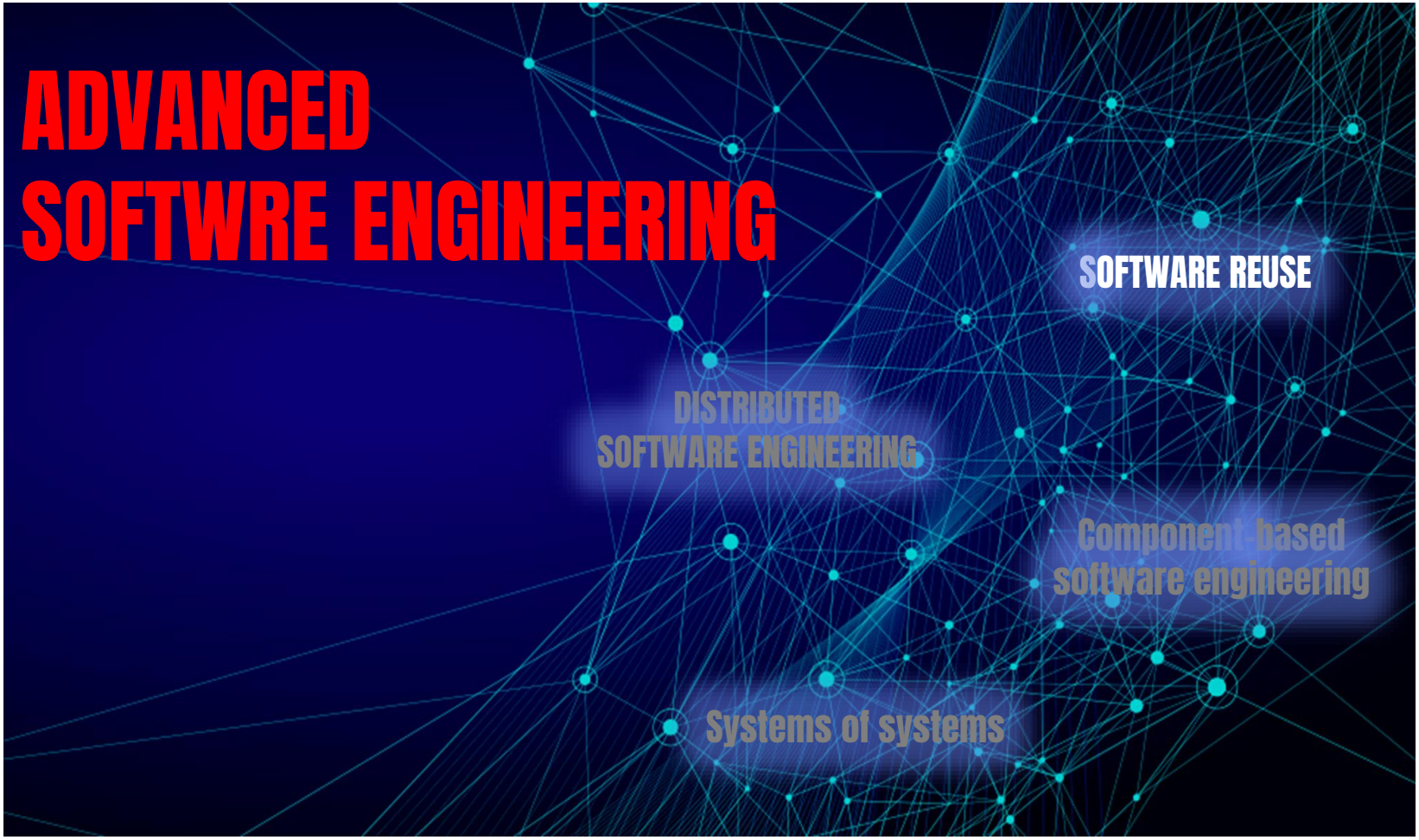
Class7: Software Reuse
Chapter 15

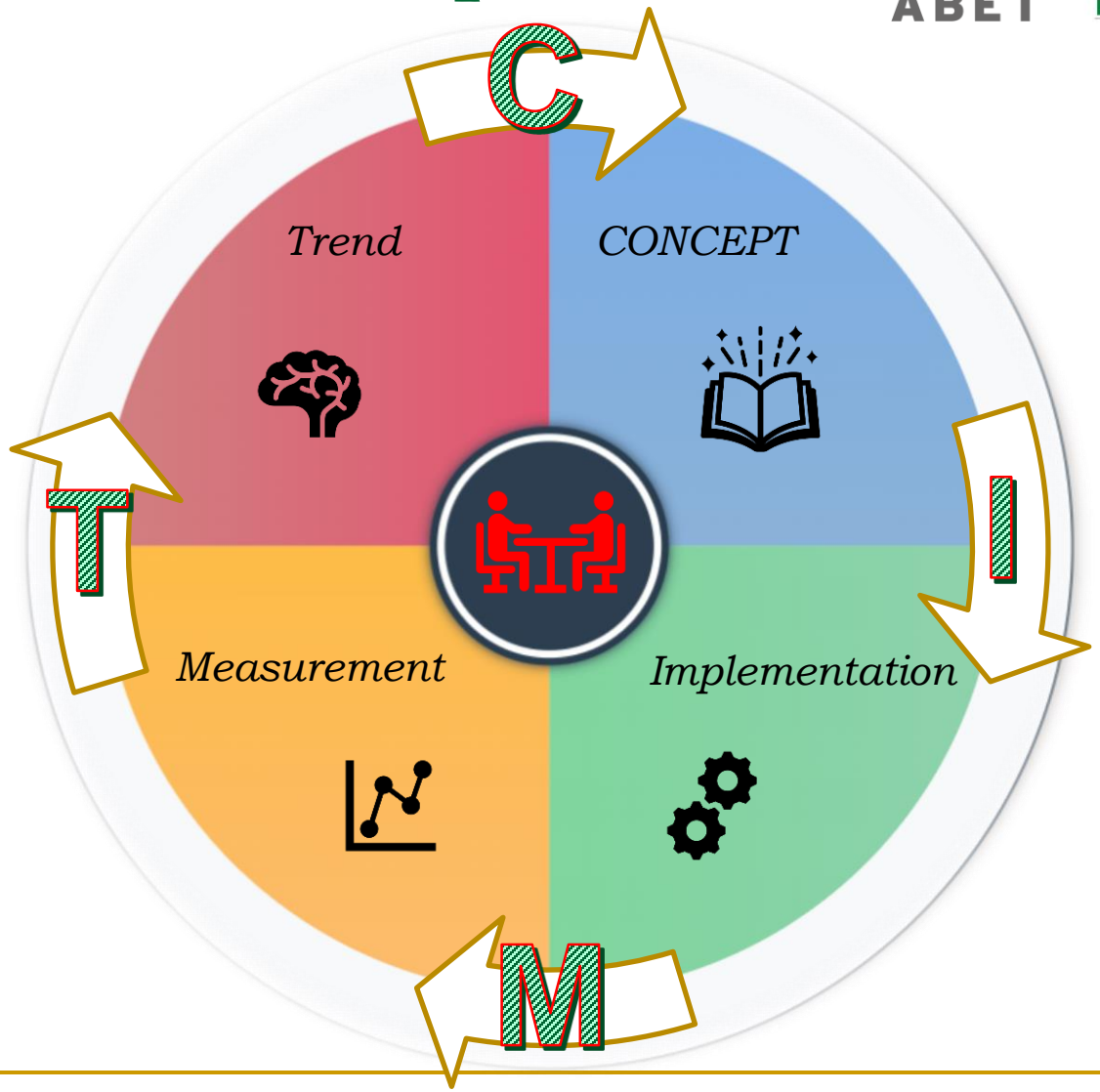
Adeeb Noor, PhD
IT Department
Faculty of Computing and Information Technology
King Abdulaziz University
Fall 2025

