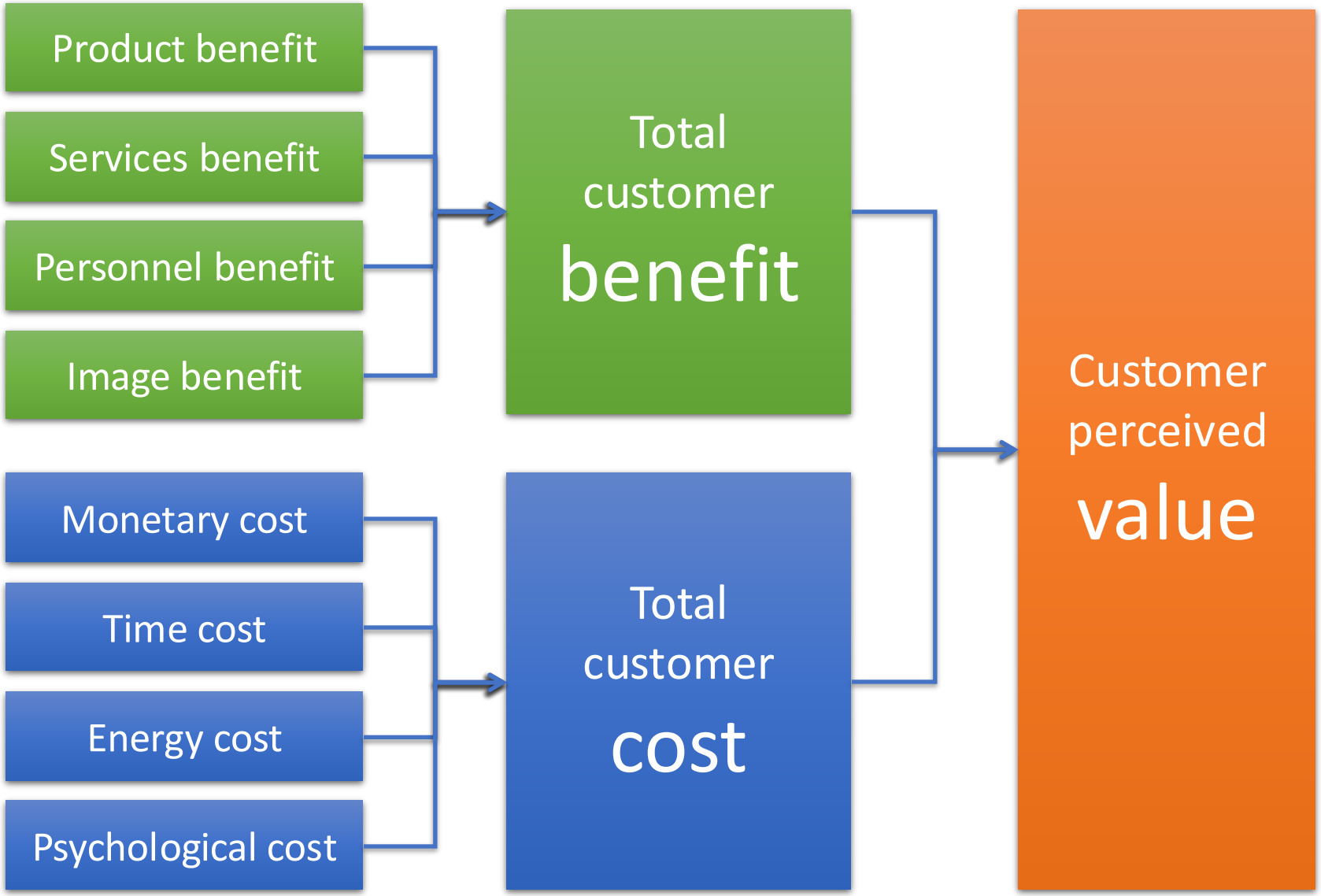
Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Value



Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

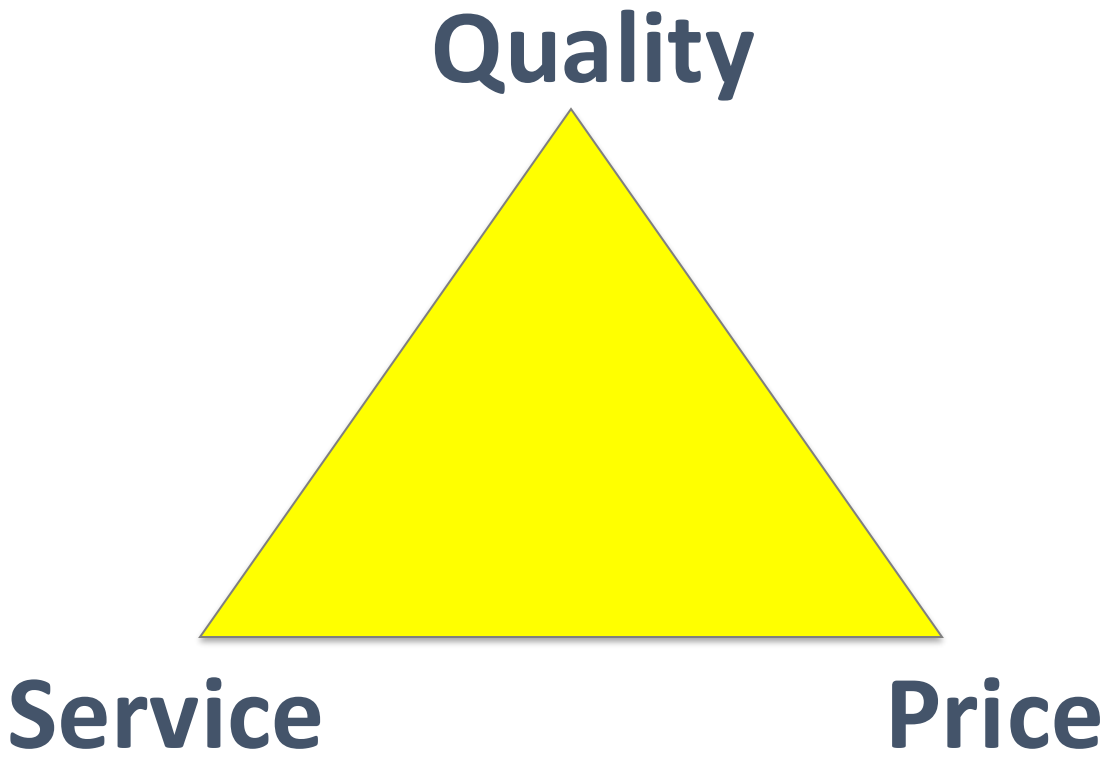
Customer Perceived Value



Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Customer Value Triad

Quality, Service, and Price (qsp)



Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Value and Satisfaction

- **Marketing**
  - **identification, creation, communication, delivery, and monitoring of customer value.**
- **Satisfaction**
  - **a person's judgment of a product's perceived performance in relationship to expectations**

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Building Customer Value, Satisfaction, and Loyalty

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Satisfaction

"a person's feelings of pleasure or disappointment that result from comparing a product's perceived performance (or outcome) to expectations"

