Software Engineering

Software Products and Project Management: Software product management and prototyping with Generative AI and Agentic AI

1142SE02
MBA, IM, NTPU (M5010) (Spring 2026)
Wed 2, 3, 4 (9:10-12:00) (B3F17)

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

Min-Yuh Day, Ph.D,
Professor and Director

Institute of Information Management, National Taipei University

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

2026-03-04

Syllabus

Week	Date	Subject/Topics
1	2026/02/25	Introduction to Software Engineering
2	2026/03/04	Software Products and Project Management: Software product management and prototyping with Generative AI and Agentic AI
3	2026/03/11	Agile Software Engineering: Agile methods, Scrum, and Extreme Programming
4	2026/03/18	Case Study on Software Engineering I
5	2026/03/25	Features, Scenarios, and Stories
6	2026/04/01	Software Architecture: Architectural design, System decomposition, and Distribution architecture

Syllabus

Week	Date	Subject/Topics
7	2026/04/08	Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service
8	2026/04/15	Midterm Project Report
9	2026/04/22	Cloud Computing and Cloud Software Architecture
10	2026/04/29	Microservices Architecture, RESTful services, Service deployment
11	2026/05/06	Case Study on Software Engineering II
12	2026/05/13	Security and Privacy; Reliable Programming; Testing: Functional testing, Test automation, Test-driven development, and Code reviews

Syllabus

Week	Date	Subject/Topics
13	2026/05/20	Industry Practices of Software Engineering
14	2026/05/27	DevOps and Code Management: Code management and DevOps automation
15	2026/06/03	Final Project Report I
16	2026/06/10	Final Project Report II

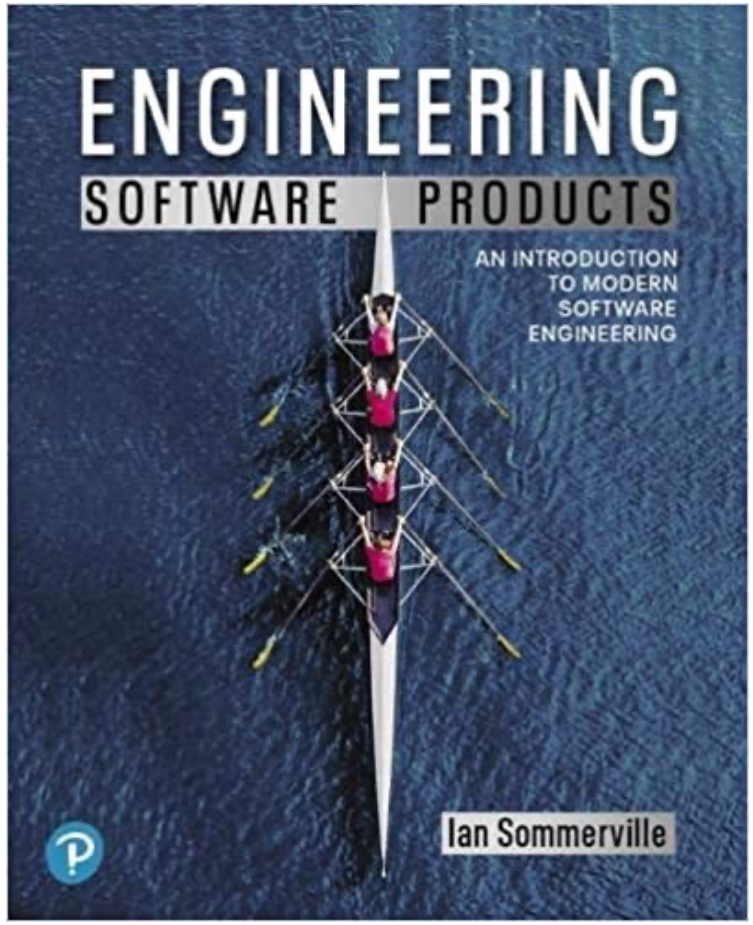
Software Products and Project Management: Software Product Management and Prototyping with Generative AI and Agentic AI

Ian Sommerville (2019),

Engineering Software Products:

An Introduction to Modern Software Engineering,

Pearson.

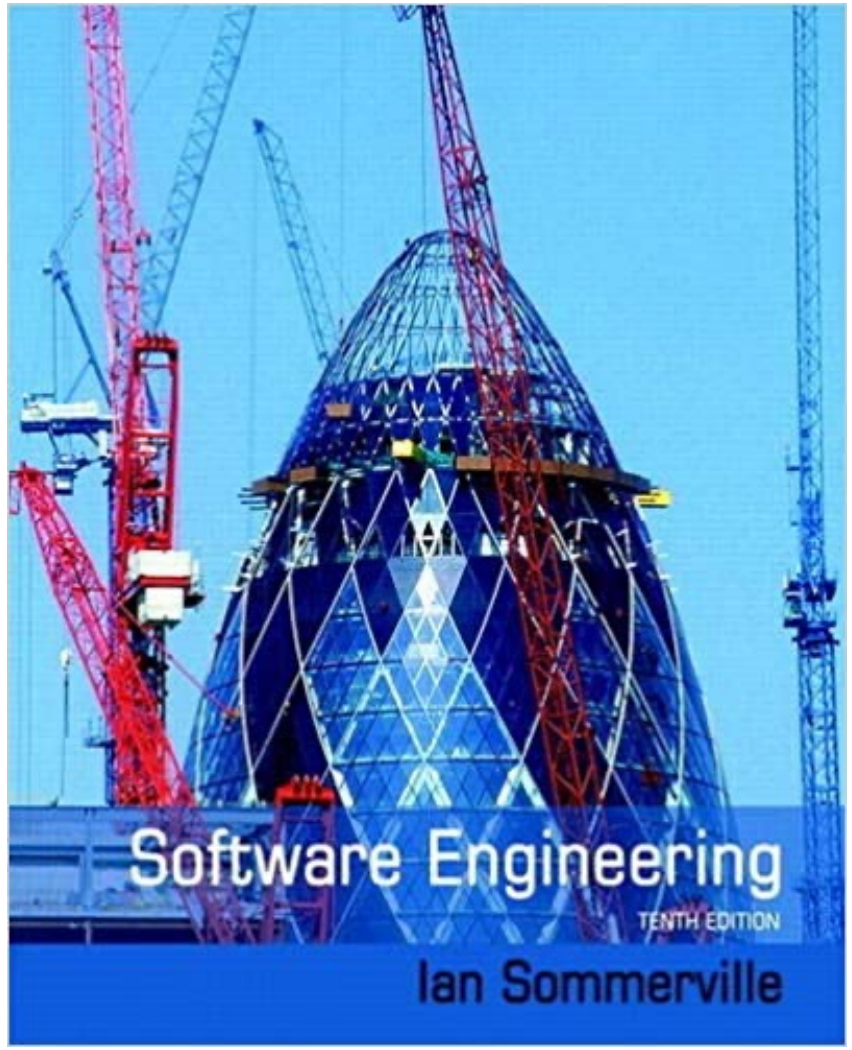


Source: https://www.amazon.com/Engineering-Software-Products-Ian-Sommerville/dp/013521064X

Ian Sommerville (2015),

Software Engineering,

10th Edition, Pearson.



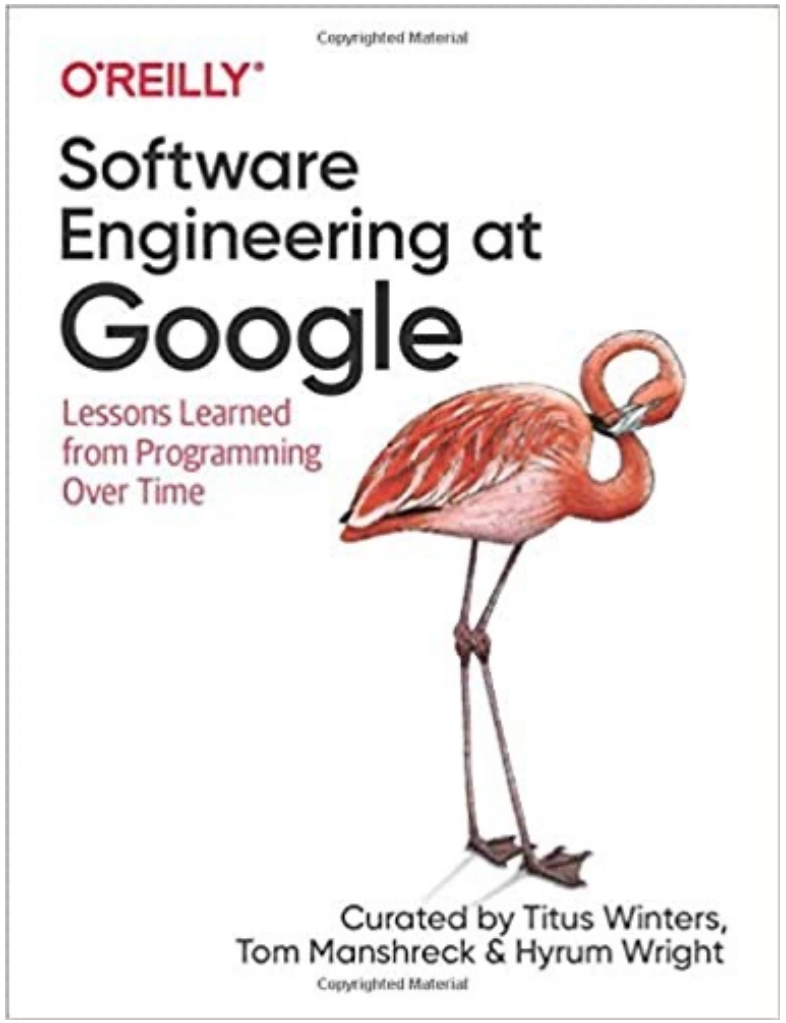
Source: https://www.amazon.com/Software-Engineering-10th-Ian-Sommerville/dp/0133943038

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

Software Engineering at Google:

Lessons Learned from Programming Over Time,

O'Reilly Media.

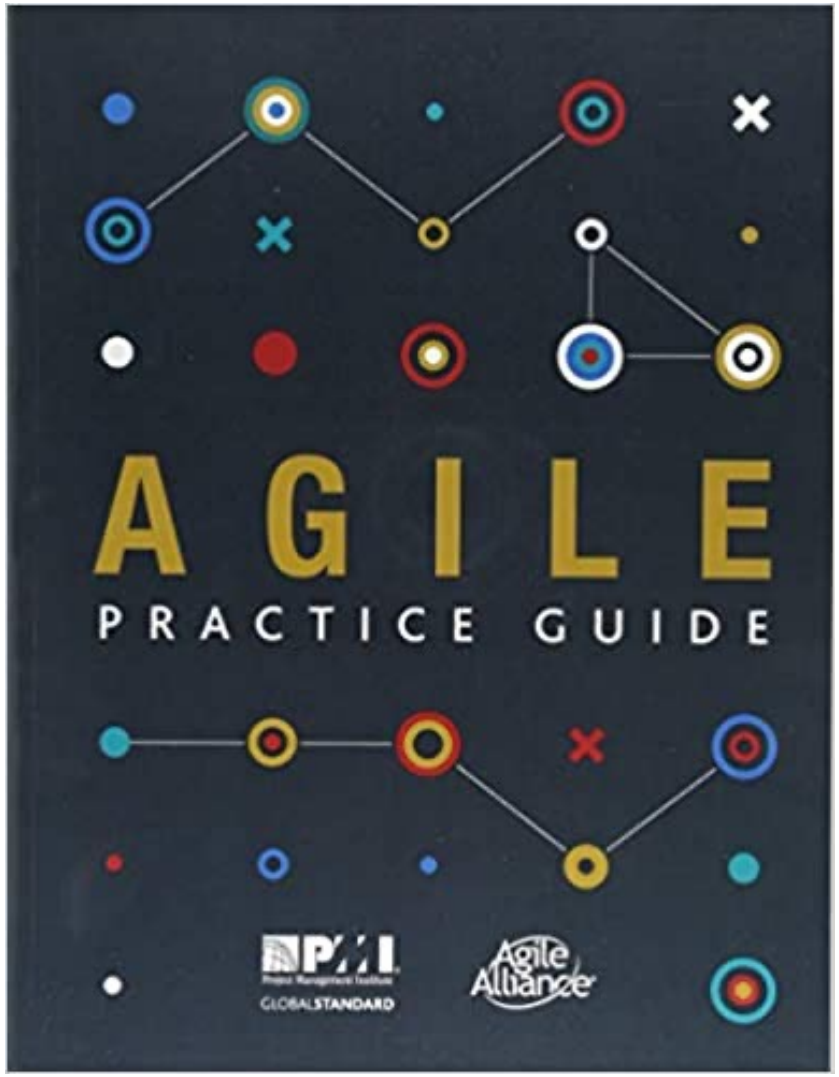


Source: https://www.amazon.com/Software-Engineering-Google-Lessons-Programming/dp/1492082791

Project Management Institute (2017),

Agile Practice Guide

PMI

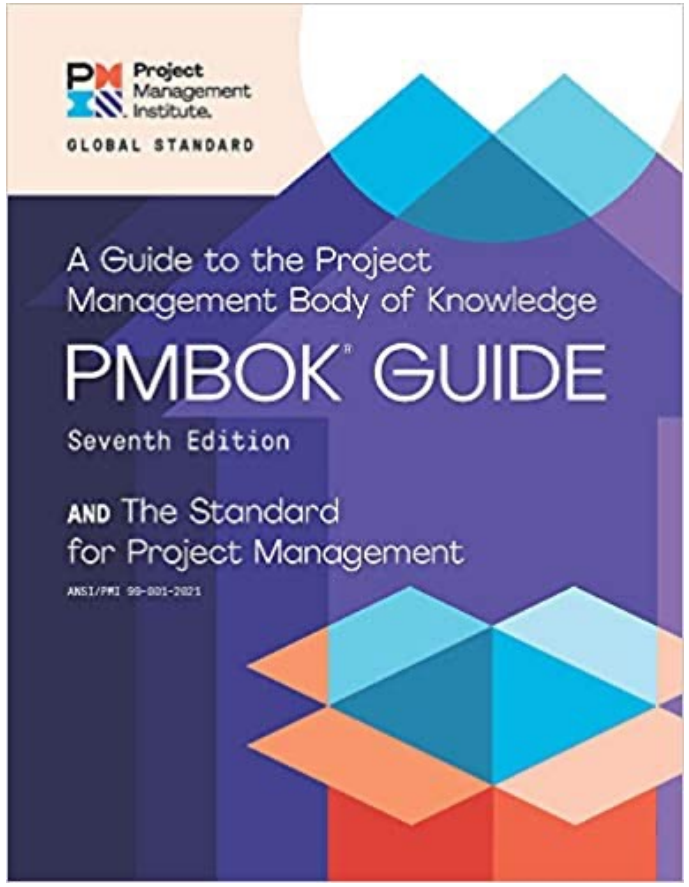


Source: https://www.amazon.com/Agile-Practice-Project-Management-Institute/dp/1628251999/

Project Management Institute (2021),

A Guide to the Project Management Body of Knowledge (PMBOK Guide) –

Seventh Edition and The Standard for Project Management

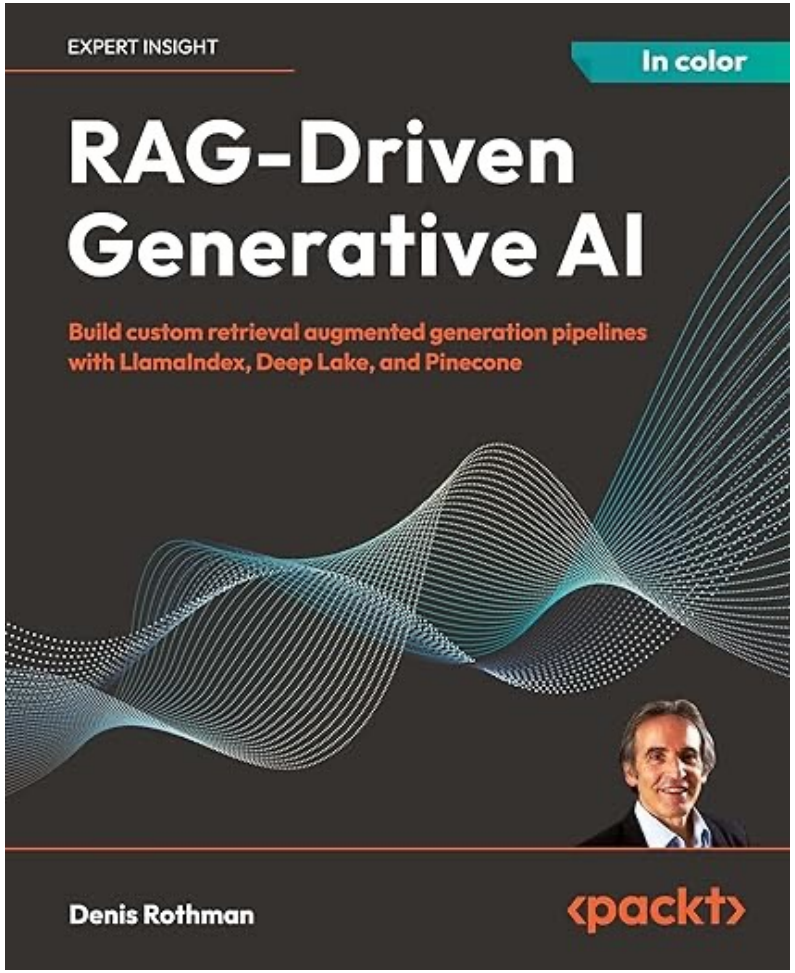


Source: https://www.amazon.com/Guide-Project-Management-Knowledge-PMBOK%C2%AE/dp/1628256648

Denis Rothman (2024),

RAG-Driven Generative AI:

Build custom retrieval augmented generation pipelines with LlamaIndex, Deep Lake, and Pinecone, Packt Publishing



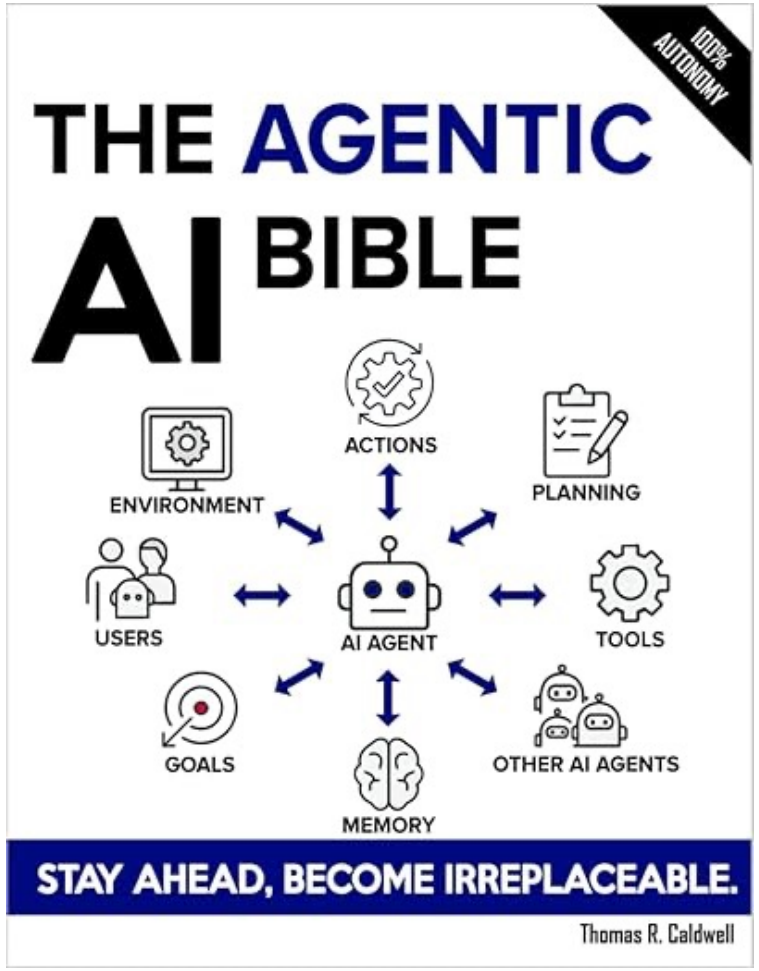
Source: https://www.amazon.com/RAG-Driven-Generative-retrieval-generation-LlamaIndex/dp/1836200919/

Thomas R. Caldwell (2025),

The Agentic AI Bible:

The Complete and Up-to-Date Guide to Design, Build, and Scale Goal-Driven, LLM-Powered Agents that Think, Execute and Evolve,

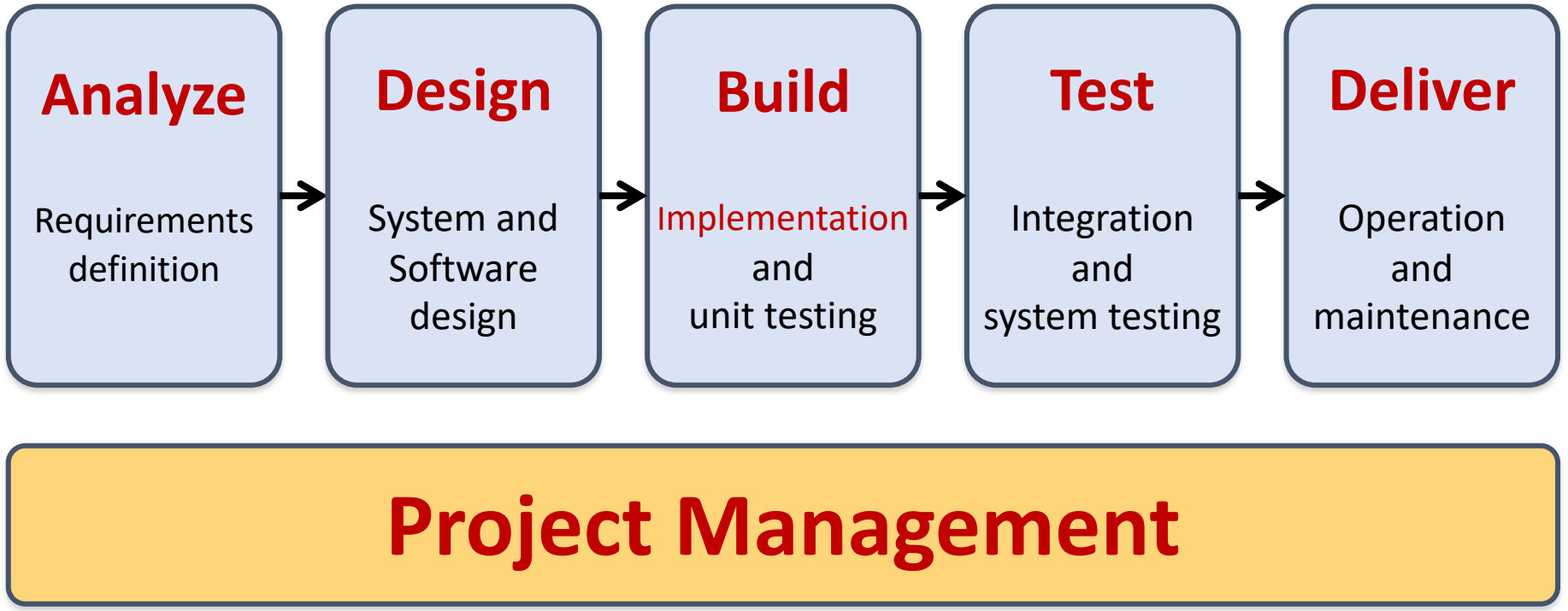
Independently published



Source: https://www.amazon.com/Agentic-Bible-Up-Date-Goal-Driven/dp/B0FL21R86Q

Software Engineering

Software Engineering
and
Project Management

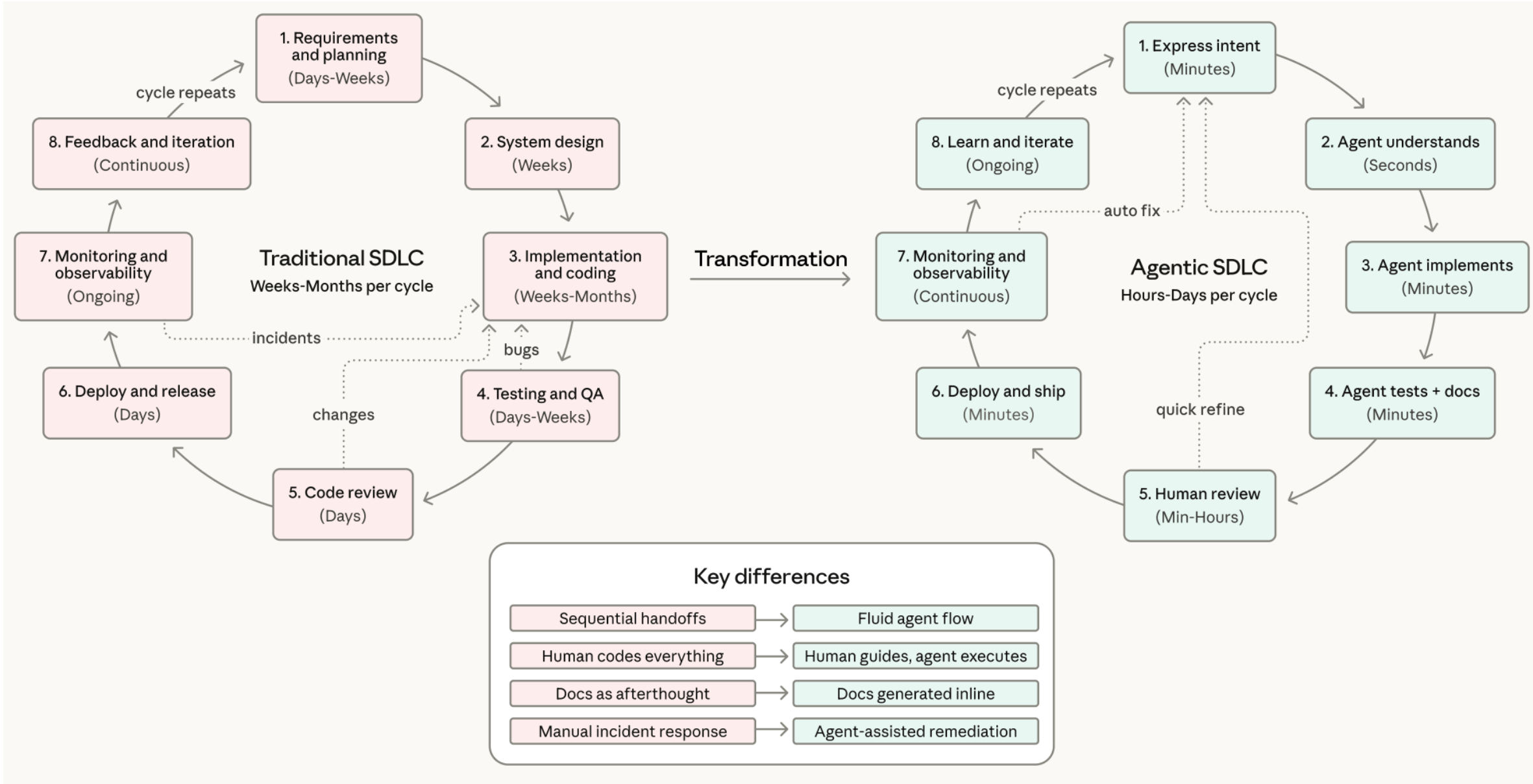


2026 Agentic Coding Trends

Foundation	1. The software development lifecycle changes dramatically
Capability	2. Single agents evolve into coordinated teams
	3. Long-running agents build complete systems
	4. Human oversight scales through intelligent collaboration
	5. Agentic coding expands to new surfaces and users
Impact	6. Productivity gains reshape software development economics
	7. Non-technical use cases expand across organizations
	8. Dual-use risk requires security-first architecture

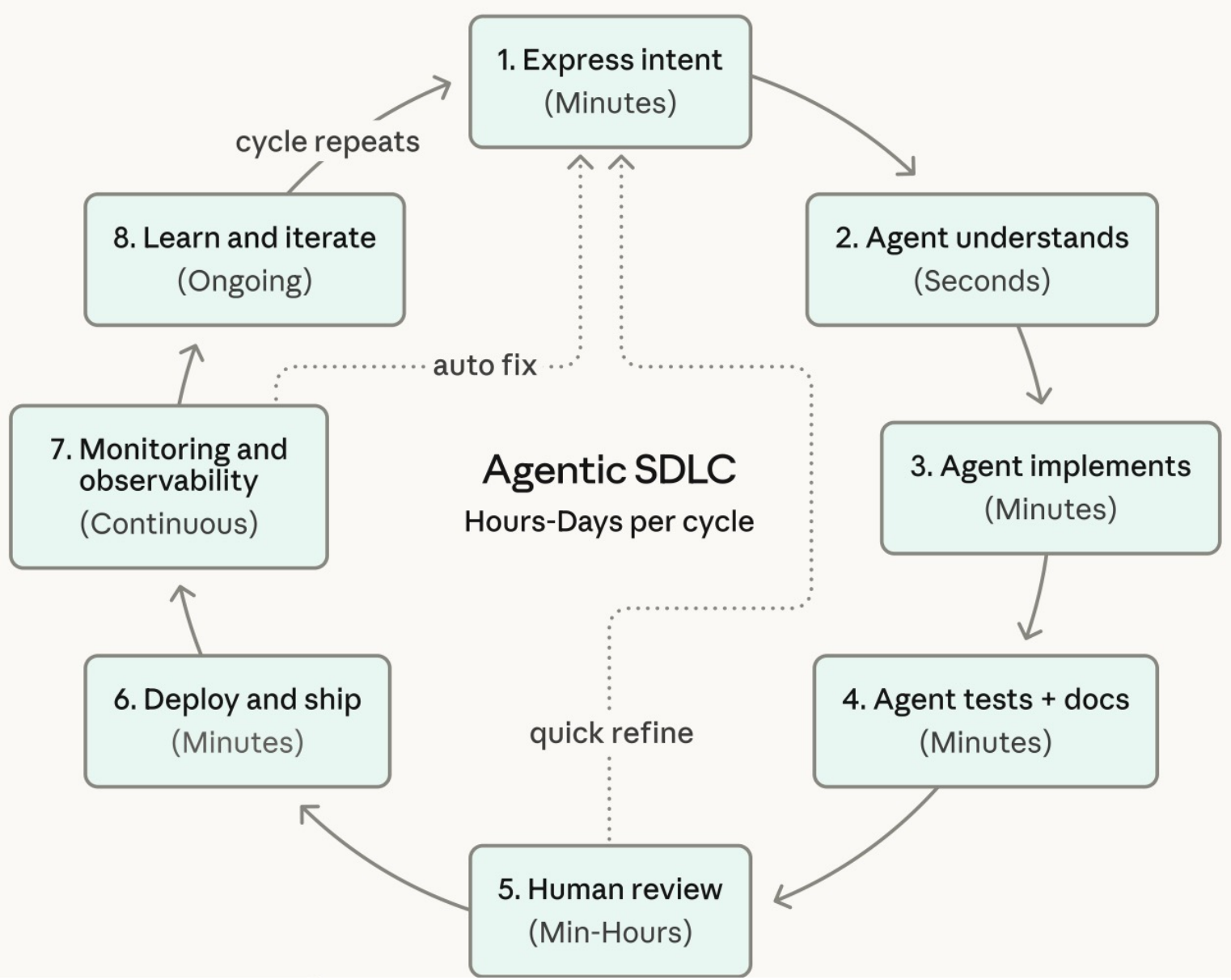
Source: Anthropic (2026), 2026 Agentic CodingTrends Report: How coding agents are reshaping software development, https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf

Agentic Coding Software Development Lifecycle



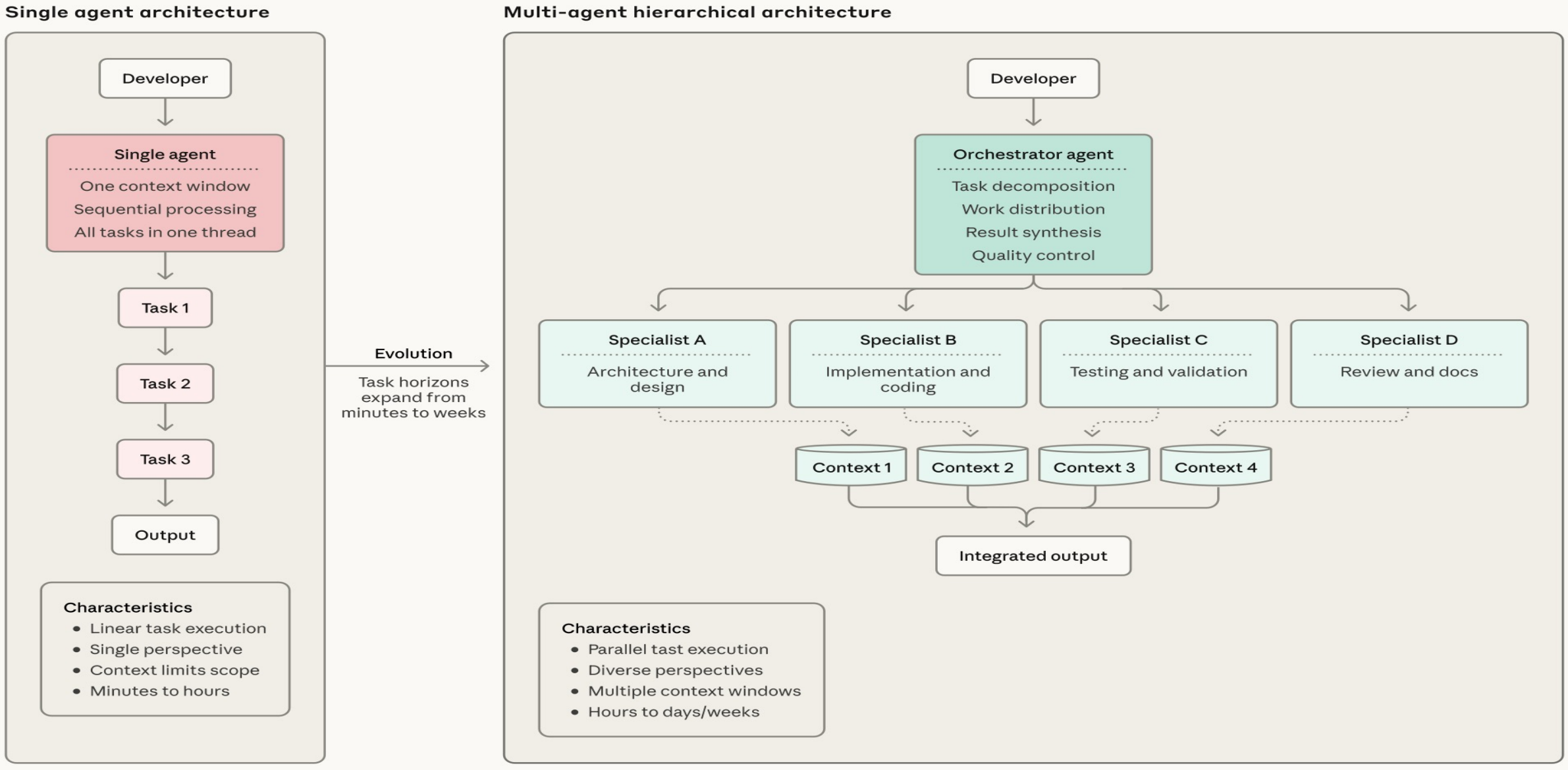
Source: Anthropic (2026), 2026 Agentic CodingTrends Report: How coding agents are reshaping software development, https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf

Agentic Coding Software Development Lifecycle



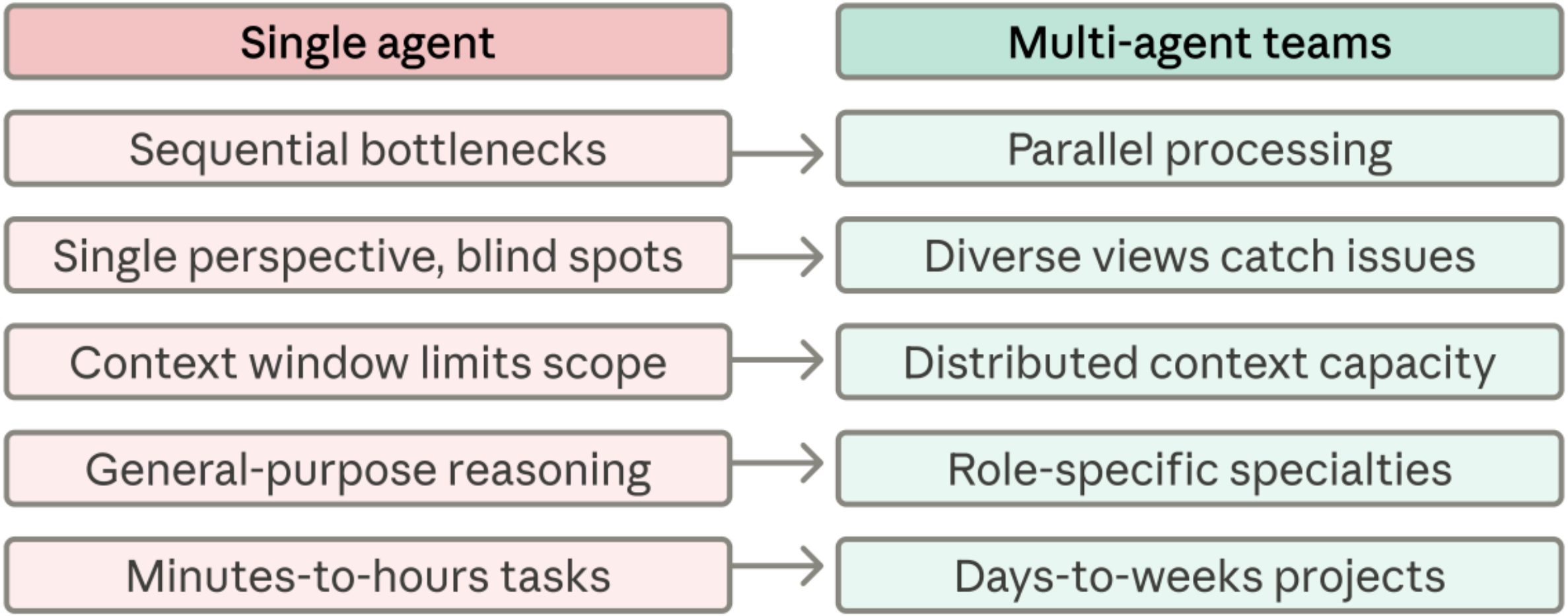
Source: Anthropic (2026), 2026 Agentic CodingTrends Report: How coding agents are reshaping software development, https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf

Coding Agent Architectures: From Single Agents to Coordinated Teams



Source: Anthropic (2026), 2026 Agentic CodingTrends Report: How coding agents are reshaping software development, https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf

Coding Agent Architectures: Single Agent to Multi-Agent Teams Performance Impact



Source: Anthropic (2026), 2026 Agentic CodingTrends Report: How coding agents are reshaping software development, https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf

Generative AI Meets Product Development

Use Case 1:

Enhancing Creativity and Design Workflows

Use Case 2:

GenAI for Customer Insights and Concept Validation

Use Case 3:

LLMs as Natural Language Interfaces to Complex Design Tools

Source: Tucker J. Marion, Mahdi Srour, and Frank Piller (2024), "When Generative AI meets product development." MIT Sloan Management Review 66, no. 1 : 14-15.