Today lecture topic – part 2

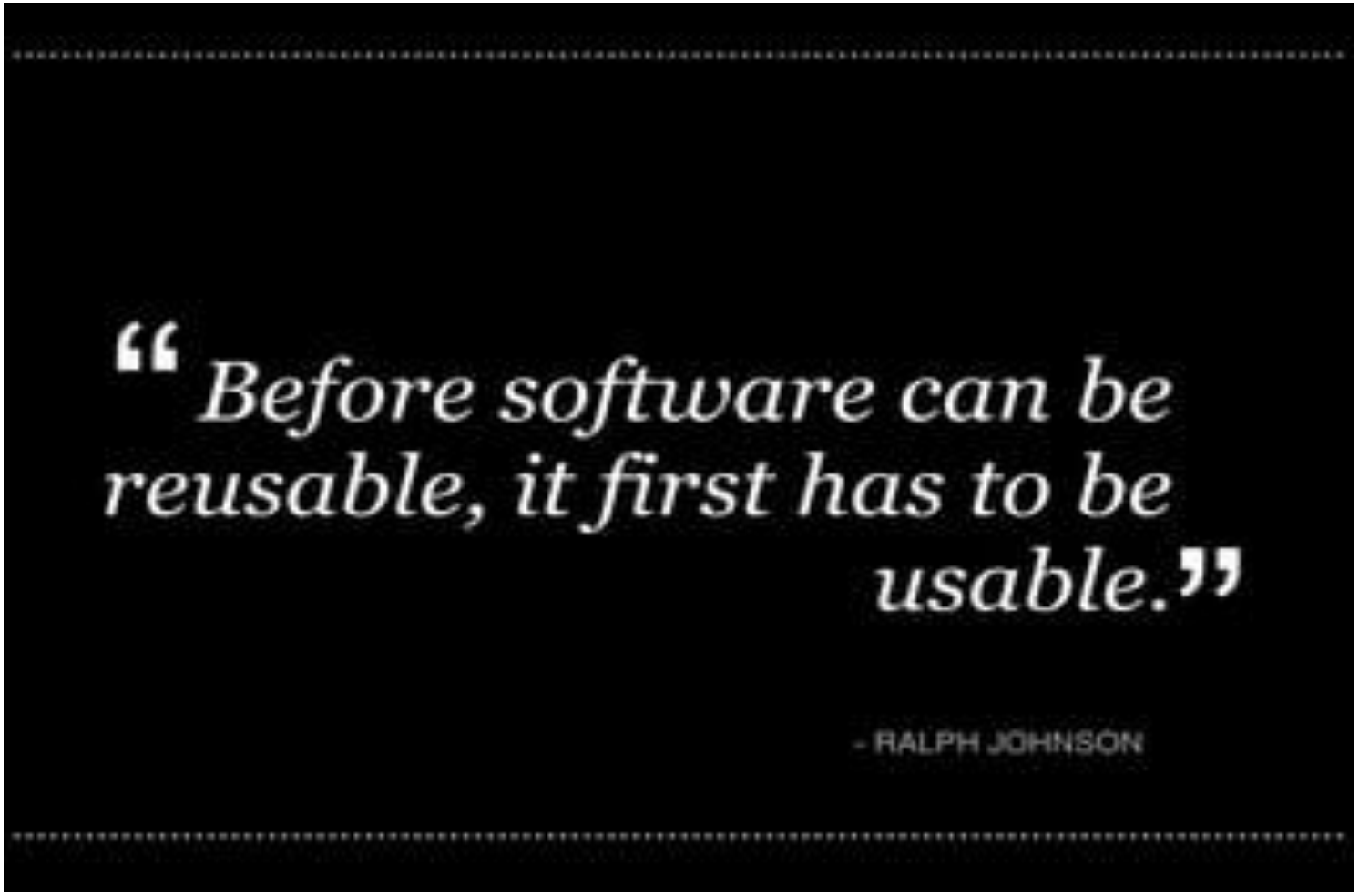


CPIT-455 CIMT compass

Where Academic Meets Business Reality



The big picture



Take-home points

- The Key CLOs: Identify the elements involved in the re-use of code and describe an approach to software reuse based on the composition of standardized, reusable components & Describe an approach to software reuse based on the composition of standardized and reusable components
- The Reuse Landscape (15.1): Software reuse spans from simple functions to entire applications, with choices influenced by project-specific factors
- Application Frameworks (15.2): Frameworks offer reusable architectures with features like security and database support, often employing design patterns and inversion of control
- Software Product Lines (15.3): These are families of applications sharing a common architecture, enabling specialization for individual customer needs through various types of modifications