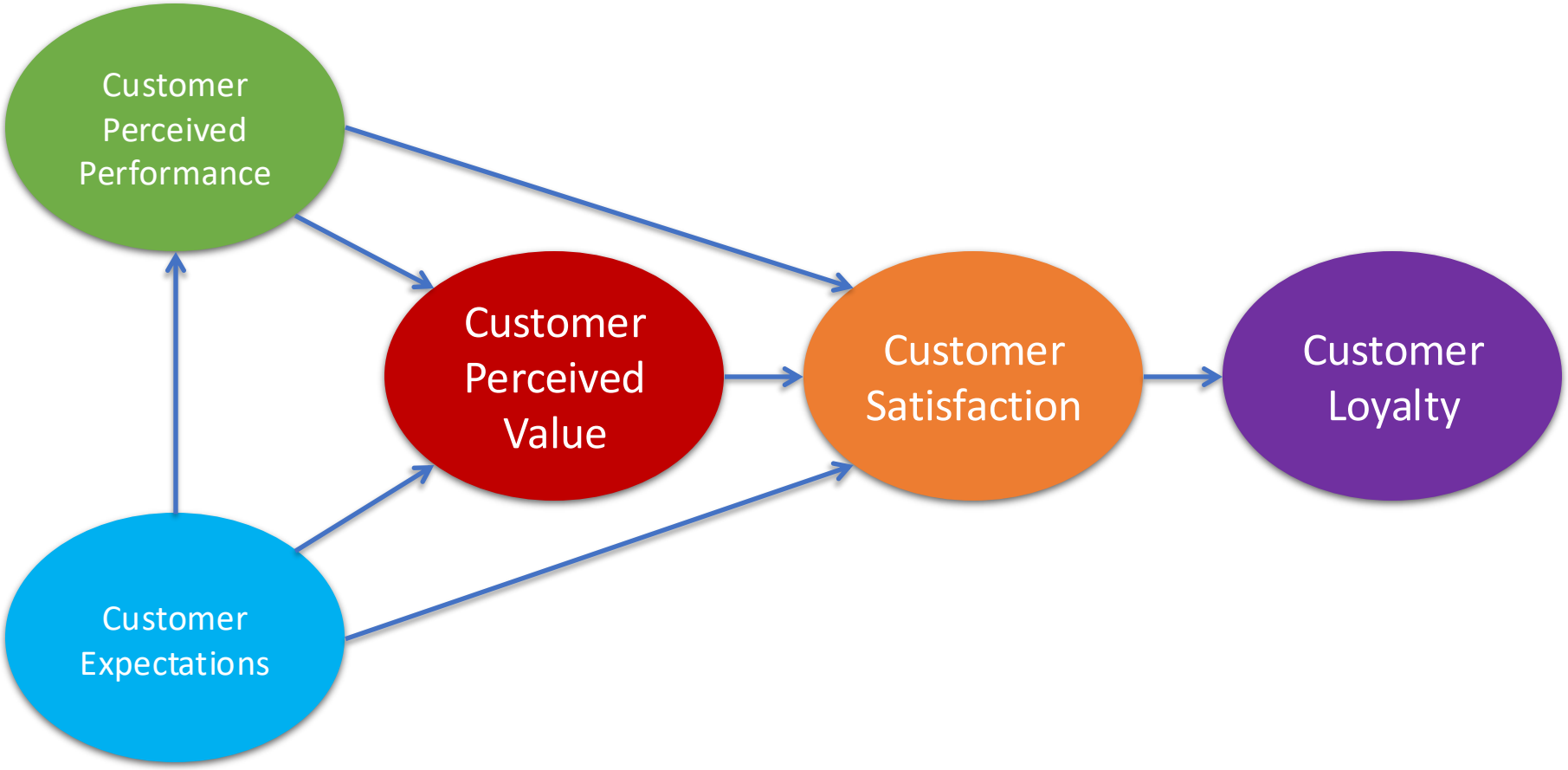
Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Loyalty

"a deeply held commitment to rebuy or repatronize a preferred product or service in the future despite situational influences and marketing efforts having the potential to cause switching behavior."

Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Customer Perceived Value, Customer Satisfaction, and Loyalty



Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Software products

- **Three factors that drive the design of software products**
  - **Business and consumer needs that are not met by current products**
  - **Dissatisfaction with existing business or consumer software products**
  - **Changes in technology that make completely new types of product possible**
- **In the early stage of product development, you are trying to understand, what product features would be useful to users, and what they like and dislike about the products that they use.**

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

Software features

- A feature is a fragment of functionality such as a ‘print’ feature, a ‘change background feature’, a ‘new document’ feature and so on.
- Before you start programming a product, you should aim to create a list of features to be included in your product.
- The feature list should be your starting point for product design and development.

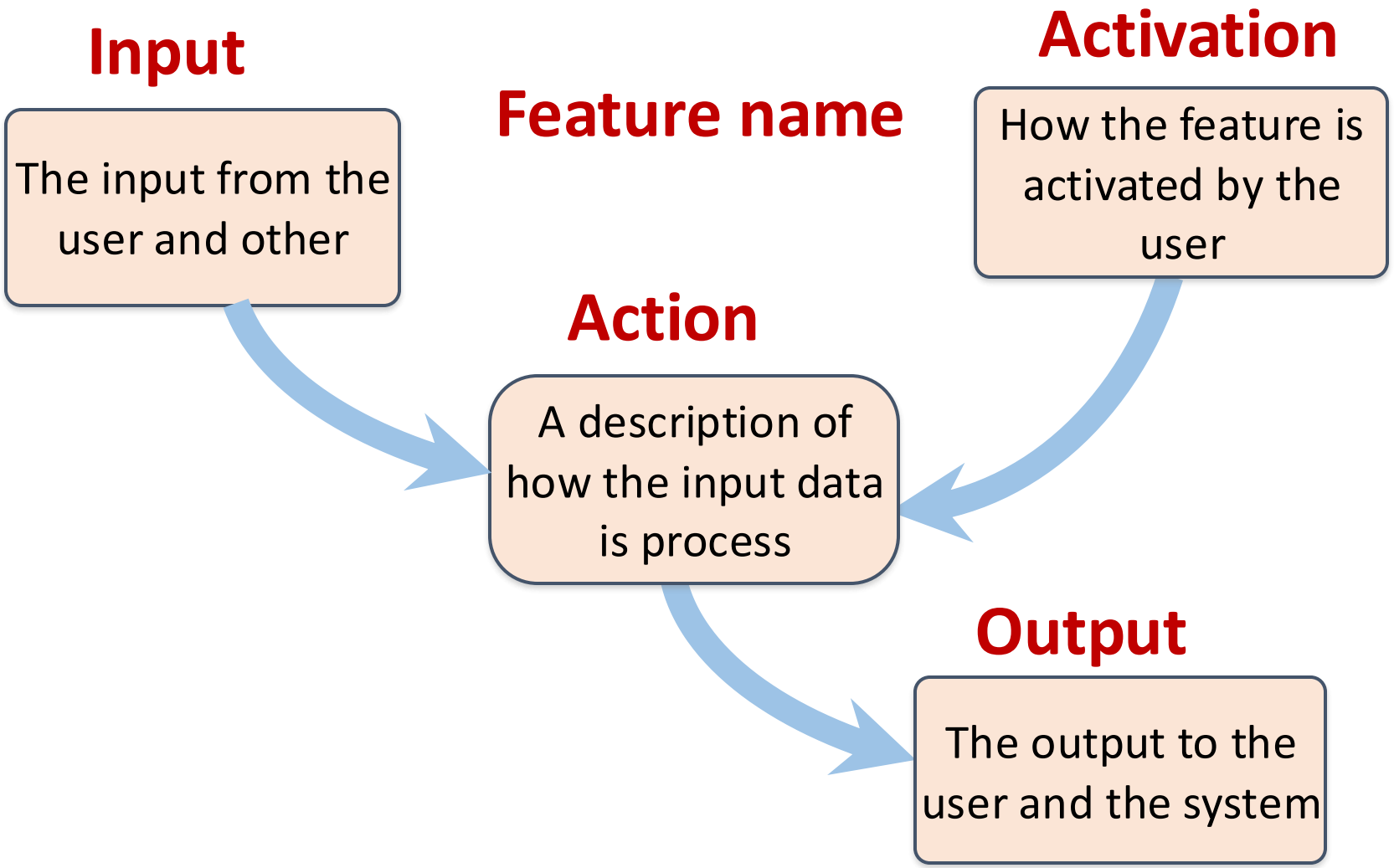
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

