軟體工程
(Software Engineering)
DevOps和程式碼管理：
程式碼管理和DevOps自動化
(DevOps and Code Management: Code management and DevOps automation)

1091SE12
MBA, IM, NTPU (M5118) (Fall 2020)
Tue 2, 3, 4 (9:10-12:00) (B8F40)

Min-Yuh Day
戴敏育
Associate Professor
副教授
Institute of Information Management, National Taipei University
國立臺北大學 資訊管理研究所
https://web.ntpu.edu.tw/~myday

2020-12-29

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2020/09/15	軟體工程概論 (Introduction to Software Engineering)
2	2020/09/22	軟體產品與專案管理：軟體產品管理，原型設計 (Software Products and Project Management: Software product management and prototyping)
3	2020/09/29	敏捷軟體工程：敏捷方法、Scrum、極限程式設計 (Agile Software Engineering: Agile methods, Scrum, and Extreme Programming)
4	2020/10/06	功能、場景和故事 (Features, Scenarios, and Stories)
5	2020/10/13	軟體架構：架構設計、系統分解、分散式架構 (Software Architecture: Architectural design, System decomposition, and Distribution architecture)
6	2020/10/20	軟體工程個案研究 I (Case Study on Software Engineering I)

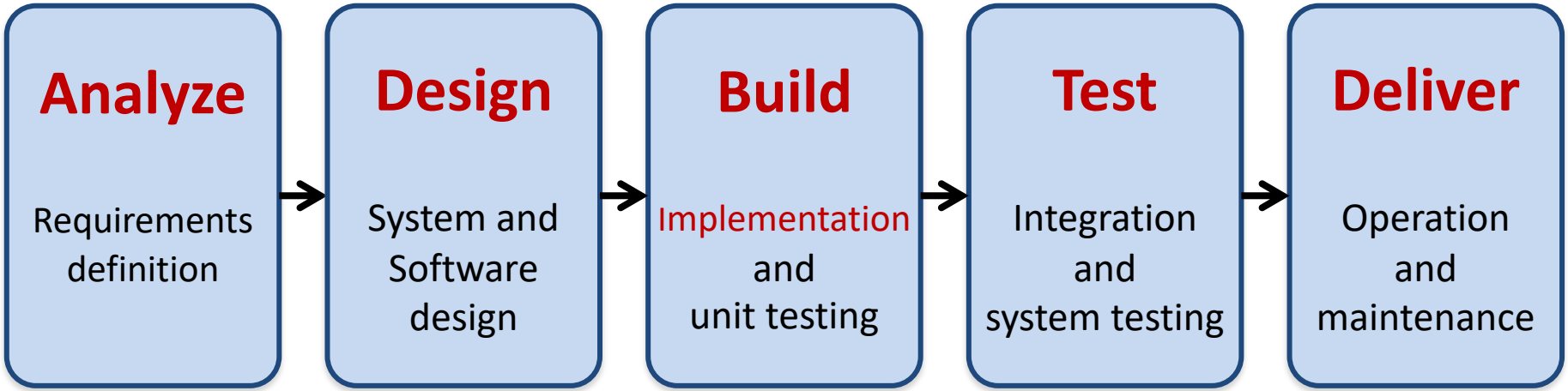
課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2020/10/27	基於雲的軟體：虛擬化和容器、軟體即服務 (Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service)
8	2020/11/03	雲端運算與雲軟體架構 (Cloud Computing and Cloud Software Architecture)
9	2020/11/10	期中報告 (Midterm Project Report)
10	2020/11/17	微服務架構：RESTful服務、服務部署 (Microservices Architecture: RESTful services, Service deployment)
11	2020/11/24	軟體工程產業實務 (Industry Practices of Software Engineering)
12	2020/12/01	安全和隱私 (Security and Privacy)

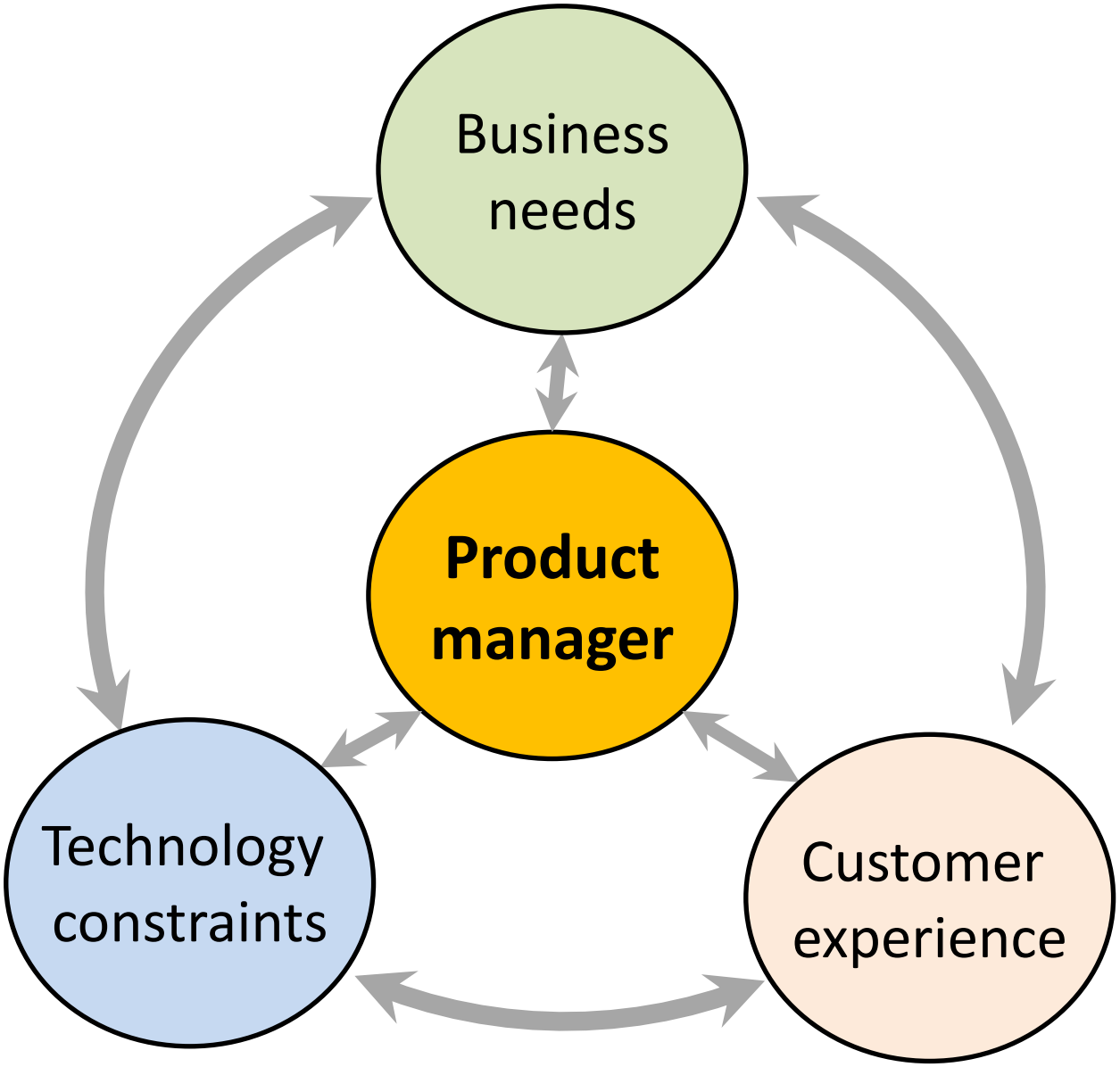
課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2020/12/08	軟體工程個案研究 II (Case Study on Software Engineering II)
14	2020/12/15	可靠的程式設計 (Reliable Programming)
15	2020/12/22	測試：功能測試、測試自動化、測試驅動的開發、程式碼審查 (Testing: Functional testing, Test automation, Test-driven development, and Code reviews)
16	2020/12/29	DevOps和程式碼管理：程式碼管理和DevOps自動化 (DevOps and Code Management: Code management and DevOps automation)
17	2021/01/05	期末報告 I (Final Project Report I)
18	2021/01/12	期末報告 II (Final Project Report I)