- Application System Reuse (15.4): This involves reusing large off-the-shelf systems, offering rapid deployment but potential limitations in flexibility and control

Chapter Summary

Software reuse offers diverse techniques, from libraries and frameworks to application system integration, to reduce development time and enhance dependability, while requiring careful consideration of project needs and vendor support.

Software reuse: BE CAREFUL

What happened:

The Ariane 5 rocket's first flight failed due to a software error

Root Causes:

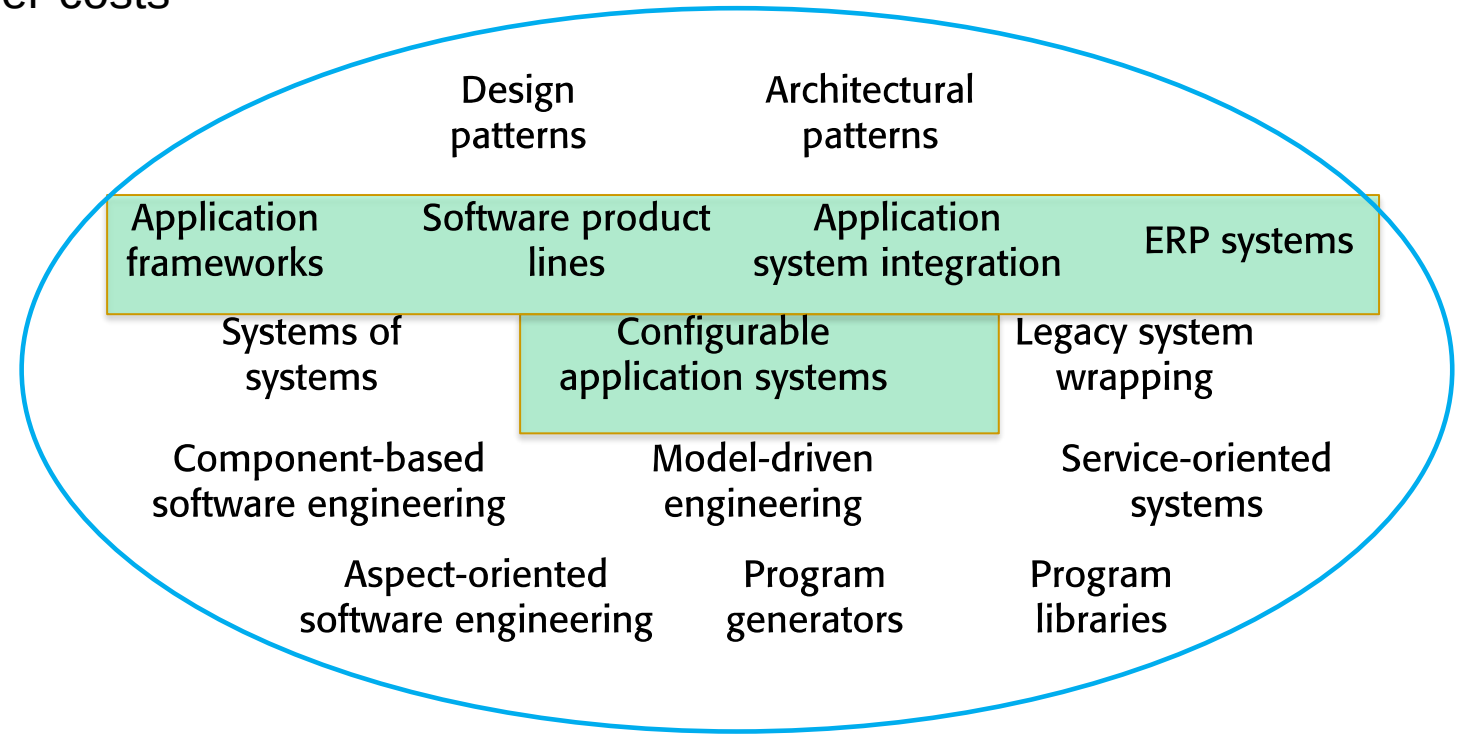
A software module from the slower Ariane 4 rocket was reused in Ariane 5, causing an overflow error due to the higher speed

How to solve Causes:

Thorough validation is crucial when reusing software, even in similar contexts, to ensure compatibility and prevent unexpected errors