Information Management

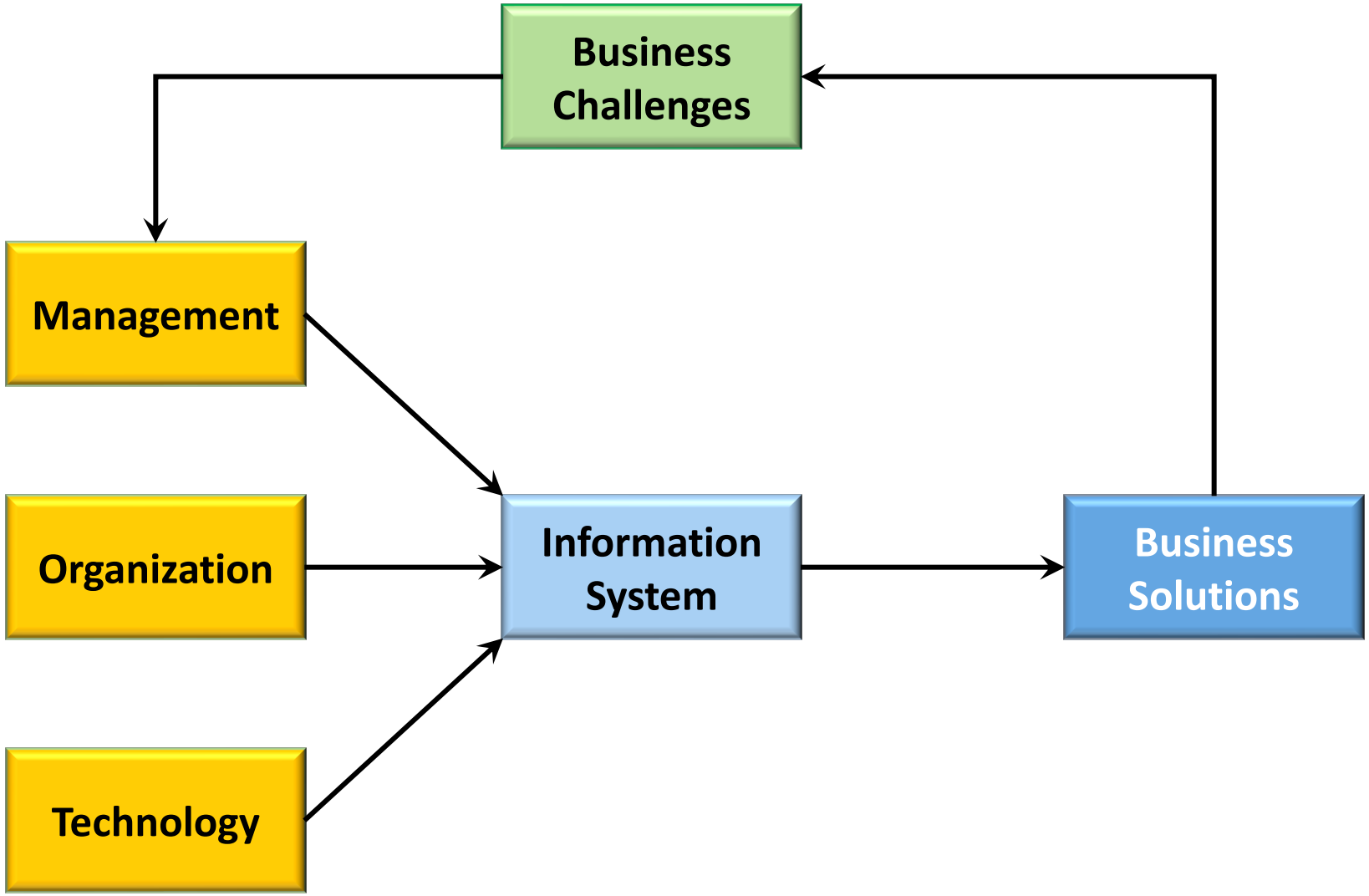
Management Information Systems (MIS)

Information Systems

Information Management (MIS)
Information Systems

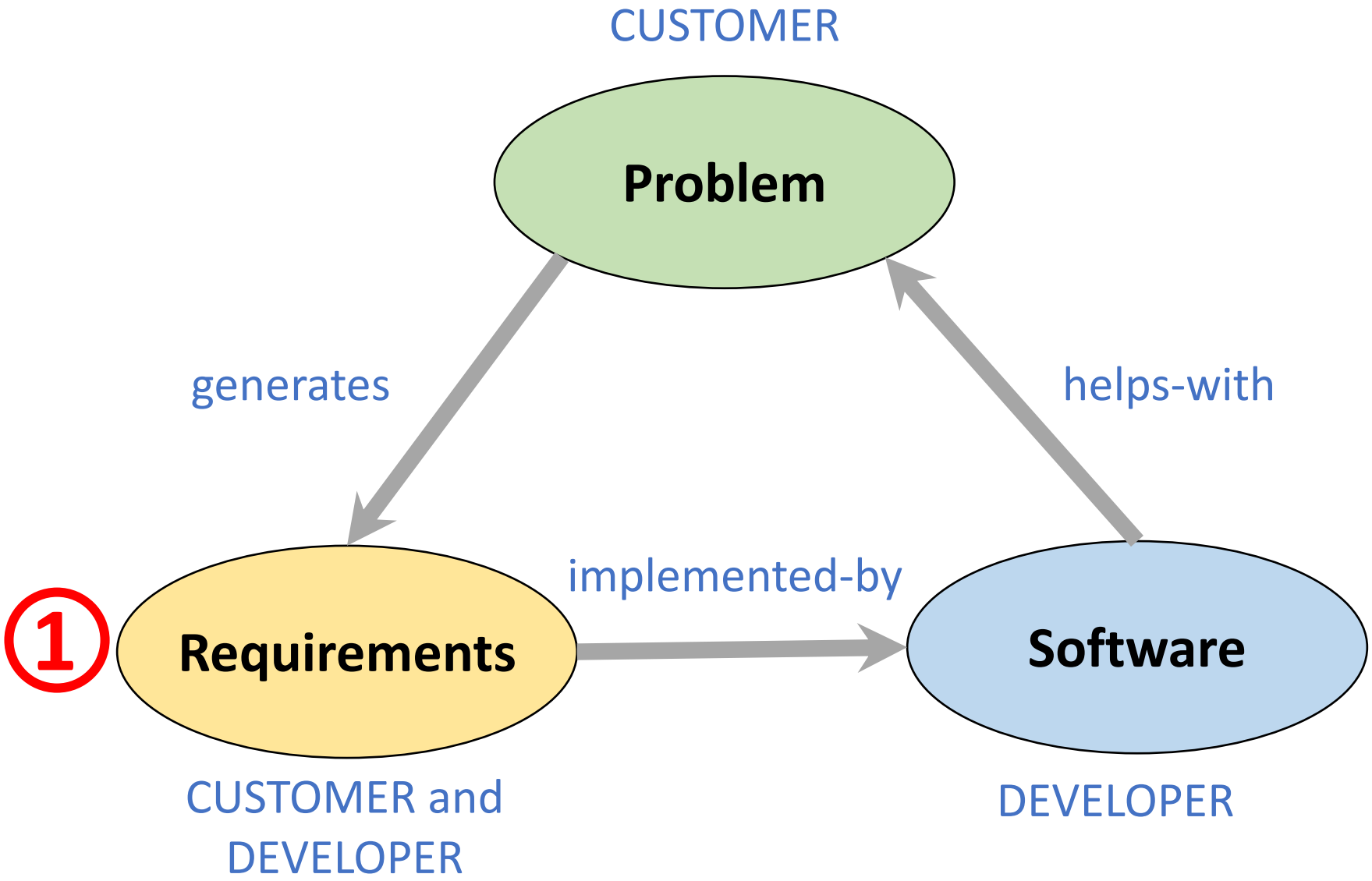
Source: Kenneth C. Laudon & Jane P. Laudon (2014), Management Information Systems: Managing the Digital Firm, Thirteenth Edition, Pearson.

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



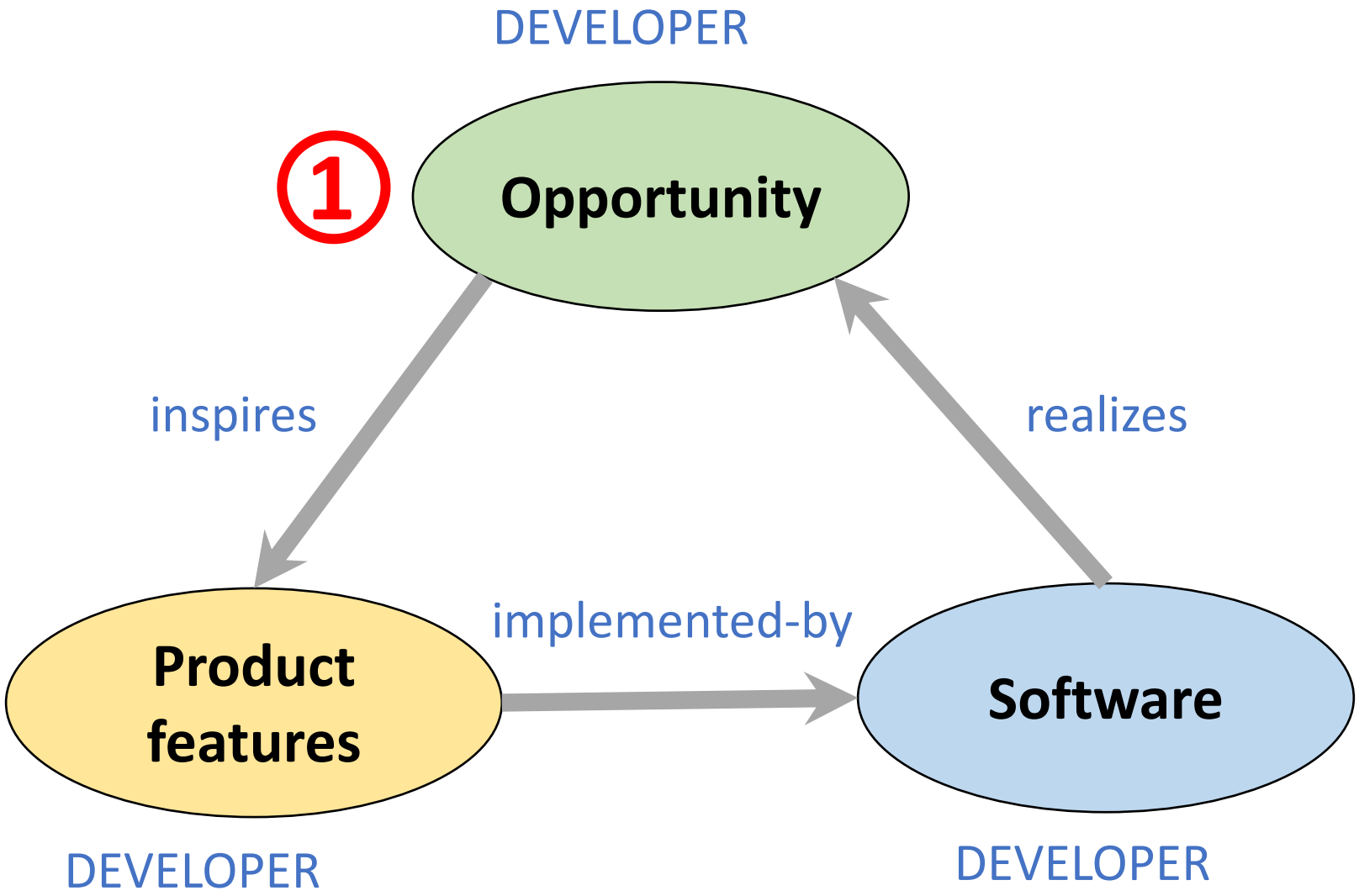
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

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.

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

Product software engineering



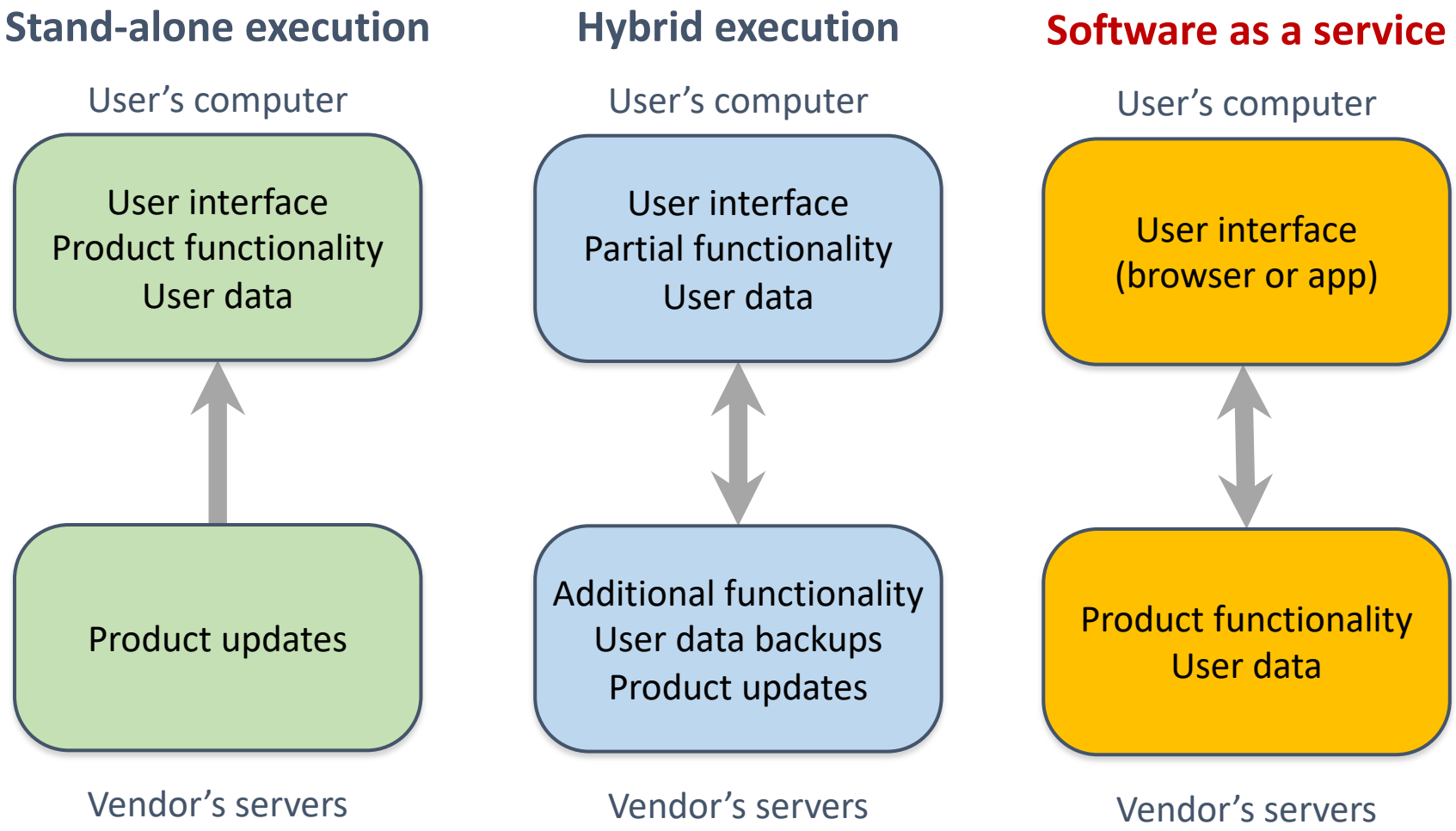
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

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.

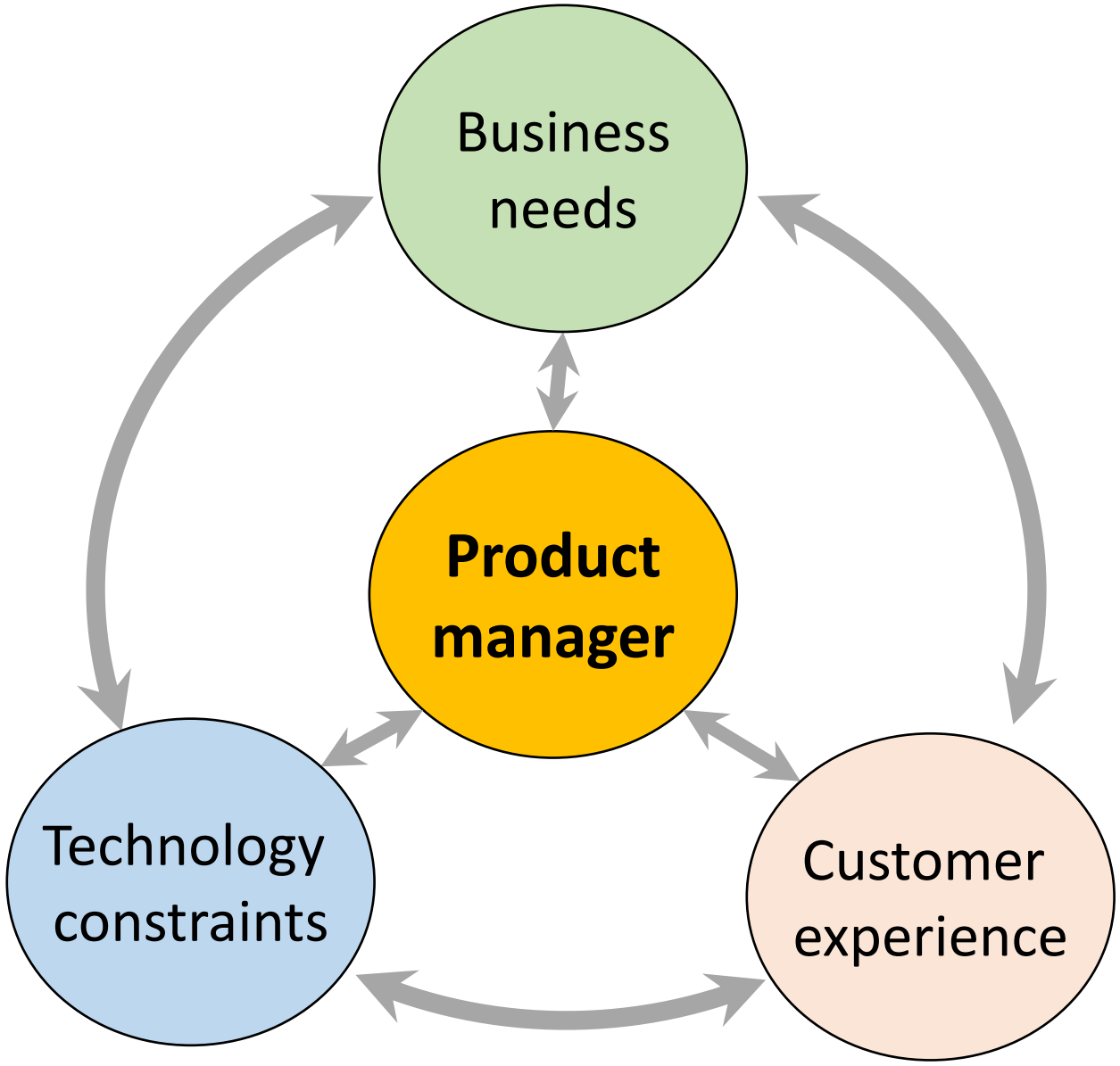
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software execution models



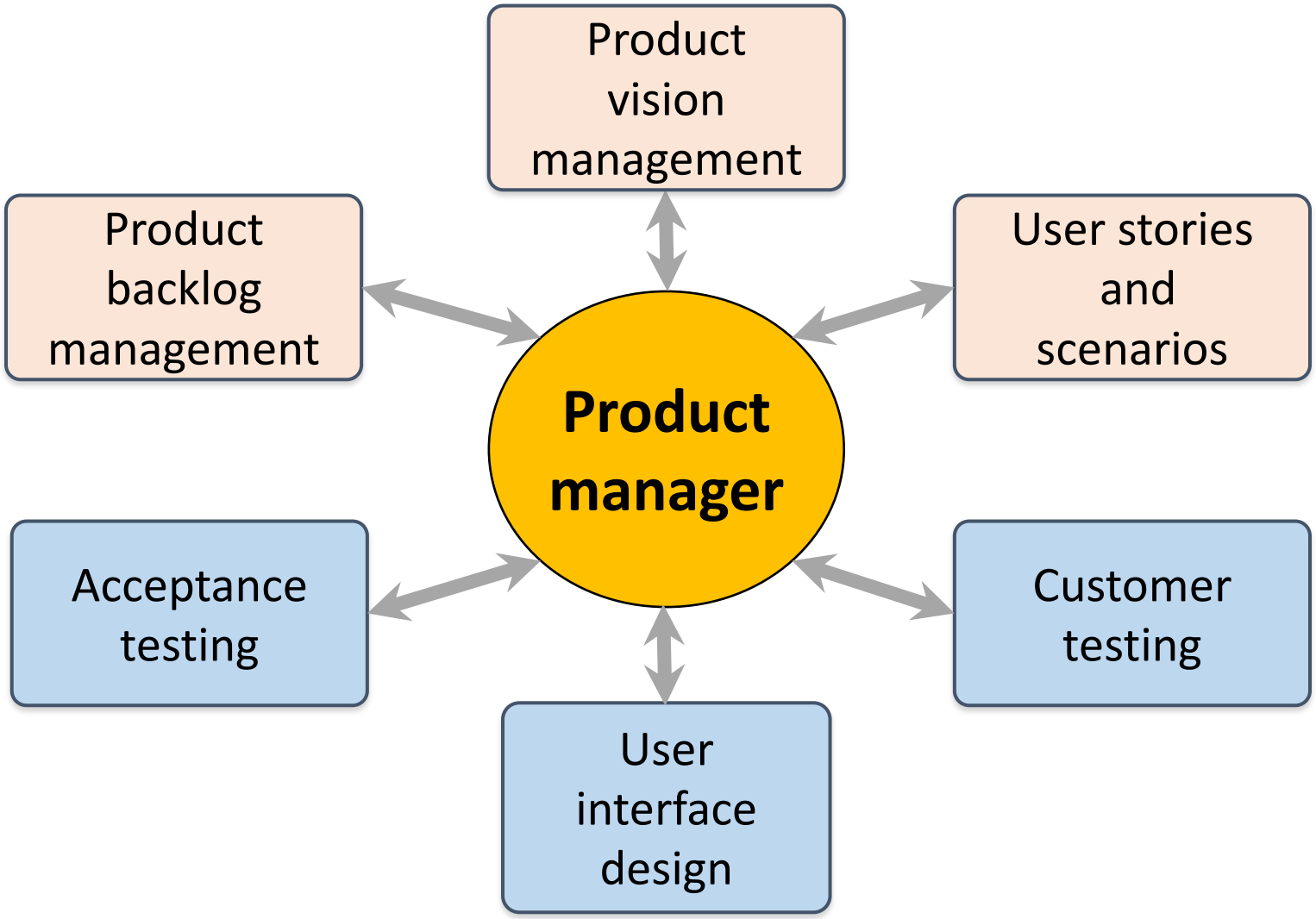
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Product management concerns



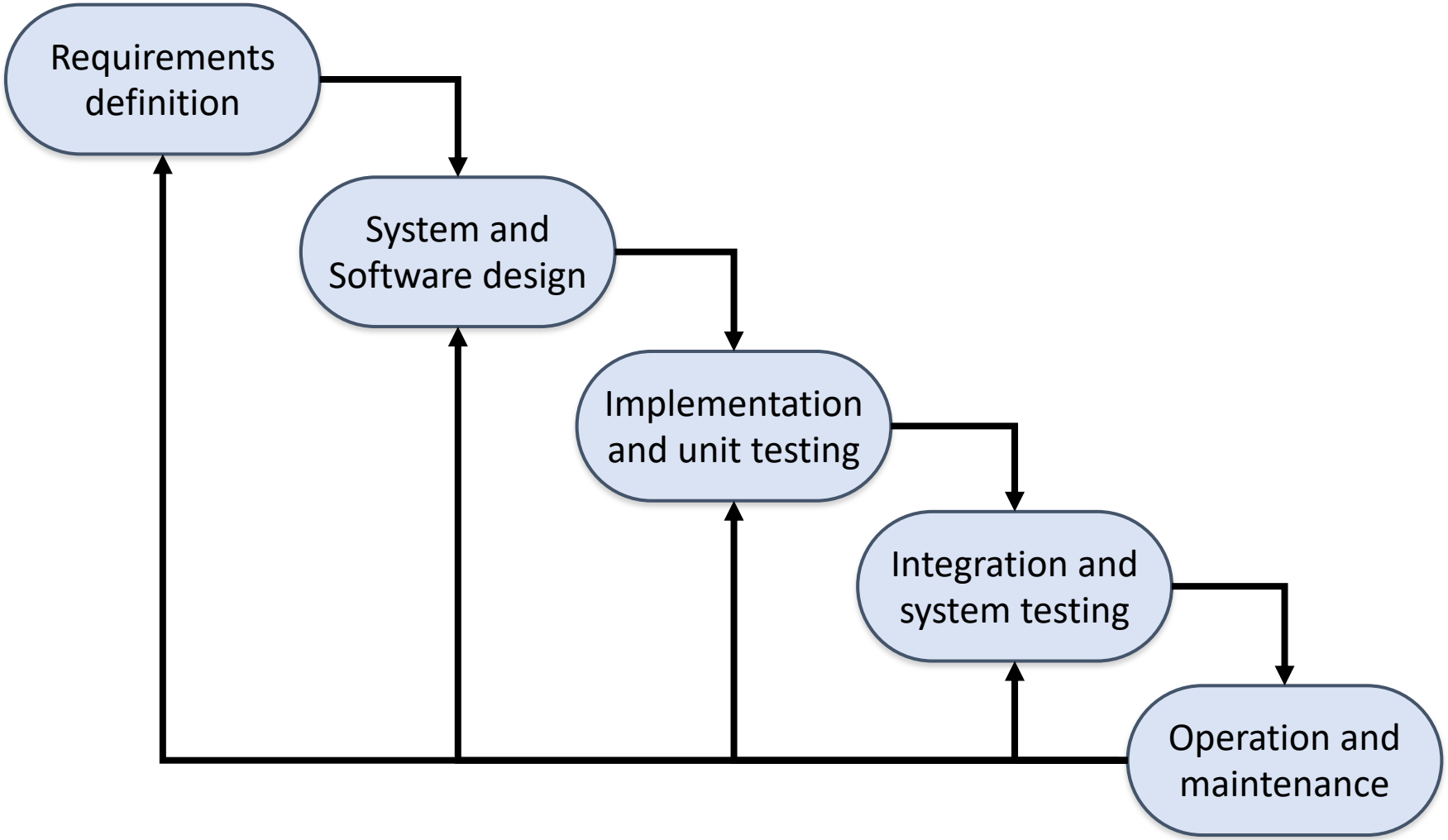
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Technical interactions of product managers



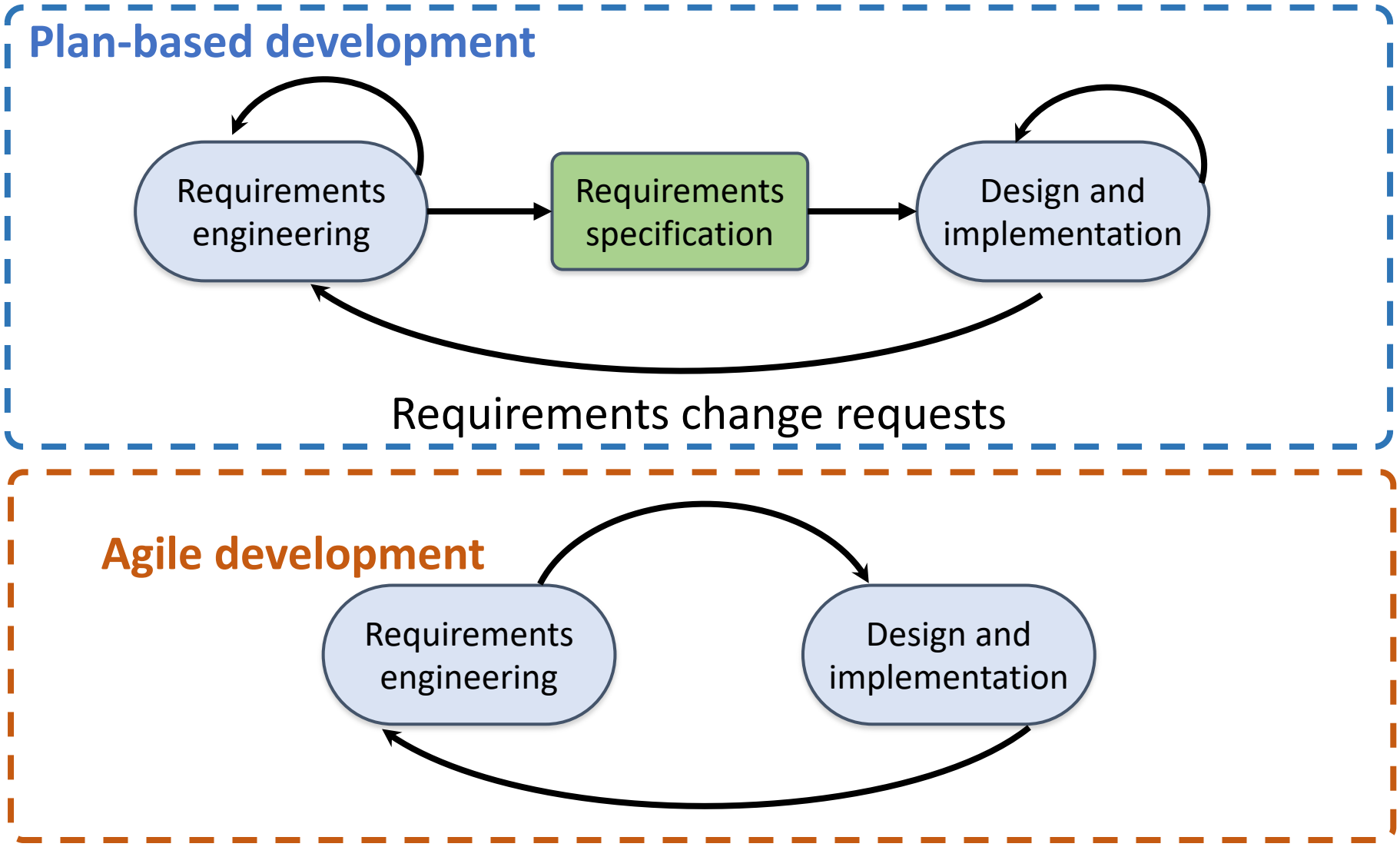
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software Development Life Cycle (SDLC)
The waterfall model