Software Engineering and Project Management

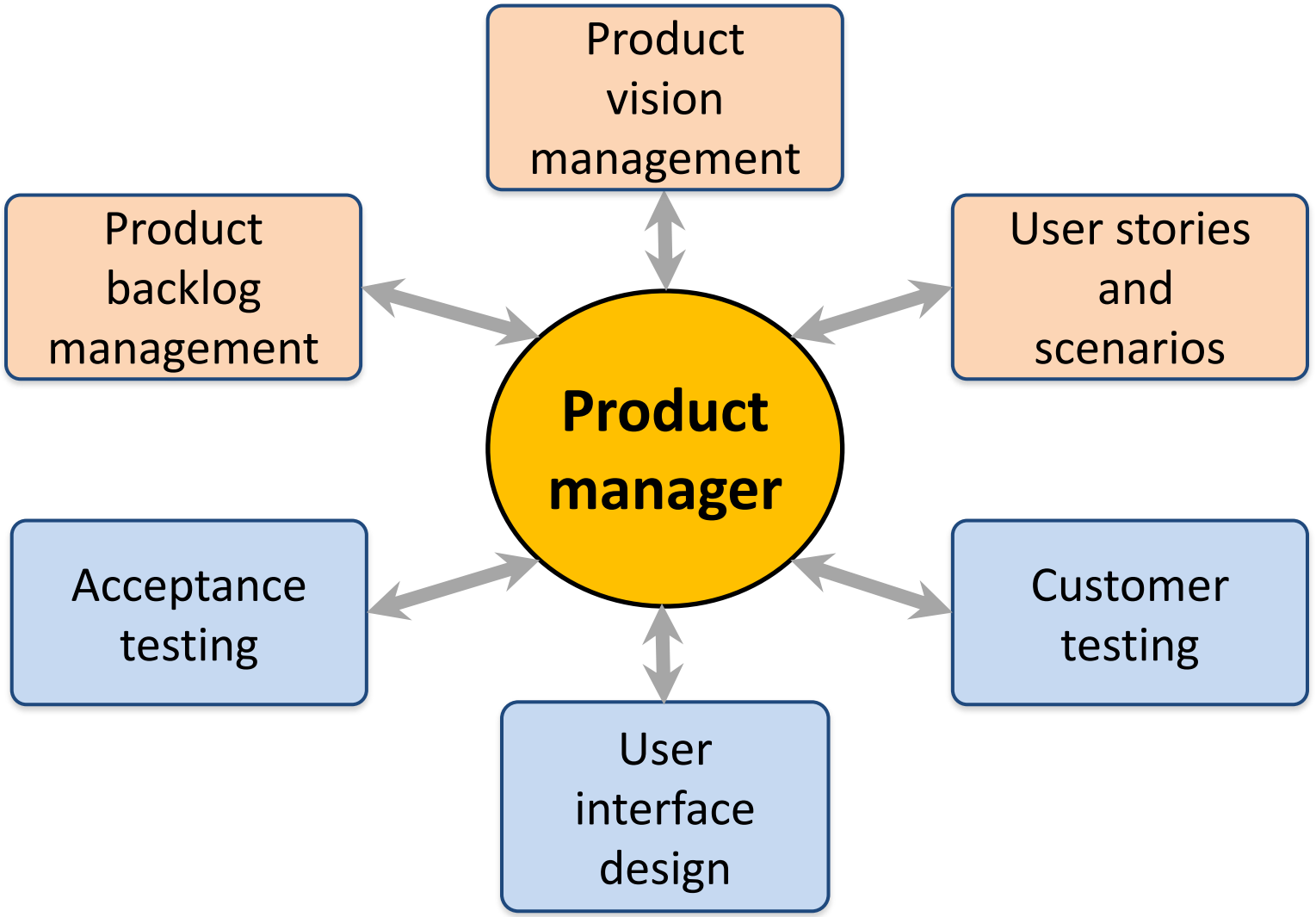


Product management concerns



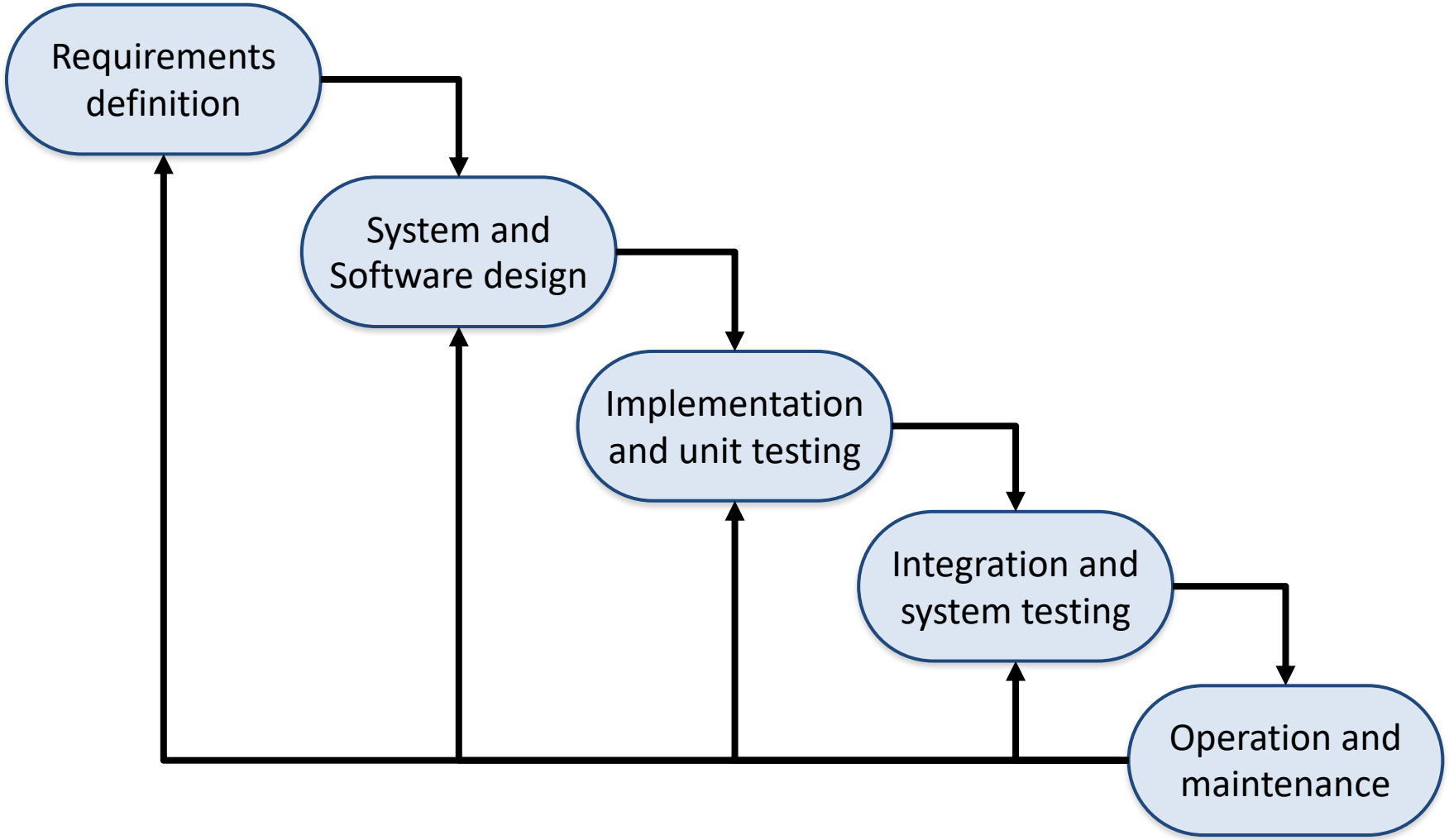
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Technical interactions of product managers



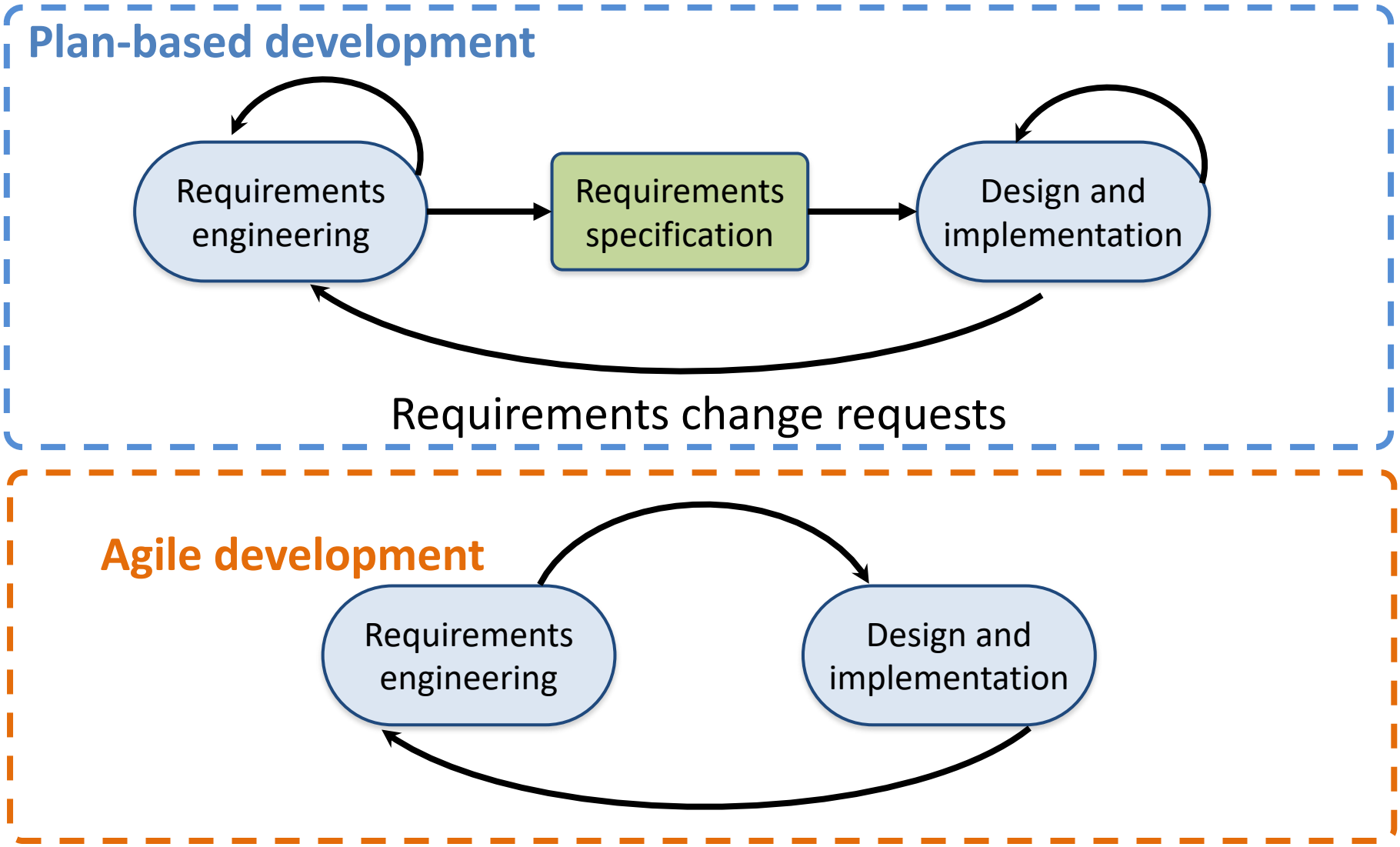
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software Development Life Cycle (SDLC)
The waterfall model



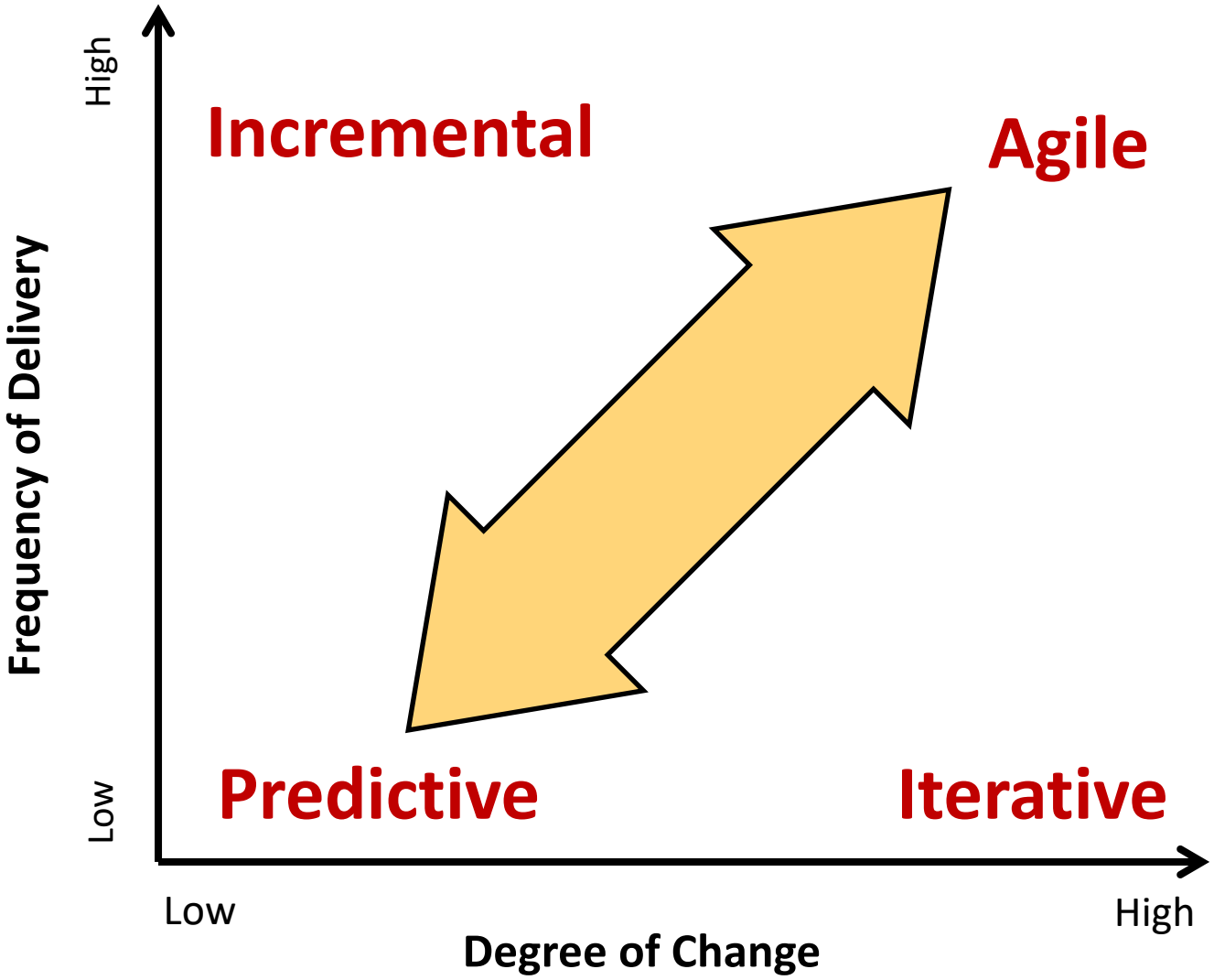
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Plan-based and Agile development



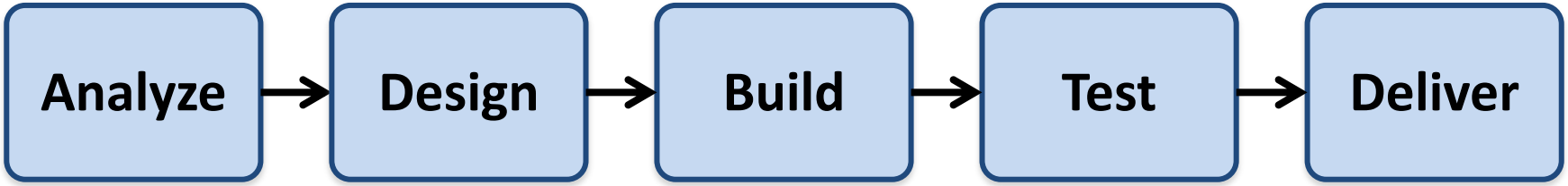
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