User understanding

- It makes sense in any product development to spend time trying to understand the potential users and customers of your product.
- A range of techniques have been developed for understanding the ways that people work and use software.
  - These include user interviews, surveys, ethnography and task analysis.
  - Some of these techniques are expensive and unrealistic for small companies.
- Informal user analysis and discussions, which simply involve asking users about their work, the software that they use, and its strengths and weaknesses are inexpensive and very valuable.

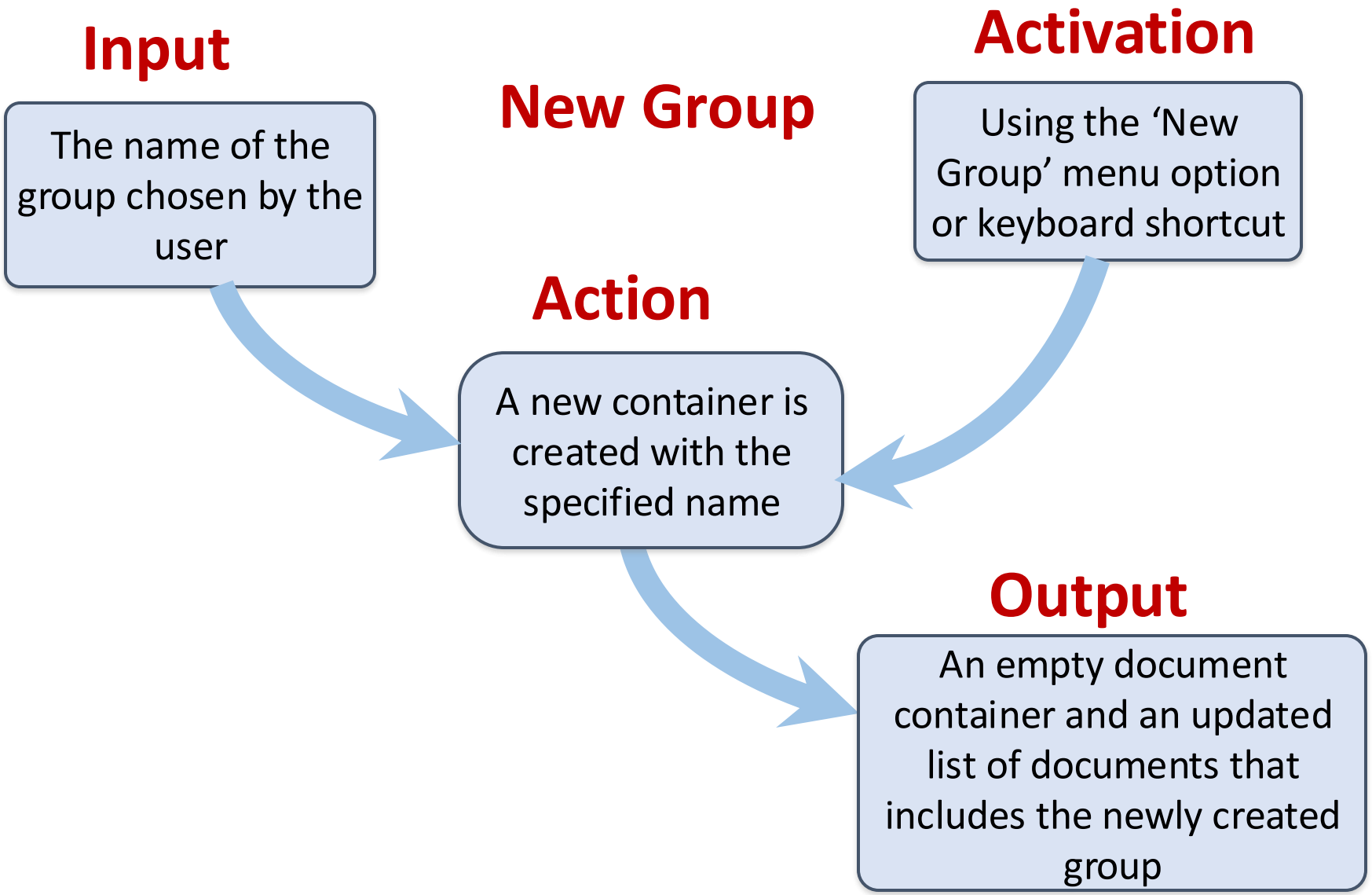
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Feature description



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

The ‘New Group’ feature description



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

Personas

- You need to have an understanding of your potential users to design features that they are likely to find useful and to design a user interface that is suited to them.
- Personas are 'imagined users' where you create a character portrait of a type of user that you think might use your product.
  - For example, if your product is aimed at managing appointments for dentists, you might create a dentist persona, a receptionist persona and a patient persona.
- Personas of different types of user help you imagine what these users may want to do with your software and how it might be used. They help you envisage difficulties that they might have in understanding and using product features.