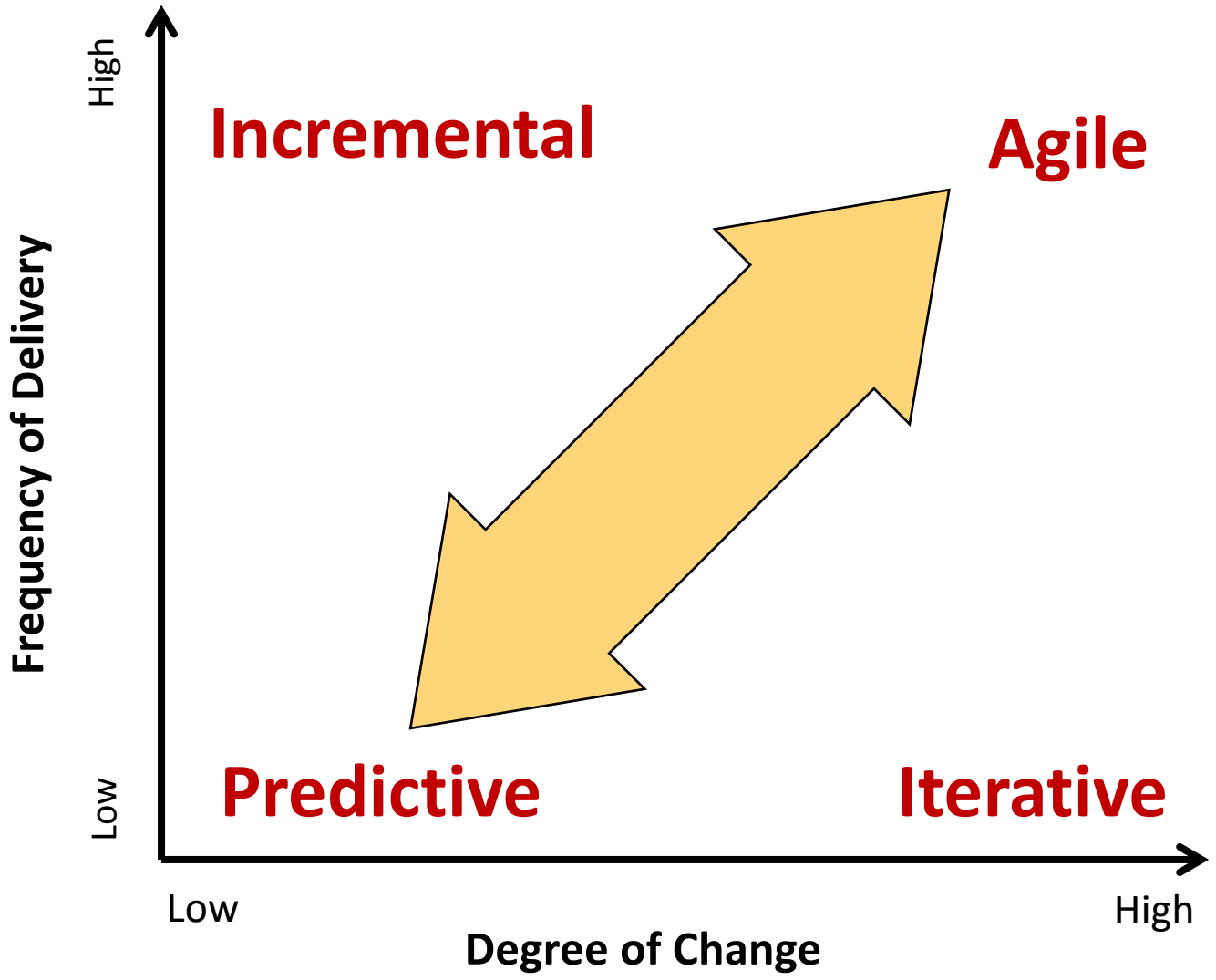
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Plan-based and Agile development



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

The Continuum of Life Cycles

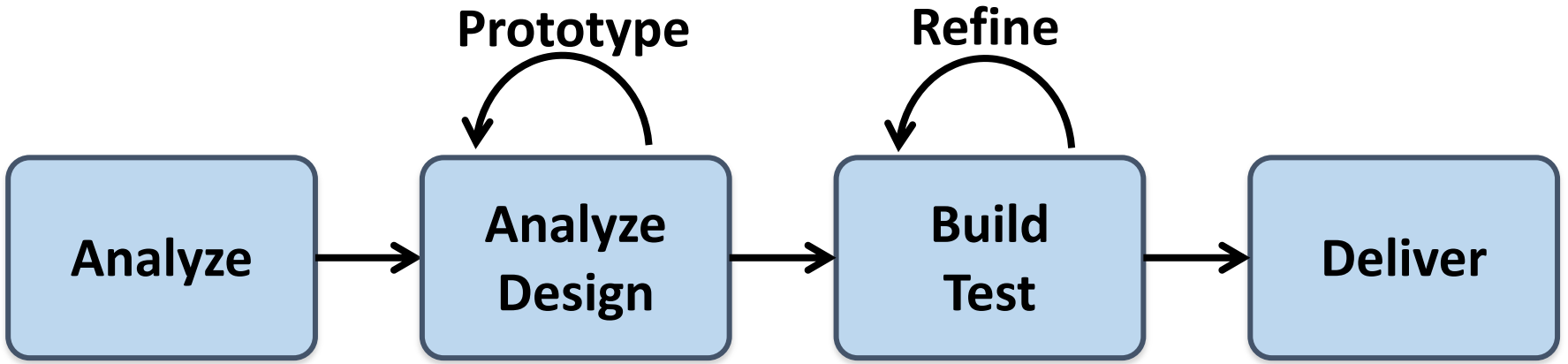


Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Predictive Life Cycle

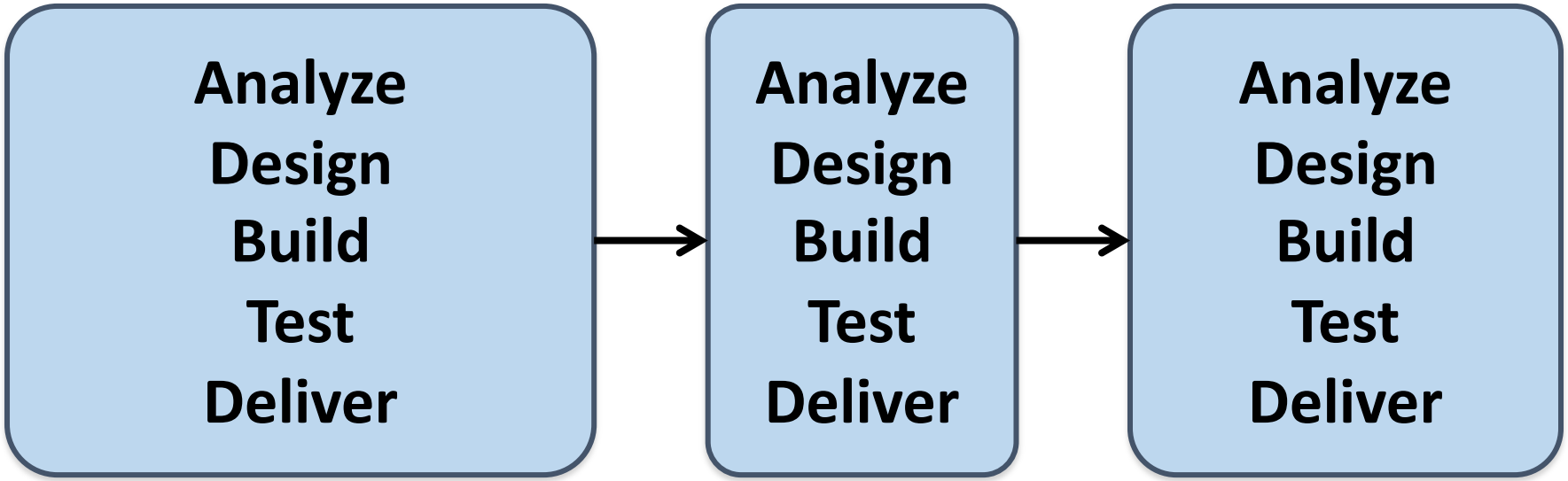
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Iterative Life Cycle



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

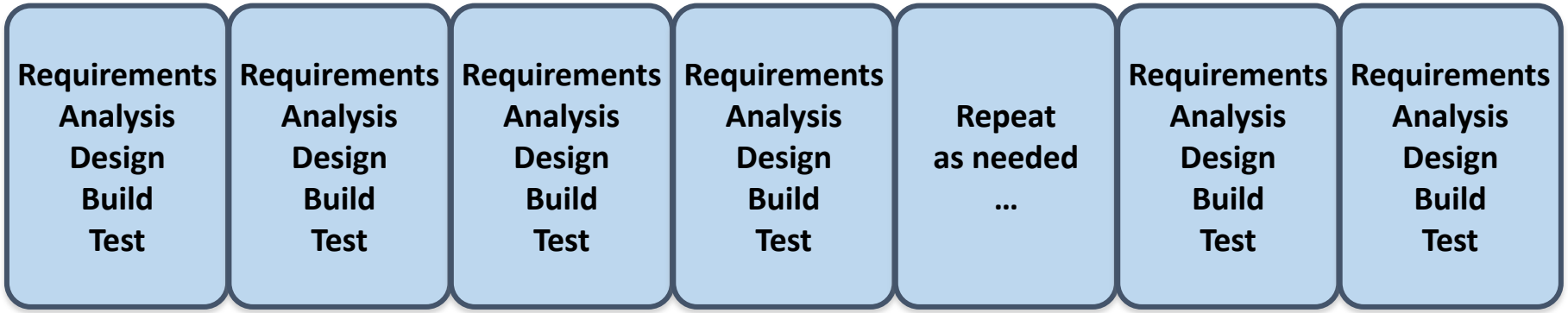
A Life Cycle of Varying-Sized Increments



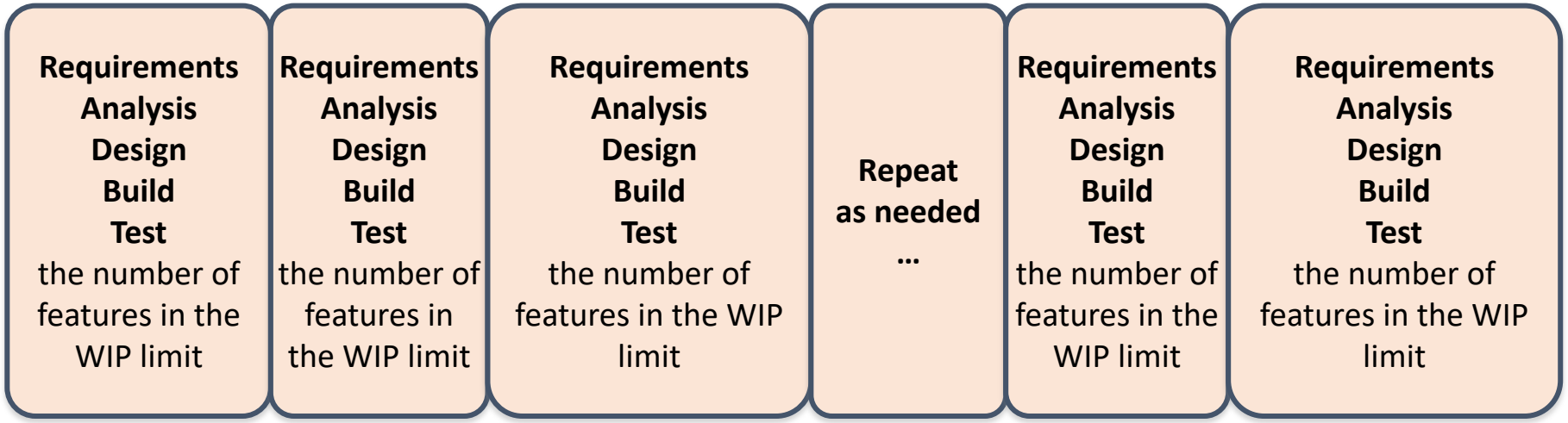
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Iteration-Based and Flow-Based Agile Life Cycles

Iteration-Based Agile

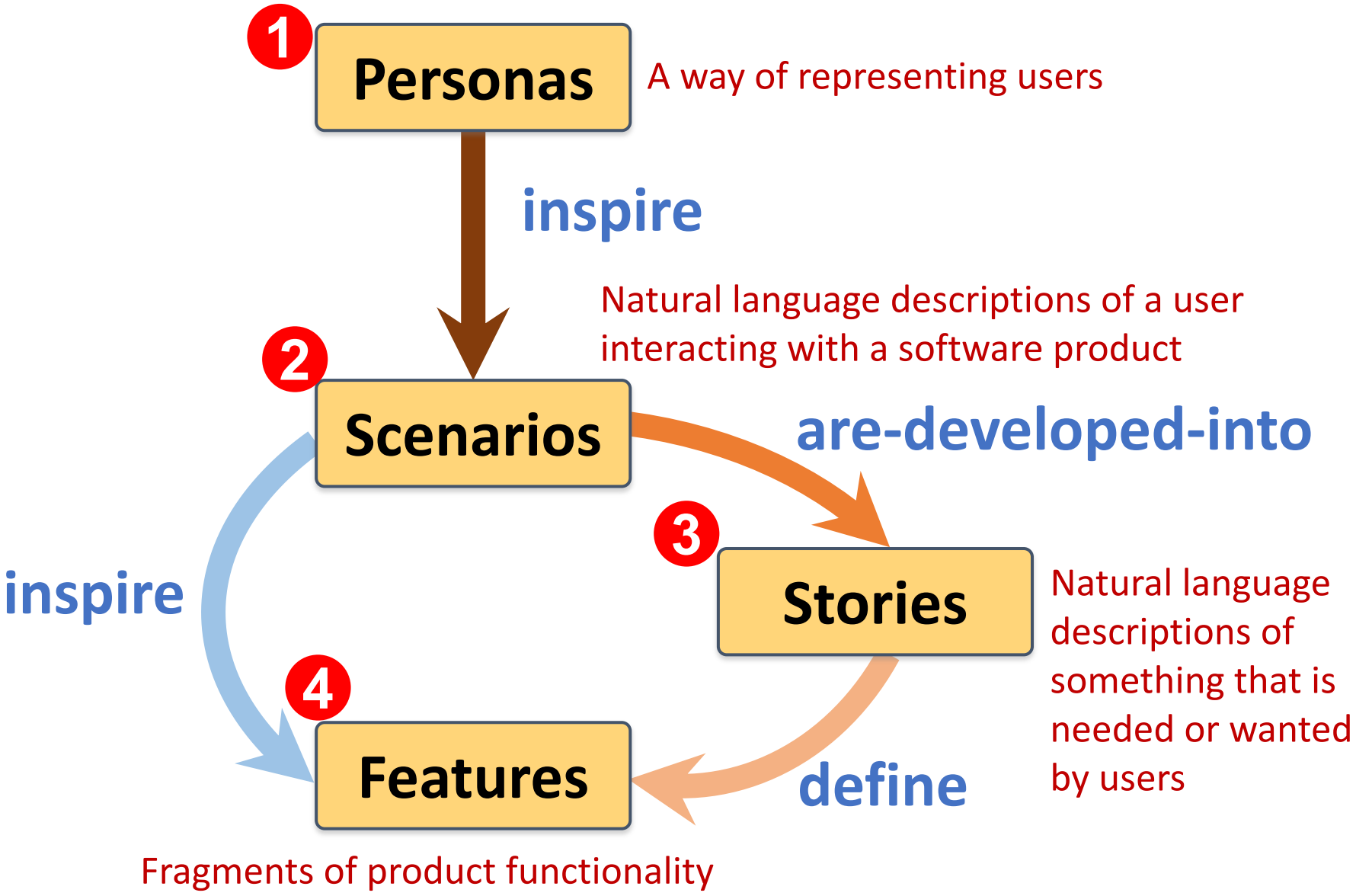


Flow-Based Agile



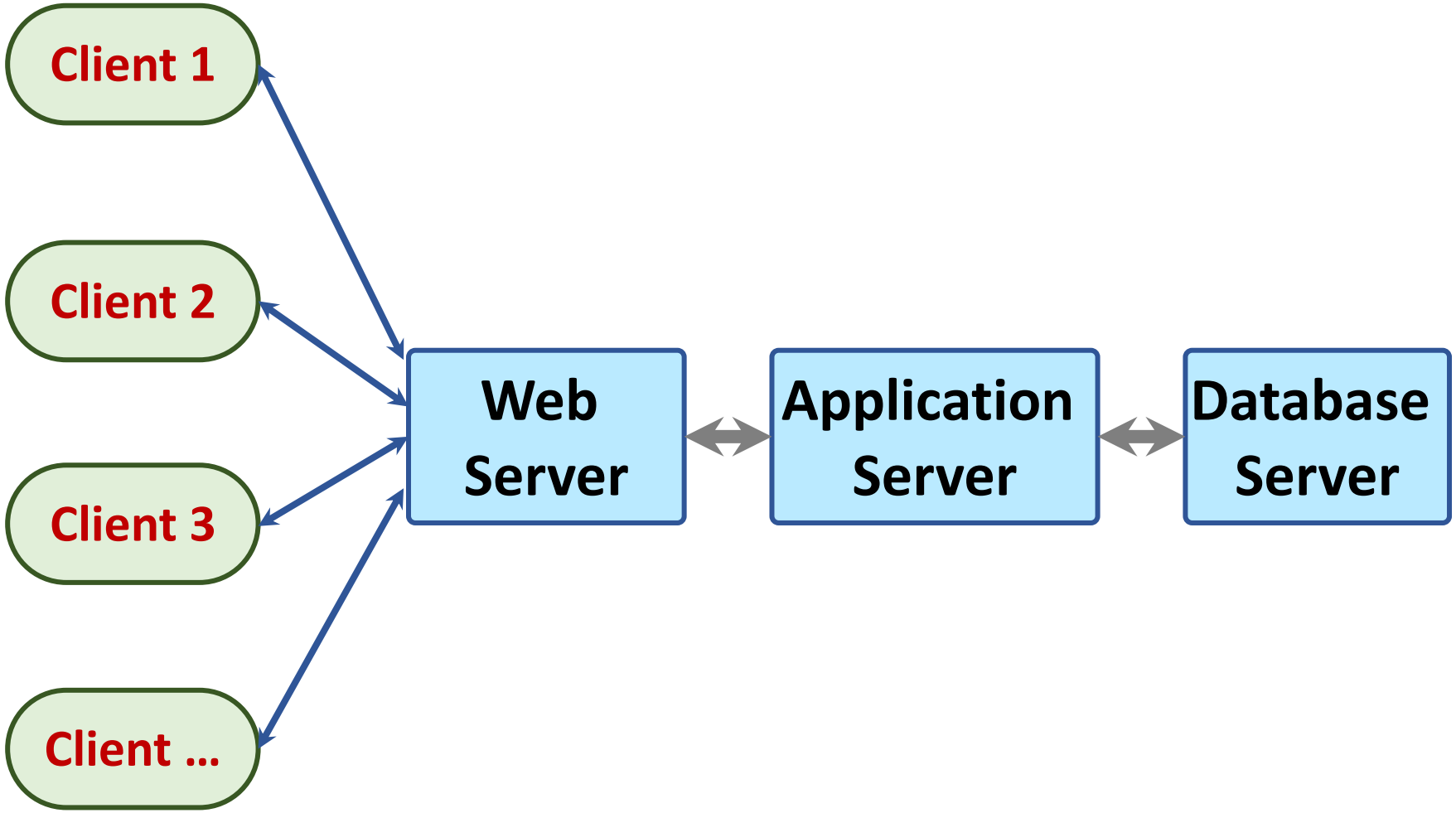
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

From personas to features



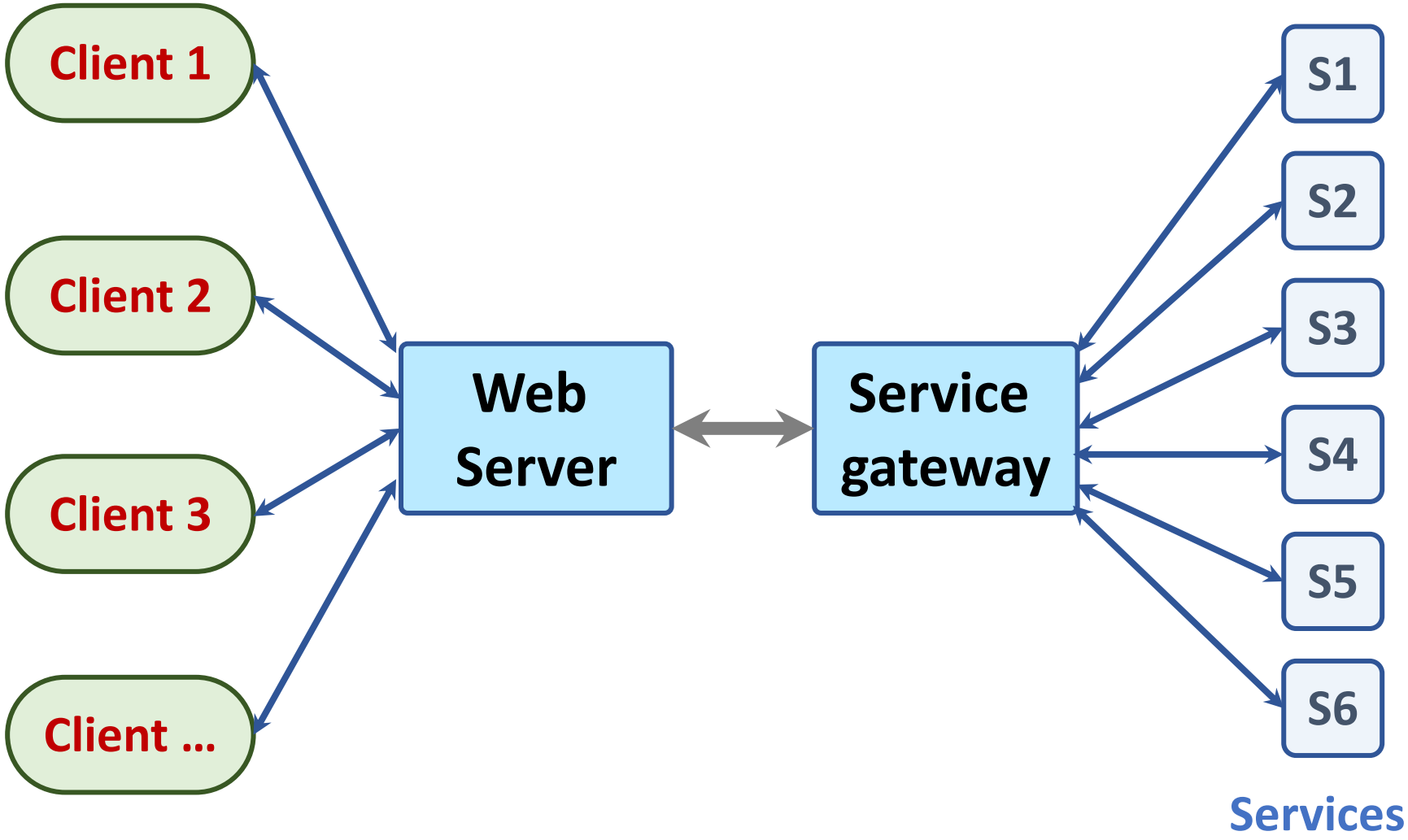
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Multi-tier client-server architecture

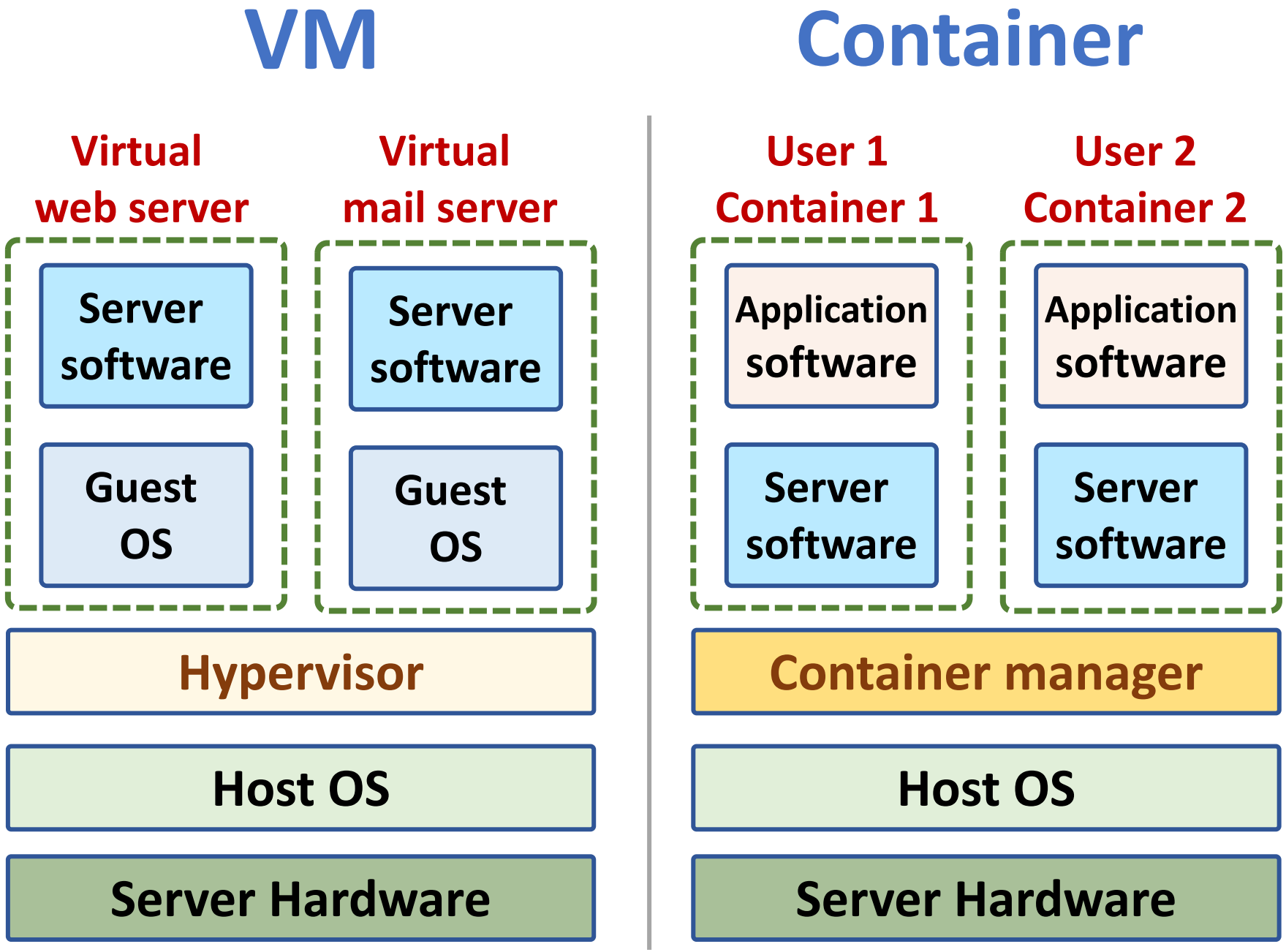


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Service-oriented Architecture

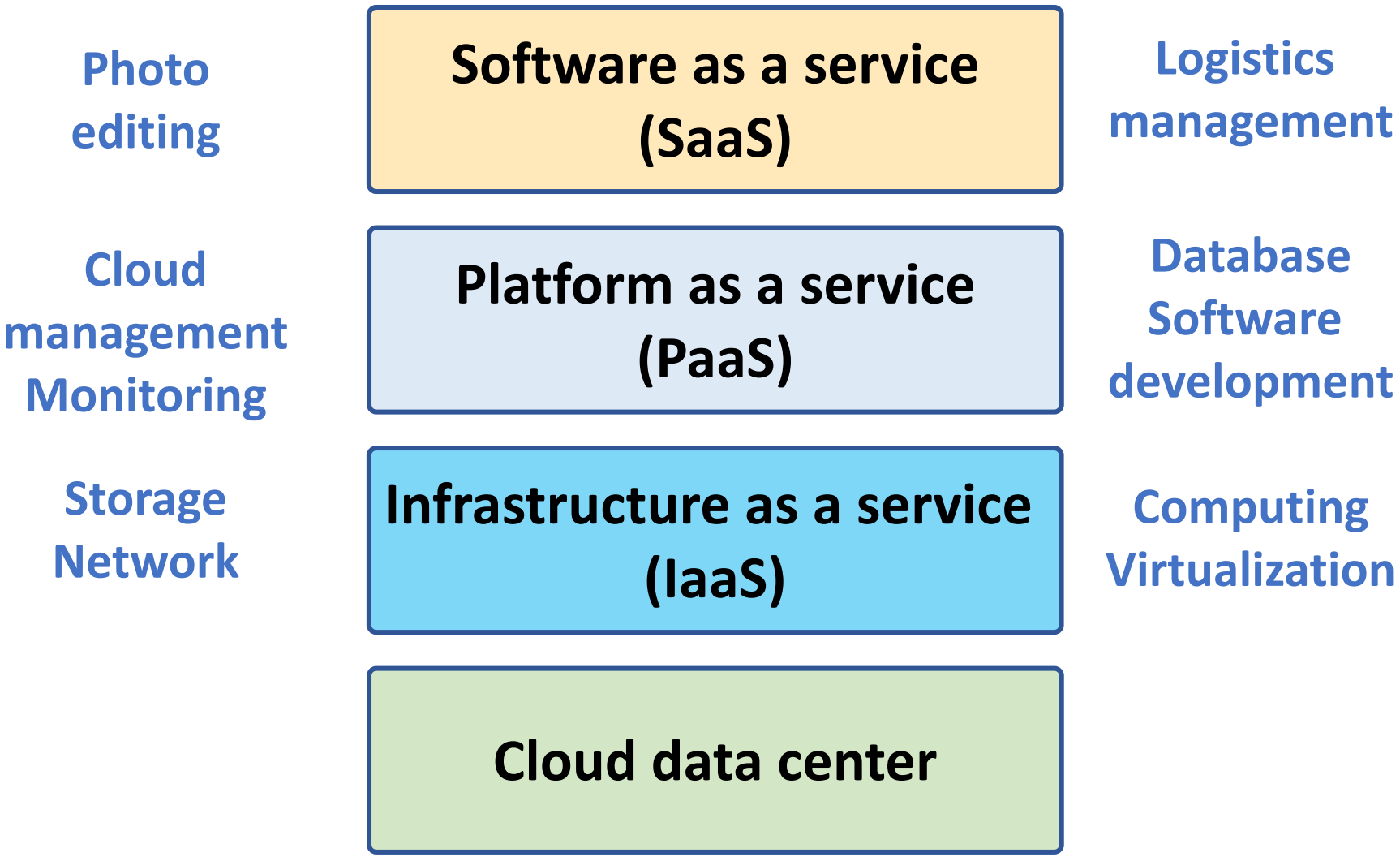


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.