The Continuum of Life Cycles



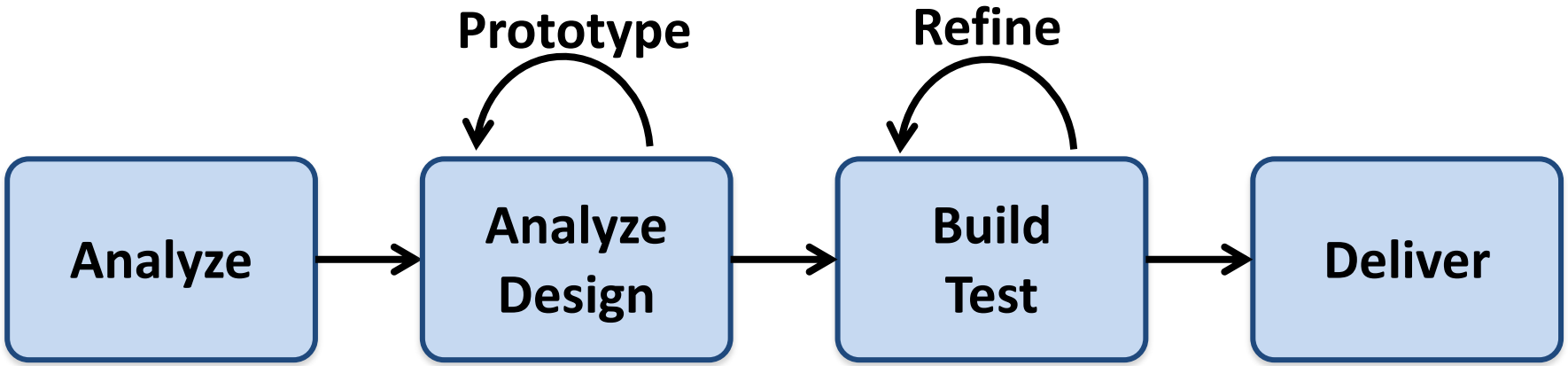
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Predictive Life Cycle



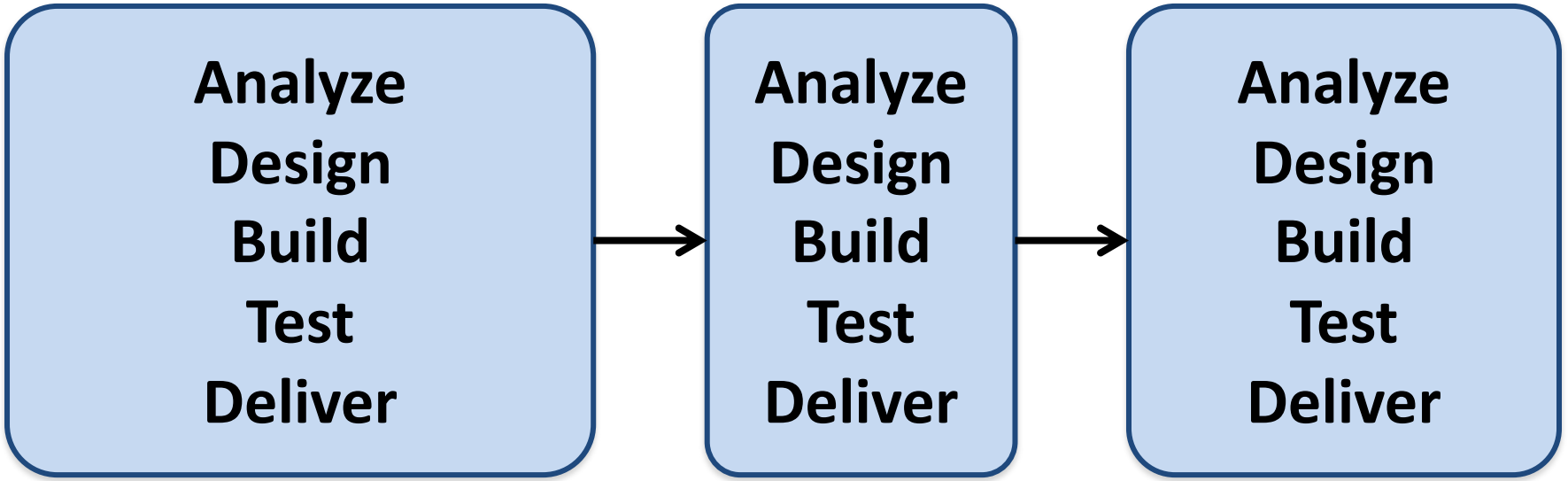
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

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

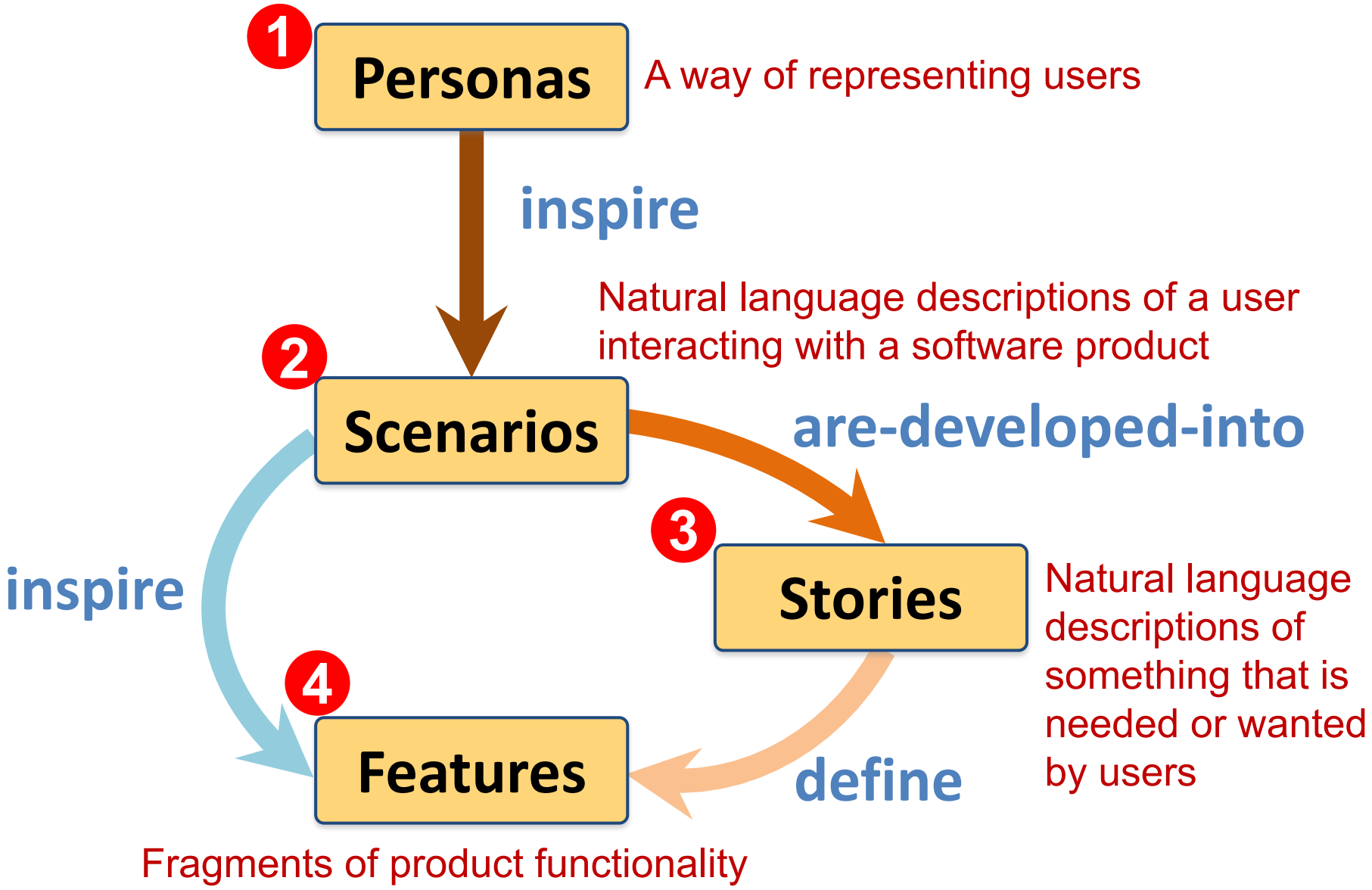
Requirements **Analysis** **Design** **Build** **Test**	**Requirements** **Analysis** **Design** **Build** **Test**	**Requirements** **Analysis** **Design** **Build** **Test**	**Requirements** **Analysis** **Design** **Build** **Test**	**Repeat** **as needed** **…**	**Requirements** **Analysis** **Design** **Build** **Test**	**Requirements** **Analysis** **Design** **Build** **Test**

Flow-Based Agile

Requirements **Analysis** **Design** **Build** **Test** the number of features in the WIP limit	**Requirements** **Analysis** **Design** **Build** **Test** the number of features in the WIP limit	**Requirements** **Analysis** **Design** **Build** **Test** the number of features in the WIP limit	**Repeat** **as needed** **…**	**Requirements** **Analysis** **Design** **Build** **Test** the number of features in the WIP limit	**Requirements** **Analysis** **Design** **Build** **Test** the number of features in the WIP limit

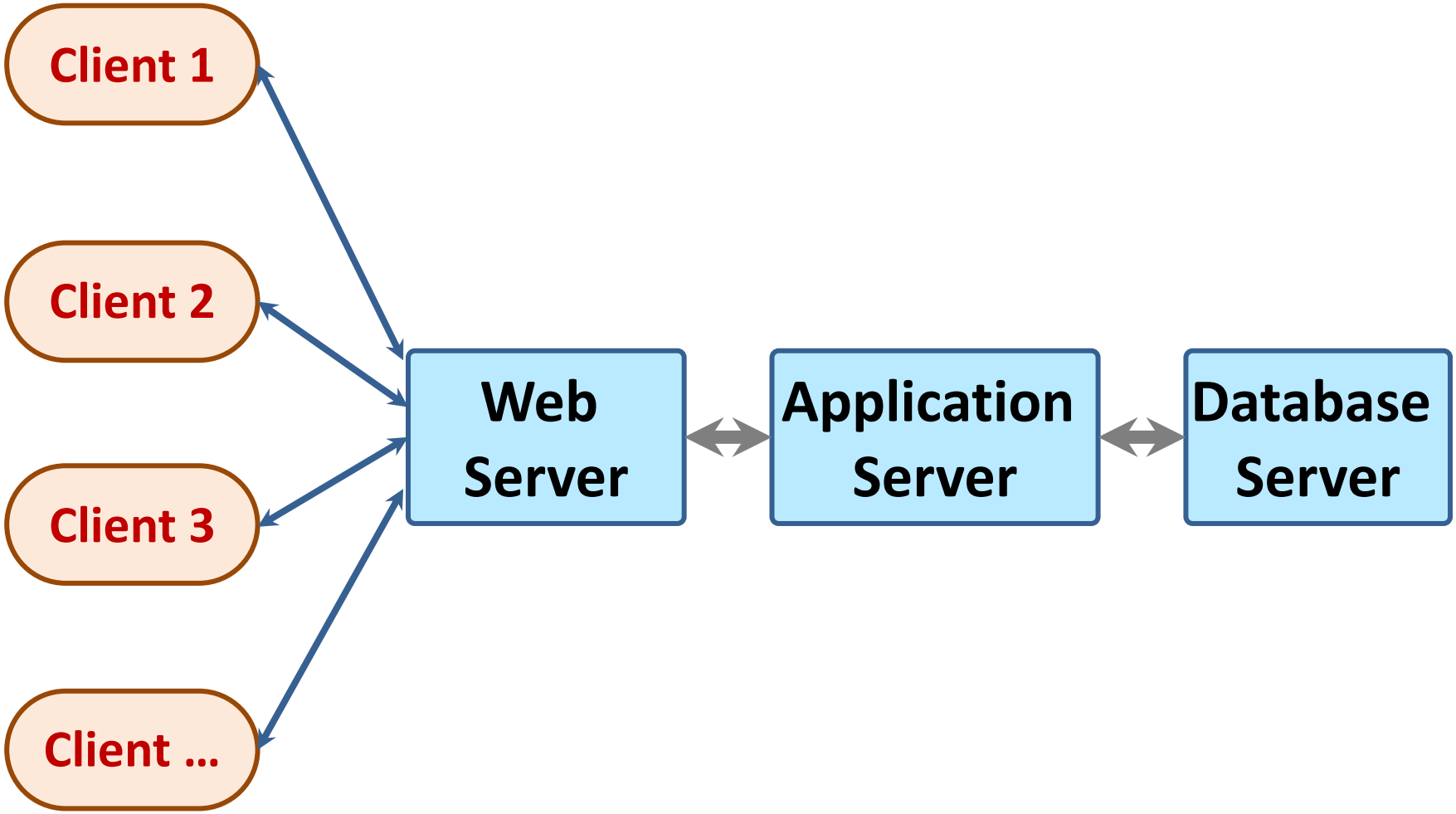
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