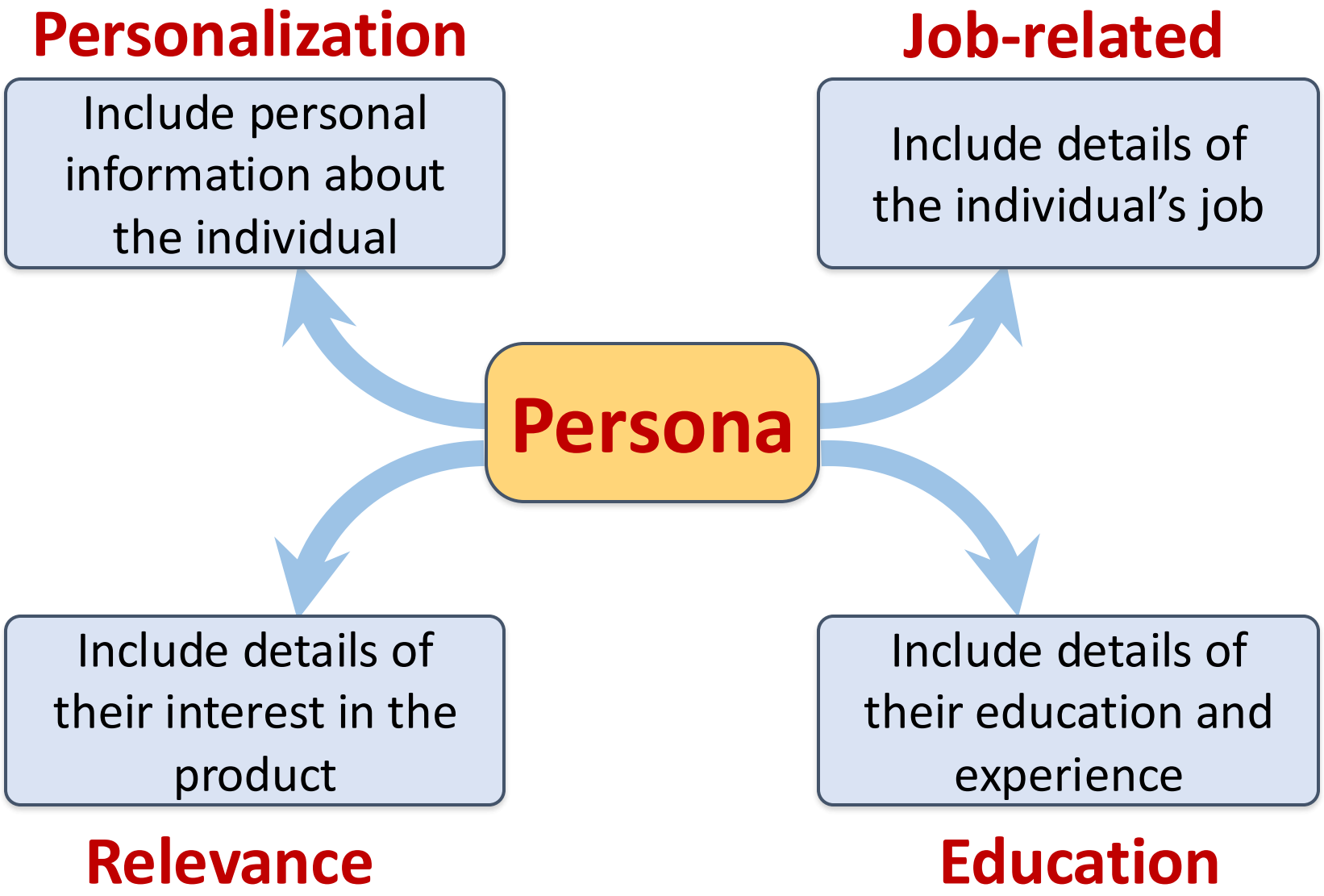
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Persona descriptions

- A persona should 'paint a picture' of a type of product user. They should be relatively short and easy-to-read.
- Describe their background and why they might want to use your product.
- Say something about their educational background and technical skills.
- These help you assess whether or not a software feature is likely to be useful, understandable and usable by typical product users.

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

Persona descriptions



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

Persona benefits

- **Personas help you and other development team members empathize with potential users of the software.**
- **Personas help because they are a tool that allows developers to 'step into the user's shoes'.**
  - **Instead of thinking about what you would do in a particular situation, you can imagine how a persona would behave and react.**
- **Personas can help you check your ideas to make sure that you are not including product features that aren't really needed.**
- **They help you to avoid making unwarranted assumptions, based on your own knowledge, and designing an over-complicated or irrelevant product.**

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

Deriving personas

- **Personas should be based on an understanding of the potential product users, their jobs, their background and their aspirations.**
- **You should study and survey potential users to understand what they want and how they might use the product.**
- **From this data, you can then abstract the essential information about the different types of product user and use this as a basis for creating personas.**
- **Personas that are developed on the basis of limited user information are called proto-personas.**
- **Proto-personas may be created as a collective team exercise using whatever information is available about potential product users. They can never be as accurate as personas developed from detailed user studies, but they are better than nothing.**

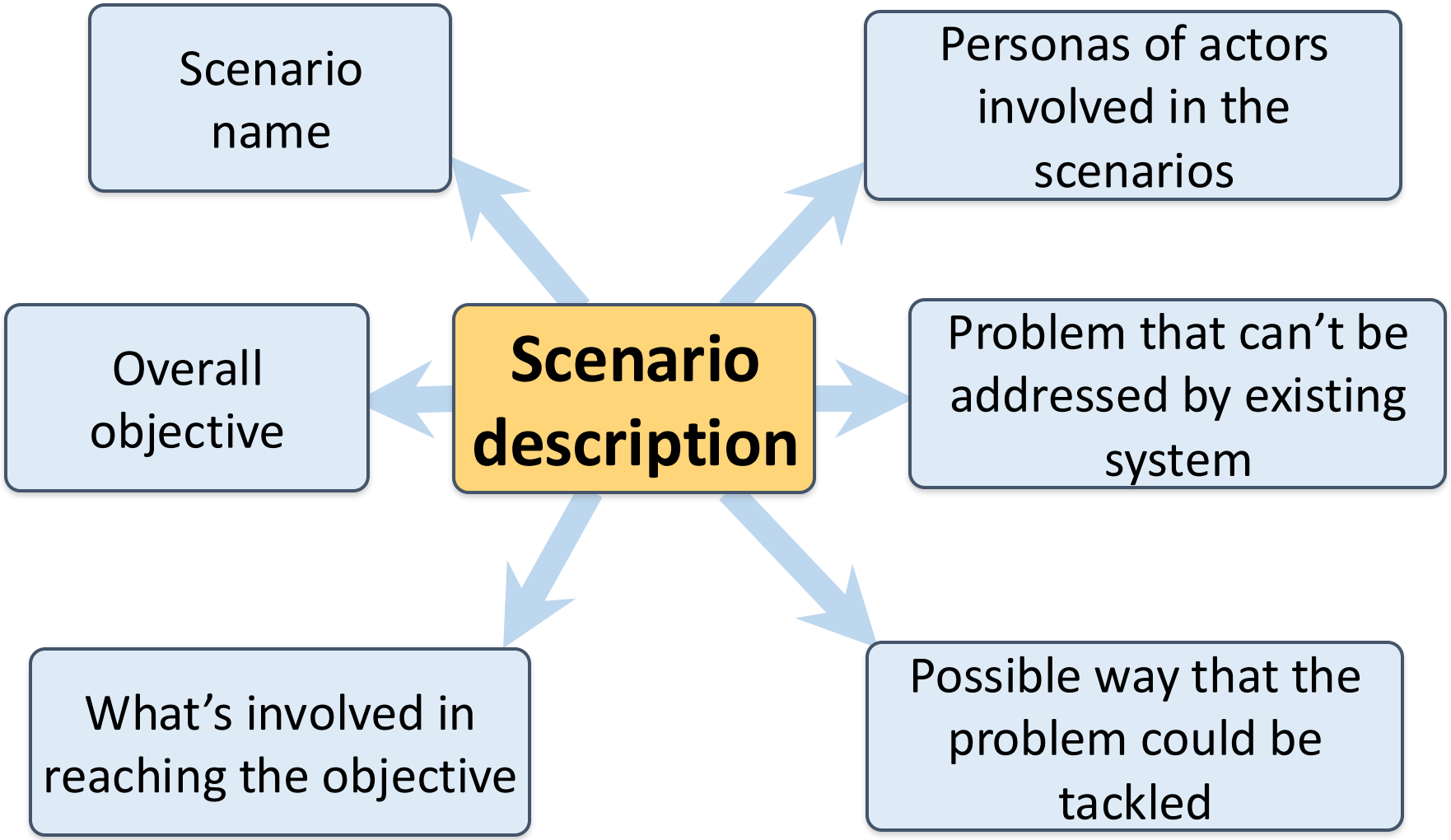
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Scenarios