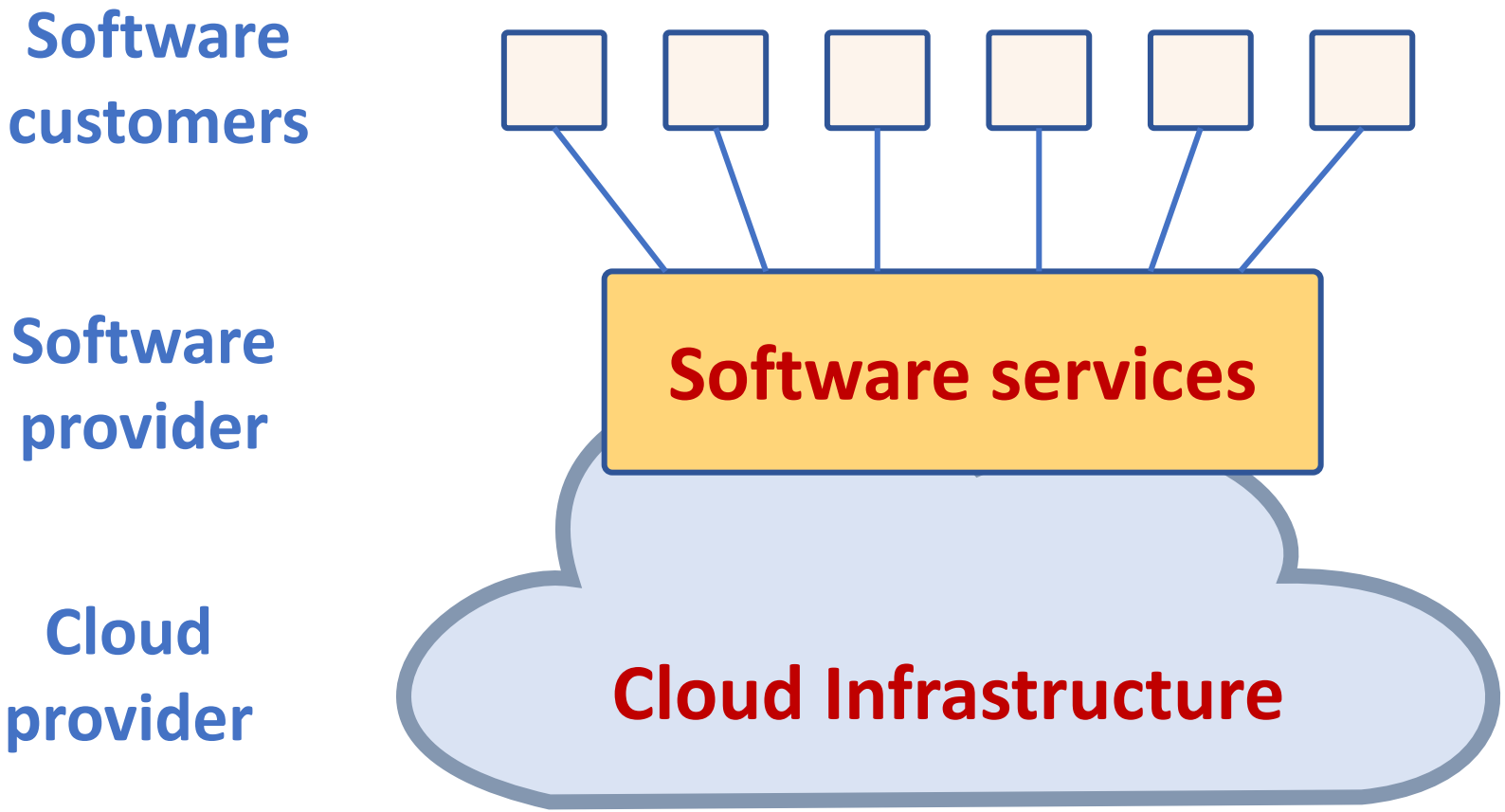
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Everything as a service



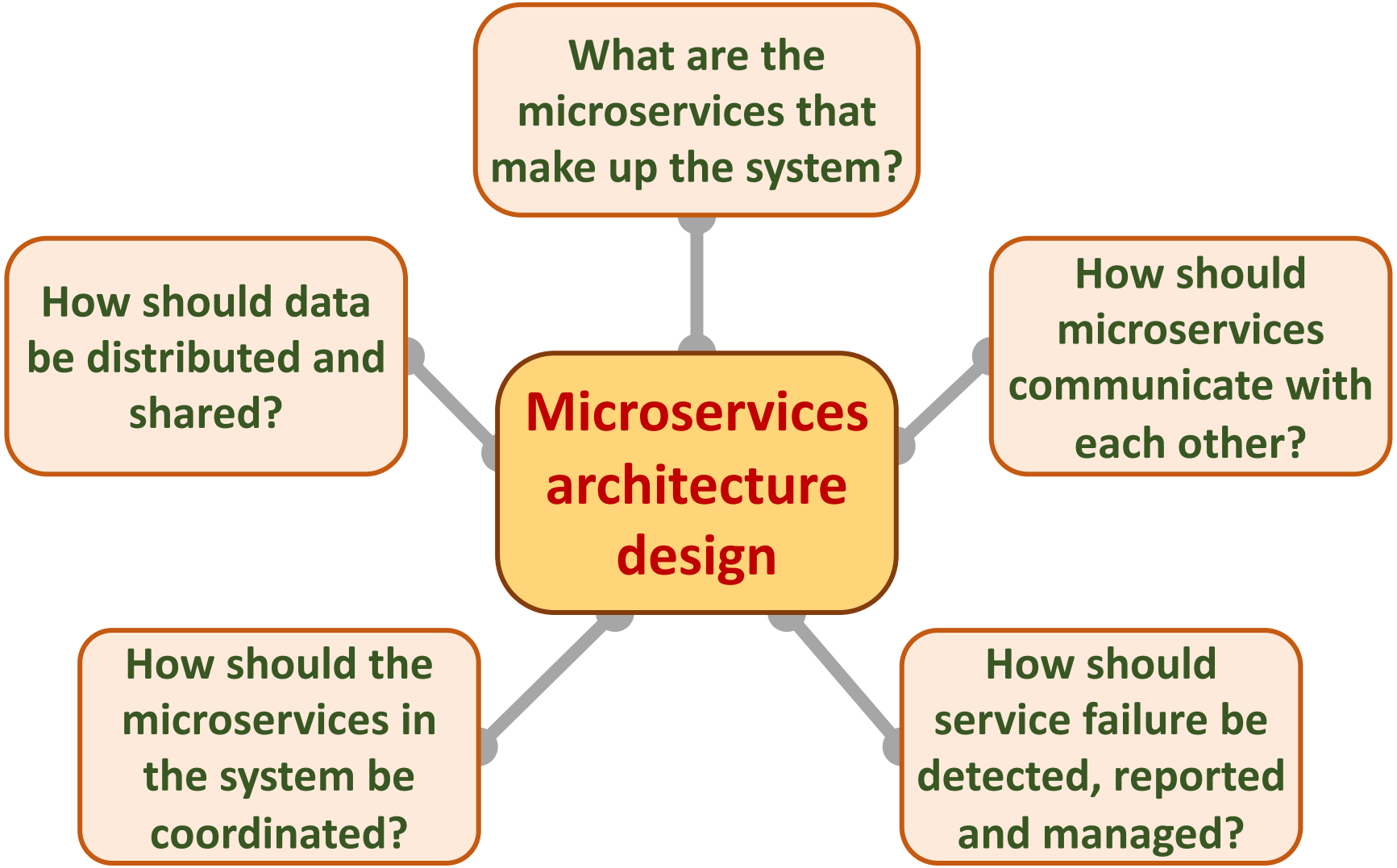
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software as a service



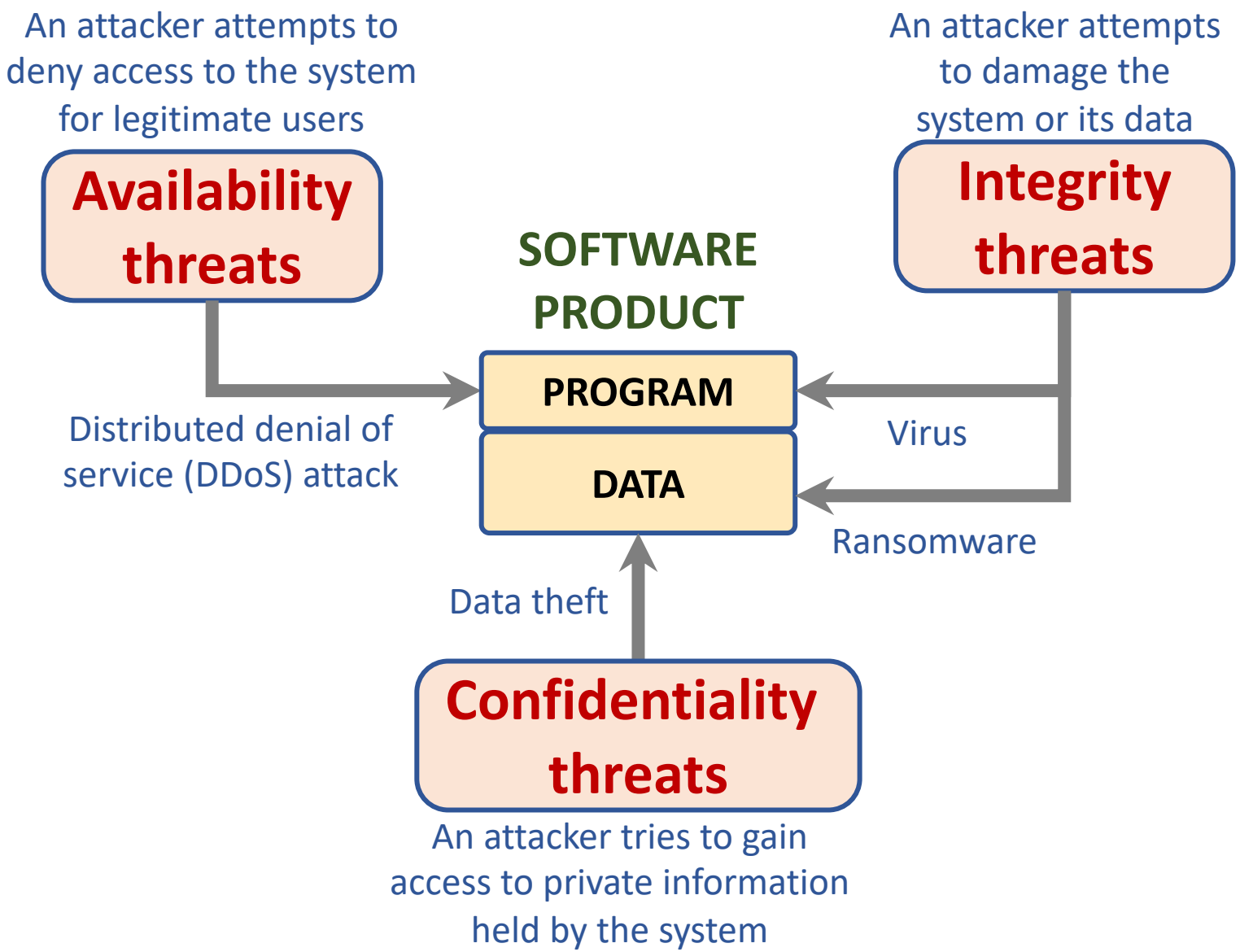
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Microservices architecture – key design questions



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

Types of security threat

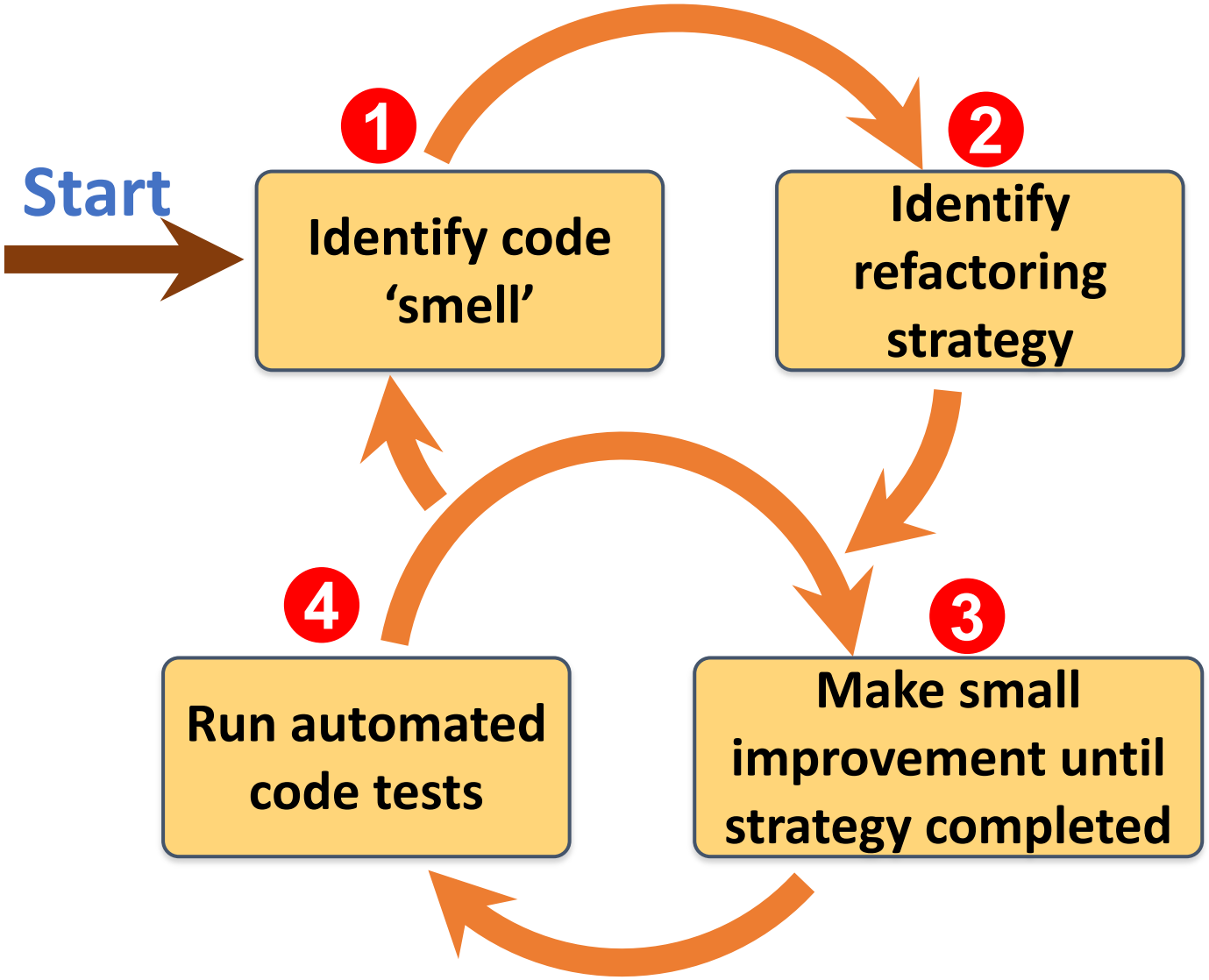


Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Software product quality attributes

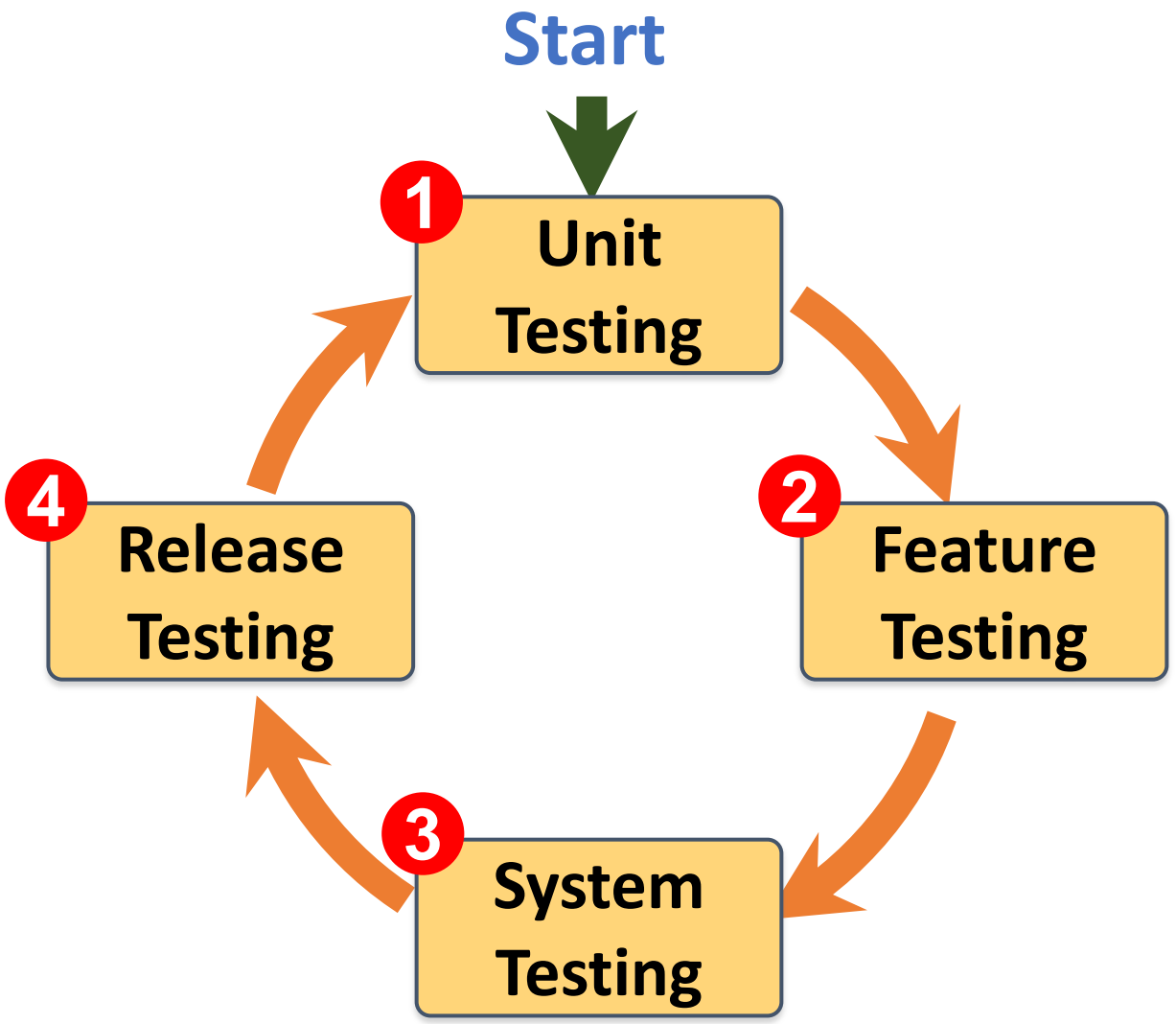
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

A refactoring process



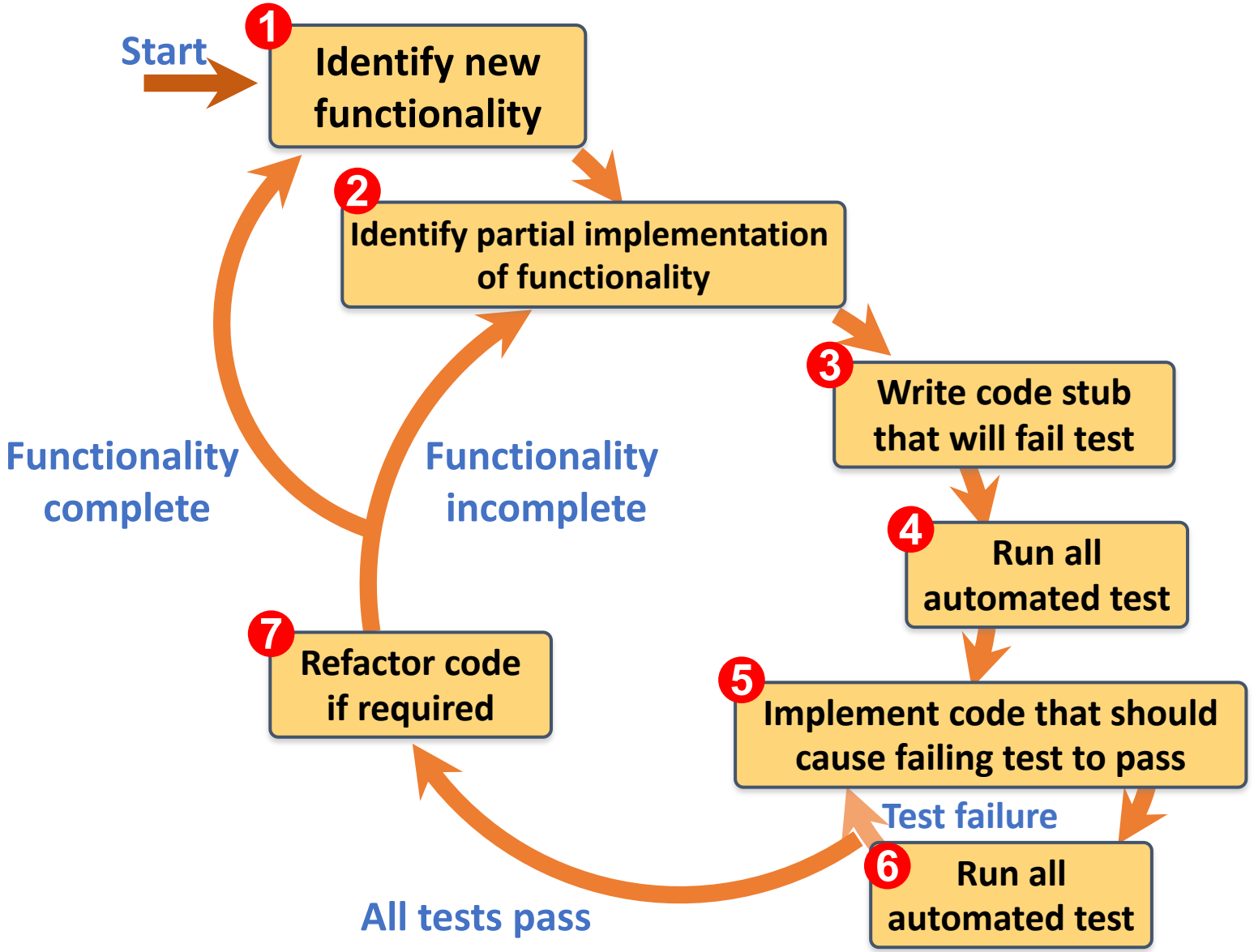
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Functional testing



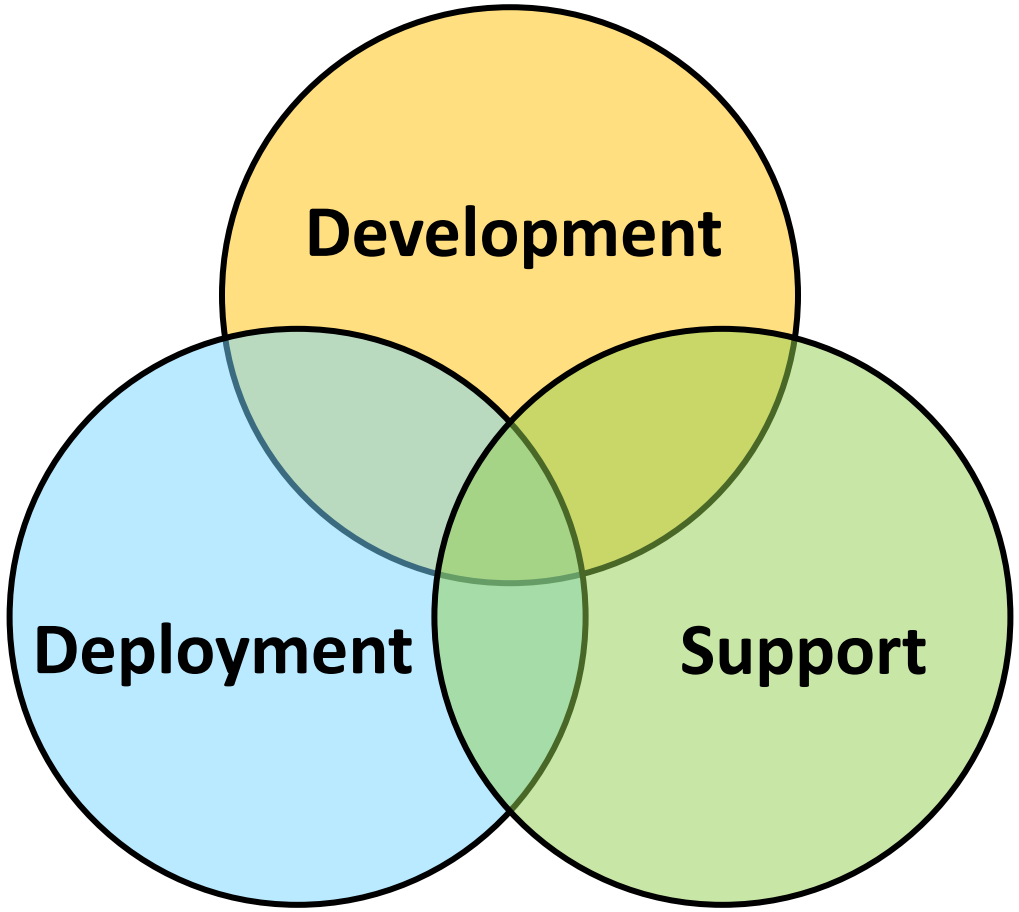
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

Test-driven development (TDD)



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

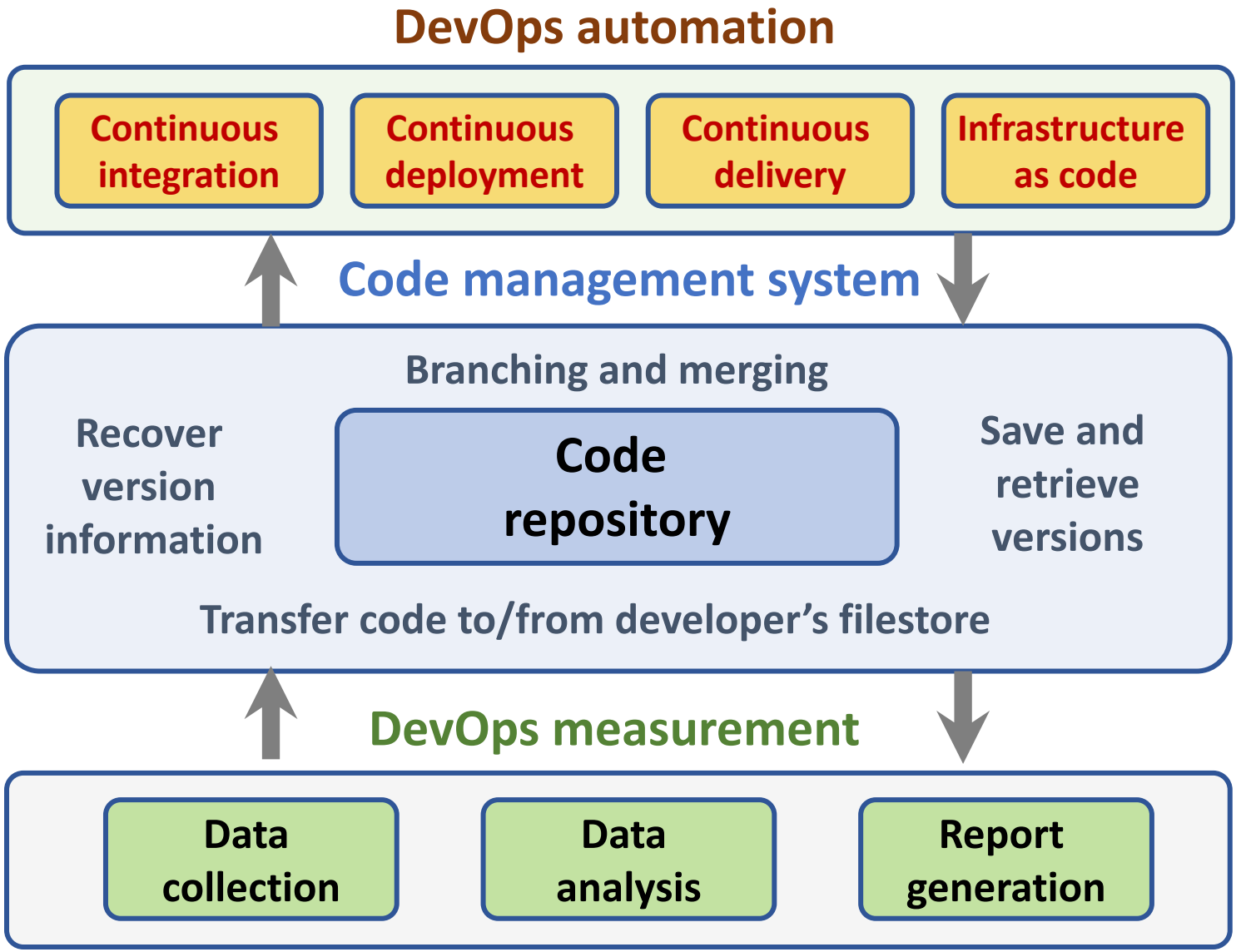
DevOps



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.

GitHub Copilot CODING AGENT

Source: https://github.blog/news-insights/product-news/github-copilot-meet-the-new-coding-agent/

SWE-bench Verified Leaderboard

AI coding agents ranking by software engineering problems resolved rate

Model	% Resolved	Org	Date
NEW live-SWE-agent + Claude 4.5 Opus medium (20251101)	79.20		2025-12-15
TRAE + Doubao-Seed-Code	78.80		2025-09-28
NEW ✅ live-SWE-agent + Gemini 3 Pro Preview (2025-11-18)	77.40		2025-11-20
NEW mini-SWE-agent + Claude 4.5 Opus (high reasoning)	76.80		2026-02-17
NEW mini-SWE-agent + Gemini 3 Flash (high reasoning)	75.80		2026-02-17
NEW mini-SWE-agent + MiniMax M2.5 (high reasoning)	75.80		2026-02-17
NEW mini-SWE-agent + Claude Opus 4.6	75.60		2026-02-17
TRAE + Claude Sonnet 4 + Opus 4 + Sonnet 3.7 + Gemini 2.5 Pro	75.20		2025-06-12
Lingxi-v1.5_claude-4-sonnet-20250514	74.60		2025-07-20
JoyCode + Claude 4 Sonnet + GPT-4.1	74.60		2025-09-15

Source: https://www.swebench.com/

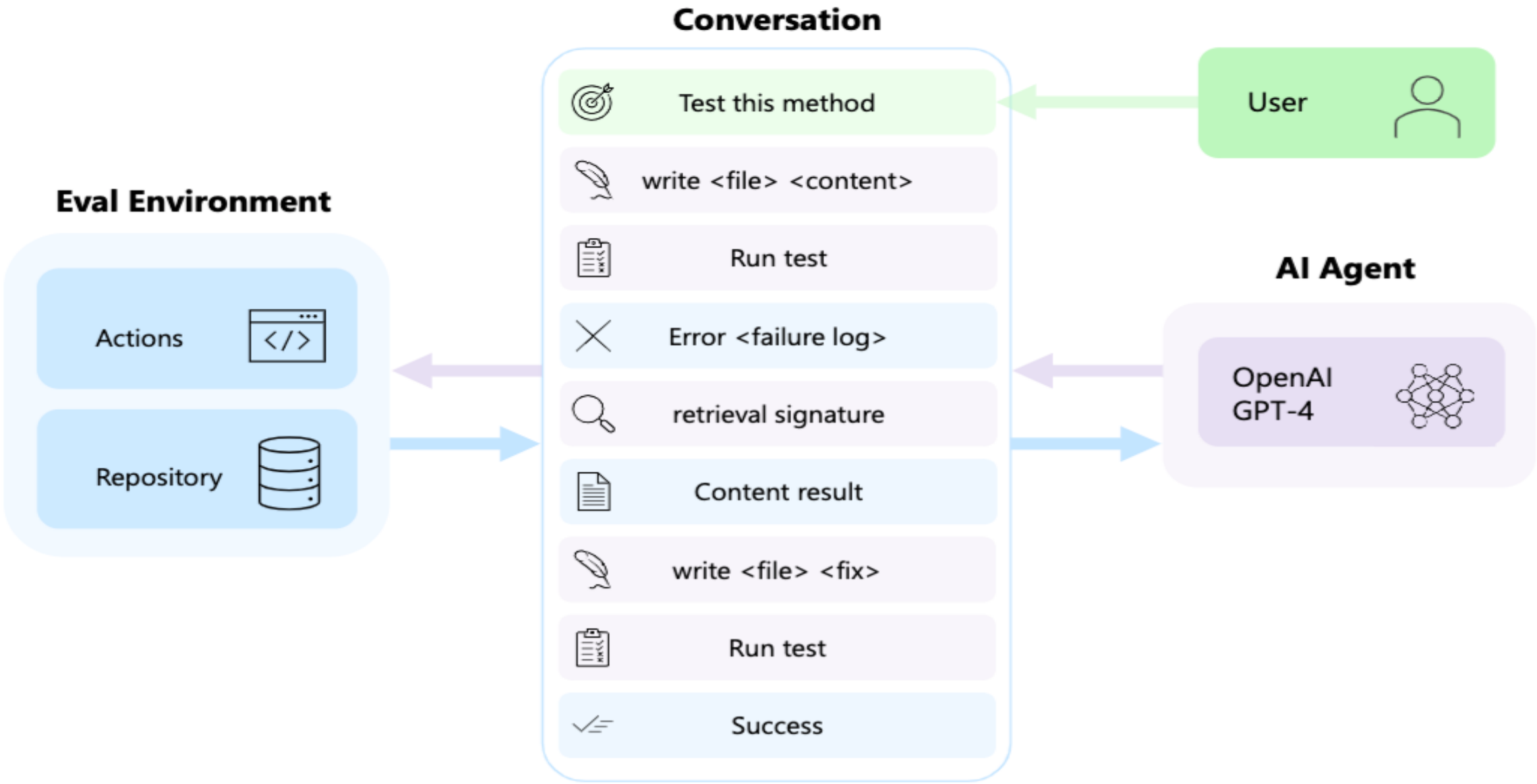
AI WebDev Code Arena

AI models on agentic coding tasks (multi-step reasoning and tool use)

Rank	Rank Spread	Model	Score	Votes
1	1 – 3	claude-opus-4-6 Anthropic · Proprietary	1560 +14/-14	2,766
2	1 – 3	claude-opus-4-6-thinking Anthropic · Proprietary	1553 +15/-15	2,115
3	1 – 3	claude-sonnet-4-6 Anthropic · Proprietary	1533 +16/-16	1,675
4	4 – 4	claude-opus-4-5-20251101-thinking-32k Anthropic · Proprietary	1499 +8/-8	11,032
5	5 – 8	gpt-5.2-high OpenAI · Proprietary	1471 +16/-16	1,696
6	5 – 8	claude-opus-4-5-20251101 Anthropic · Proprietary	1470 +8/-8	11,113
7	5 – 13	gemini-3.1-pro-preview Google · Proprietary	1461 +15/-15	1,826
8	5 – 13	glm-5 Z.ai · MIT	1452 +13/-13	2,520
9	7 – 13	gemini-3-pro Google · Proprietary	1444 +7/-7	16,948
10	7 – 13	gemini-3-flash Google · Proprietary	1440 +8/-8	12,778

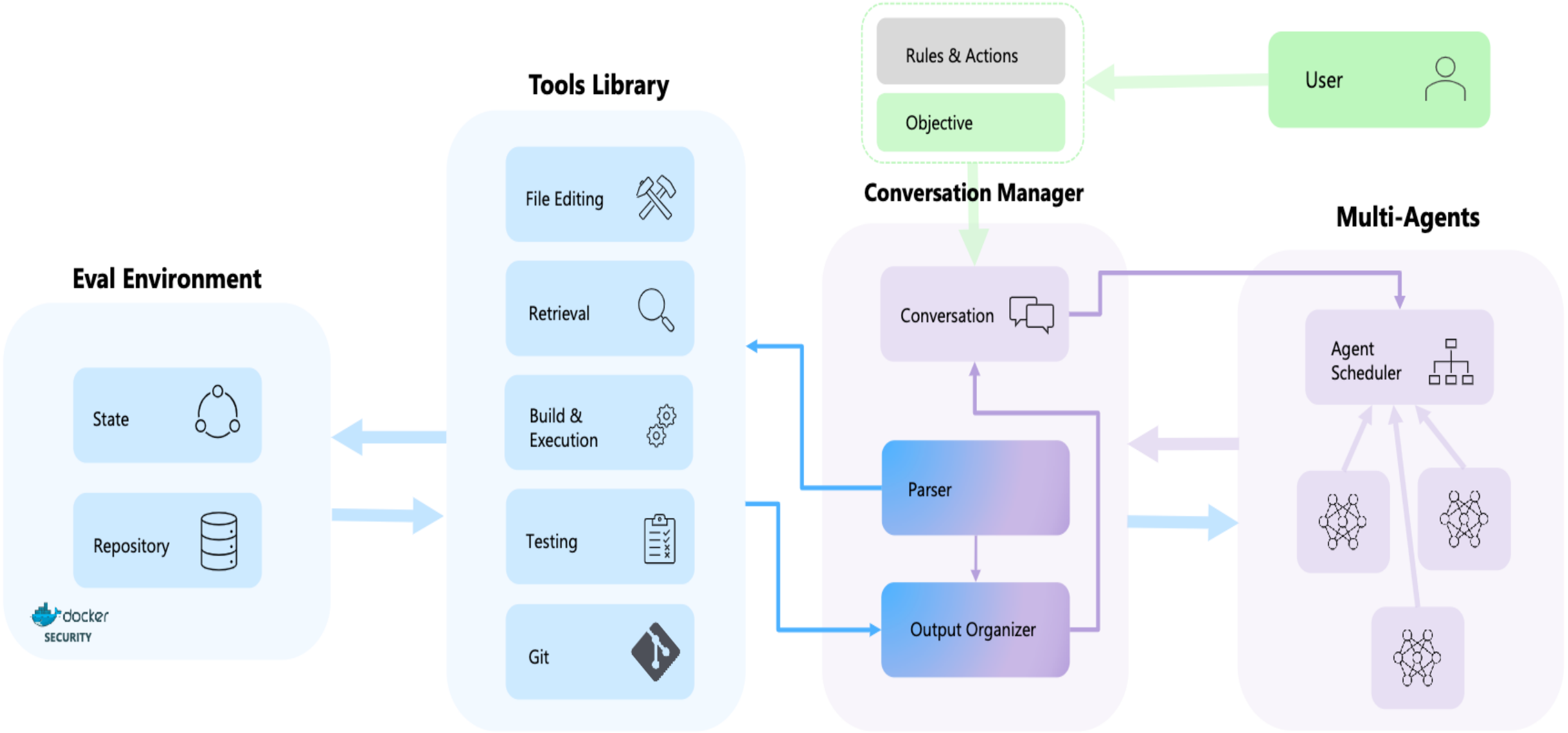
Source: https://arena.ai/leaderboard/code

AutoDev: Automated AI-Driven Development



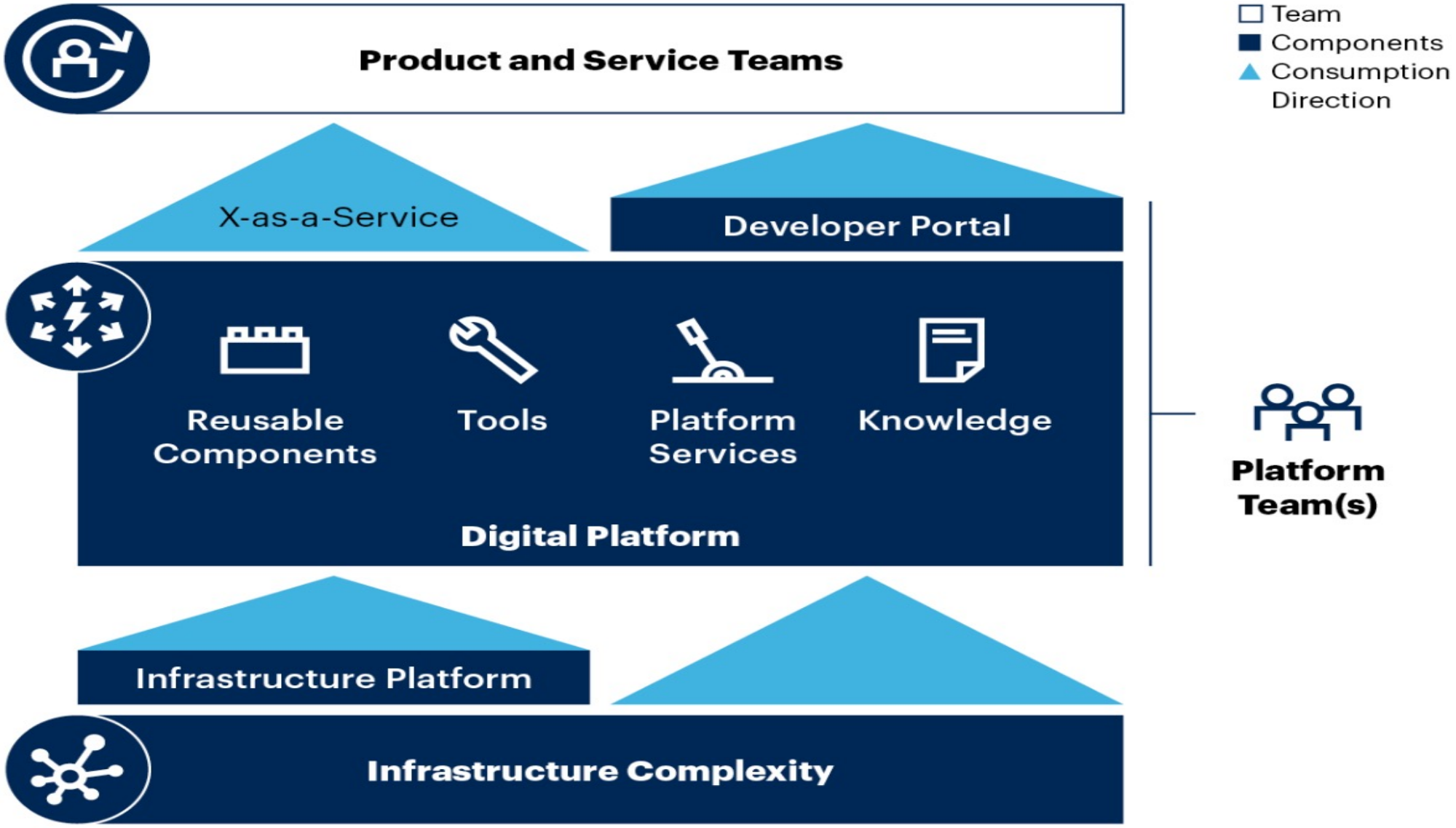
Source: Tufano, Michele, Anisha Agarwal, Jinu Jang, Roshanak Zilouchian Moghaddam, and Neel Sundaresan. (2024) "AutoDev: Automated AI-Driven Development." arXiv preprint arXiv:2403.08299 (2024).