From personas to features



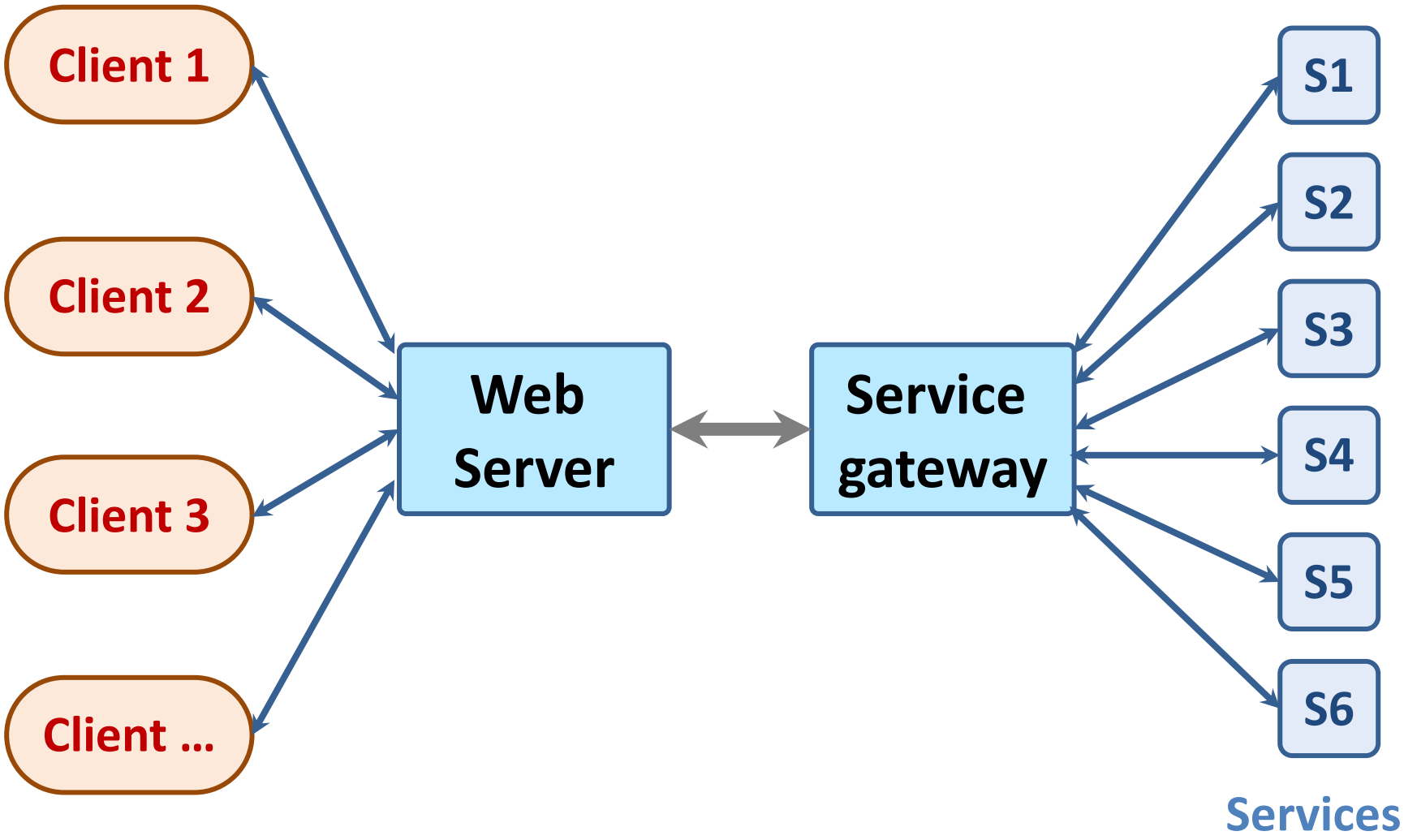
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Multi-tier client-server architecture

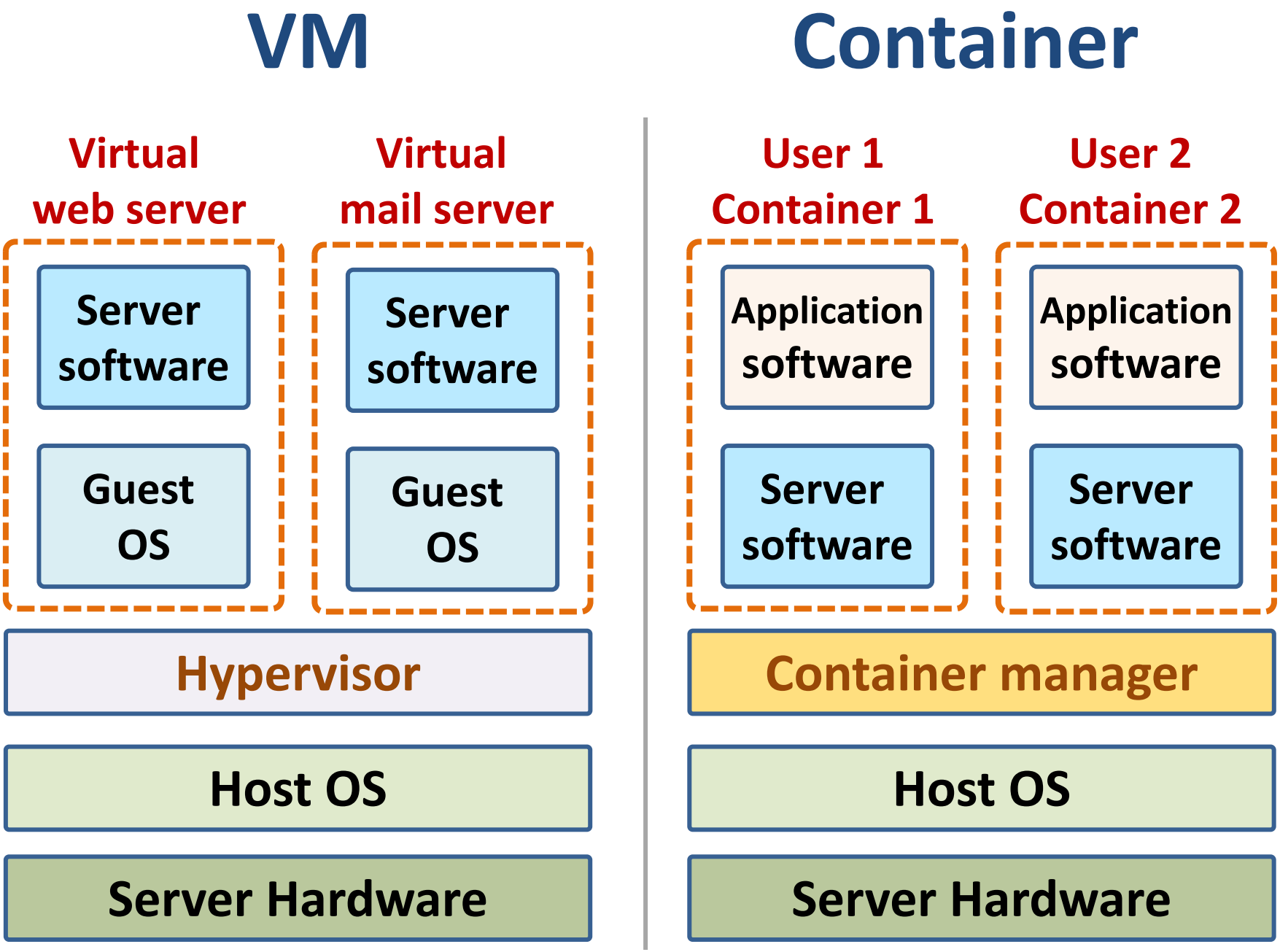


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Service-oriented Architecture

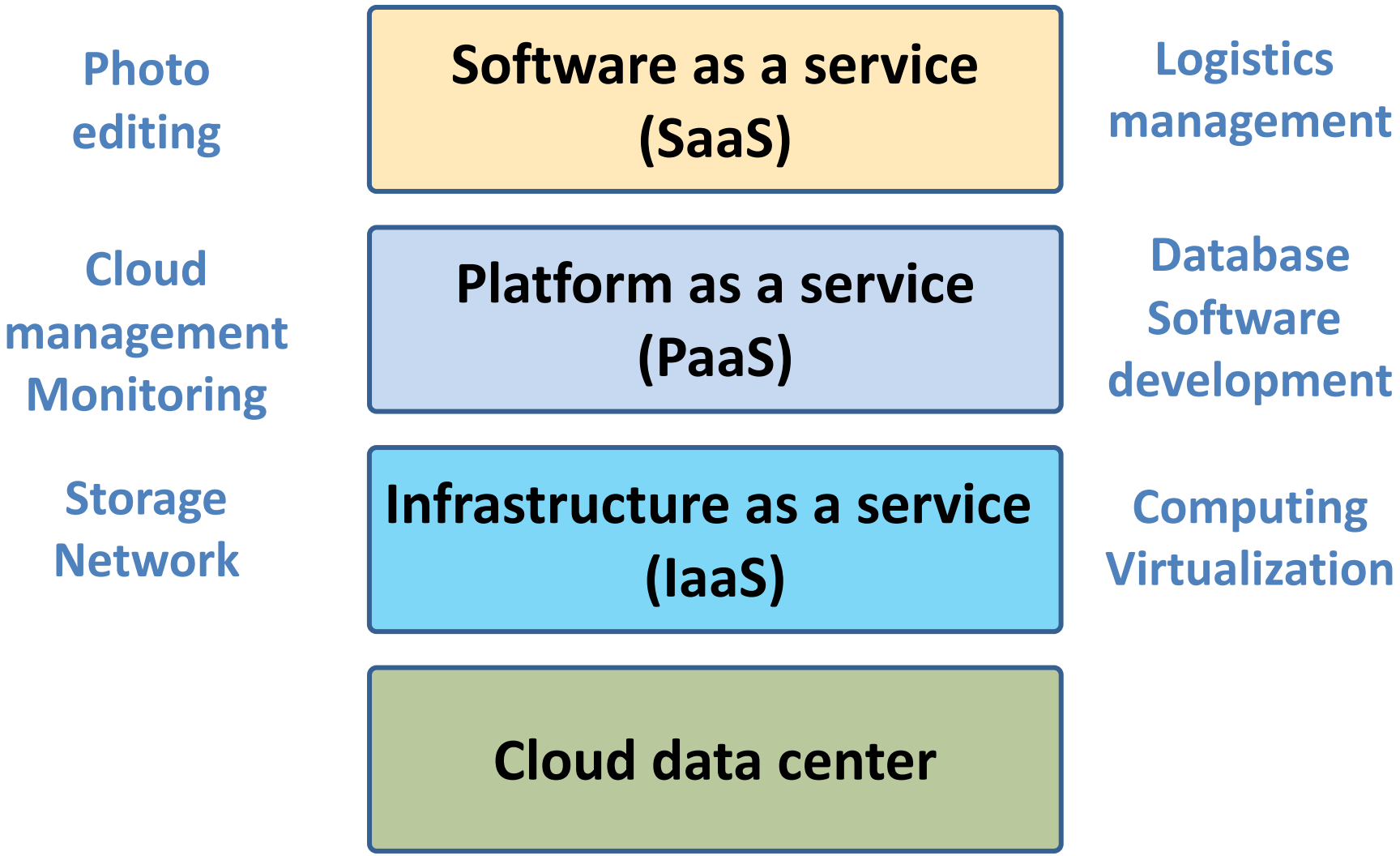


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.