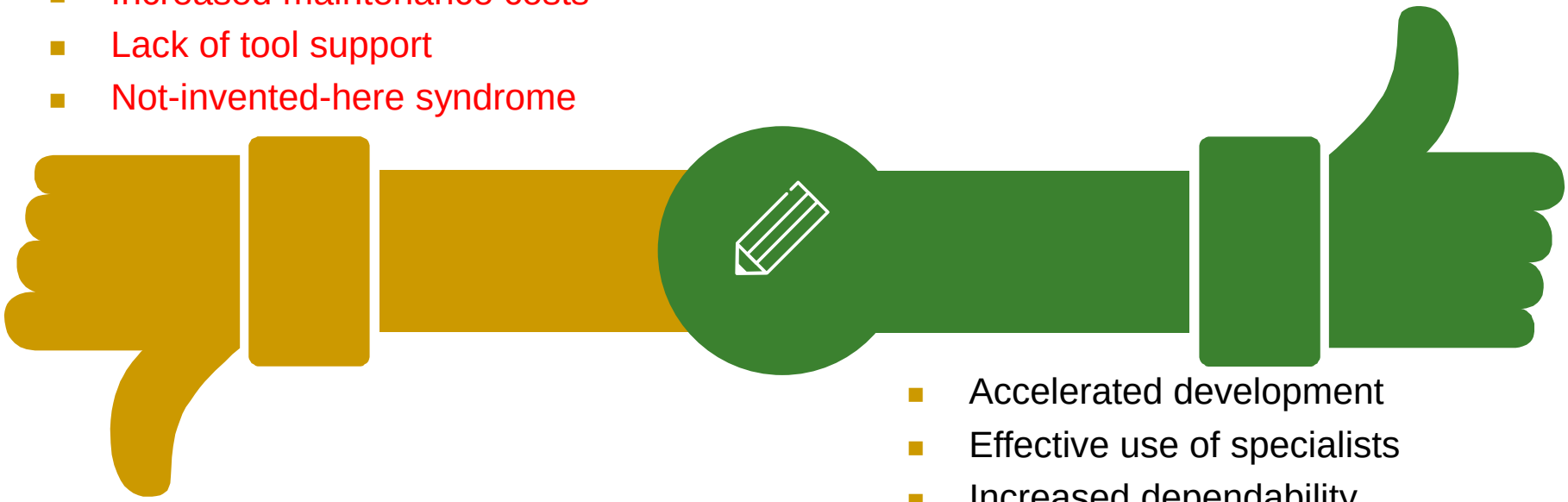
What is software reuse

- Software reuse is the process of incorporating existing software components, which can range from entire system reuse or application reuse to smaller component reuse or even individual object and function reuse, into new software systems to reduce development time, improve software quality, and lower costs