AutoDev: Automated AI-Driven Development



Source: Tufano, Michele, Anisha Agarwal, Jinu Jang, Roshanak Zilouchian Moghaddam, and Neel Sundaresan. (2024) "AutoDev: Automated AI-Driven Development." arXiv preprint arXiv:2403.08299 (2024).

Platform Engineering



Source: https://www.gartner.com/en/articles/what-is-platform-engineering

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.

OKRs, CFRs, KPIs
Agile Performance Management

- **OKRs**
  - **Objectives and Key Results**
- **CFRs**
  - **Conversations, Feedback, Recognition**
- **KPIs**
  - **Key Performance Indicators**

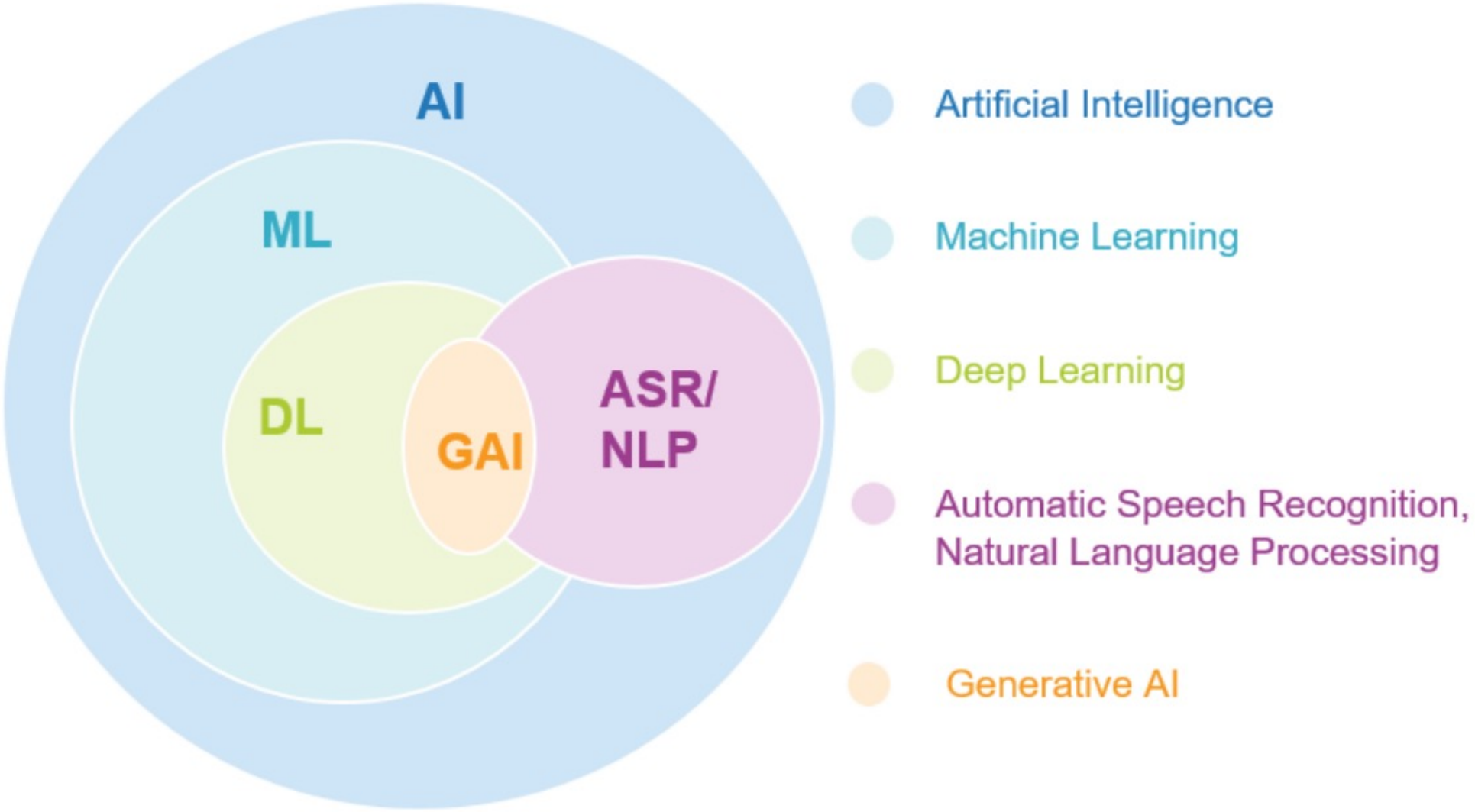
Agentic AI

for

Agile AI Software Engineering

Generative AI
Agentic AI
Physical AI

AI, ML, DL, Generative AI



Source: Jeong, Cheonsu. "A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture." arXiv preprint arXiv:2309.01105 (2023).

Generative AI, Agentic AI, Physical AI

2012 AlexNet
Deep learning breakthrough

Perception AI
Speech recognition
Deep recommender systems
Medical imaging

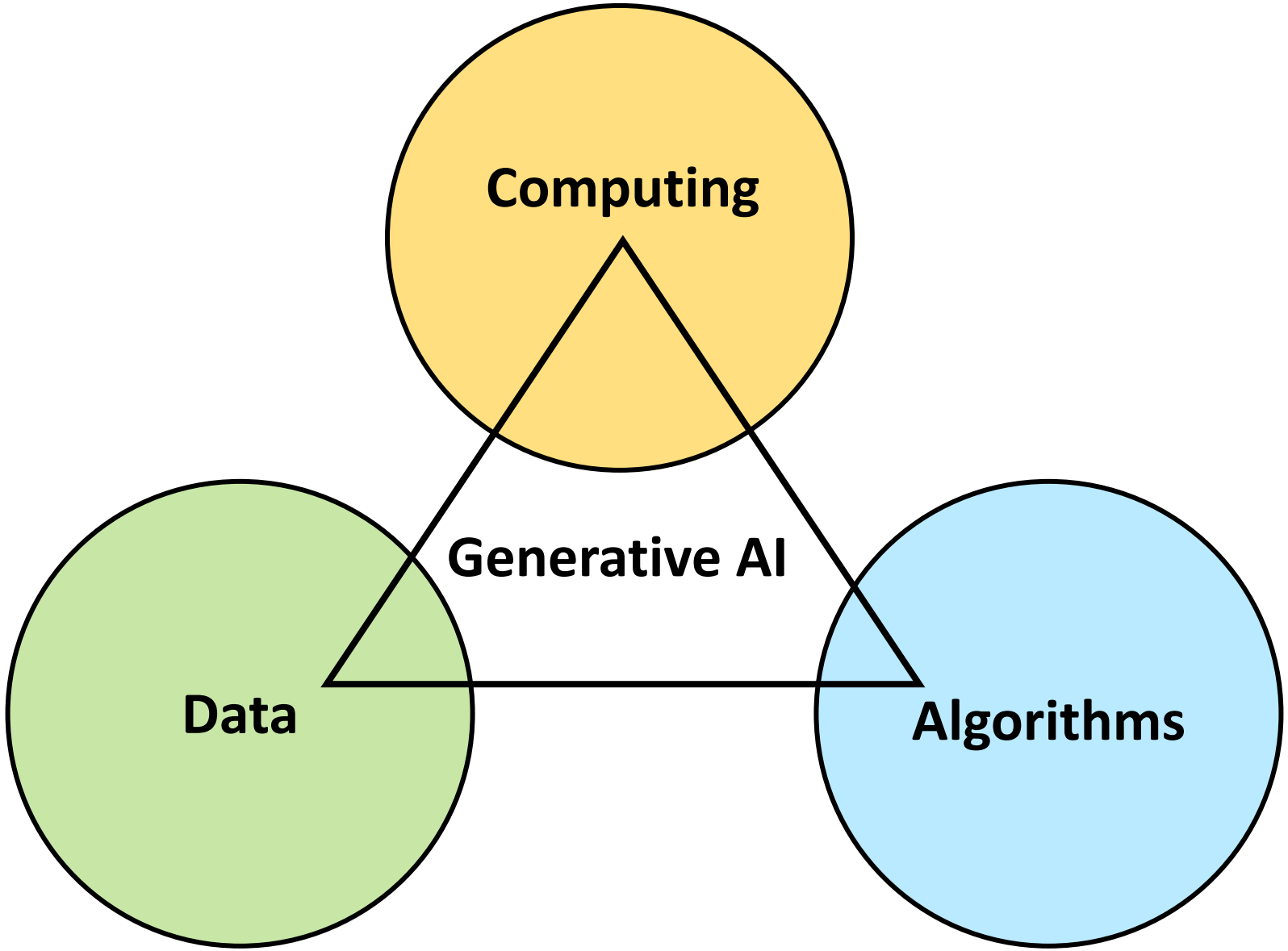
Generative AI
Digital marketing
Content creation

Agentic AI
Coding assistants
Customer service
Patient care

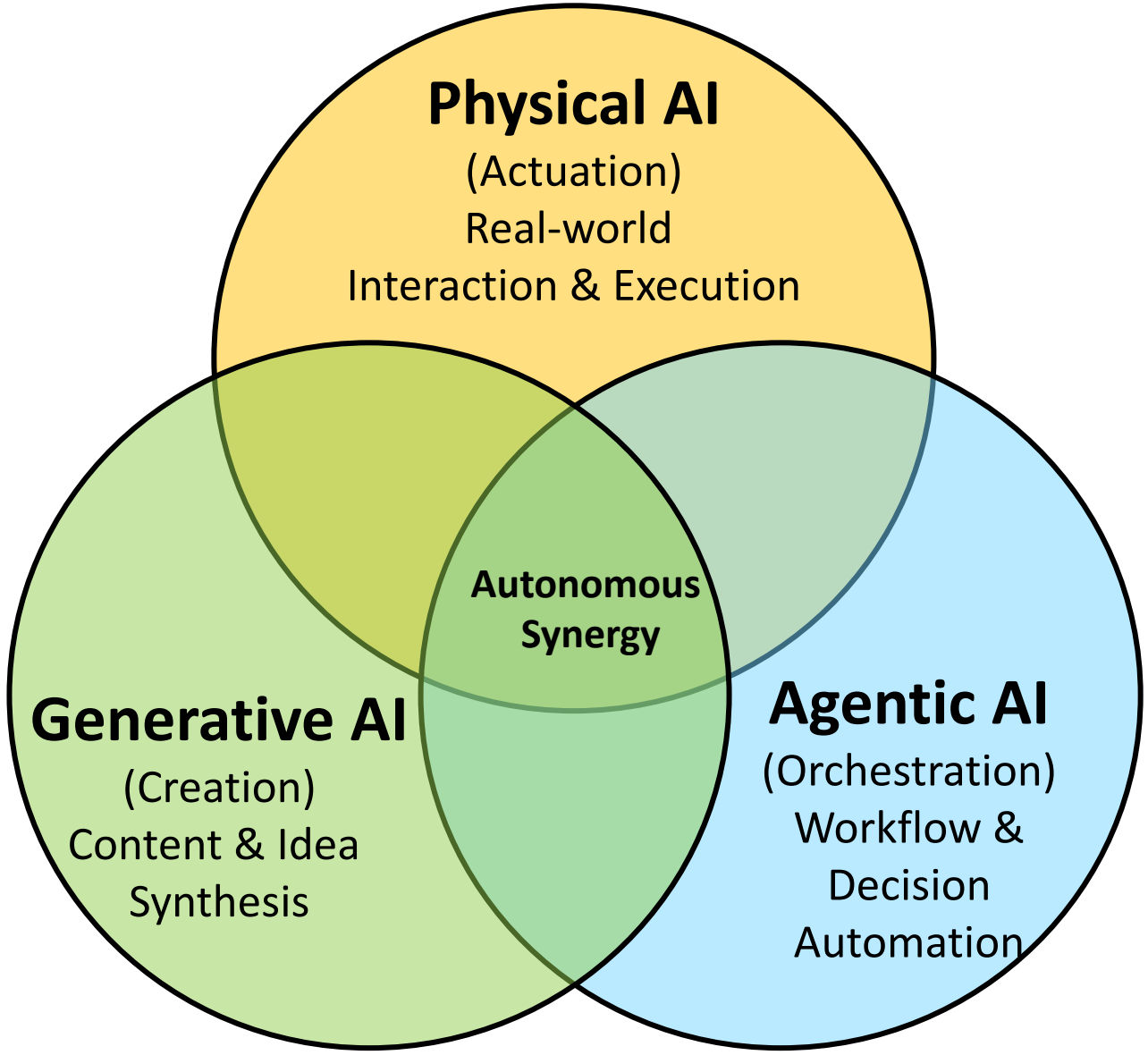
Physical AI
Self-driving cars
General robotics

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

Generative AI

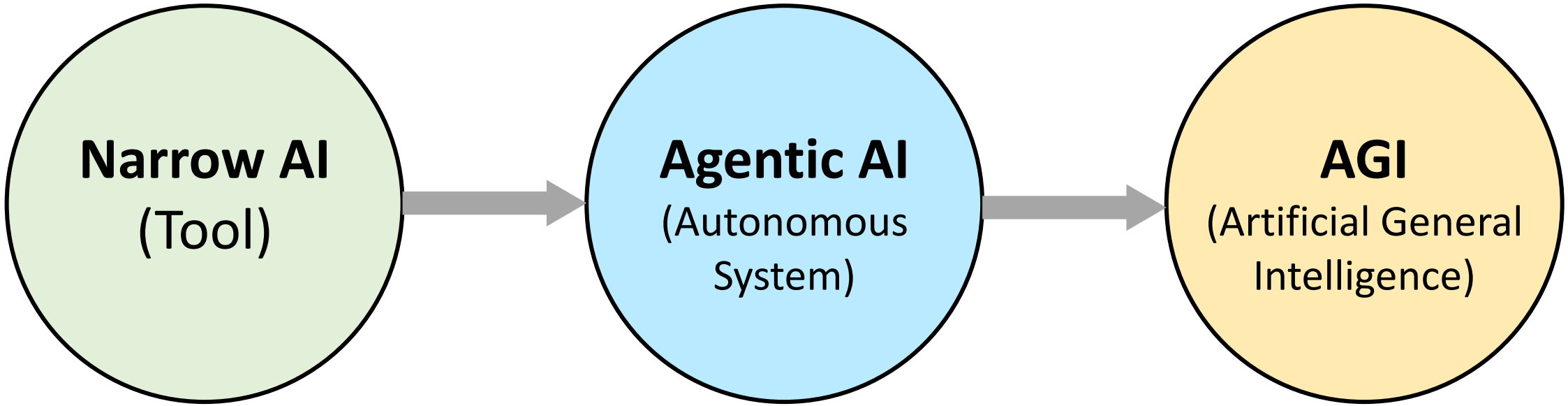


Generative AI, Agentic AI, Physical AI



New Economic Paradigm Shift: From Creation to Execution

The Future of AI
From Tools to Agents:
The Rise and Autonomy of Agentic AI

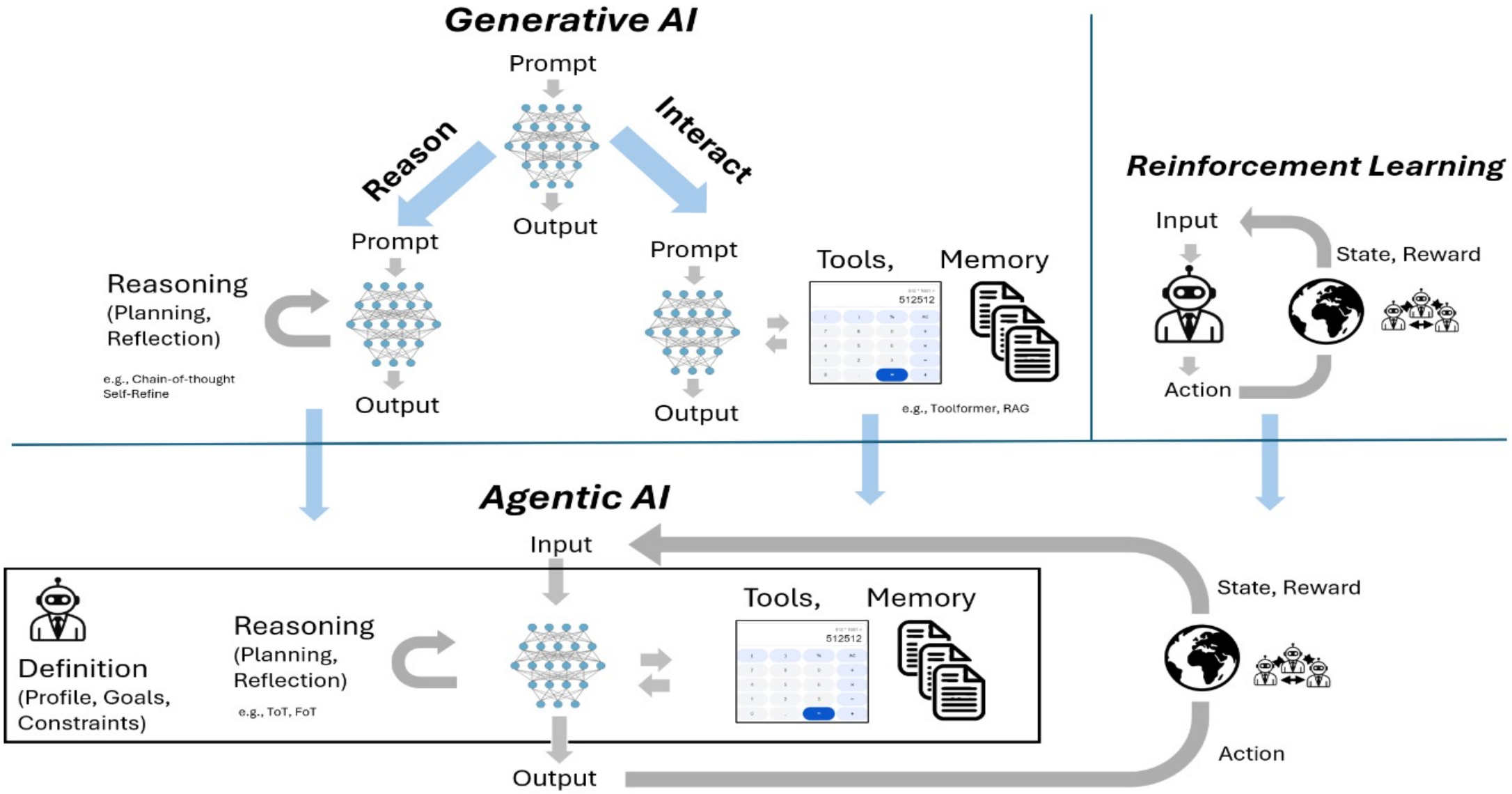


Executes Specific Tasks,
Content Generation

Goal-Oriented Behavior,
Autonomous Decision-Making,
Multi-Step Workflows,
Self-Improvement within Parameters

Connecting Multimodal GenAI
to Future AGI

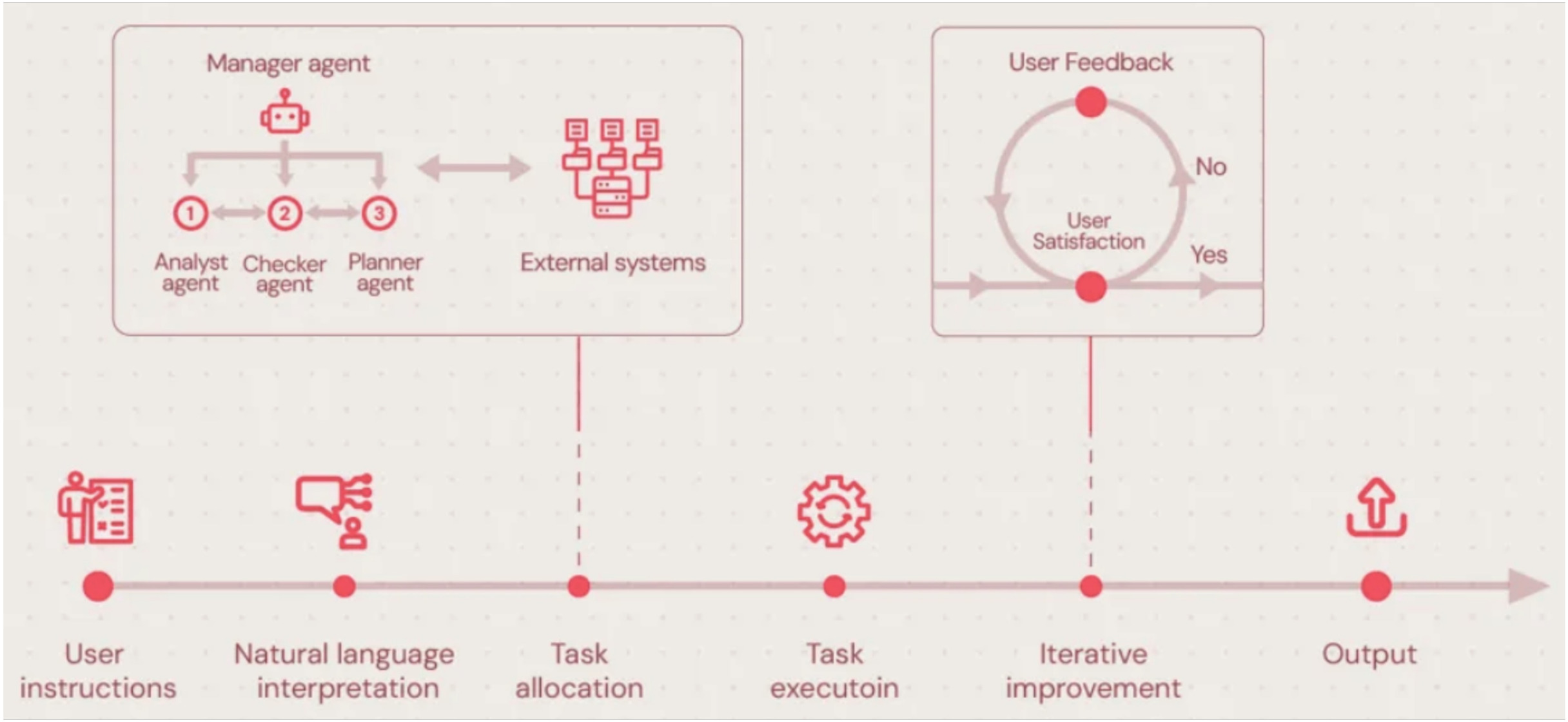
From Generative AI to Agentic AI



Source: Schneider, Johannes. "Generative to Agentic AI: Survey, Conceptualization, and Challenges." arXiv preprint arXiv:2504.18875 (2025).

Agentic AI Systems
RAG Agents with LLM
Dialogue Systems

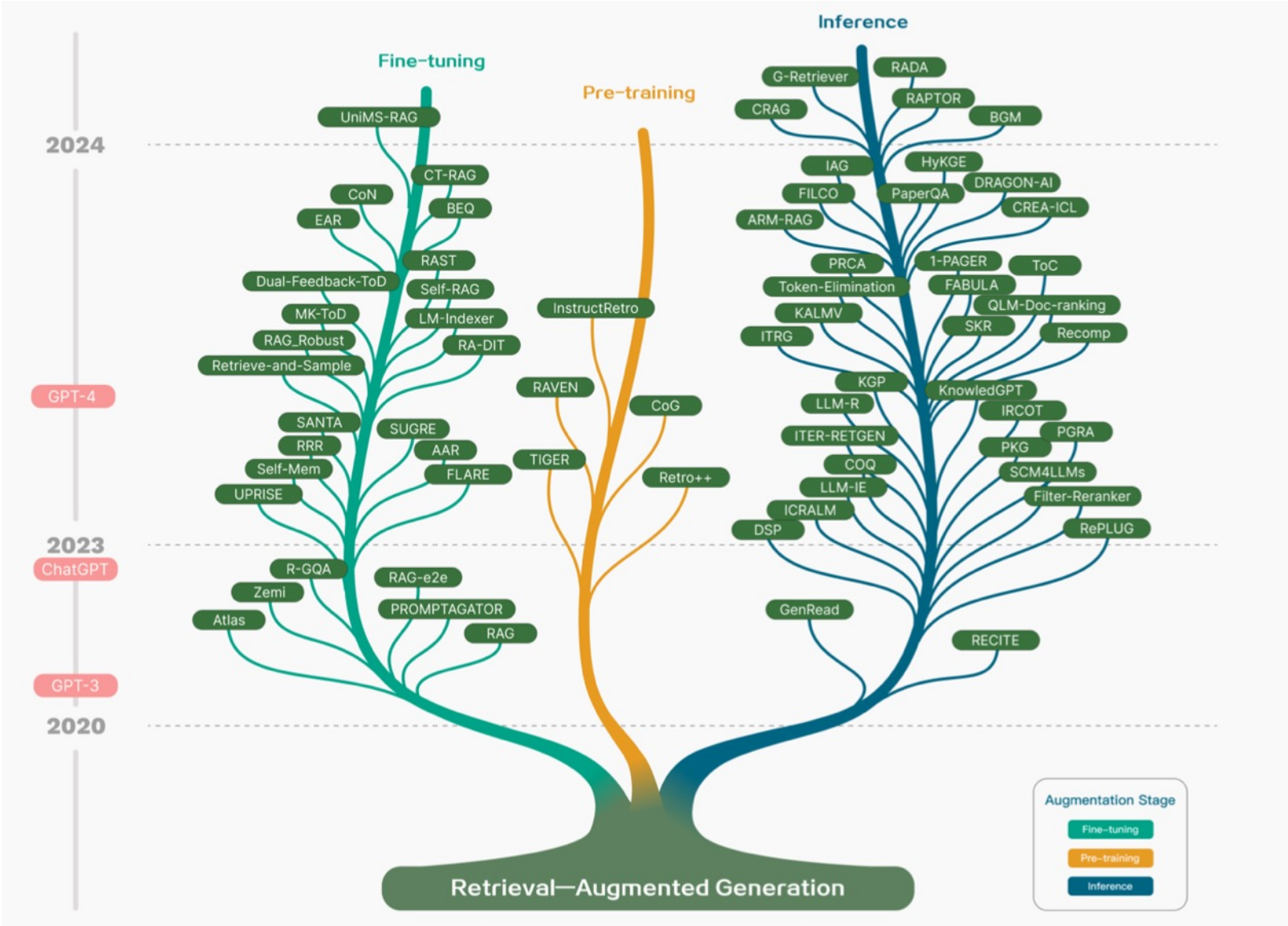
Agentic AI Systems



Source: https://neontri.com/blog/agentic-ai/

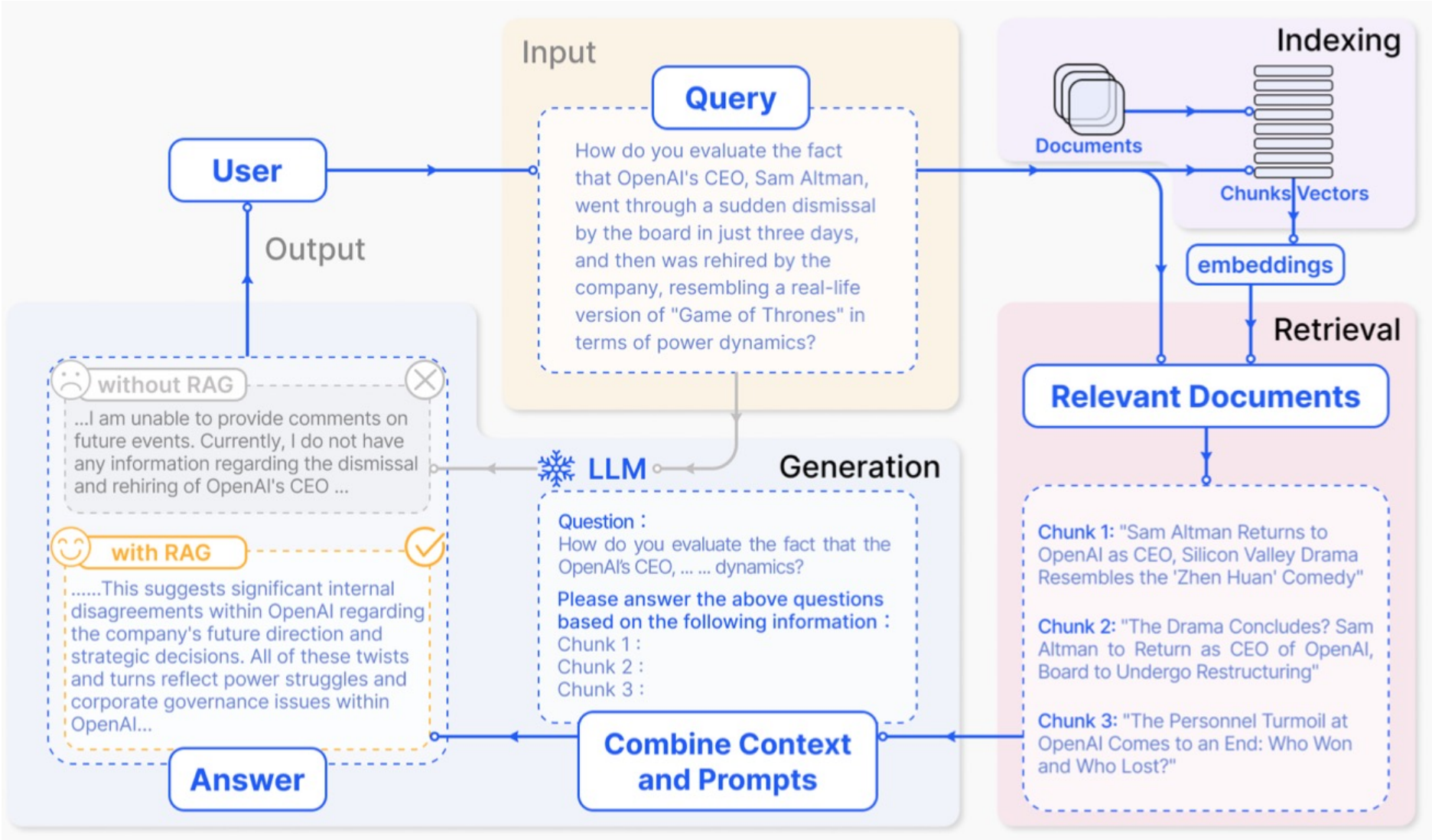
Technology Tree of RAG Research

Retrieval-Augmented Generation (RAG) for Large Language Models (LLMs)



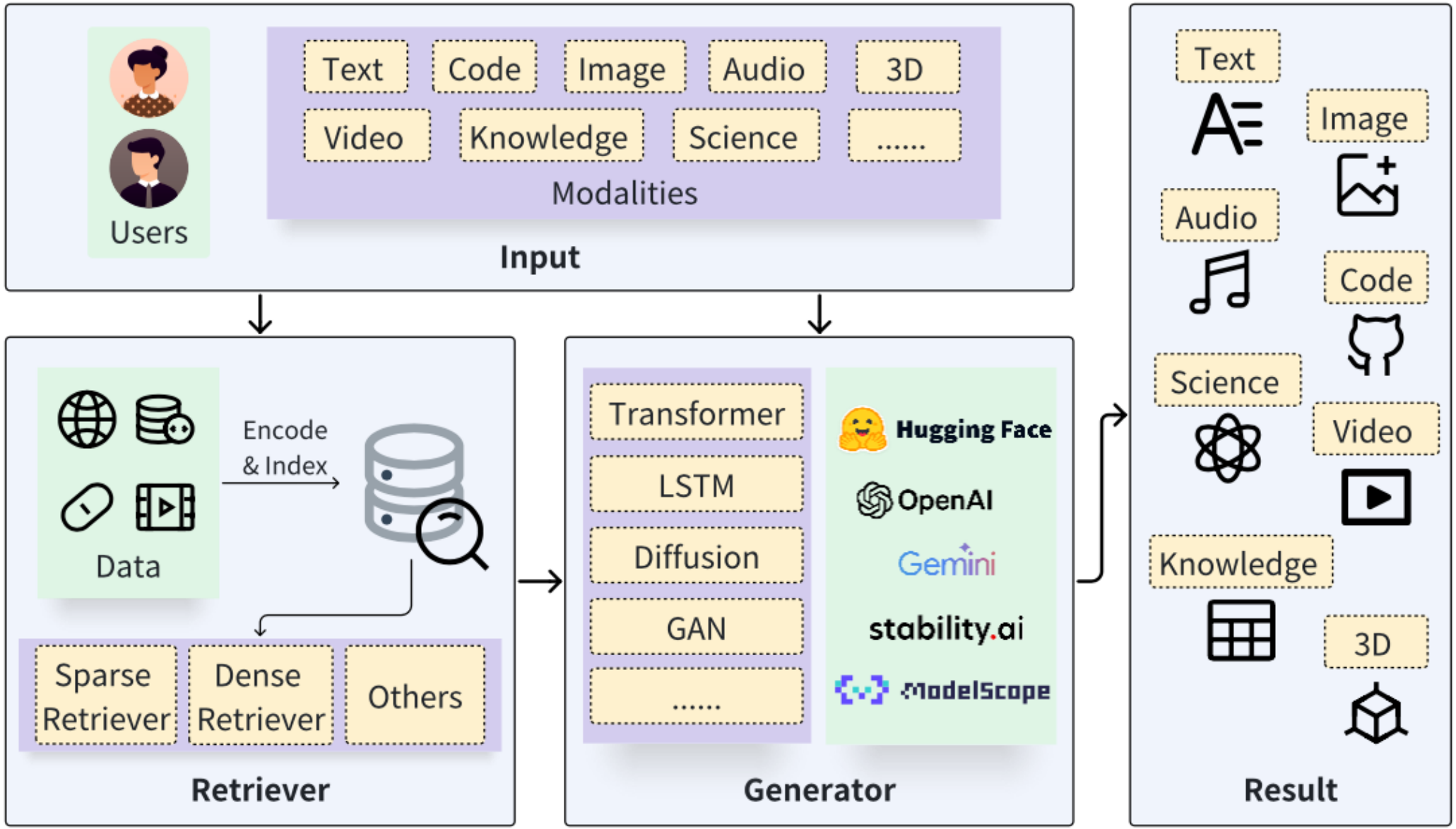
Source: Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

Retrieval-Augmented Generation (RAG) for Large Language Models (LLMs)



Source: Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

Retrieval-Augmented Generation (RAG) Architecture



Source: Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., ... & Cui, B. (2024). Retrieval-augmented generation for ai-generated content: A survey. arXiv preprint arXiv:2402.19473.