- A scenario is a narrative that describes how a user, or a group of users, might use your system.
- There is no need to include everything in a scenario – the scenario isn't a system specification.
- It is simply a description of a situation where a user is using your product's features to do something that they want to do.
- Scenario descriptions may vary in length from two to three paragraphs up to a page of text.

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

Elements of a scenario description



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

Writing scenarios

- **Scenarios should always be written from the user's perspective and based on identified personas or real users.**
- **Your starting point for scenario writing should be the personas that you have created. You should normally try to imagine several scenarios from each persona.**
- **Ideally, scenarios should be general and should not include implementation information.**
- **However, describing an implementation is often the easiest way to explain how a task is done.**
- **It is important to ensure that you have coverage of all of the potential user roles when describing a system.**

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

User involvement

- It is easy for anyone to read and understand scenarios, so it is possible to get users involved in their development.
- The best approach is to develop an imaginary scenario based on our understanding of how the system might be used then ask users to explain what you have got wrong.
- They might ask about things they did not understand and suggest how the scenario could be extended and made more realistic.
- Our experience was that users are not good at writing scenarios.
- The scenarios that they created were based on how they worked at the moment. They were far too detailed and the users couldn't easily generalize their experience.

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

User stories

1. **WHO** — **As a** <role>
2. **WHAT** — **I** <want | need> **to** <do something>
3. **WHY** — **so that** <reason>

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

User stories

- As a <role>,
  I <want | need> to <do something>
  - As a teacher,
    I want to tell all members of my group when new information is available
- As a <role>
  I <want | need> to <do something>
  so that <reason>
  - As a teacher,
    I need to be able to report who is attending a class trip
    so that the school maintains the required health and safety records.

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

User stories

- **Scenarios** are **high-level stories** of system use.
  They should describe a sequence of interactions with the system but should not include details of these interactions.
- **User stories** are **finer-grain narratives** that set out in a more detailed and structured way a single thing
  that a user wants from a software system.
  - **As** an author,
    I need a way **to** organize the book
    that I'm writing into chapters and sections.

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

User stories

- This story reflects what has become the standard format of a user story:
- As a <role>, I <want | need> to <do something>
  - As a teacher, I want to tell all members of my group when new information is available
- A variant of this standard format adds a justification for the action:
  - As a <role> I <want | need> to <do something> so that <reason>
    - As a teacher,
      I need to be able to report who is attending a class trip
      so that the school maintains the required health and safety records.

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

User stories in planning

- An important use of user stories is in planning.
  - Many users of the Scrum method represent the product backlog as a set of user stories.
- User stories should focus on
  a clearly defined system feature or
  aspect of a feature
  that can be implemented
  within a single sprint.

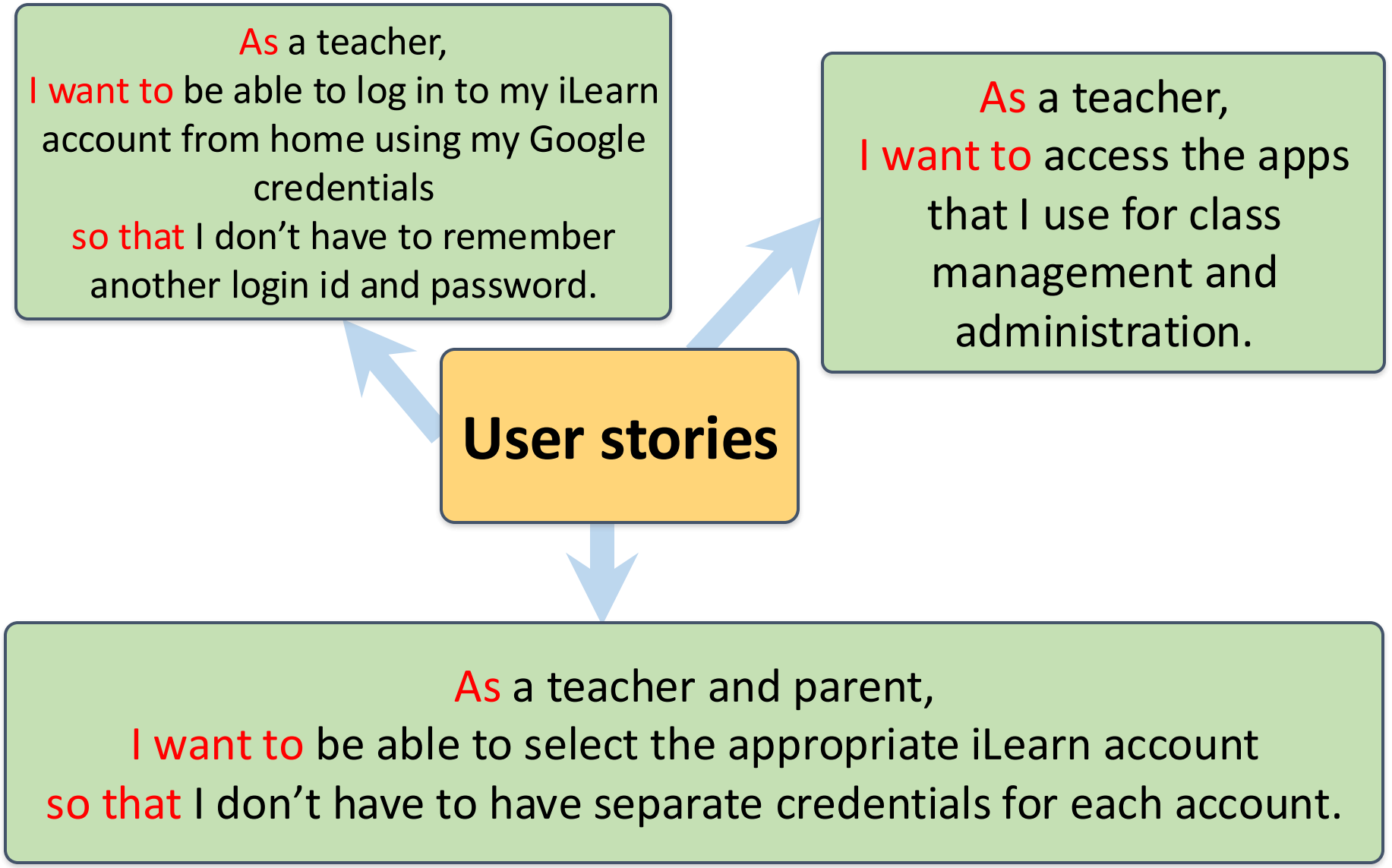
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

User stories in planning

- If the story is about a more complex feature that might take several sprints to implement, then it is called an epic.
  - As a system manager, I need a way to backup the system and restore either individual applications, files, directories or the whole system.
  - There is a lot of functionality associated with this user story. For implementation, it should be broken down into simpler stories with each story focusing on a single aspect of the backup system.