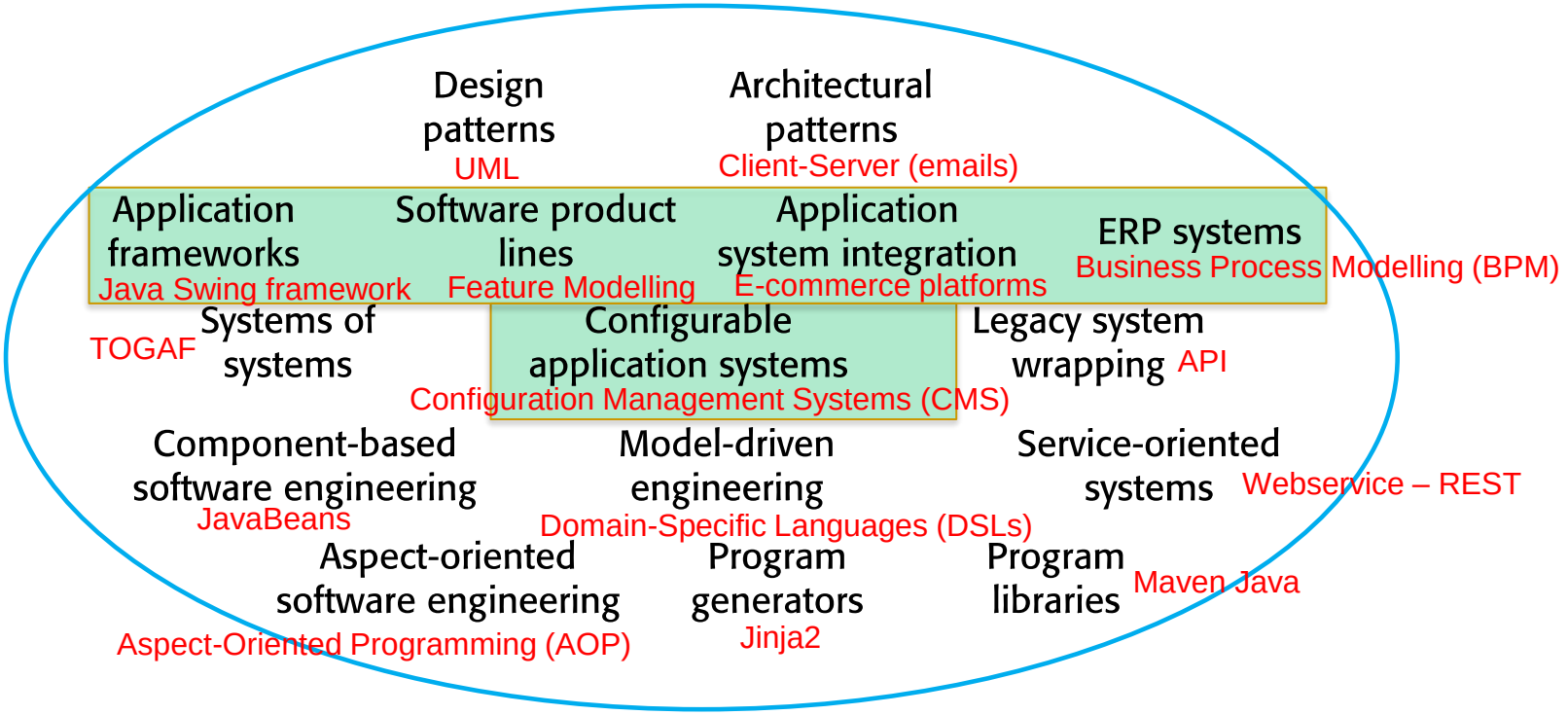
Problems vs. Benefits

- Creating, maintaining, and using a component library
- Finding, understanding, and adapting reusable components
- Increased maintenance costs
- Lack of tool support
- Not-invented-here syndrome



- Accelerated development
- Effective use of specialists
- Increased dependability
- Lower development costs
- Reduced process risk
- Standards compliance

15.1 Approaches that support software reuse



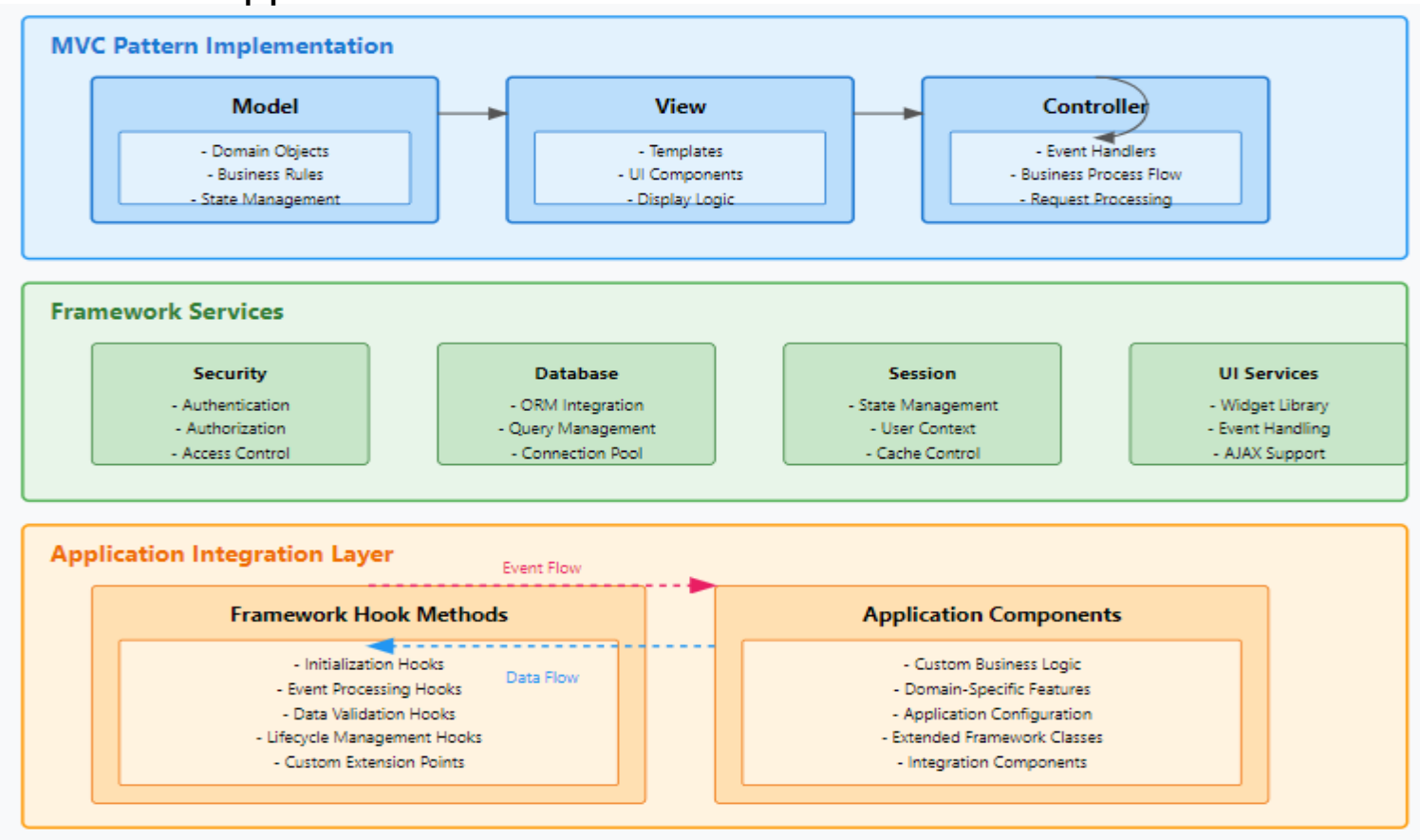
The Reuse Landscape

Reuse planning factors

- The development schedule for the software
- The expected software lifetime
- The background, skills and experience of the development team
- The criticality of the software and its non-functional requirements
- The application domain
- The execution platform for the software

15.2 Application frameworks

Framework definition: an integrated set of software artefacts (such as classes, objects and components) that collaborate to provide a reusable architecture for a family of related applications