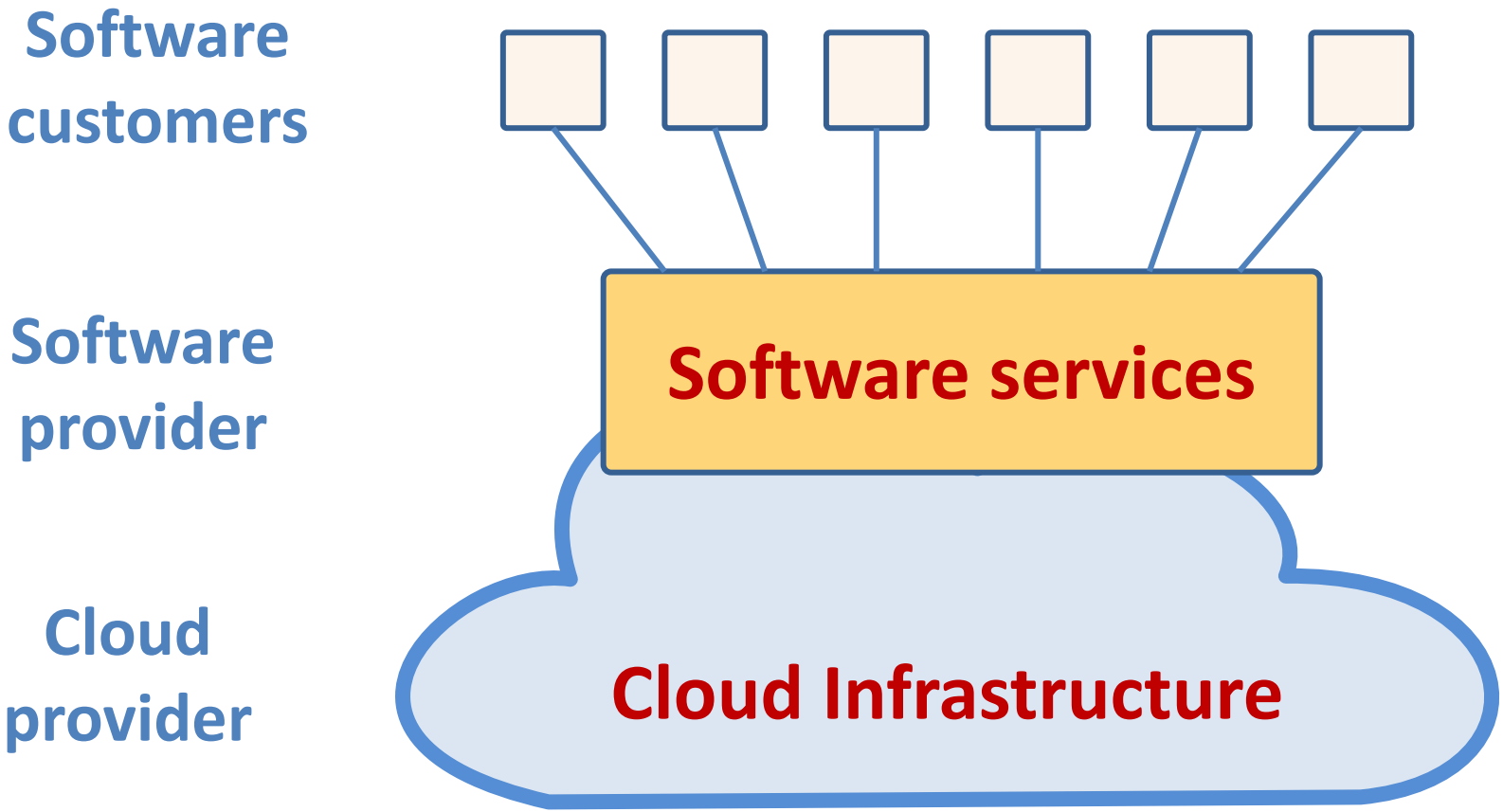
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Everything as a service



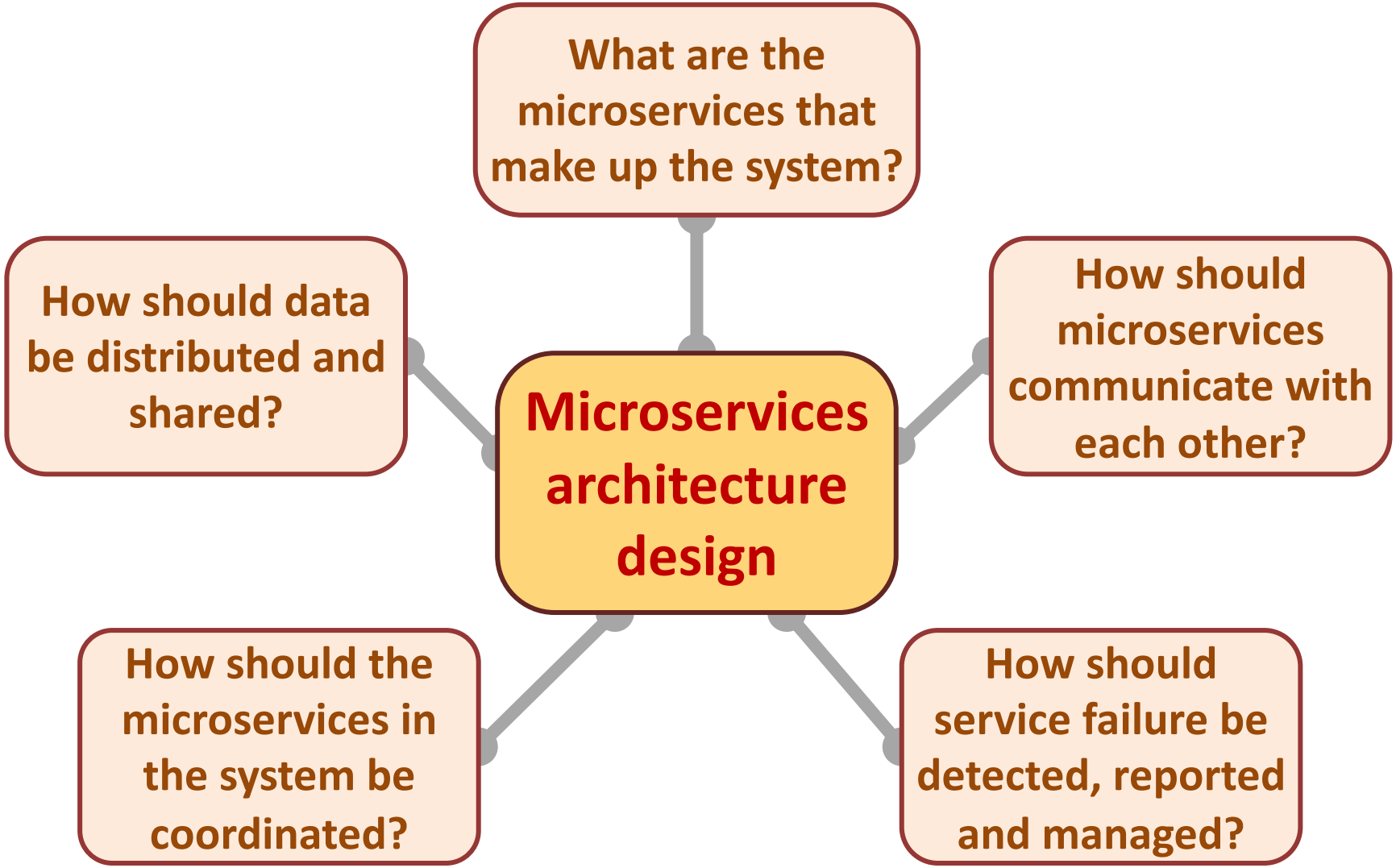
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software as a service



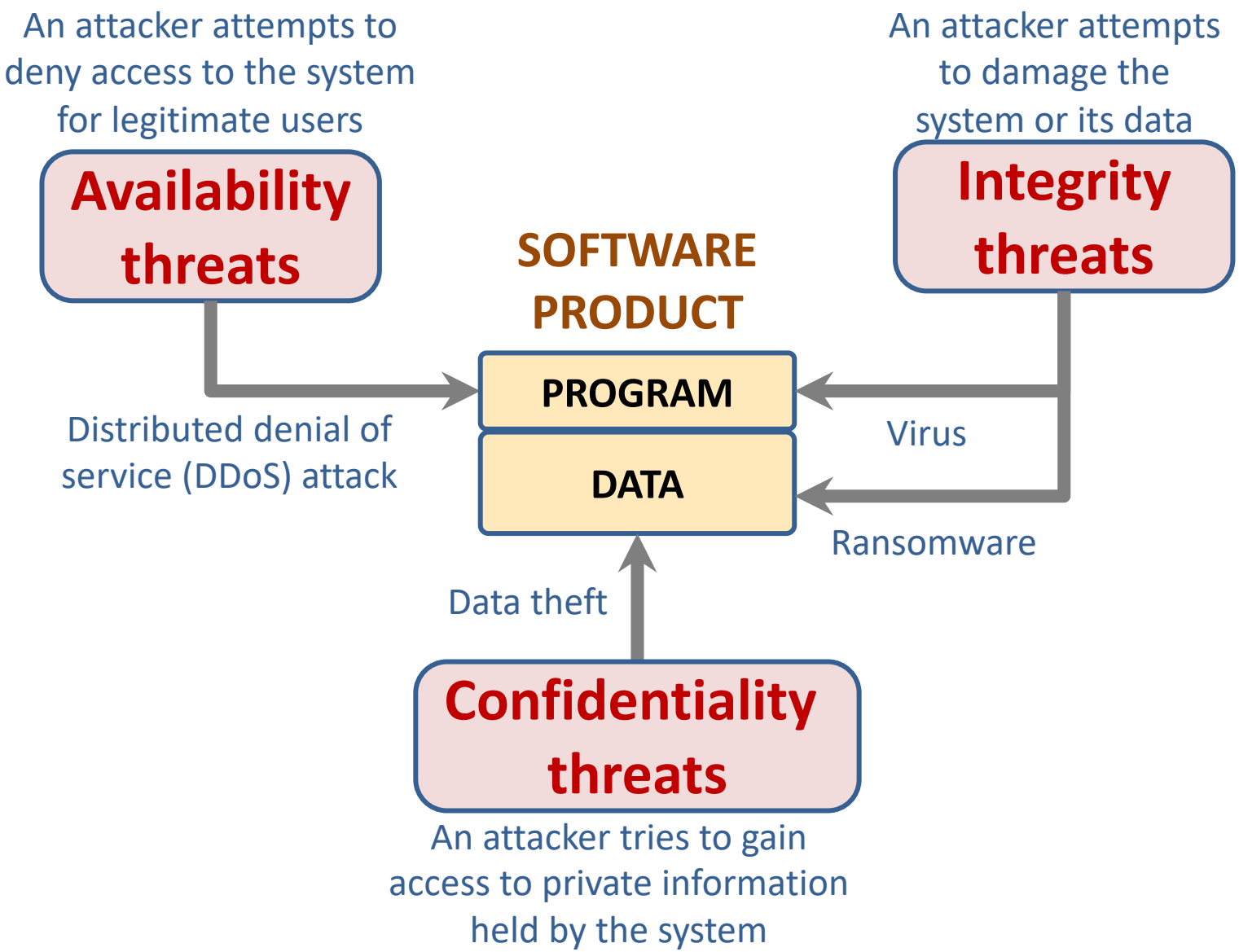
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Microservices architecture – key design questions