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

User stories from Emma's scenario



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

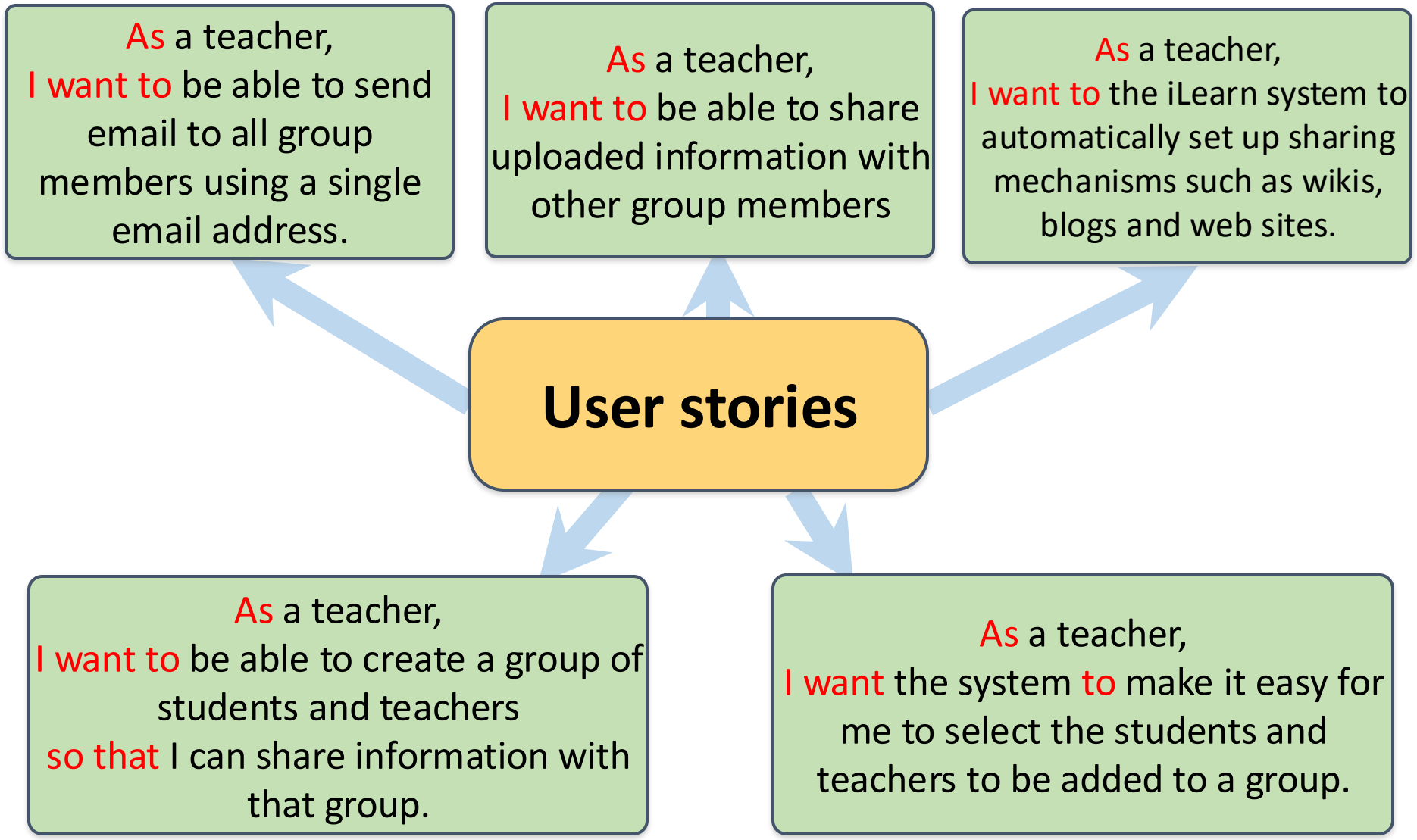
Feature description using user stories

- **Stories can be used to describe features in your product that should be implemented.**

- **Each feature can have a set of associated stories that describe how that feature is used.**

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

User stories describing the Groups feature



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

Stories and scenarios

- As you can express all of the functionality described in a scenario as user stories, do you really need scenarios?'
- Scenarios are more natural and are helpful for the following reasons:
  - Scenarios read more naturally because they describe what a user of a system is actually doing with that system. People often find it easier to relate to this specific information rather than the statement of wants or needs set out in a set of user stories.
  - If you are interviewing real users or are checking a scenario with real users, they don't talk in the stylized way that is used in user stories. People relate better to the more natural narrative in scenarios.
  - Scenarios often provide more context - information about what the user is trying to do and their normal ways of working. You can do this in user stories, but it means that they are no longer simple statements about the use of a system feature.

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

Feature identification

- Your aim in the initial stage of product design should be to create a list of features that define your product.
- A feature is a way of allowing users to access and use your product's functionality so the feature list defines the overall functionality of the system.
- Features should be independent, coherent and relevant.