Evaluating RAG with Ragas Metrics

ragas score

generation

faithfulness

how factually acurate is
the generated answer

answer relevancy

how relevant is the generated
answer to the question

retrieval

context precision

the signal to noise ratio of retrieved
context

context recall

can it retrieve all the relevant information
required to answer the question

Source: https://blog.langchain.dev/evaluating-rag-pipelines-with-ragas-langsmith/

NVIDIA Developer Program

https://developer.nvidia.com/join-nvidia-developer-program

NVIDIA Deep Learning Institute (DLI)

https://learn.nvidia.com/

NVIDIA Certified Instructor

Min-Yuh Day

Dr. Min-Yuh Day is a Professor in the Graduate Institute of Information Management at National Taipei University, Taiwan. He holds a Ph.D. degree in Information Management from National Taiwan University, Taiwan. His research focuses on AI, generative AI, ESG green fintech, big data, e-commerce, and biomedical informatics, with publications in top journals and conferences.

Organization

National Taipei University (NTPU)

Location

Taiwan

Instructor Tier

Silver

Workshop Certifications

- Building RAG Agents with LLMs, English
- Generative AI with Diffusion Models, English
- Fundamentals of Deep Learning, English
- Building Agentic AI Applications with LLMs, English

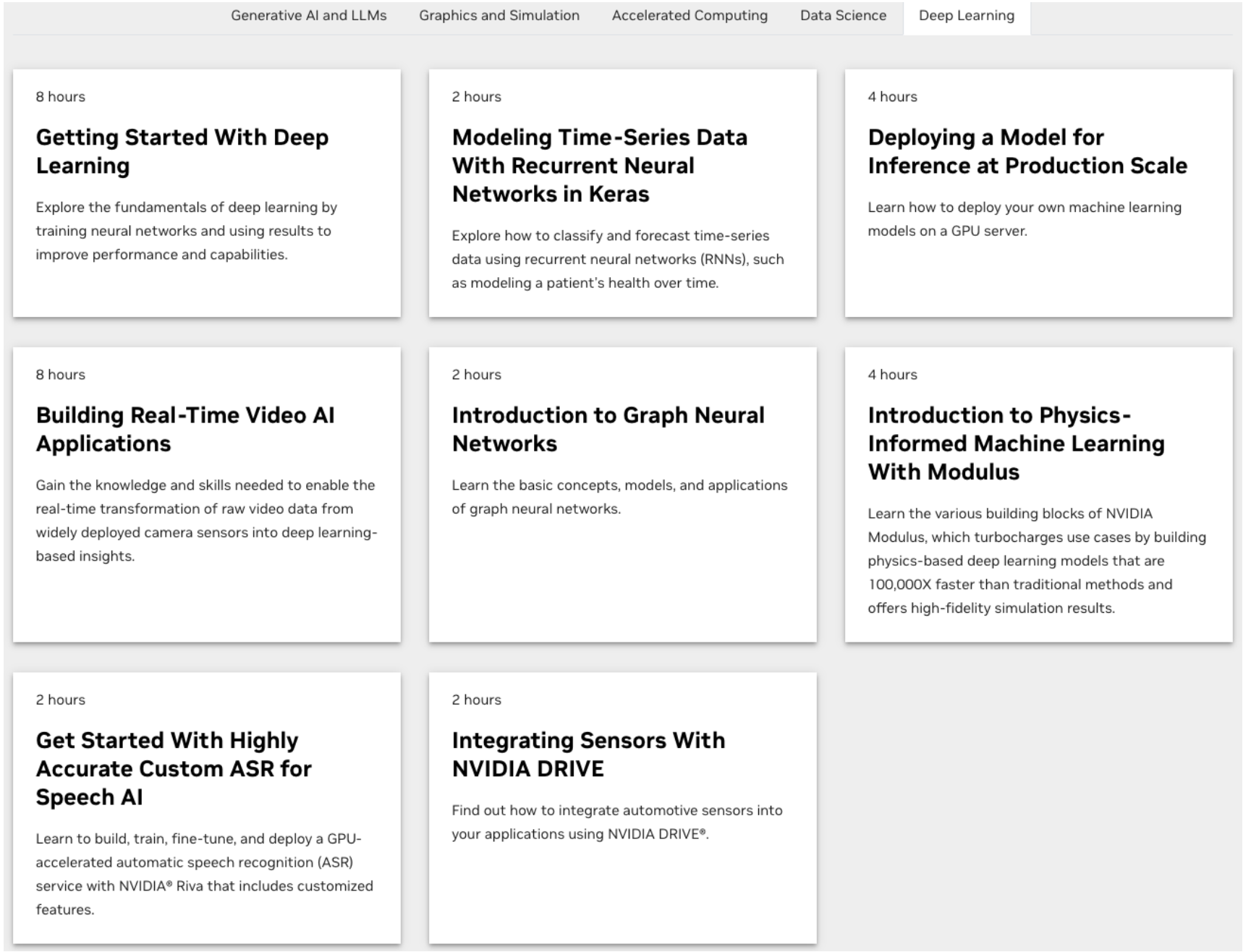
https://www.nvidia.com/en-us/training/instructor-directory/search/

Get NVIDIA DLI Certificate Before the NVIDIA Workshop

- **Step 1. Join NVIDIA Developer Program (Free)**
  https://developer.nvidia.com/join-nvidia-developer-program
- **Step 2. Visit NVIDIA Deep Learning Institute (DLI)**
  https://learn.nvidia.com/
- **Step 3. Enroll "Building RAG Agents with LLMs" Self-Paced Course (Free)**
  https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Join the NVIDIA Developer Program

take one of the complimentary technical self-paced courses (worth up to $90)



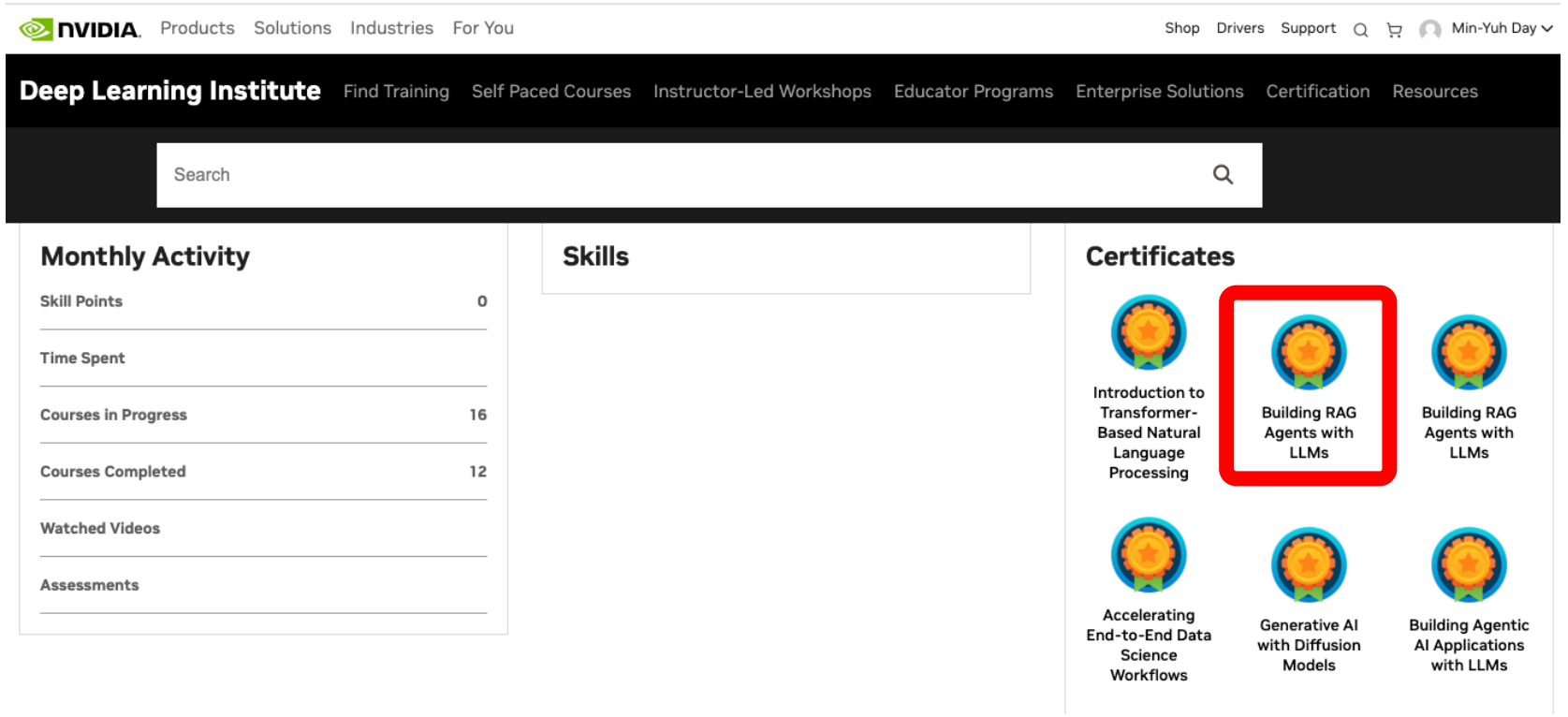
https://developer.nvidia.com/join-nvidia-developer-program

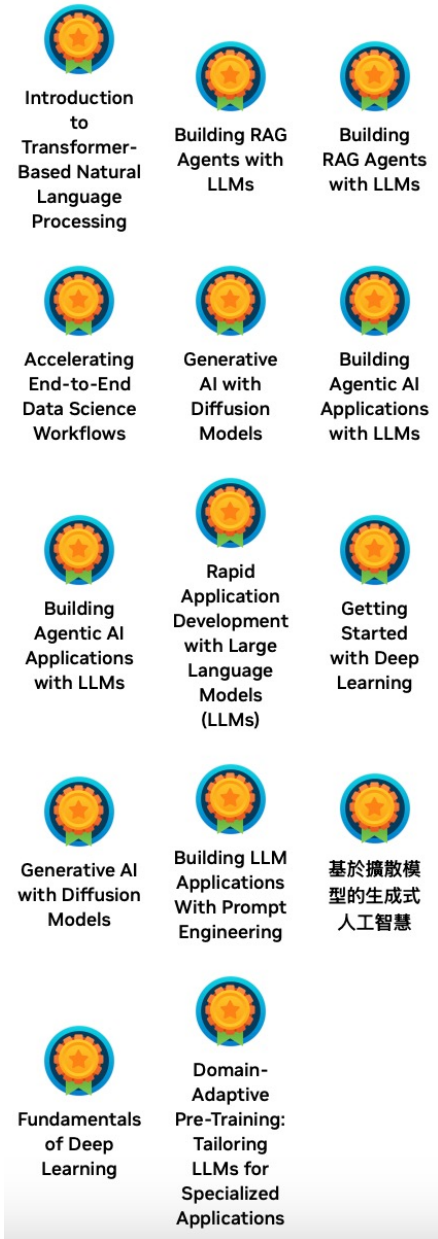
NVIDIA Deep Learning Institute (DLI)

Self-Paced Course **Generative AI Explained** Free 2 hours	Self-Paced Course **Getting Started With Deep Learning** Certificate available $90 8 hours	Instructor-Led Workshop **Fundamentals of Deep Learning** Certificate available $500 8 hours	Self-Paced Course **Introduction to Transformer-Based Natural Language Processing** Certificate available $30 6 hours
Self-Paced Course **Building RAG Agents With LLMs** Certificate available Free 8 hours	Instructor-Led Workshop **Building RAG Agents With LLMs** Certificate available $500 8 hours	Self-Paced Course **Generative AI with Diffusion Models** Certificate available $90 8 hours	Instructor-Led Workshop **Generative AI with Diffusion Models** Certificate available $500 8 hours

https://www.nvidia.com/en-us/learn/learning-path/generative-ai-llm/

Deep Learning Institute (DLI)





Completed Courses

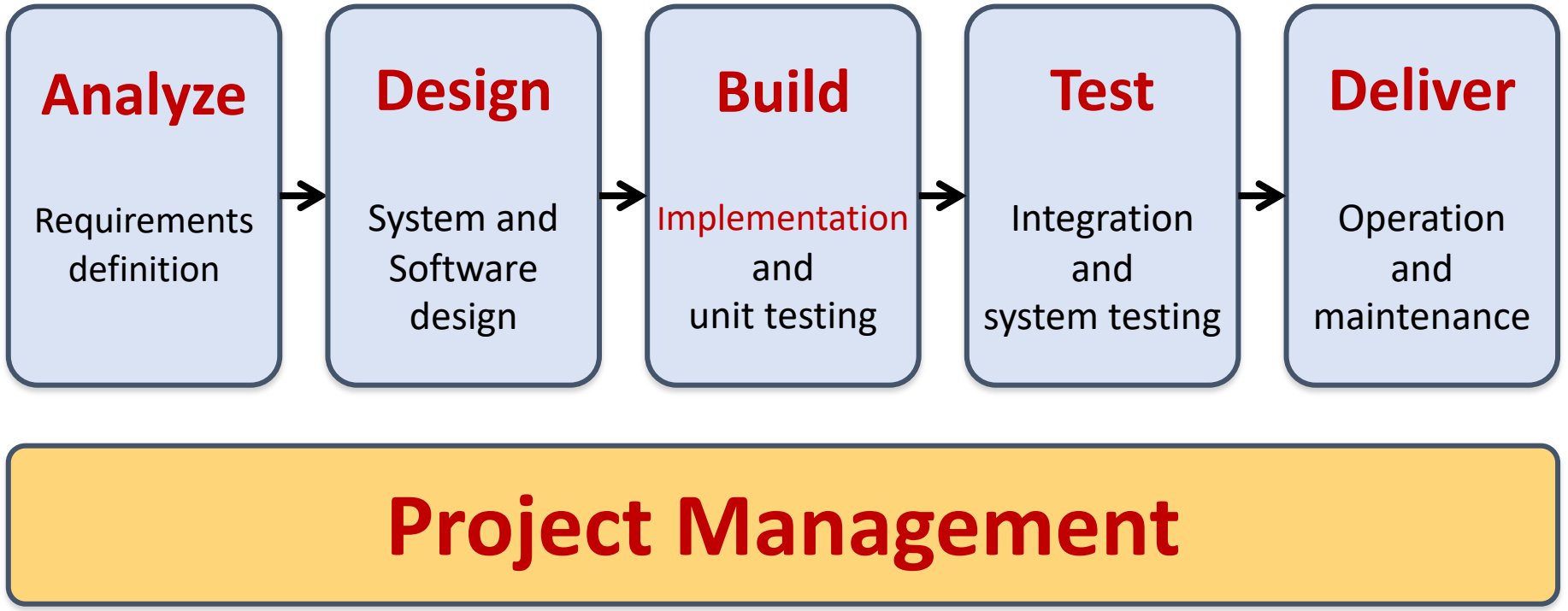
View more

Self-paced	Self-paced	Self-paced	Self-paced	Self-paced
Sizing LLM Inference Systems	Augment your LLM Using Retrieval Augmented Generation	Building RAG Agents with LLMs	Generative AI Explained	Introduction to Transform Based Natural Language Processing
100% Completed 03:00	100% Completed 01:00	100% Completed 08:00	100% Completed 02:00	100% Completed 06:00

https://learn.nvidia.com/my-learning

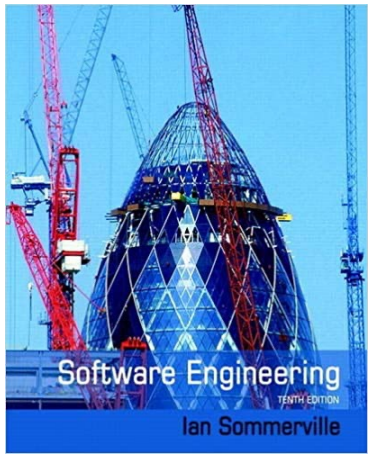
Software Engineering

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.



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

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.**

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

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.**

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

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.**

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

What are the fundamental software engineering activities?

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

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

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.**

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

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.**

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

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.**

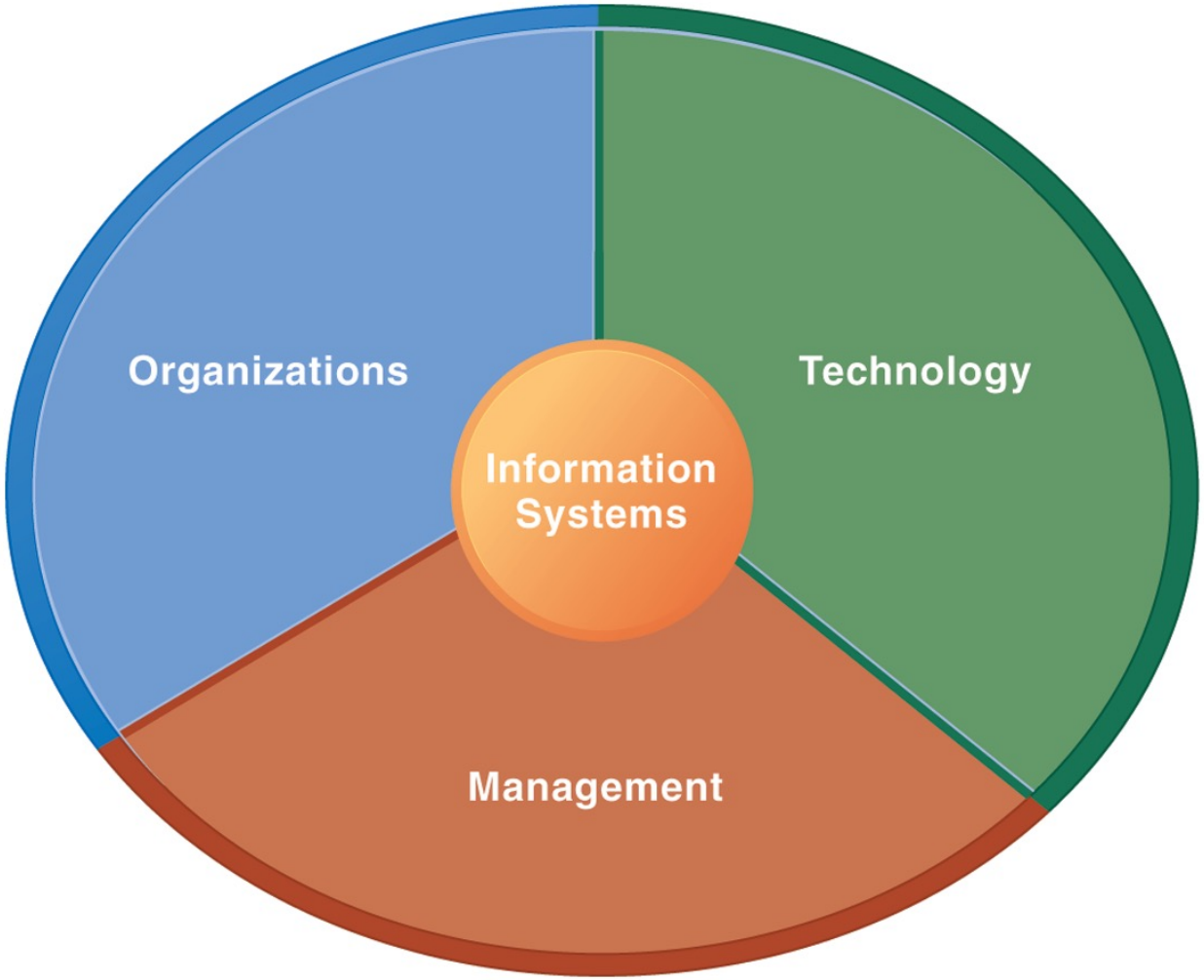
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Information Management

Management Information Systems (MIS)

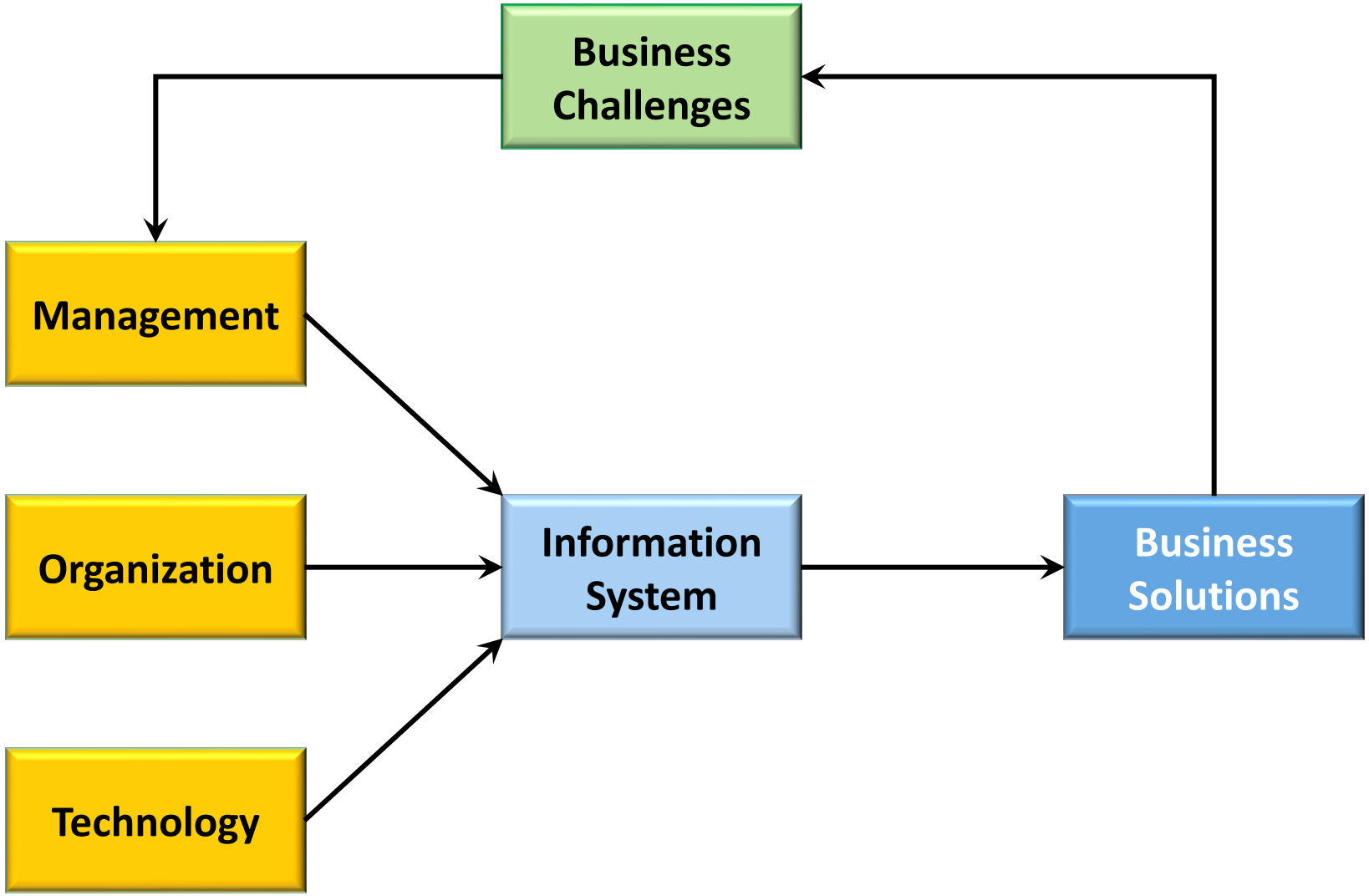
Information Systems

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.

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).**

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

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.**

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

Software projects

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

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

Project

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

Source: Project Management Institute (2017), A Guide to the Project Management Body of Knowledge (PMBOK Guide), Sixth Edition, PMI

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

PMBOK Guide v6

Project Management Body of Knowledge:

- Introduction
- Project Environment
- Role of the Project Manager
- 10 Knowledge Areas
  - Integration
  - Scope
  - Schedule
  - Cost
  - Quality
  - Resources
  - Communications
  - Risk
  - Procurement
  - Stakeholders

The Standard for Project Management (5 Process Groups):

- Initiating
- Planning
- Executing
- Monitoring and Controlling
- Closing

PMBOK Guide v7

The Standard for Project Management :

- Introduction
- System for Value Delivery
- 12 Project Management Principles:
  - 1. Stewardship, 2. Team
  - 3. Stakeholders, 4. Value
  - 5. Systems Thinking, 5. Leadership
  - 7. Tailoring, 8. Quality
  - 9. Complexity, 10, Risk
  - 11. Adaptability and Resiliency
  - 12. Change

Project Management Body of Knowledge:

- 8 Project Performance Domains:
  - 1. Stakeholders, 2. Team,
  - 3. Development approach and Life Cycle
  - 4. Planning, 5. Project Work, 6. Delivery,
  - 7. Measurement, 8. Uncertainty
- Tailoring
- Models, Methods, and Artifacts

Source: 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 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

Source: Project Management Institute (2017), A Guide to the Project Management Body of Knowledge (PMBOK Guide), Sixth Edition, PMI

Project Management Process Groups

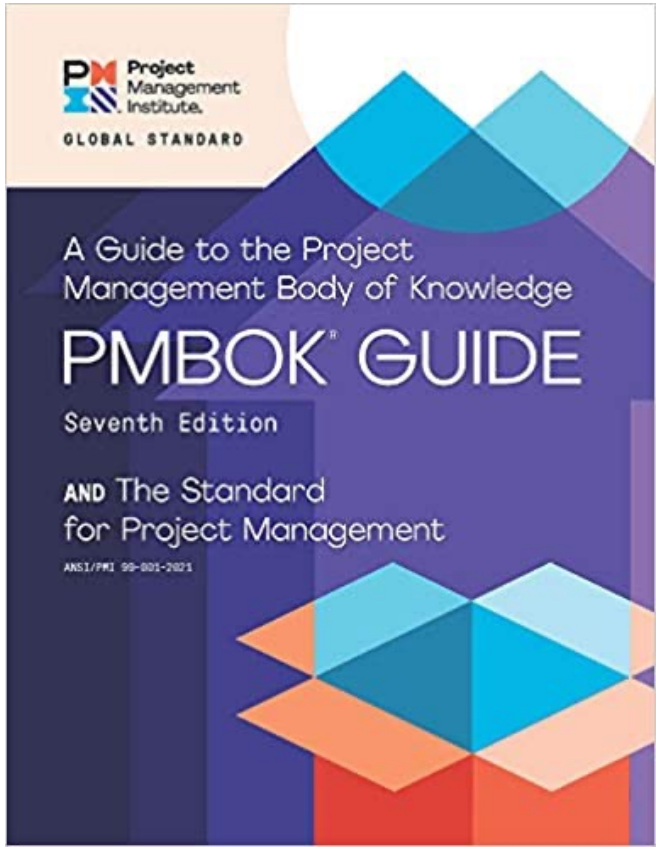
(PMBOK v6)

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

Source: Project Management Institute (2017), A Guide to the Project Management Body of Knowledge (PMBOK Guide), Sixth Edition, PMI

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



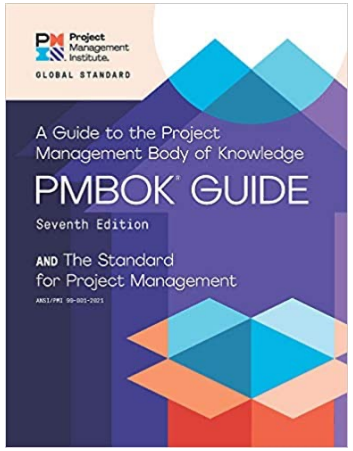
Source: Project Management Institute (2021), A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Seventh Edition and The Standard for Project Management, PMI

Project Management

8 Project Performance Domains

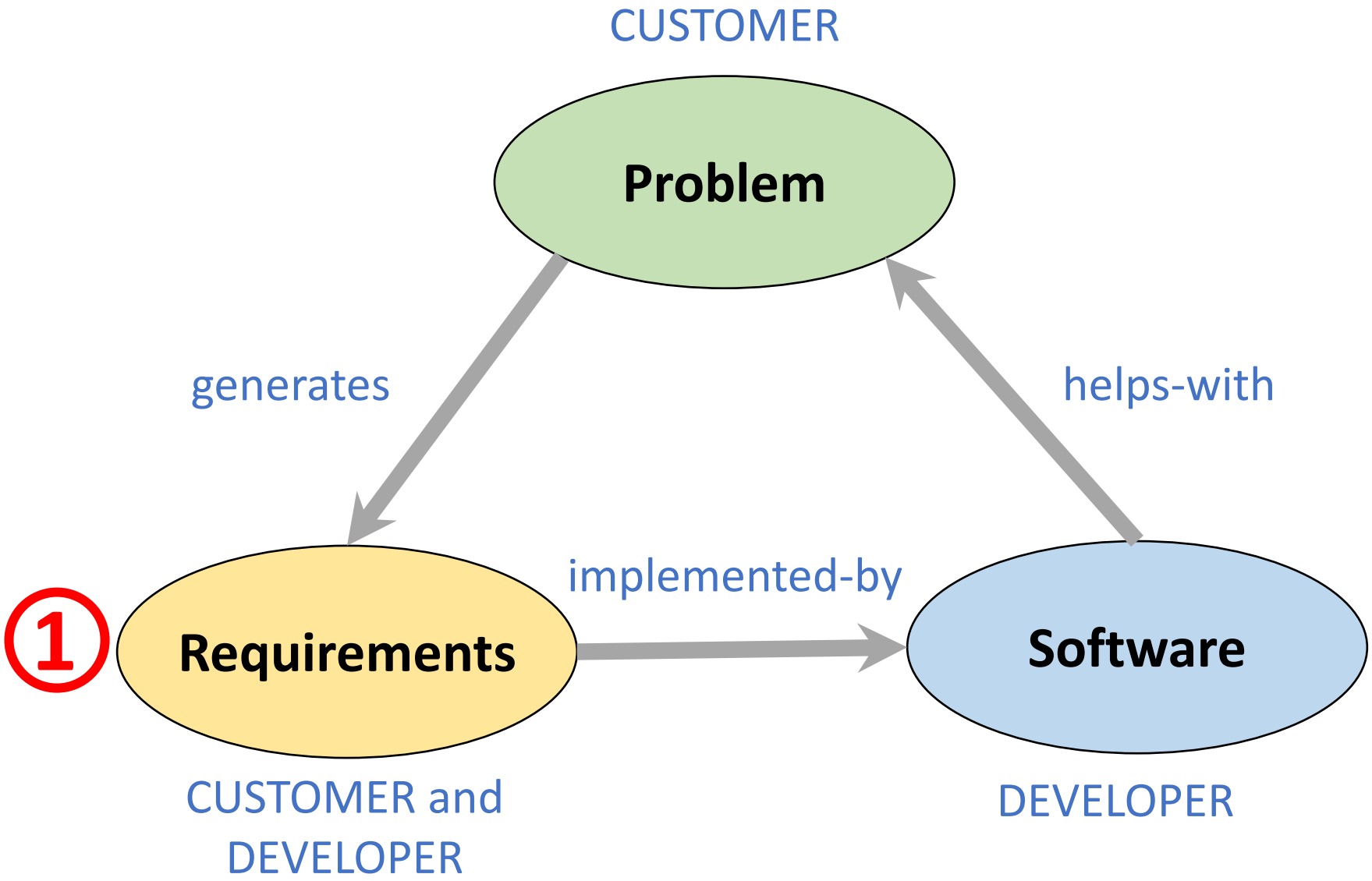
(PMBOK v7)

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



Source: 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-based software engineering



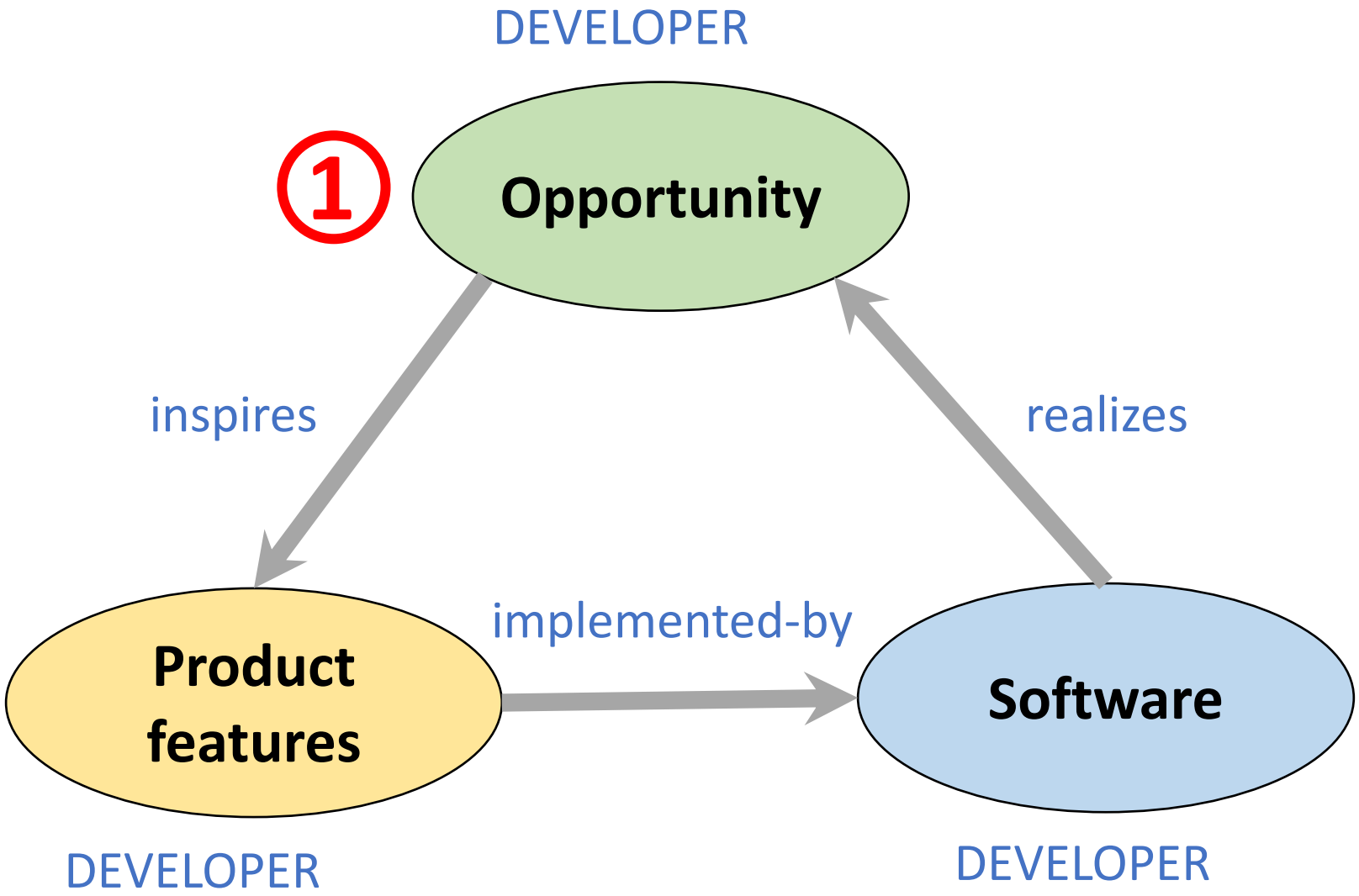
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

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.