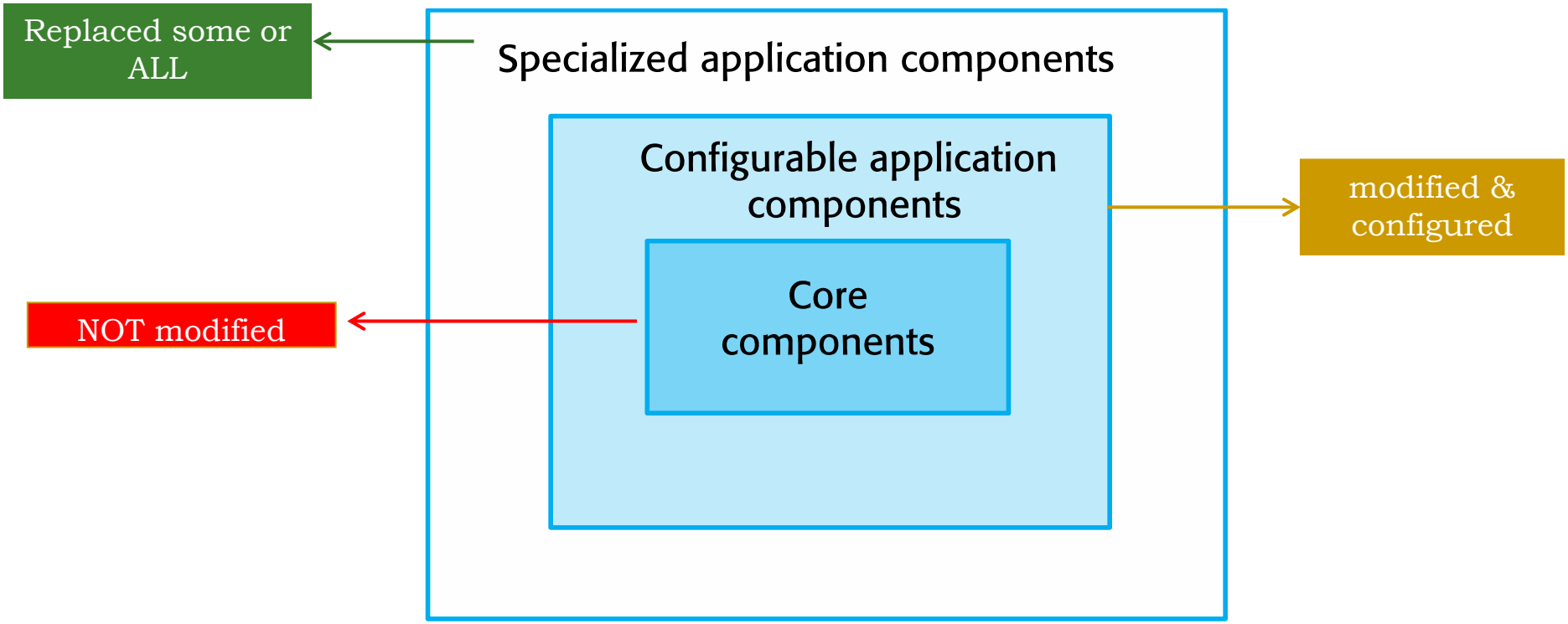


Framework classes

- System infrastructure frameworks
  - Support the development of system infrastructures such as communications, user interfaces and compilers
- Middleware integration frameworks
  - Standards and classes that support component communication and information exchange
- Enterprise application frameworks
  - Support the development of specific types of application such as telecommunications or financial systems

15.3 Software product lines

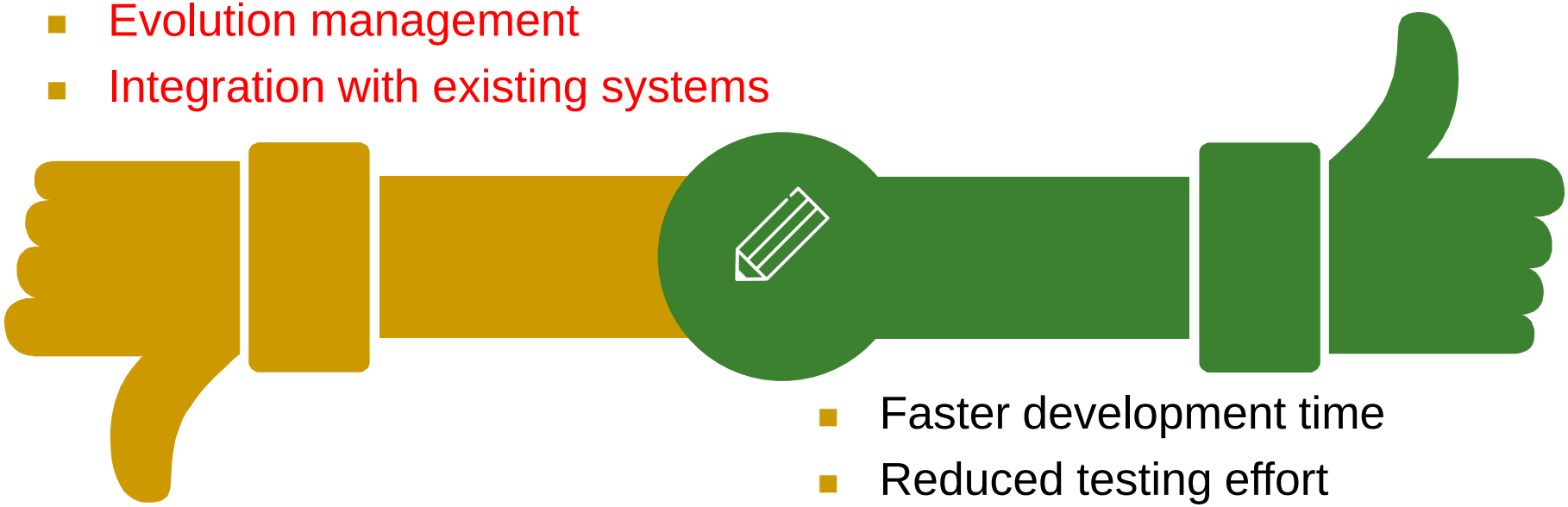
- Software product lines or application families are applications with generic functionality that can be adapted and configured for use in a specific context