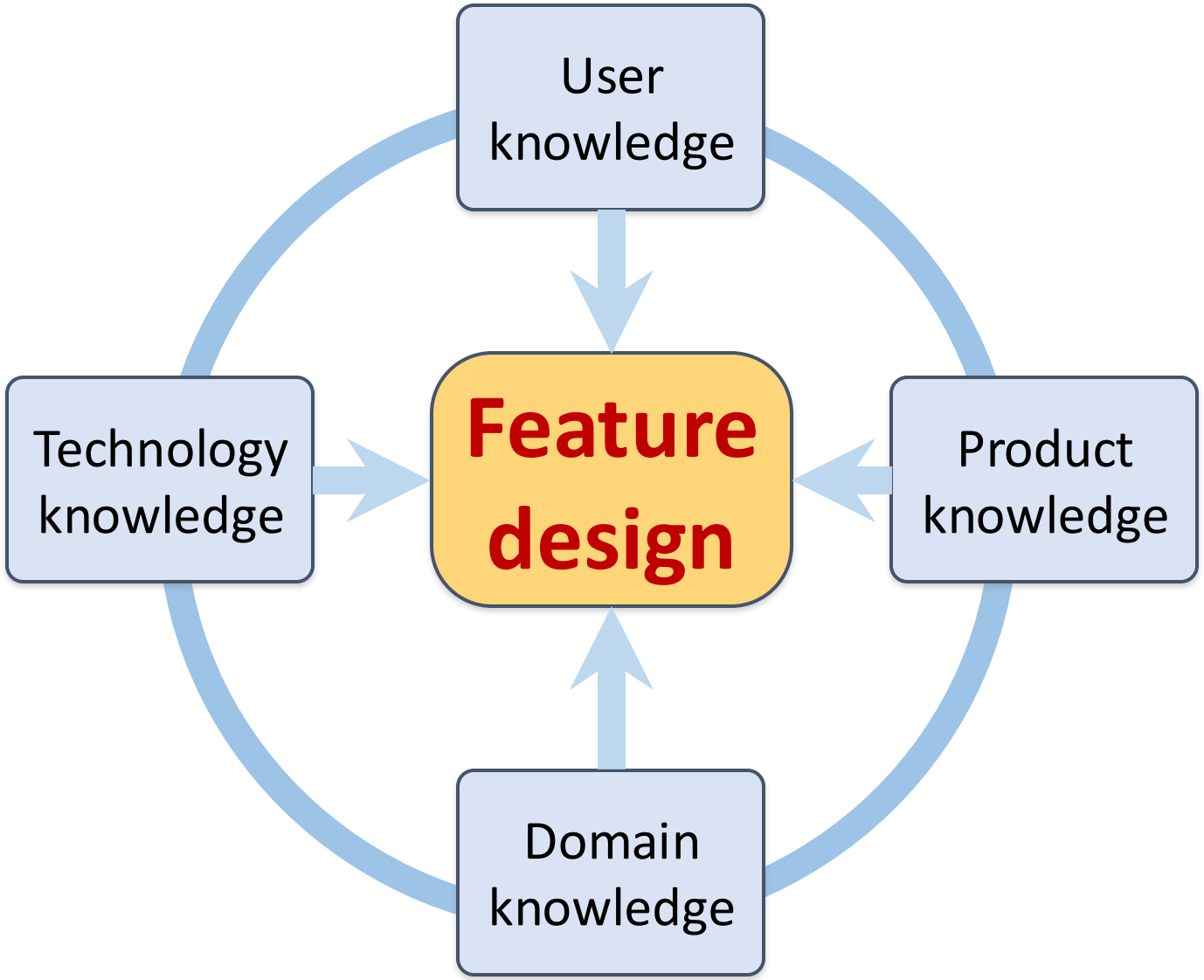
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Feature identification

- **Features should be independent, coherent and relevant:**
  - **Independence**
    Features should not depend on how other system features are implemented and should not be affected by the order of activation of other features.
  - **Coherence**
    Features should be linked to a single item of functionality.
    They should not do more than one thing and they should never have side-effects.
  - **Relevance**
    Features should reflect the way that users normally carry out some task.
    They should not provide obscure functionality that is hardly ever required.

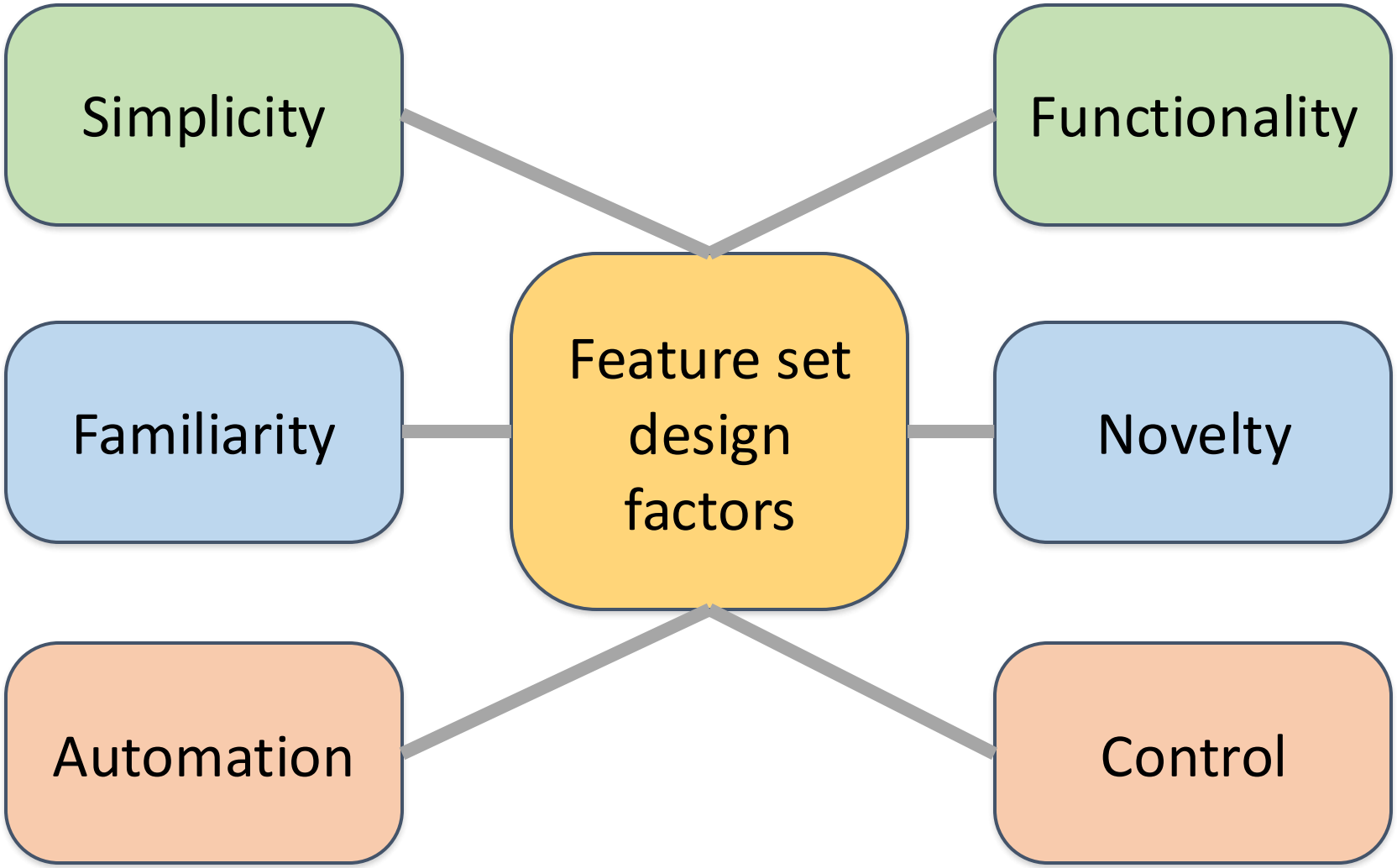
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Feature design



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

Factors in feature set design



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

Feature trade-offs

- **Simplicity and functionality**
  - **You need to find a balance between providing a simple, easy-to-use system and including enough functionality to attract users with a variety of needs.**
- **Familiarity and novelty**
  - **Users prefer that new software should support the familiar everyday tasks that are part of their work or life.**
    To encourage them to adopt your system, you need to find a balance between familiar features and new features that convince users that your product can do more than its competitors.
- **Automation and control**
  - **Some users like automation, where the software does things for them. Others prefer to have control.**
    You have to think carefully about what can be automated, how it is automated and how users can configure the automation so that the system can be tailored to their preferences.