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

Types of security threat

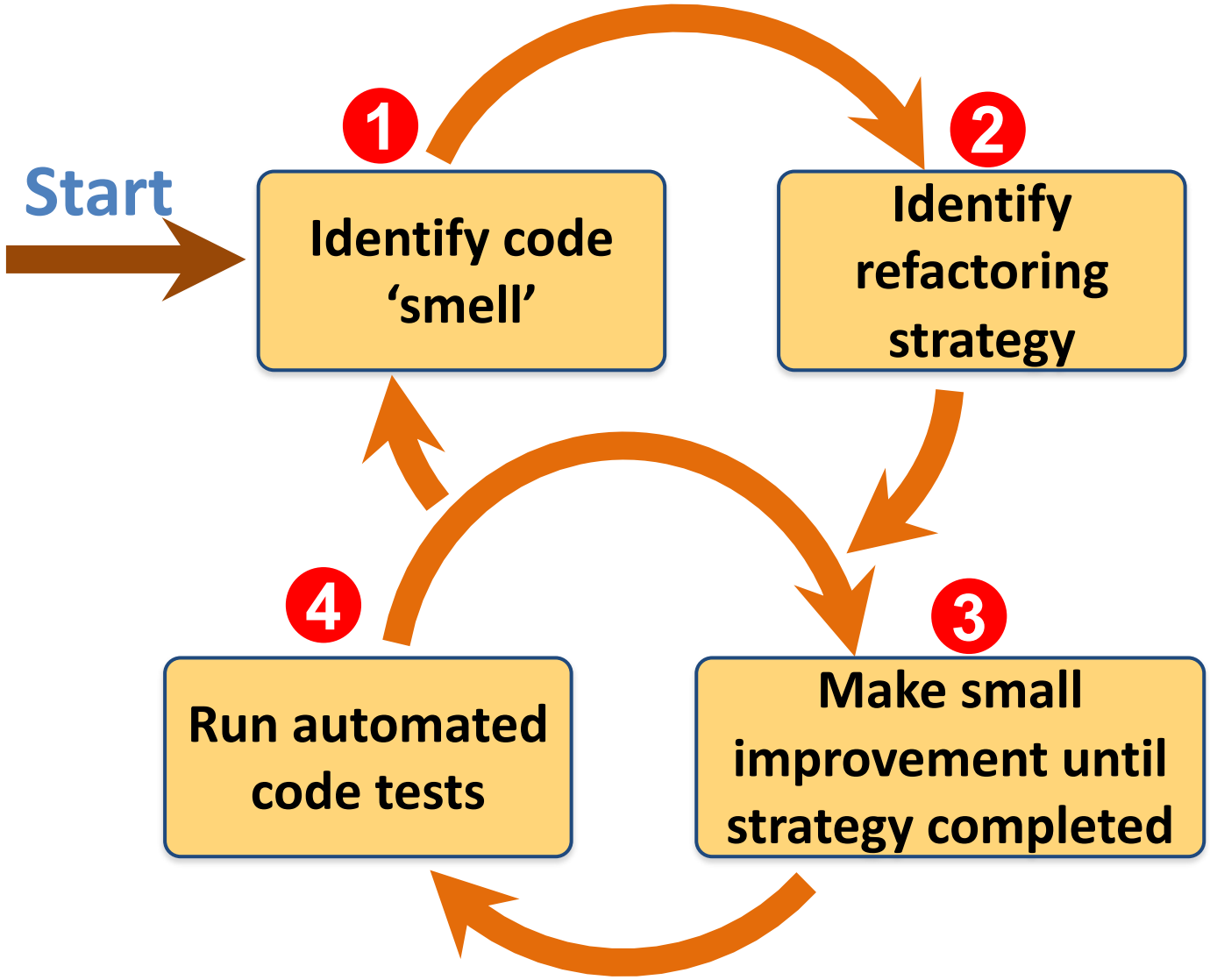


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software product quality attributes

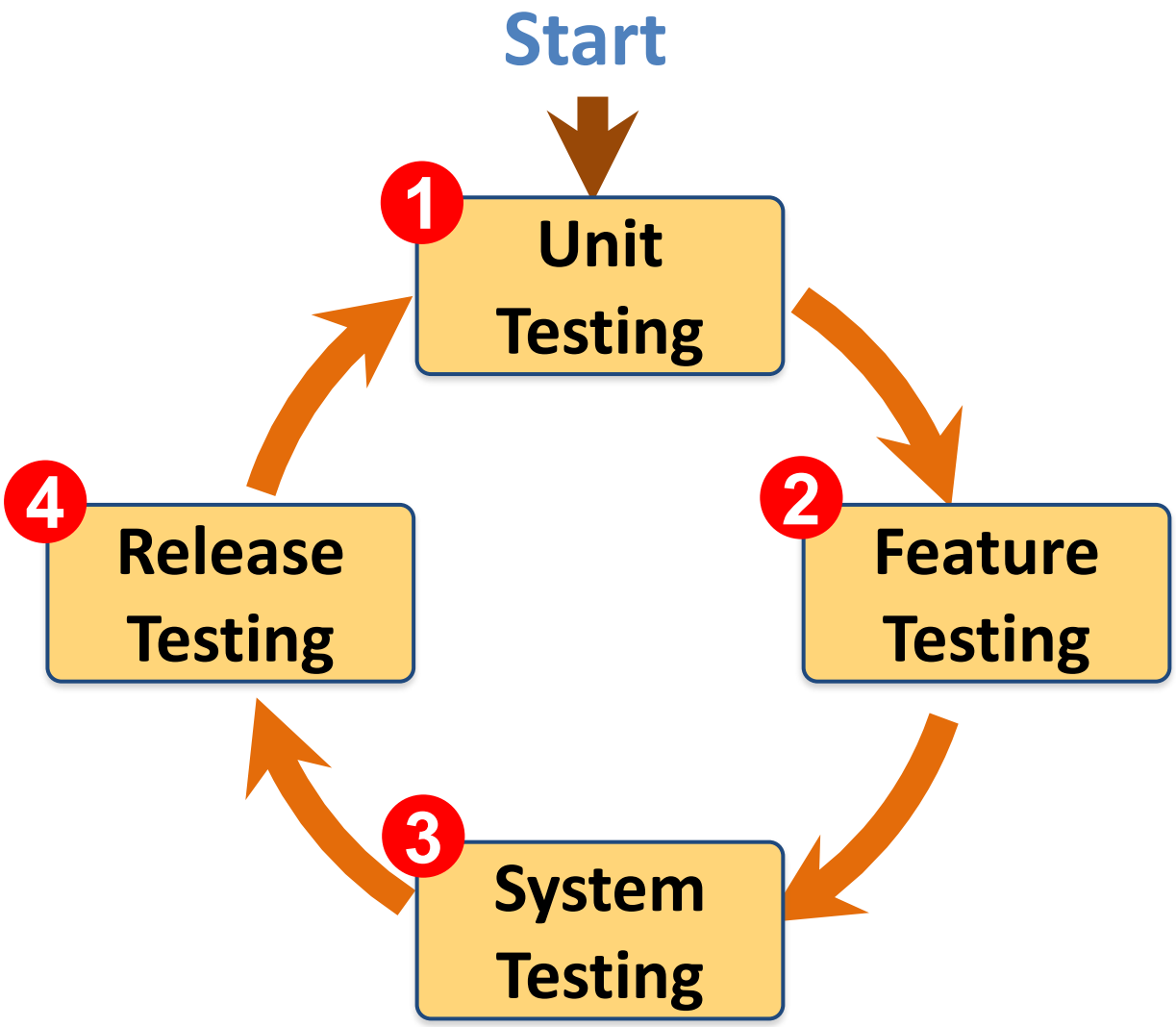
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

A refactoring process



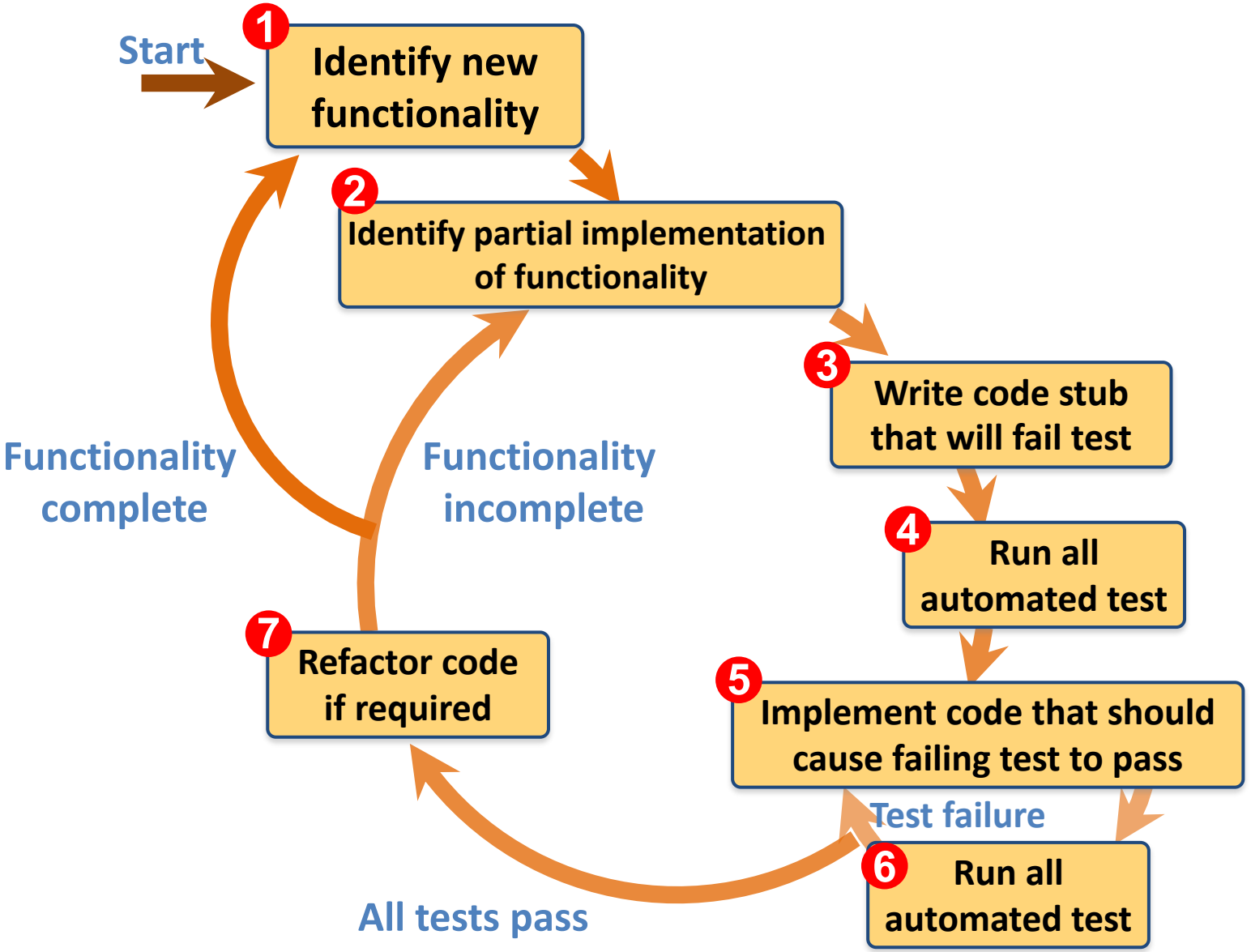
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

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 and Code Management: Code management and DevOps automation