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.

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.

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

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.**

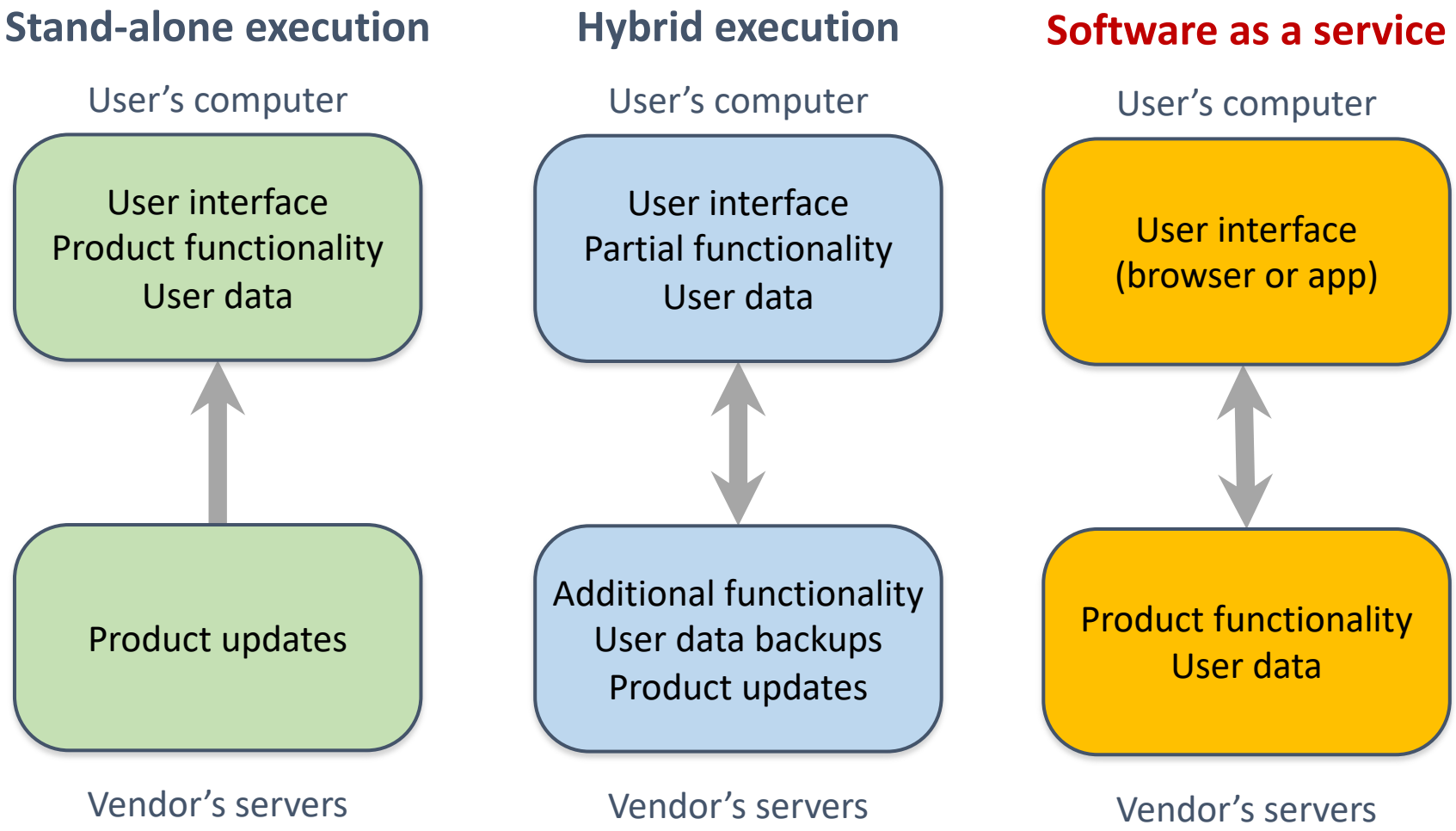
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

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.

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.

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.

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

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**

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

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?

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

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)**

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

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."

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

Information sources for developing a product vision

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

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

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.

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

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.

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

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.

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

The Essence of
Strategic Marketing (STP)

Segmentation

Targeting

Positioning

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

Customer Value

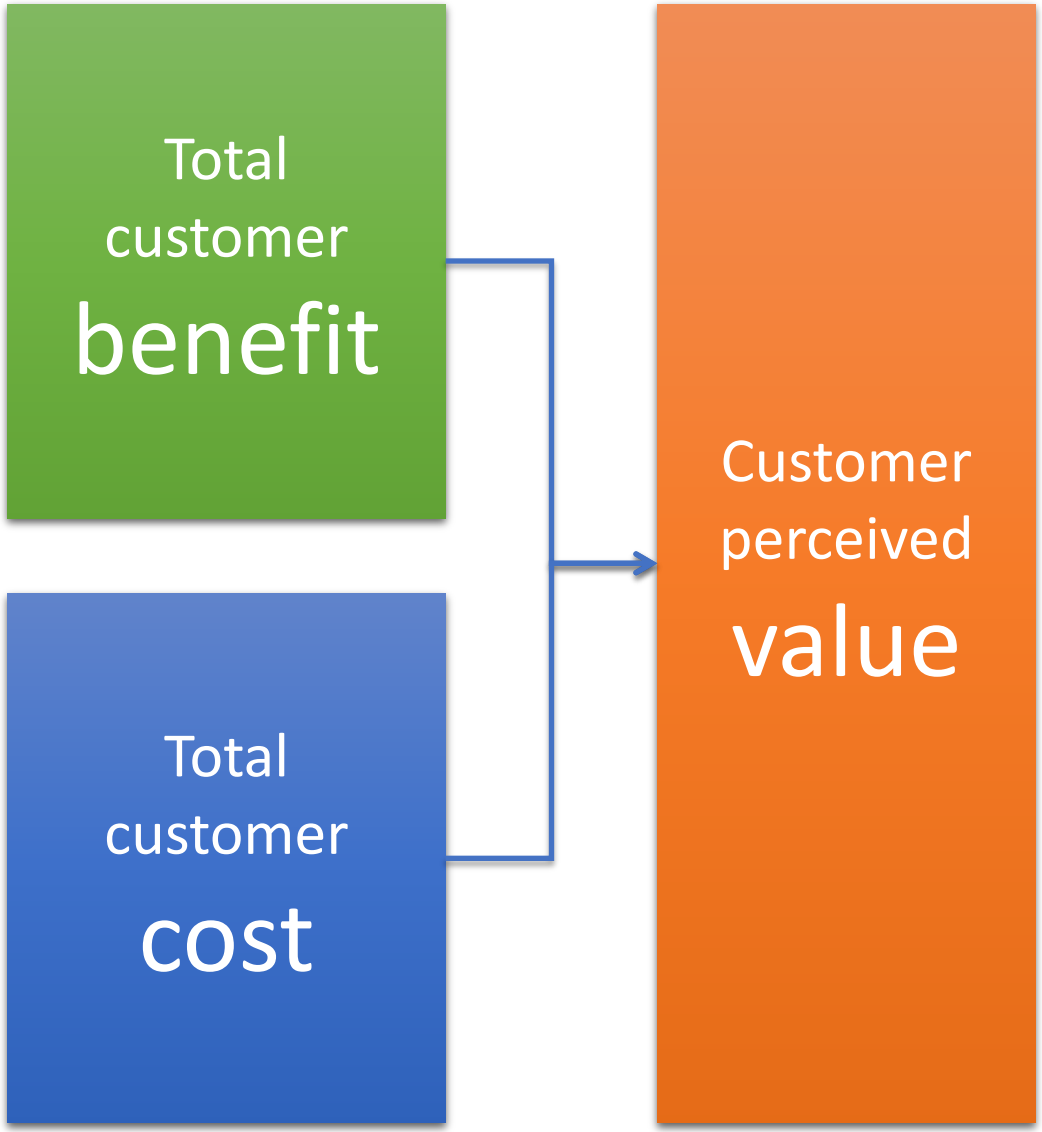
Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Value

the sum of the tangible and intangible benefits and costs

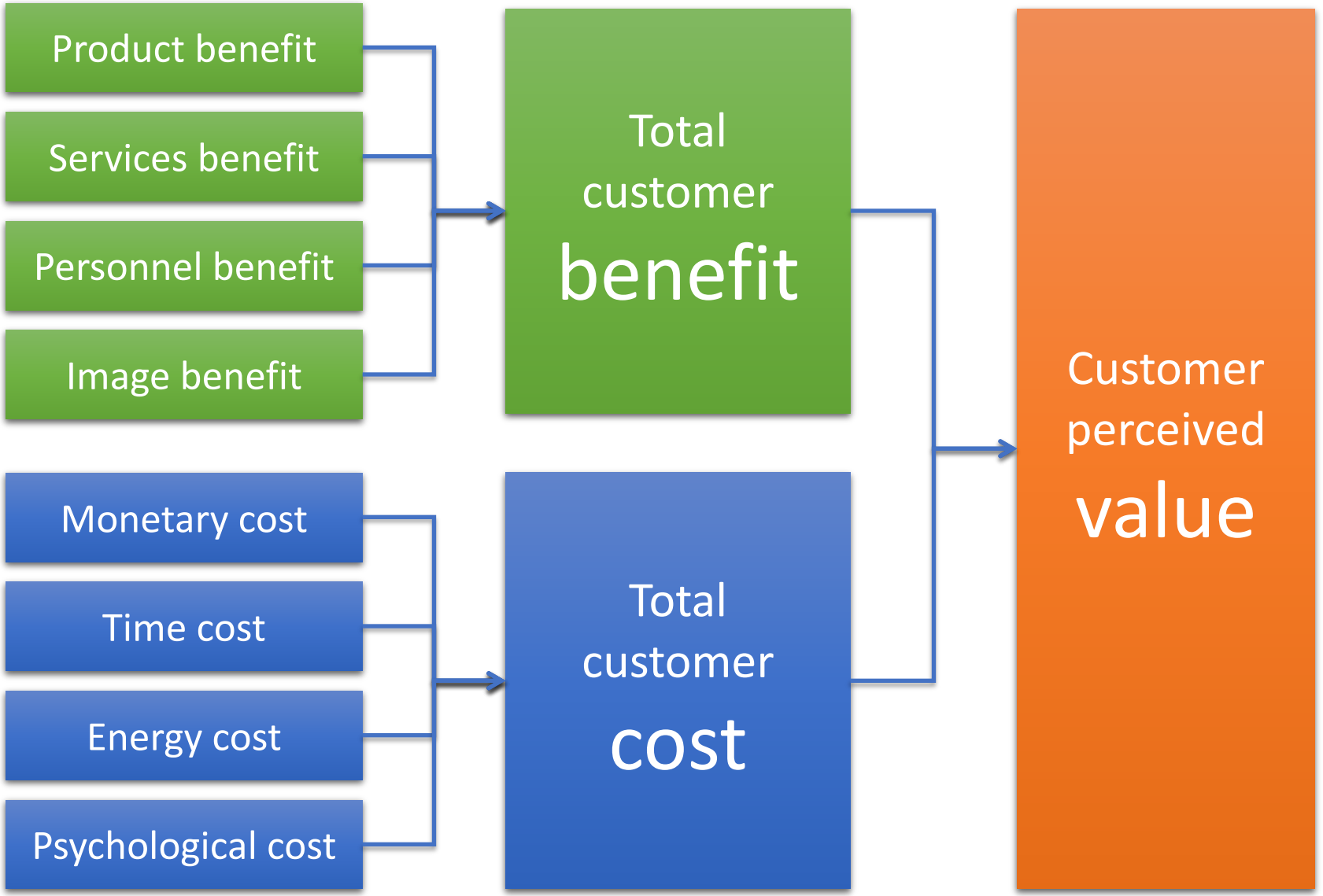
Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Value



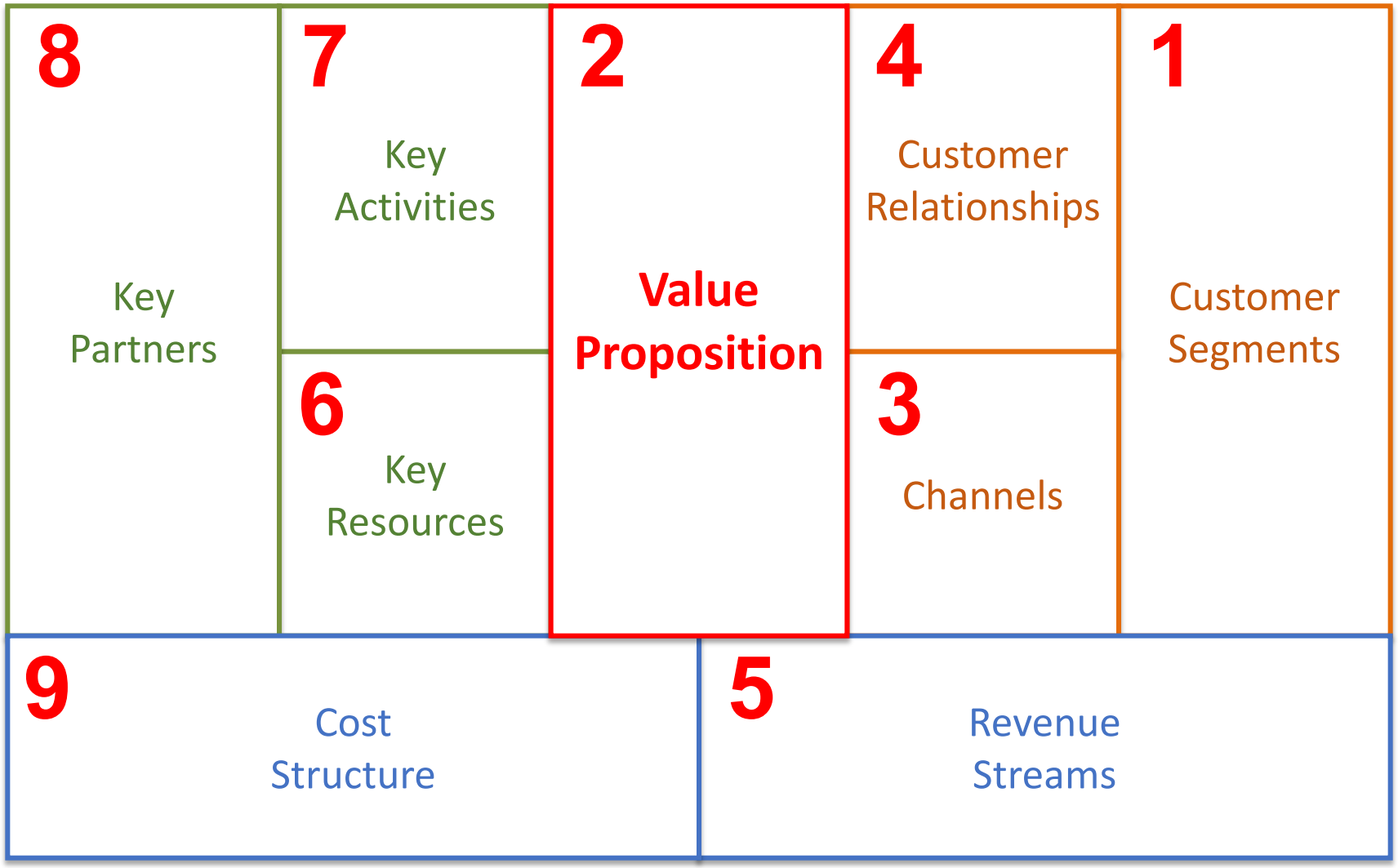
Source: Philip Kotler and Kevin Lane Keller (2016), Marketing Management, 15th edition, Pearson.

Customer Perceived Value



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

Business Model



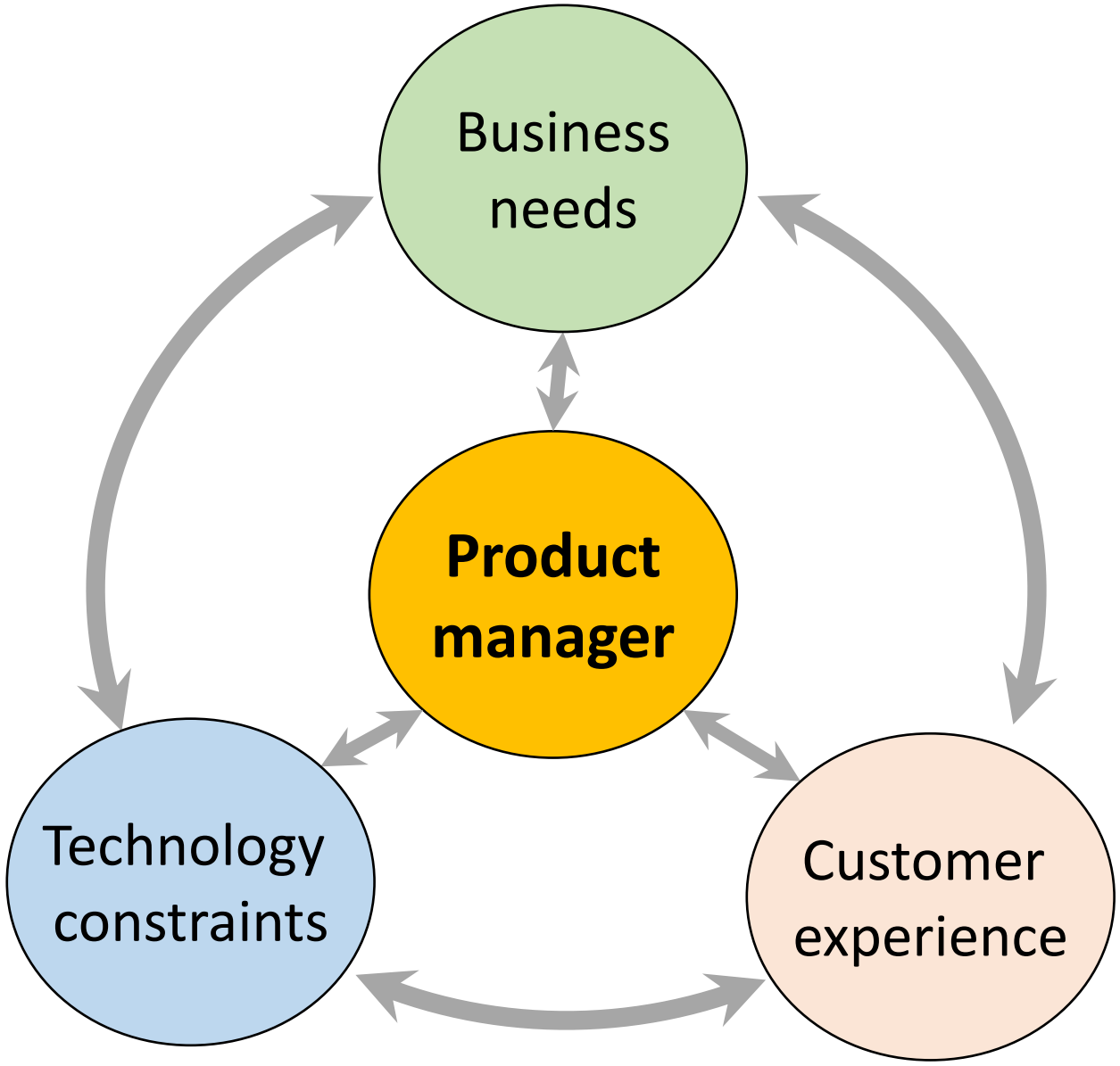
Source: Alexander Osterwalder & Yves Pigneur, Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers, Wiley, 2010.

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**

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

Product management concerns



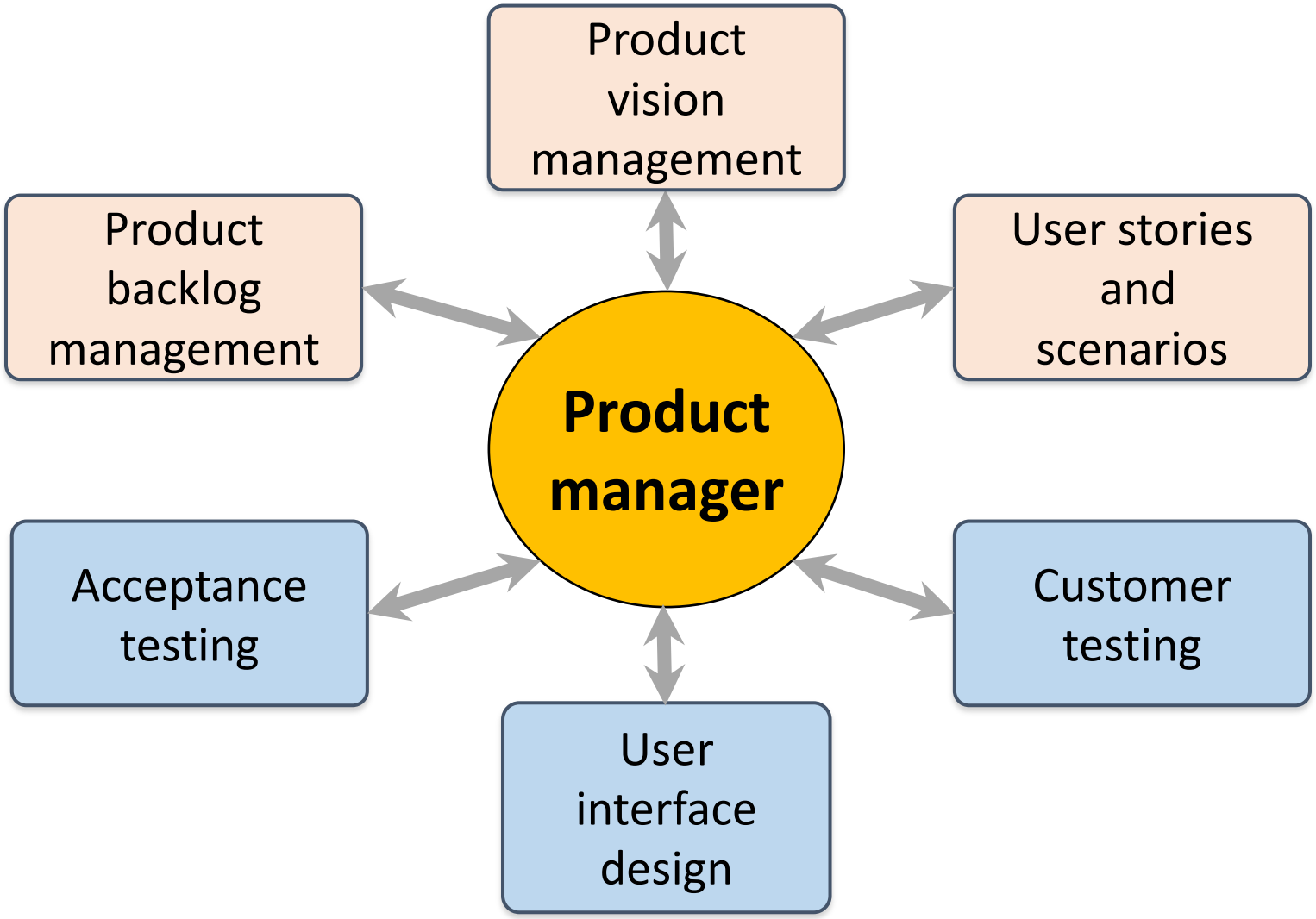
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

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.**

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.

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’**

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

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.**

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

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.**

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

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.**

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

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.**

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

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.**

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

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.**

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

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.**

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

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.**

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

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.**

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

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

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

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.**

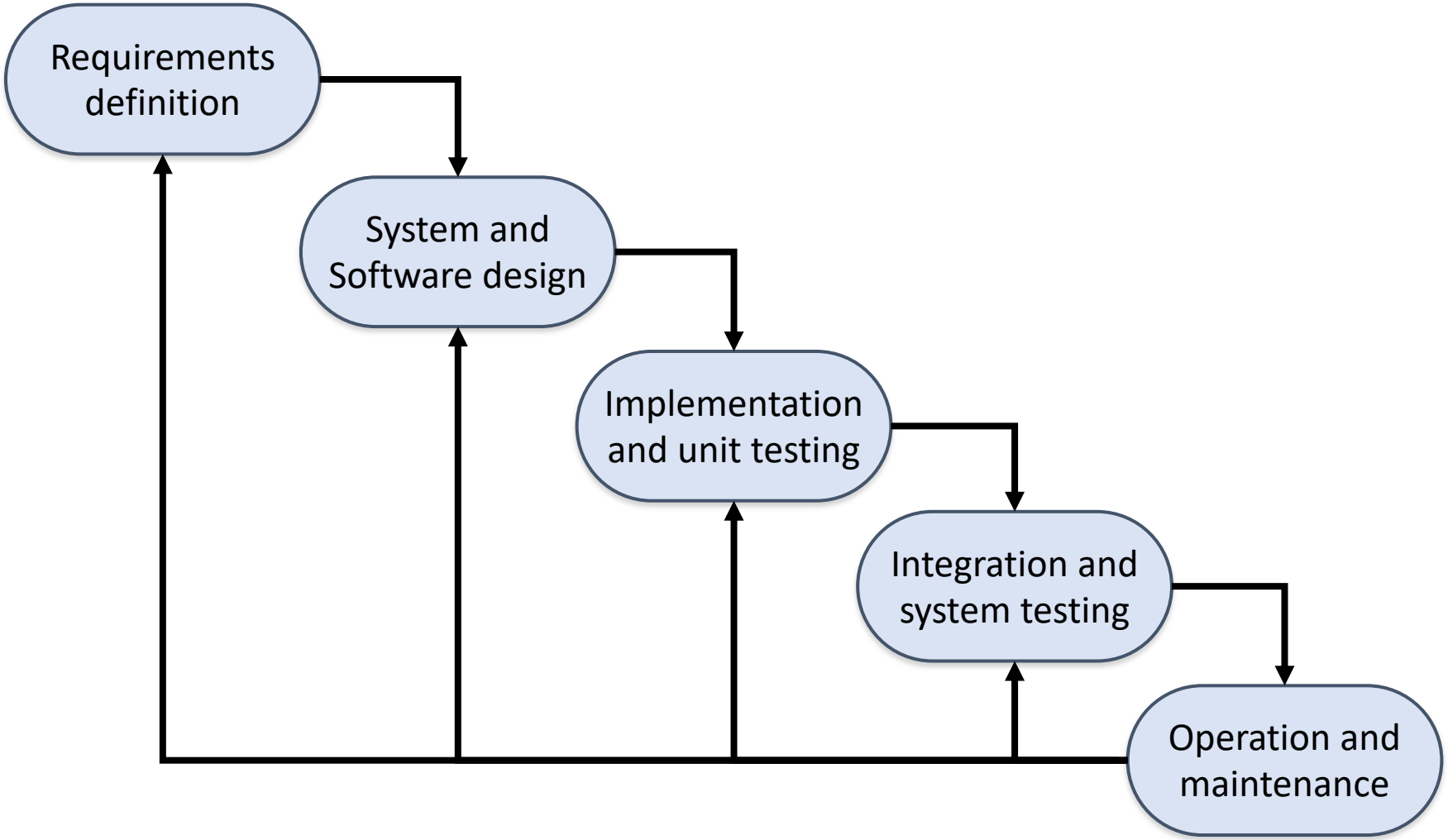
Source: Ian Sommerville (2019), Engineering Software Products: An Introduction to Modern Software Engineering, Pearson.

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.**

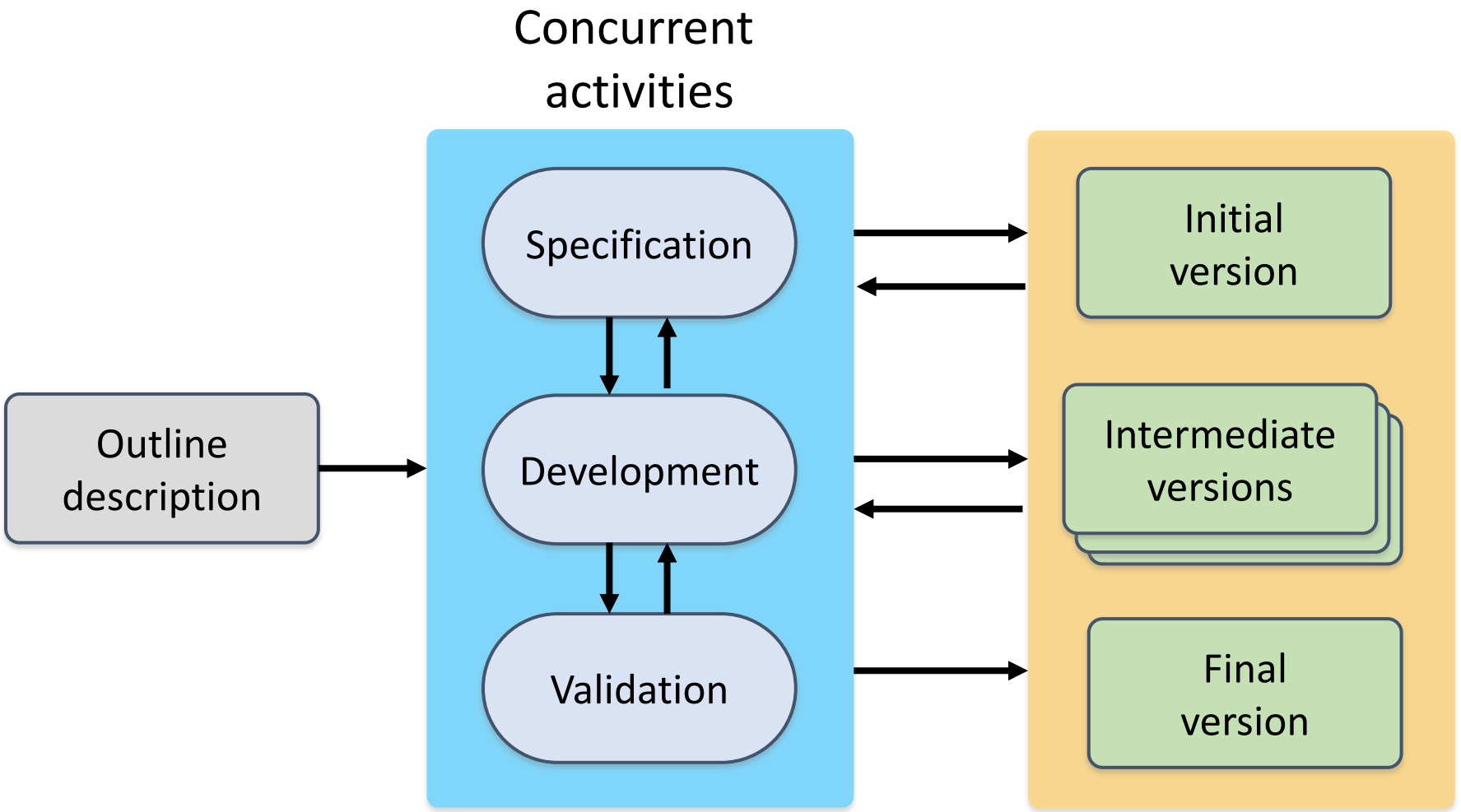
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Software Development Life Cycle (SDLC)
The waterfall model



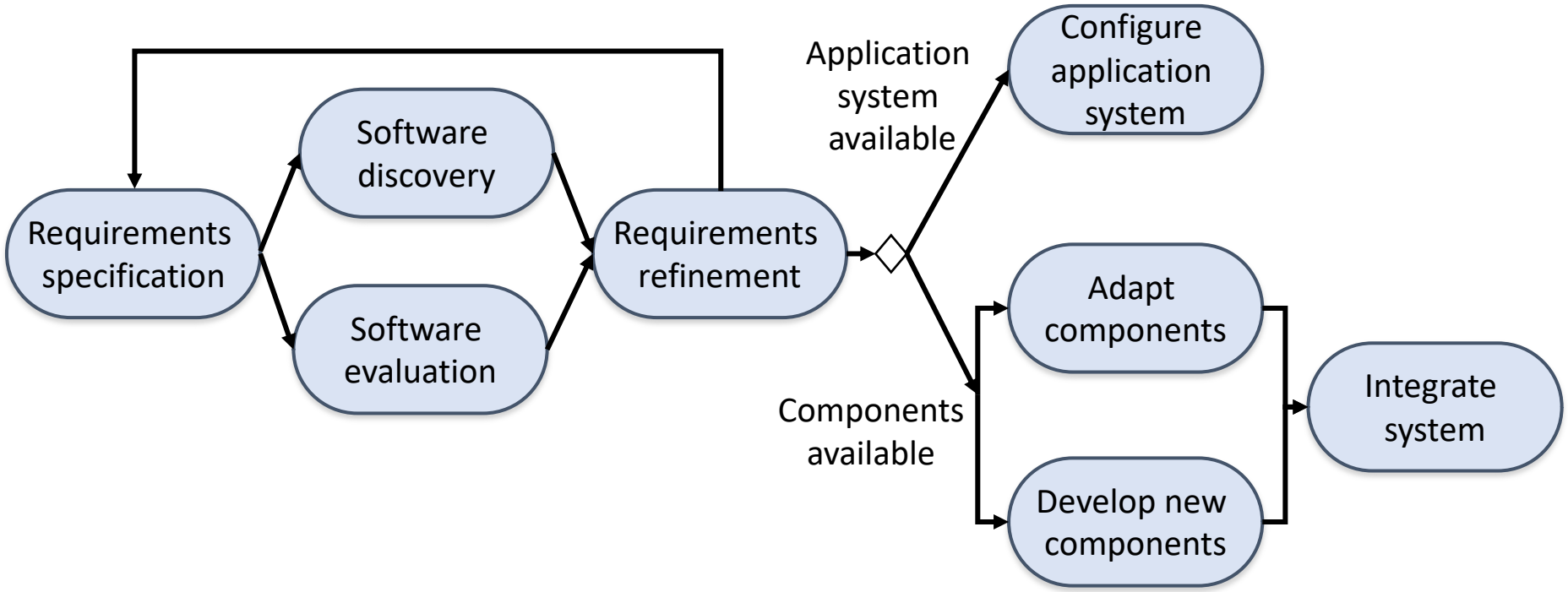
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Incremental development