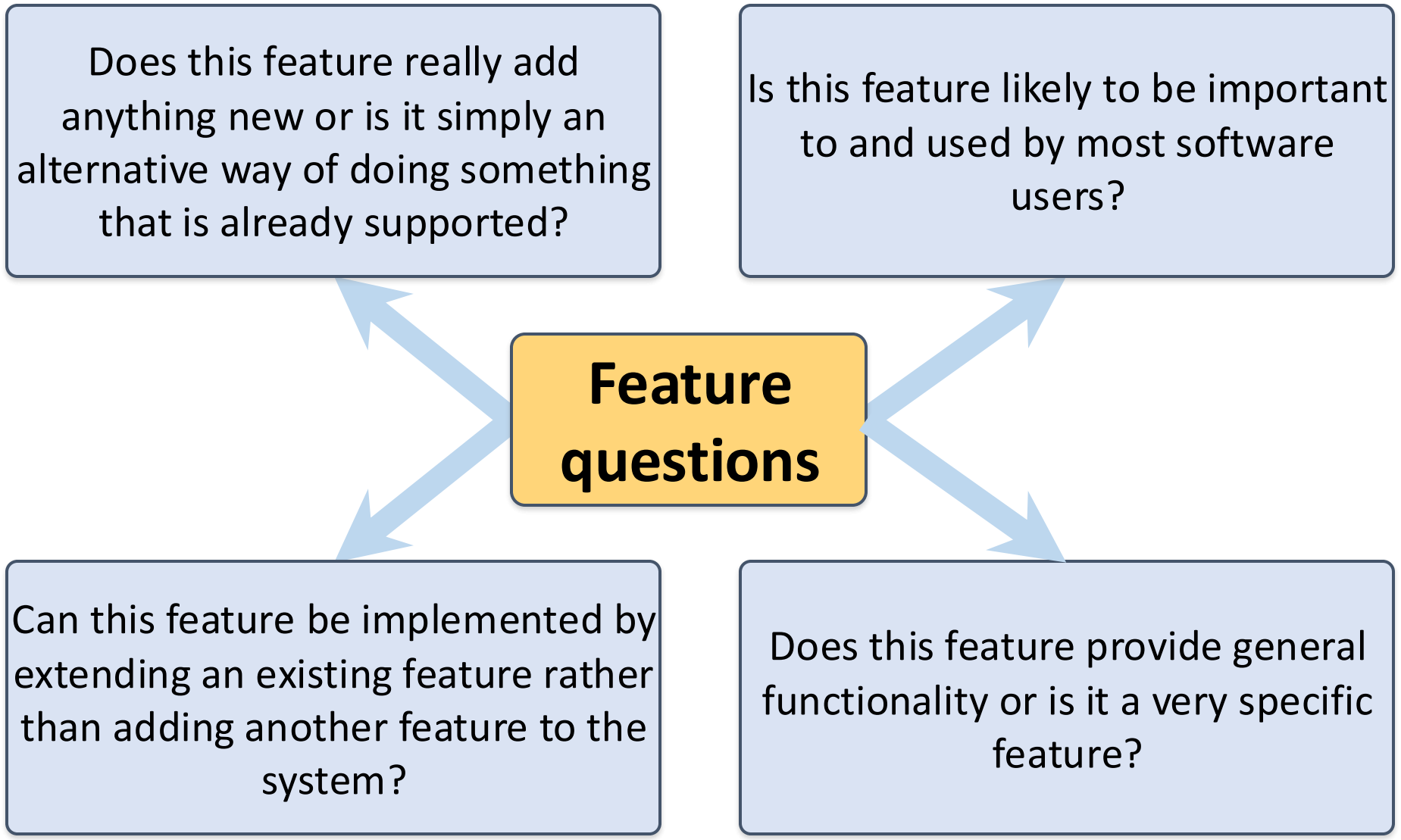
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Feature creep

- **Feature creep occurs when new features are added in response to user requests without considering whether or not these features are generally useful or whether they can be implemented in some other way.**
- **Too many features make products hard to use and understand**
- **There are three reasons why feature creep occurs:**
  - **Product managers are reluctant to say 'no' when users ask for specific features.**
  - **Developers try to match features in competing products.**
  - **The product includes features to support both inexperienced and experienced users.**

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

Avoiding feature creep



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

Feature derivation

- **Features can be identified directly from the product vision or from scenarios.**
- **You can highlight phrases in narrative description to identify features to be included in the software.**
  - **You should think about the features needed to support user actions, identified by active verbs, such as use and choose.**

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

The iLearn system vision

- **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 itself, 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

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

Features from the product vision

- **A feature that allows users to access and use existing web-based resources;**
- **A feature that allows the system to exist in multiple different instantiations;**
- **A feature that allows user configuration of the system to create a specific instantiation.**

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

Feature description using user stories

- Description
  - As a system manager, I want to create and configure an iLearn environment by adding and removing services to/from that environment so that I can create environments for specific purposes.
  - As a system manager, I want to set up sub-environments that include a subset of services that are included in another environment.
  - As a system manager, I want to assign administrators to created environments.
  - As a system manager, I want to limit the rights of environment administrators so that they cannot accidentally or deliberately disrupt the operation of key services.
  - As a teacher, I want to be able to add services that are not integrated with the iLearn authentication system.
- Constraints
  - The use of some tools may be limited for license reasons so there may be a need to access license management tools during configuration.
- Comments
  - Based on Elena's and Jack's scenarios.

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

Innovation and feature identification

- **Scenarios and user stories should always be your starting point for identifying product features.**
- **Scenarios tell you how users work at the moment. They don't show how they might change their way of working if they had the right software to support them.**
- **Stories and scenarios are 'tools for thinking' and they help you gain an understanding of how your software might be used. You can identify a feature set from stories and scenarios.**
- **User research, on its own, rarely helps you innovate and invent new ways of working.**
- **You should also think creatively about alternative or additional features that help users to work more efficiently or to do things differently.**

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

Summary

- A **software product feature** is a fragment of functionality that implements something that a user may need or want when using the product.
- The **first stage** of product development is to identify the **list of product features** in which you identify each feature and give a brief description of its functionality.
- **Personas** are '**imagined users**' where you create a character portrait of a type of user that you think might use your product.

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

Summary

- A persona description should 'paint a picture' of a typical product user. It should describe their educational background, technology experience and why they might want to use your product.
- A scenario is a narrative that describes a situation where a user is accessing product features to do something that they want to do.

Summary

- **Scenarios** should always be written from the user's perspective and should be based on identified personas or real users.
- **User stories** are finer-grain narratives that set out, in a structured way, something that a user wants from a software system.

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

Summary

- **User stories may be used as a way of extending and adding detail to a scenario or as part of the description of system features.**
- **The key influences in feature identification and design are user research, domain knowledge, product knowledge, and technology knowledge.**
- **You can identify features from scenarios and stories by highlighting user actions in these narratives and thinking about the features that you need to support these actions.**

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

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.