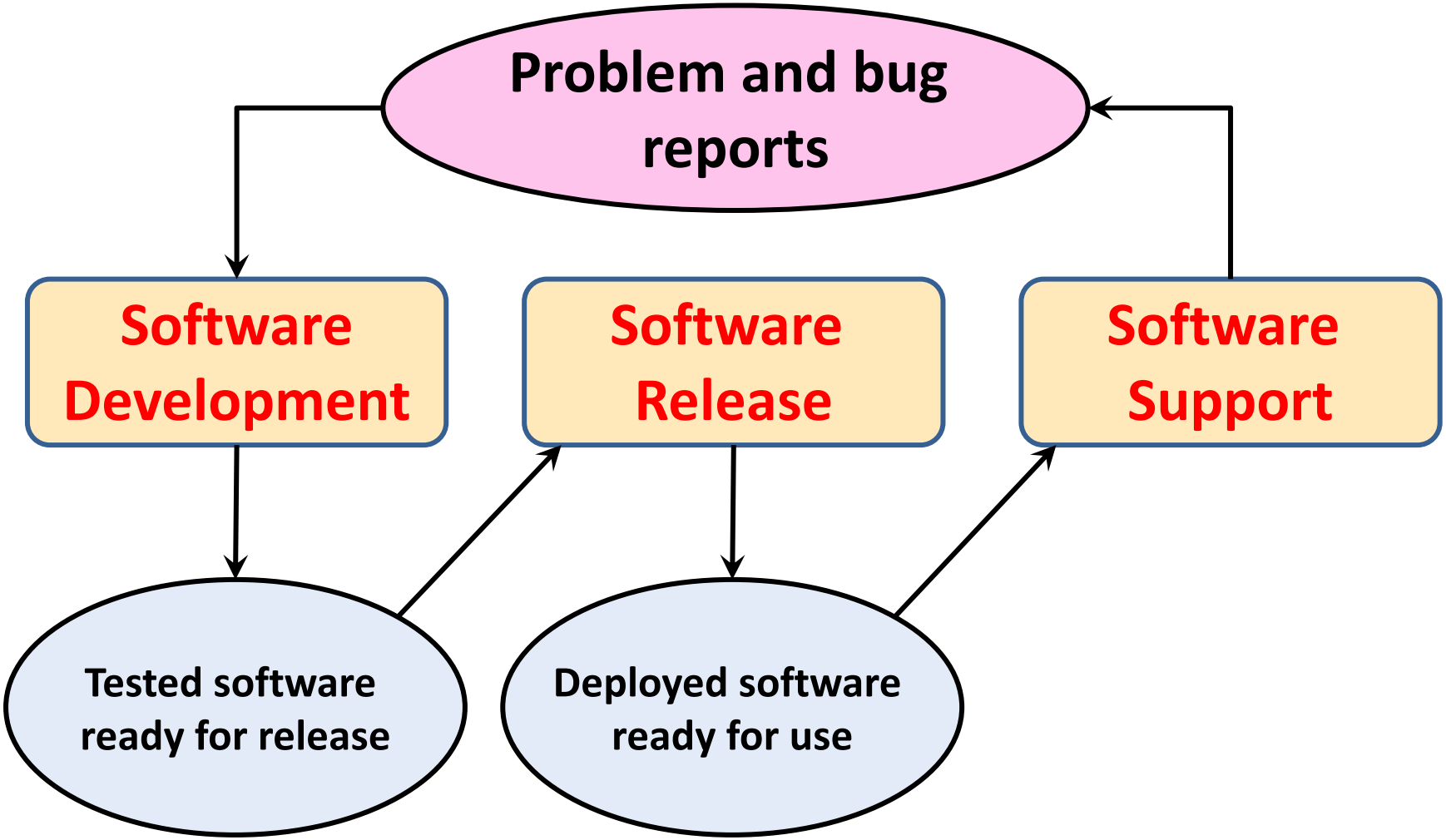
Outline

- Source code management
- DevOps automation
- DevOps measurement

Software support

- Traditionally, separate teams were responsible software development, software release and software support.
- The development team passed over a 'final' version of the software to a release team.
  - Built a release version, tested this and prepared release documentation before releasing the software to customers.
- A third team was responsible for providing customer support.
  - The original development team were sometimes also responsible for implementing software changes.
  - Alternatively, the software may have been maintained by a separate 'maintenance team'.

Software Development, release and support



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

DevOps

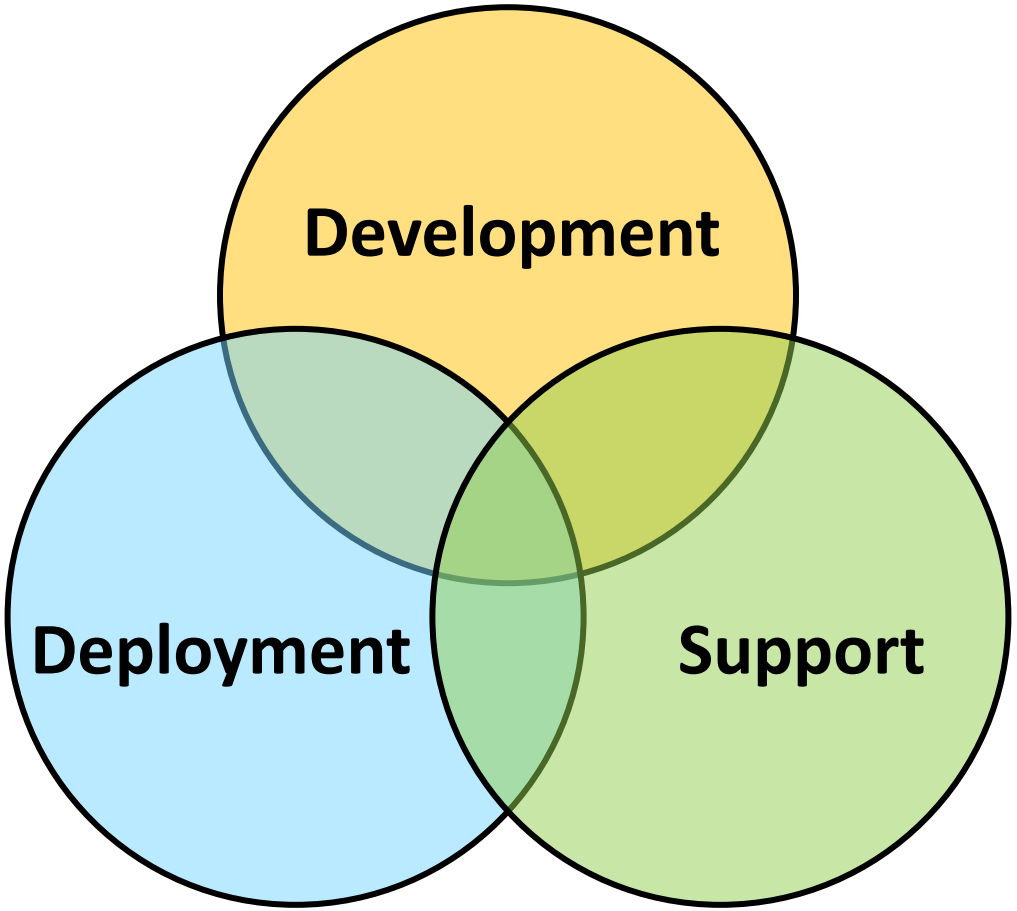
- There are inevitable delays and overheads in the traditional support model.
- To speed up the release and support processes, an alternative approach called DevOps (Development+Operations) has been developed.

DevOps

- Three factors led to the development and widespread adoption of DevOps:
  - Agile software engineering reduced the development time for software, but the traditional release process introduced a bottleneck between development and deployment.
  - Amazon re-engineered their software around services and introduced an approach in which a service was developed and supported by the same team. Amazon's claim that this led to significant improvements in reliability was widely publicized.
  - It became possible to release software as a service, running on a public or private cloud. Software products did not have to be released to users on physical media or downloads.

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.

DevOps principles

- **Everyone is responsible for everything**
  All team members have joint responsibility for developing, delivering and supporting the software.
- **Everything that can be automated should be automated**
  All activities involved in testing, deployment and support should be automated if it is possible to do so. There should be minimal manual involvement in deploying software.
- **Measure first, change later**
  DevOps should be driven by a measurement program where you collect data about the system and its operation. You then use the collected data to inform decisions about changing DevOps processes and tools.

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

Benefits of DevOps

Faster deployment	Software can be deployed to production more quickly because communication delays between the people involved in the process are dramatically reduced.
Reduced risk	The increment of functionality in each release is small so there is less chance of feature interactions and other changes causing system failures and outages.
Faster repair	DevOps teams work together to get the software up and running again as soon as possible.
More productive teams	DevOps teams are happier and more productive than the teams involved in the separate activities.

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

Code management

- Code management is a set of software-supported practices that is used to manage an evolving codebase.
- During the development of a software product, the development team will probably create tens of thousands of lines of code and automated tests.
- These will be organized into hundreds of files. Dozens of libraries may be used, and several, different programs may be involved in creating and running the code.

Code management

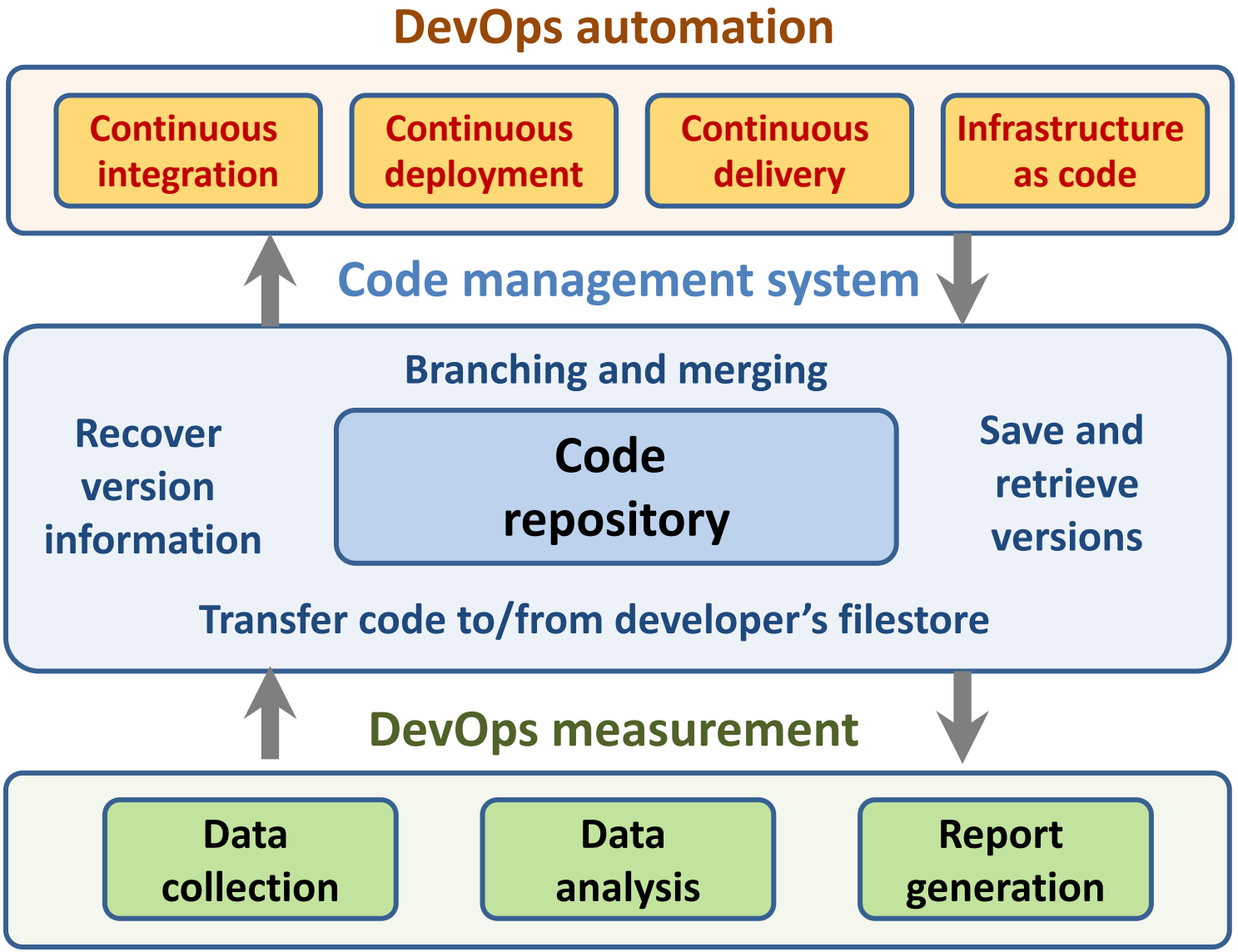
- You need code management to ensure that changes made by different developers do not interfere with each other, and to create different product versions.
- Code management tools make it easy to create an executable product from its source code files and to run automated tests on that product.

Code management and DevOps

- Source code management, combined with automated system building, is essential for professional software engineering.
- In companies that use DevOps, a modern code management system is a fundamental requirement for 'automating everything'.
- Not only does it store the project code that is ultimately deployed, it also stores all other information that is used in DevOps processes.
- DevOps automation and measurement tools all interact with the code management system

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.

Code management fundamentals

- Code management systems provide a set of features that support four general areas:
- **Code transfer**
  - Developers take code into their personal file store to work on it then return it to the shared code management system.
- **Version storage and retrieval**
  - Files may be stored in several different versions and specific versions of these files can be retrieved.
- **Merging and branching**
  - Parallel development branches may be created for concurrent working. Changes made by developers in different branches may be merged.
- **Version information**
  - Information about the different versions maintained in the system may be stored and retrieved

Code repository

- All source code management systems have the general form with a shared repository and a set of features to manage the files in that repository:
  - All source code files and file versions are stored in the repository, as are other artefacts such as configuration files, build scripts, shared libraries and versions of tools used.
  - The repository includes a database of information about the stored files such as version information, information about who has changed the files, what changes were made at what times, and so on.