Concept

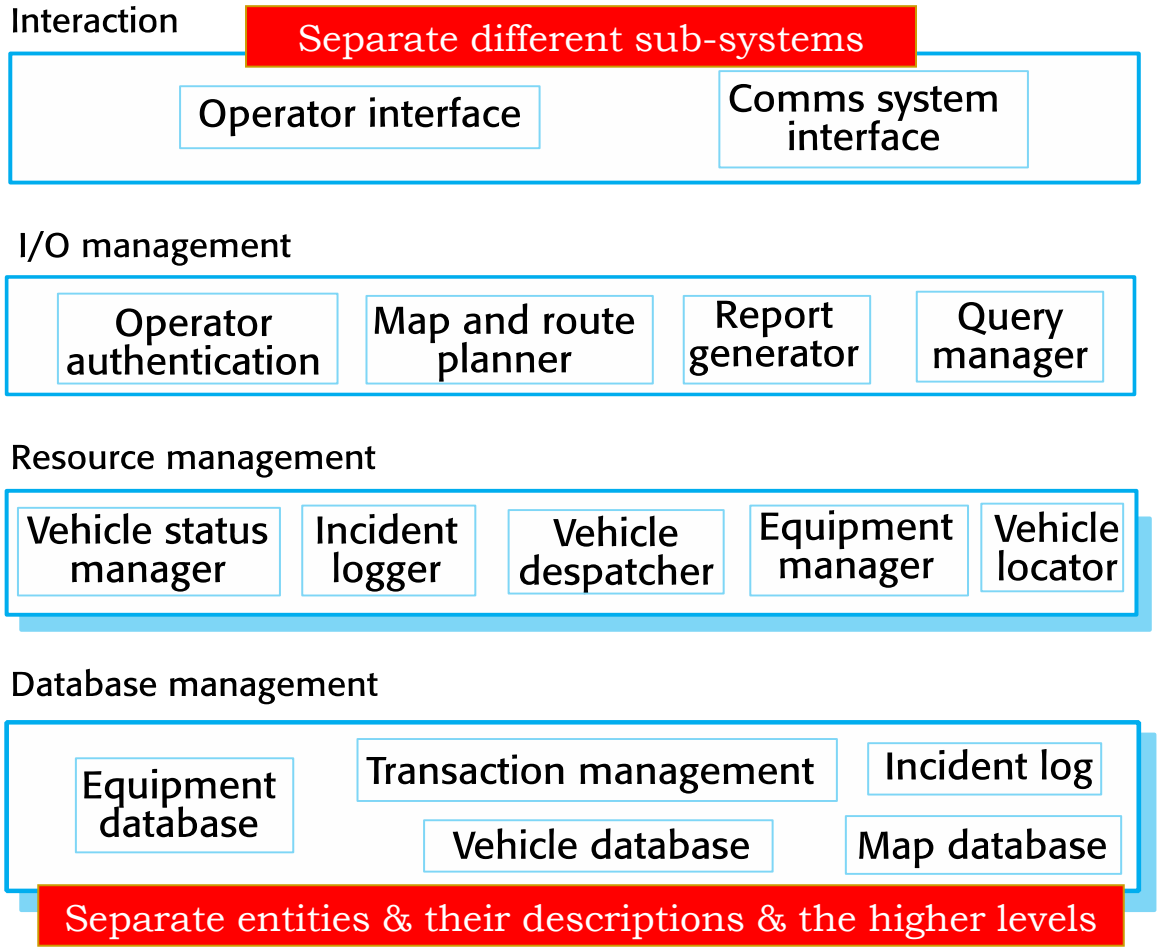
Challenges vs. Benefits of SPL

- Requirement compromises
- Complex configuration
- Evolution management
- Integration with existing systems



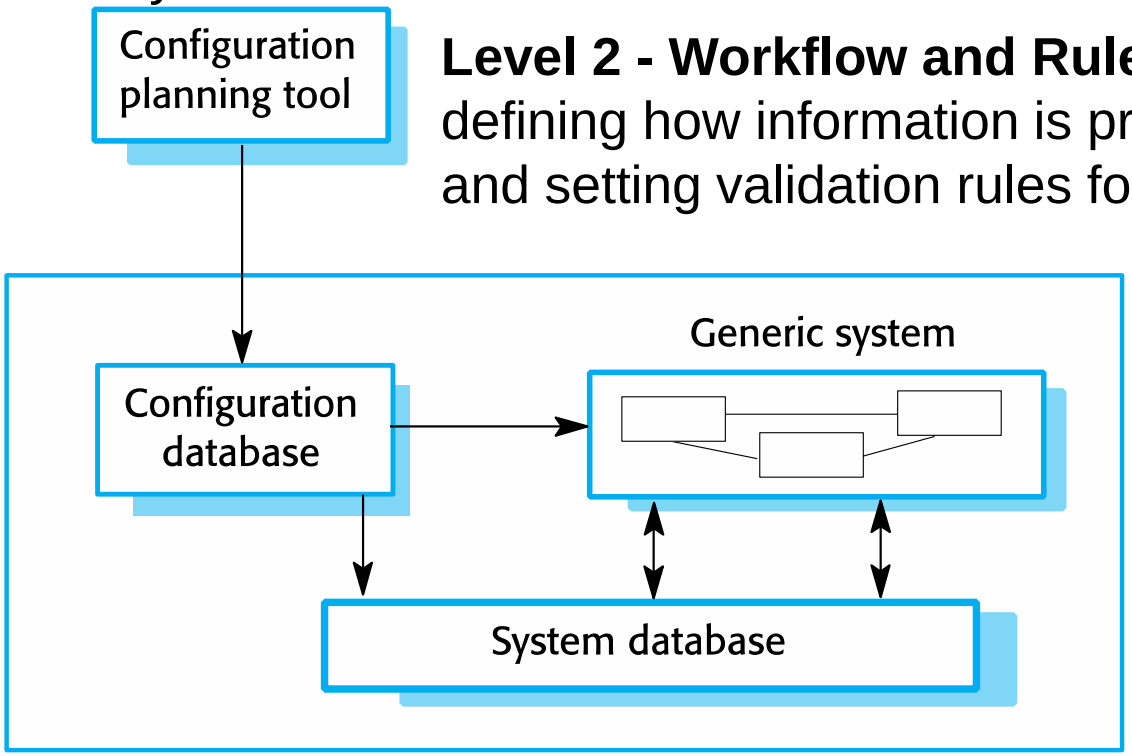
- Faster development time
- Reduced testing effort
- Consistent architecture
- Domain expertise accumulation