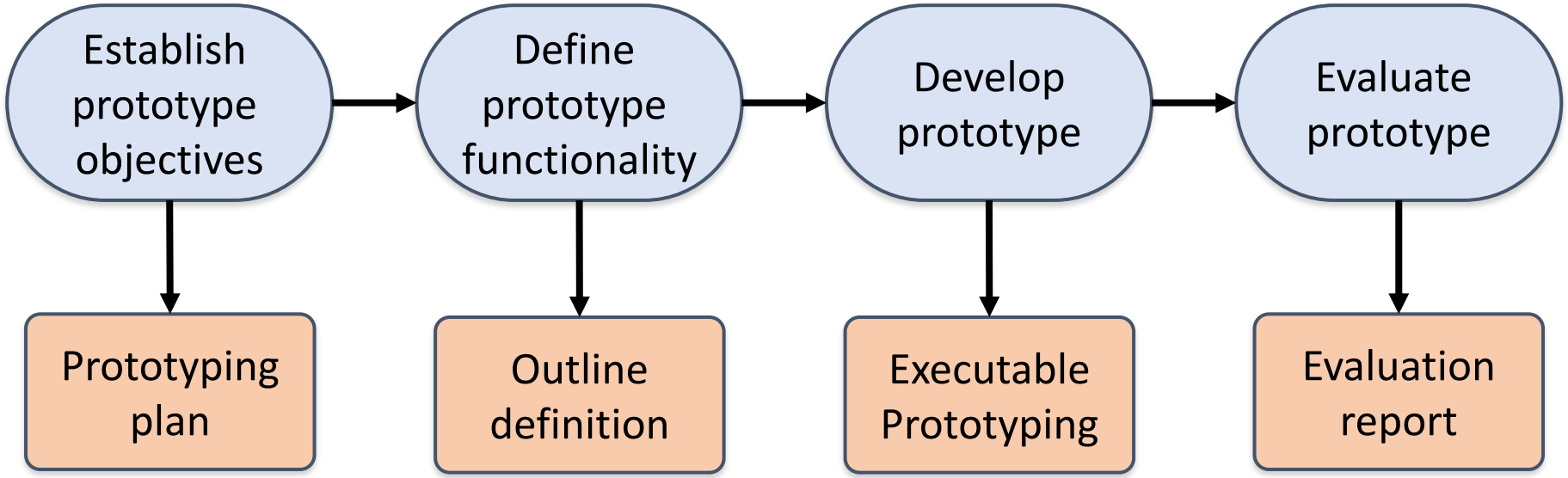
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Reuse-oriented software engineering



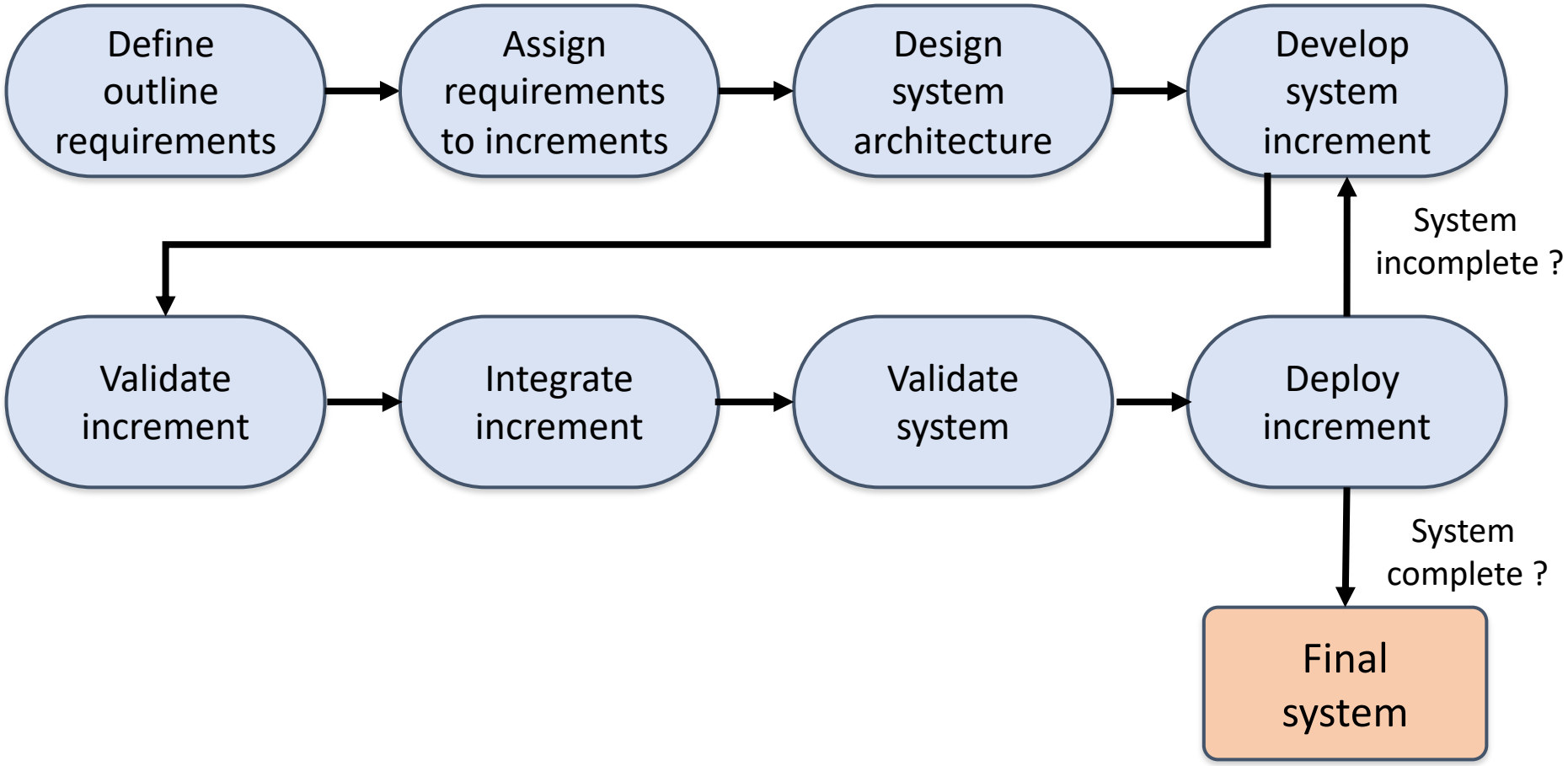
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Prototype development



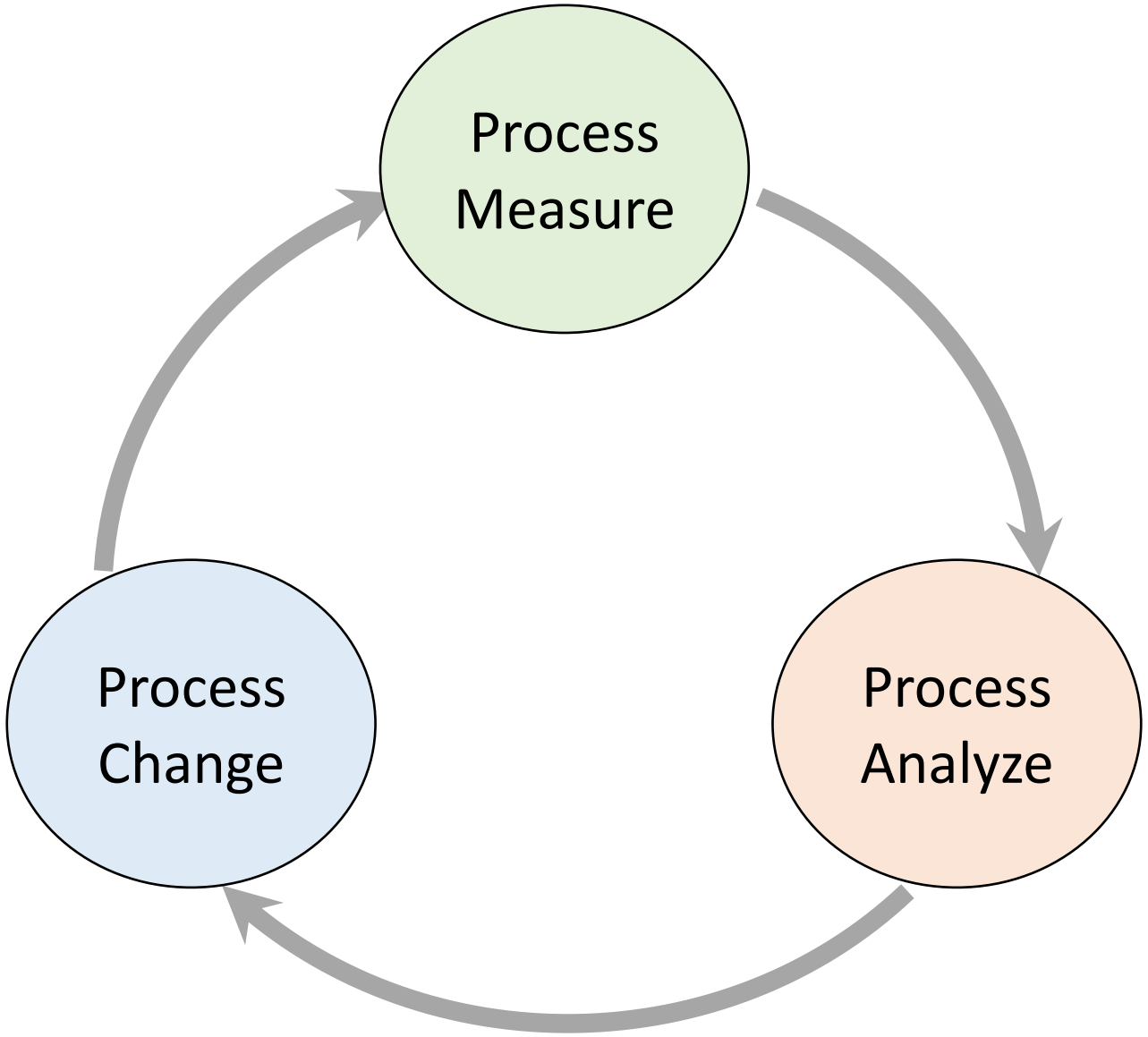
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Incremental delivery



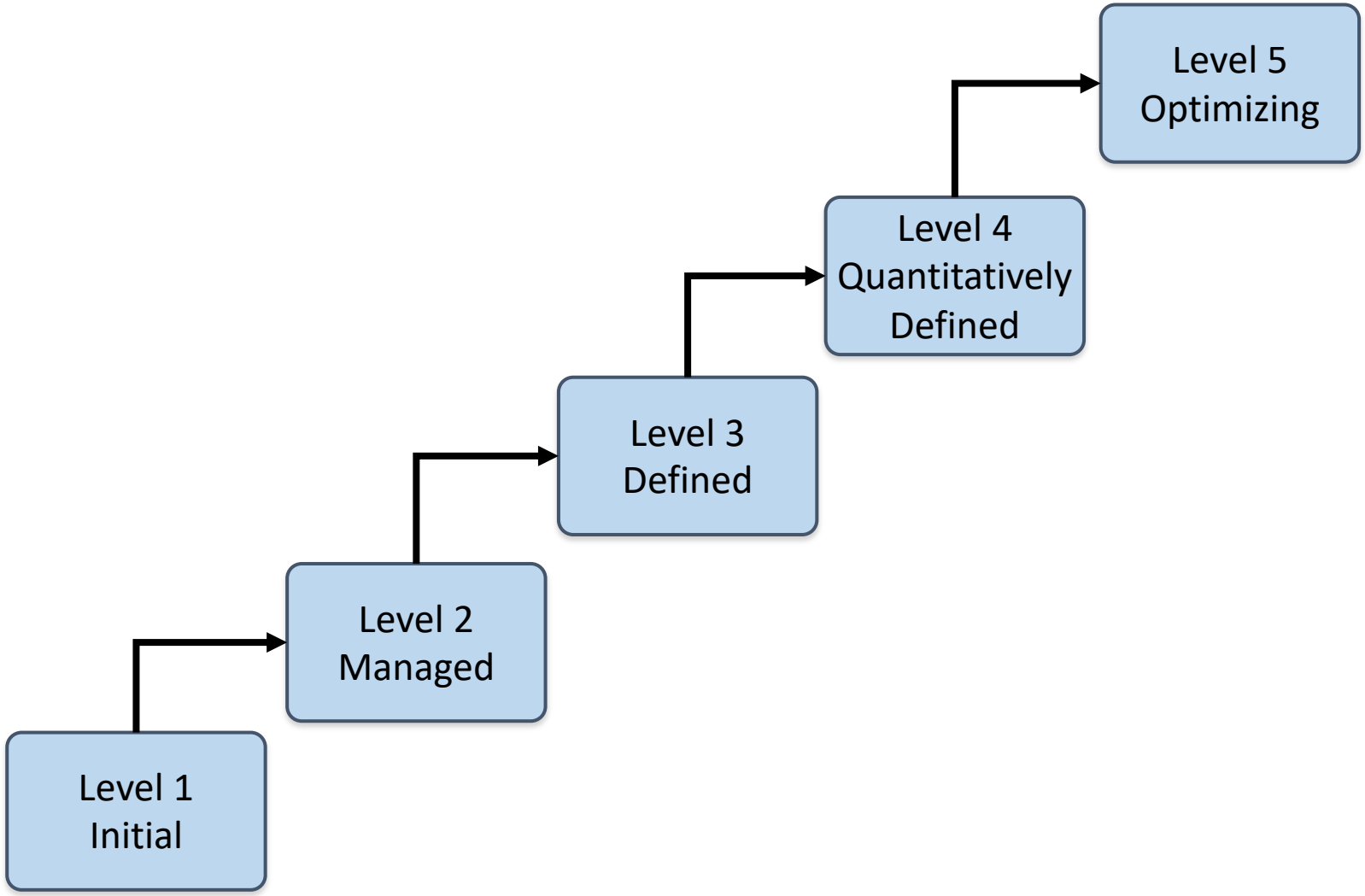
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

The process improvement model



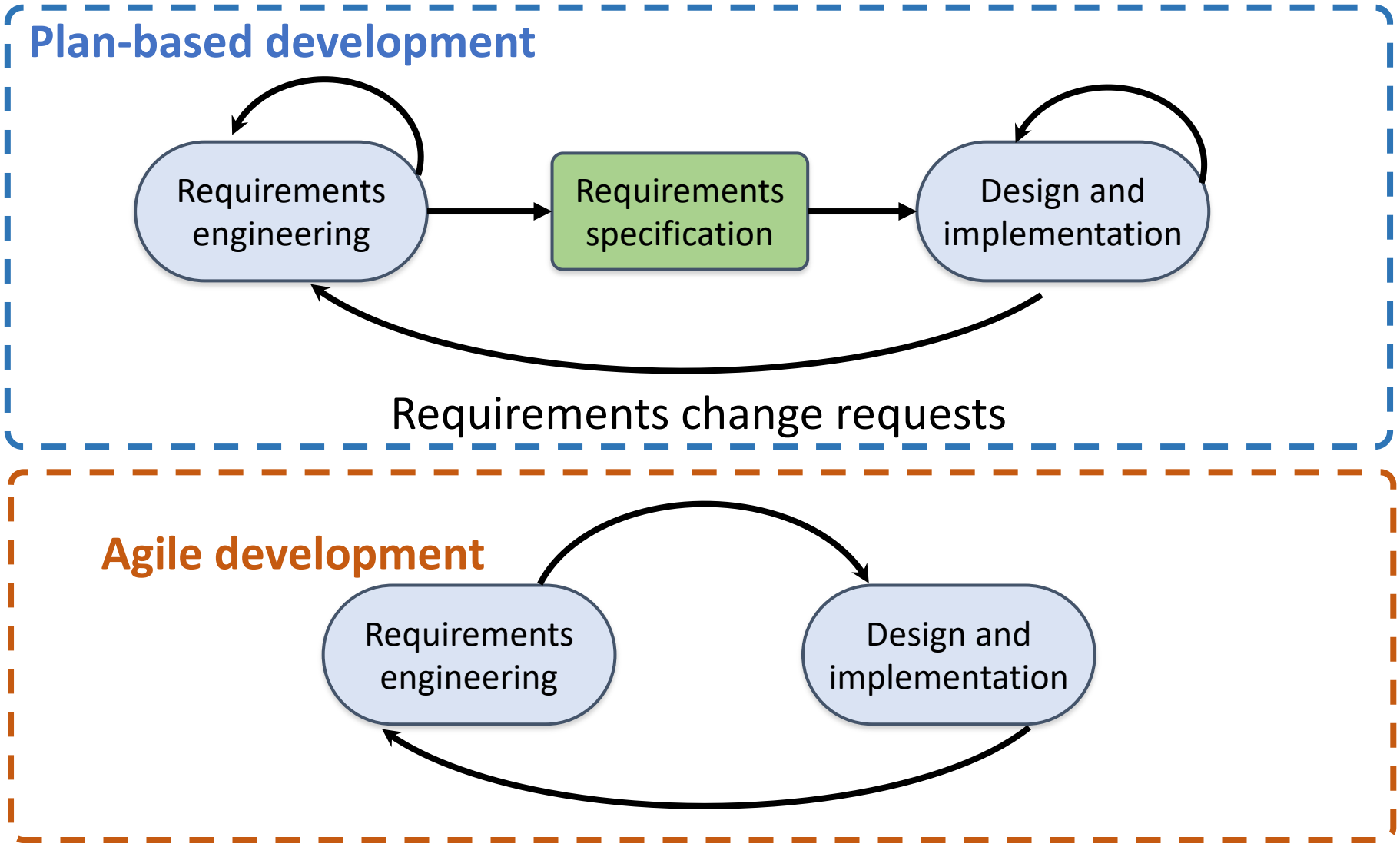
Source: Ian Sommerville (2015), Software Engineering, 10th Edition, Pearson.

Capability maturity levels



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

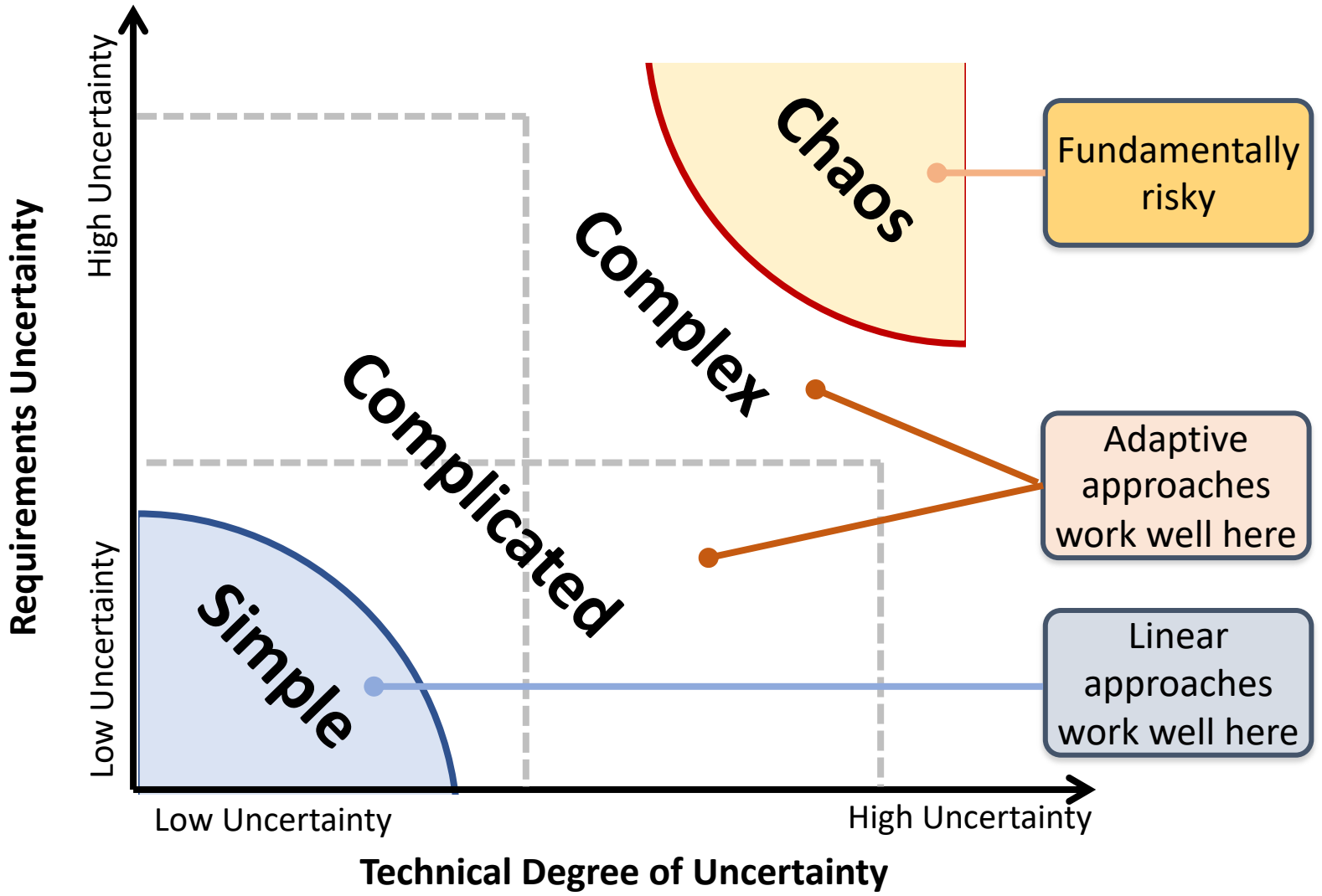
Plan-based and Agile development



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

Uncertainty and Complexity Model

Inspired by the Stacey Complexity Model



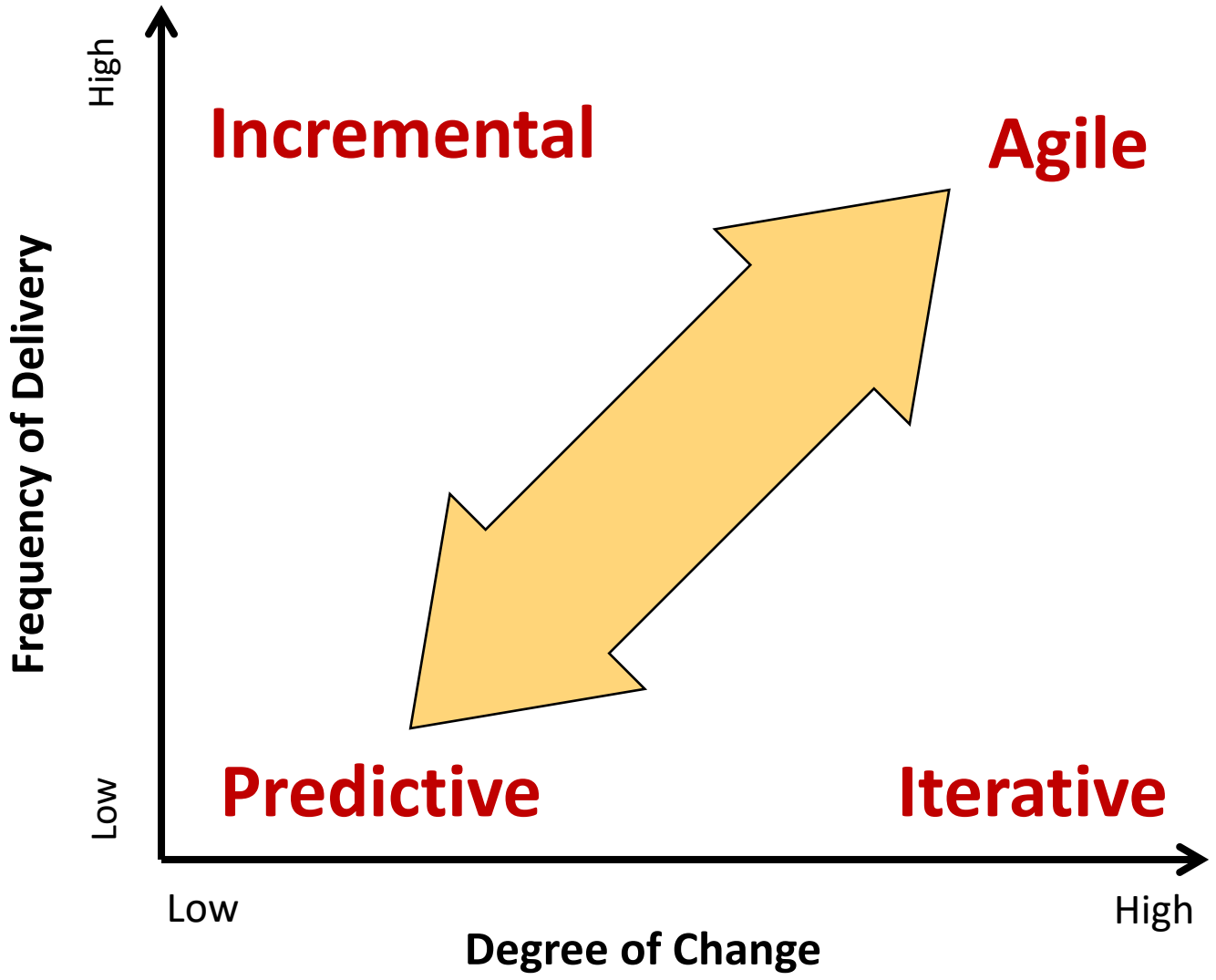
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Characteristics of Four Categories of Life Cycles

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

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

The Continuum of Life Cycles

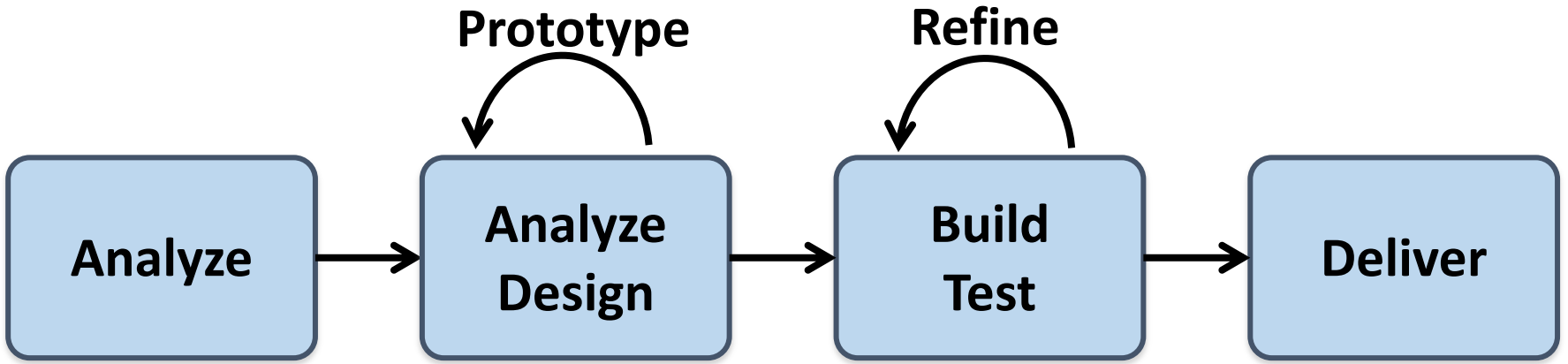


Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Predictive Life Cycle

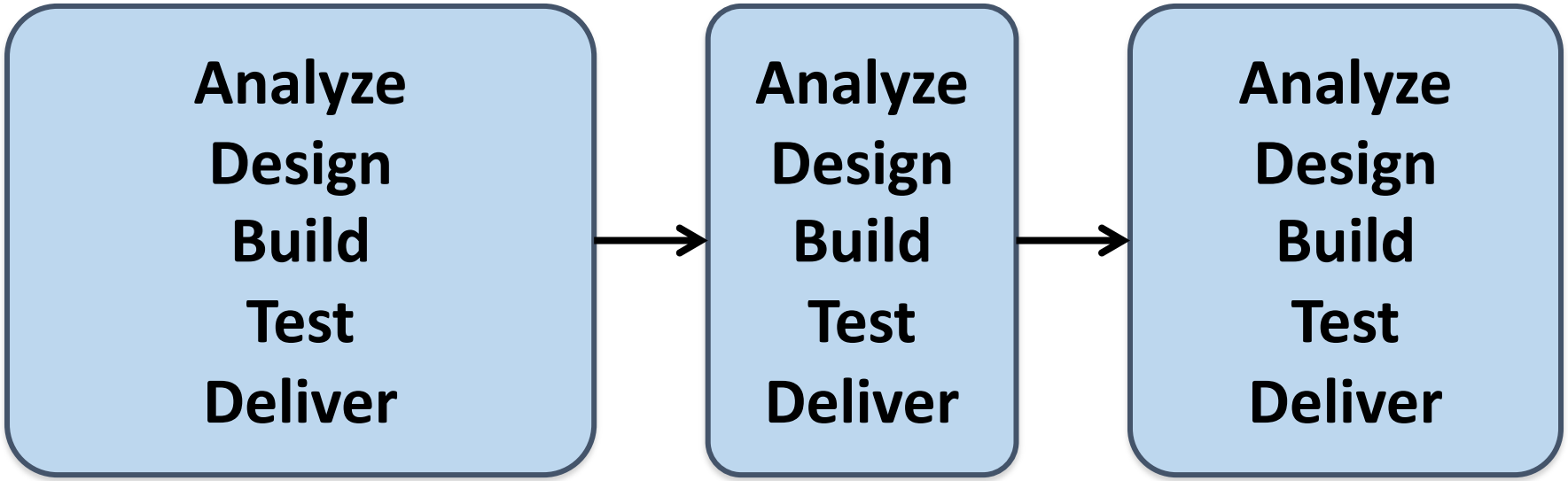
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

Iterative Life Cycle



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

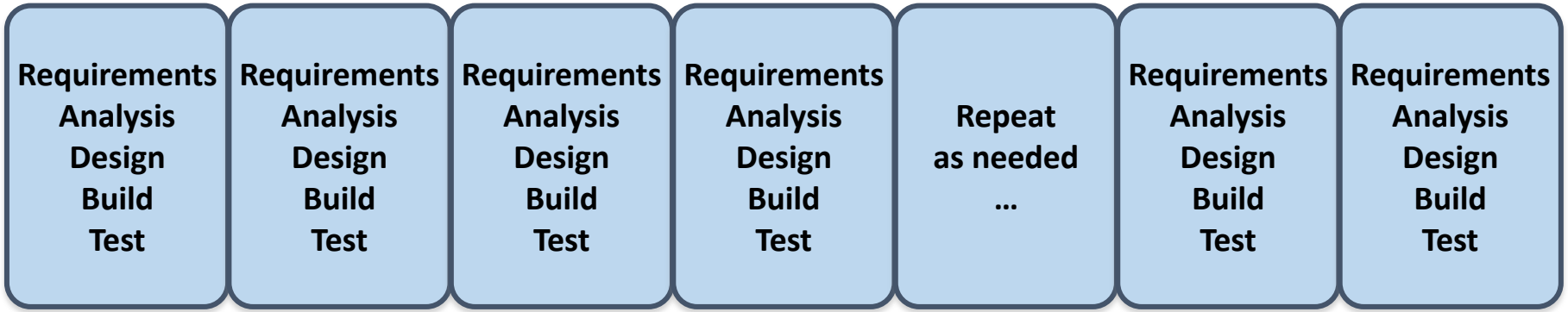
A Life Cycle of Varying-Sized Increments



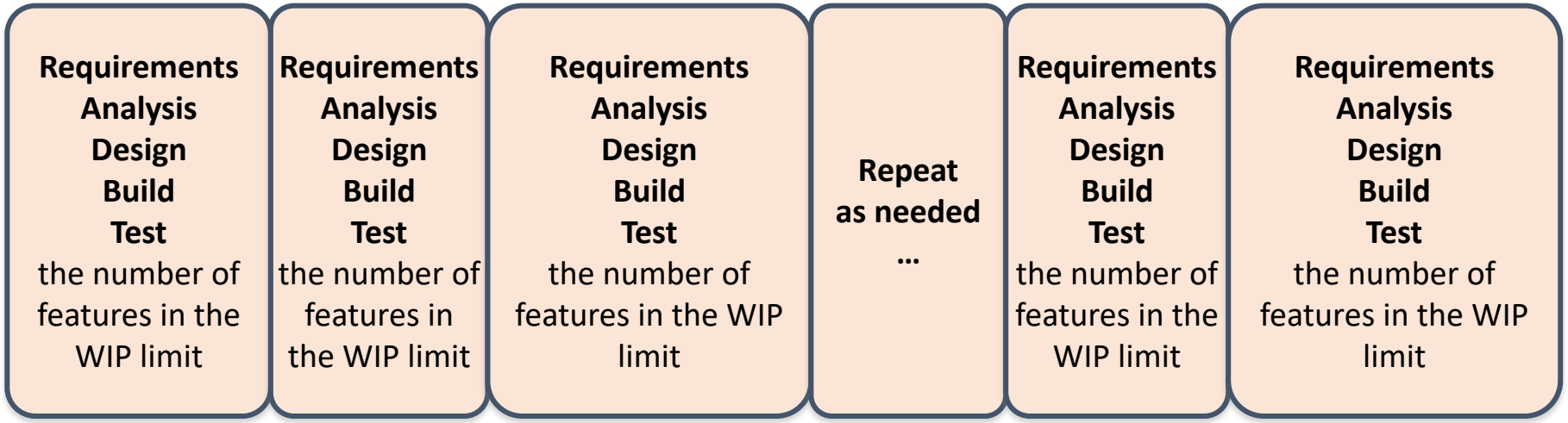
Source: Project Management Institute (2017), Agile Practice Guide, Project Management Institute

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

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.
- Denis Rothman (2024), RAG-Driven Generative AI: Build custom retrieval augmented generation pipelines with LlamaIndex, Deep Lake, and Pinecone, Packt Publishing
- Thomas R. Caldwell (2025), The Agentic AI Bible: The Complete and Up-to-Date Guide to Design, Build, and Scale Goal-Driven, LLM-Powered Agents that Think, Execute and Evolve, Independently published
- Yilei Zhao, Wentao Zhang, Xiao Lei, Yandan Zheng, Mengpu Liu, and Wei Yang Bryan Lim. (2026) "Advancing ESG Intelligence: An Expert-level Agent and Comprehensive Benchmark for Sustainable Finance." arXiv preprint arXiv:2601.08676 (2026).
- NVIDIA DLI (2026), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1
- NVIDIA DLI (2026), Generative AI with Diffusion Models, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-14+V1
- Tucker J. Marion, Mahdi Srour, and Frank Piller (2024), "When Generative AI meets product development." MIT Sloan Management Review 66, no. 1 : 14-15.