Code repository

- Files can be transferred to and from the repository and information about the different versions of files and their relationships may be updated.
  - Specific versions of files and information about these versions can always be retrieved from the repository.

Features of code management systems

- Version and release identification
- Change history recording
- Independent development
- Project support
- Storage management

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

Git

- In 2005, Linus Torvalds, the developer of Linux, revolutionized source code management by developing a distributed version control system (DVCS) called Git to manage the code of the Linux kernel.
- This was geared to supporting large-scale open source development. It took advantage of the fact that storage costs had fallen to such an extent that most users did not have to be concerned with local storage management.
- Instead of only keeping the copies of the files that users are working on, Git maintains a clone of the repository on every user's computer

Source: https://git-scm.com/

Benefits of distributed code management

Resilience

Speed

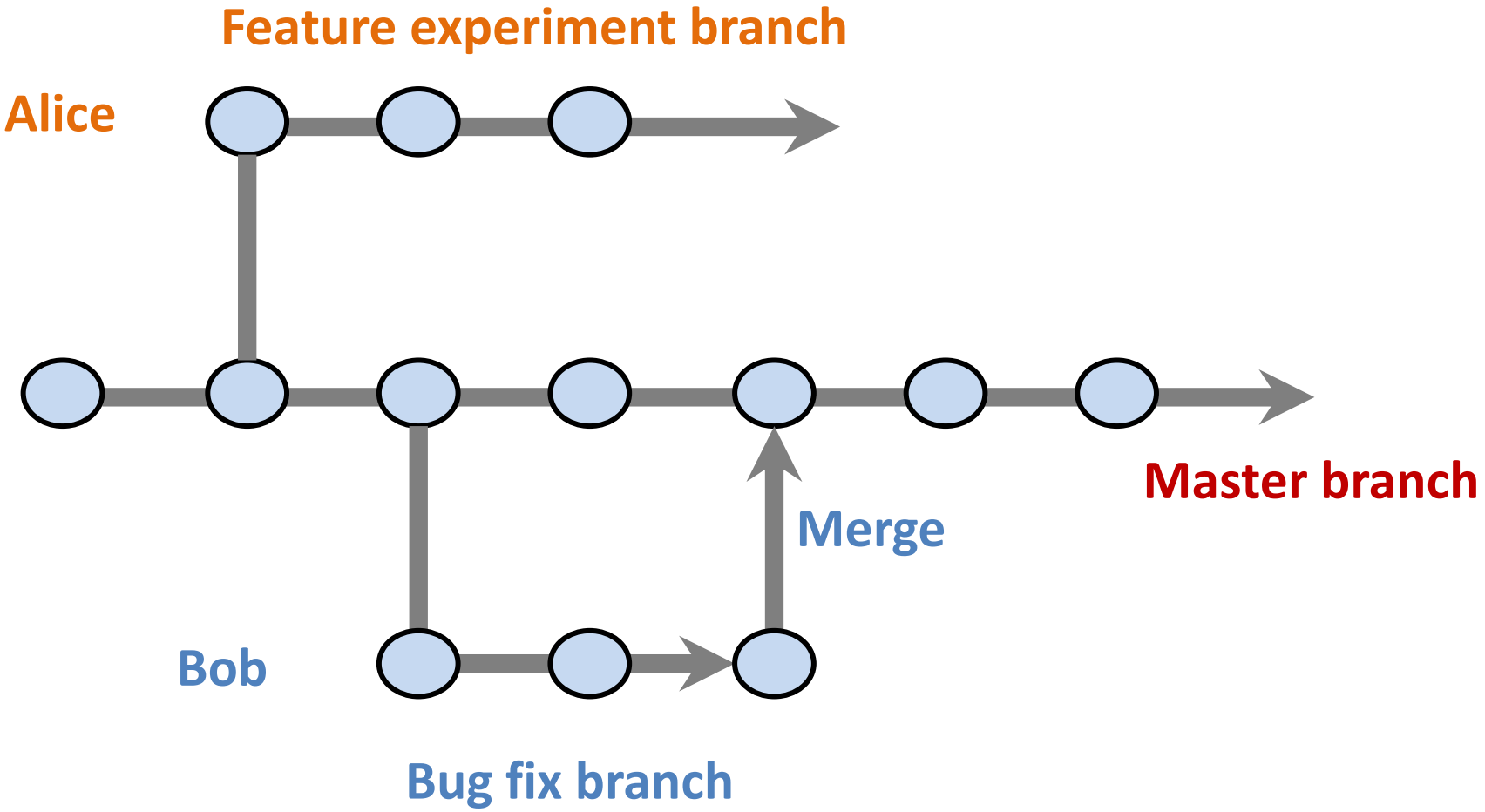
Flexibility

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

Branching and merging

- Branching and merging are fundamental ideas that are supported by all code management systems.
- A branch is an independent, stand-alone version that is created when a developer wishes to change a file.
- The changes made by developers in their own branches may be merged to create a new shared branch.
- The repository ensures that branch files that have been changed cannot overwrite repository files without a merge operation.

Branching and merging



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

DevOps automation

- By using DevOps with automated support, you can dramatically reduce the time and costs for integration, deployment and delivery.
- Everything that can be, should be automated is a fundamental principle of DevOps.
- As well as reducing the costs and time required for integration, deployment and delivery, process automation also makes these processes more reliable and reproducible.
- Automation information is encoded in scripts and system models that can be checked, reviewed, versioned and stored in the project repository.

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

Aspects of DevOps automation

Continuous integration	Each time a developer commits a change to the project's master branch, an executable version of the system is built and tested.
Continuous delivery	A simulation of the product's operating environment is created and the executable software version is tested.
Continuous deployment	A new release of the system is made available to users every time a change is made to the master branch of the software.
Infrastructure as code	Machine-readable models of the infrastructure (network, servers, routers, etc.) on which the product executes are used by configuration management tools to build the software's execution platform.

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

Characteristics of infrastructure as code

Visibility

Reproducibility

Reliability

Recovery

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

DevOps measurement

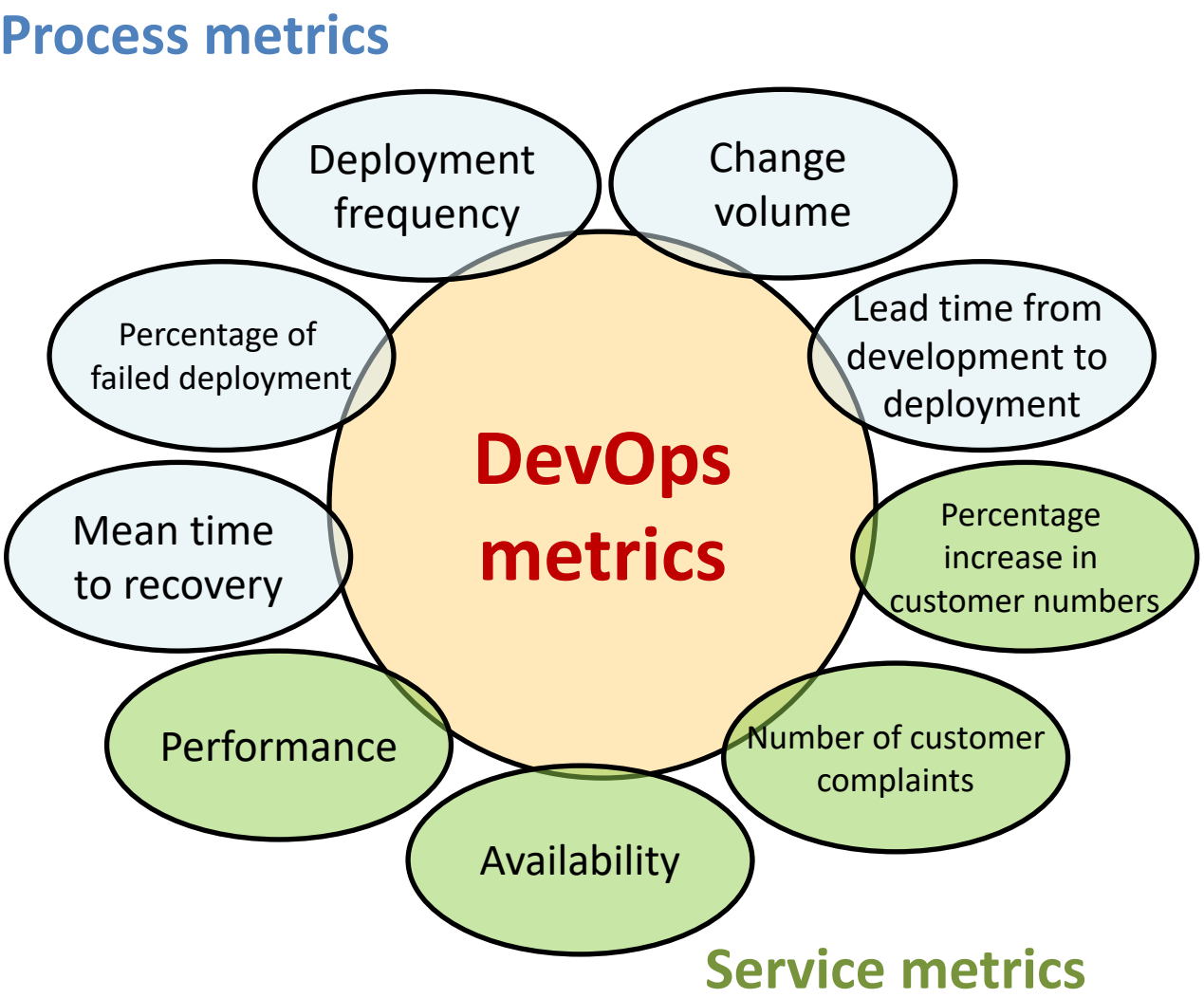
- After you have adopted DevOps, you should try to continuously improve your DevOps process to achieve faster deployment of better-quality software.
- There are four types of software development measurement:
  - Process measurement
  - Service measurement
  - Usage measurement
  - Business success measurement

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

Automating measurement

- As far as possible, the DevOps principle of automating everything should be applied to software measurement.
- You should instrument your software to collect data about itself and you should use a monitoring system to collect data about your software's performance and availability.
- Some process measurements can also be automated.
  - However, there are problems in process measurement because people are involved. They work in different ways, may record information differently and are affected by outside influences that affect the way they work.

Metrics used in the DevOps scorecard



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

Summary

- DevOps is the integration of software development and the management of that software once it has been deployed for use. The same team is responsible for development, deployment and software support.
- The benefits of DevOps are faster deployment, reduced risk, faster repair of buggy code and more productive teams.
- Source code management is essential to avoid changes made by different developers interfering with each other.

Summary

- All code management systems are based around a shared code repository with a set of features that support code transfer, version storage and retrieval, branching and merging and maintaining version information.
- Git is a distributed code management system that is the most widely used system for software product development. Each developer works with their own copy of the repository which may be merged with the shared project repository.

Summary

- DevOps is the integration of software development and the management of that software once it has been deployed for use. The same team is responsible for development, deployment and software support.
- The benefits of DevOps are faster deployment, reduced risk, faster repair of buggy code and more productive teams.
- Source code management is essential to avoid changes made by different developers interfering with each other.

Summary

- Continuous integration means that as soon as a change is committed to a project repository, it is integrated with existing code and a new version of the system is created for testing.
- Automated system building tools reduce the time needed to compile and integrate the system by only recompiling those components and their dependents that have changed.
- Continuous deployment means that as soon as a change is made, the deployed version of the system is automatically updated. This is only possible when the software product is delivered as a cloud-based service.

Summary

- Infrastructure as code means that the infrastructure (network, installed software, etc.) on which software executes is defined as a machine-readable model. Automated tools, such as Chef and Puppet, can provision servers based on the infrastructure model.
- Measurement is a fundamental principle of DevOps. You may make both process and product measurements. Important process metrics are deployment frequency, percentage of failed deployments, and mean time to recovery from failure.

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 (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