The product line architecture of a vehicle dIspatcher



Deployment-time configuration

Level 1- Component Selection: choosing the modules that provide the needed functionality



Level 2 - Workflow and Rule Definition: defining how information is processed and setting validation rules for user inputs

Level 3 - Parameter Definition: setting specific values for system parameters to tailor the application

15.4 Application system reuse

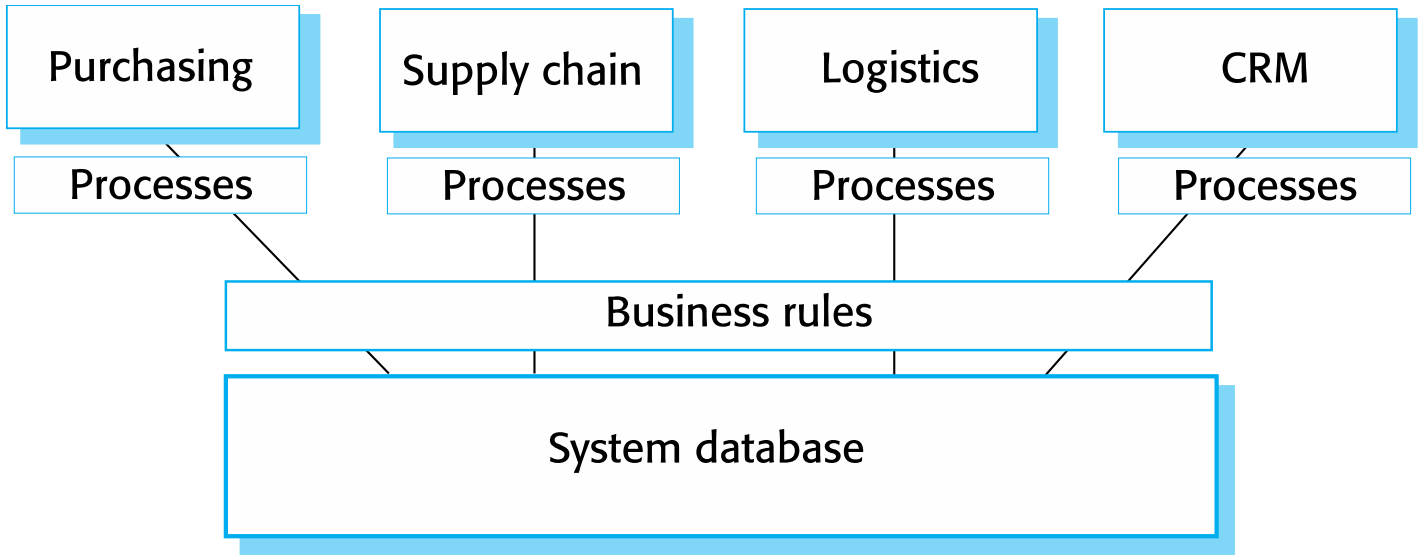
- **An application system product** is a software system can be adapted for different customers without changing the source code of the system

Configurable application systems	Application system integration
Single product that provides the functionality required by a customer	Several heterogeneous system products are integrated to provide customized functionality
Based around a generic solution and standardized processes	Flexible solutions may be developed for customer processes
Development focus is on system configuration	Development focus is on system integration
System vendor is responsible for maintenance	System owner is responsible for maintenance
System vendor provides the platform for the system	System owner provides the platform for the system

- Benefits: rapid deployment, easier assessment of functionality
- Problems: need to adapt requirements, difficulties in changing product

15.4.1 Configurable application systems

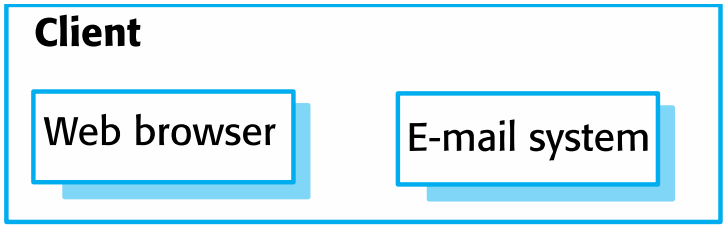
- **An Enterprise Resource Planning (ERP)** system is a generic system that supports common business processes such as ordering and invoicing, manufacturing, etc
- The generic core is adapted by including modules and by incorporating knowledge of business processes and rules



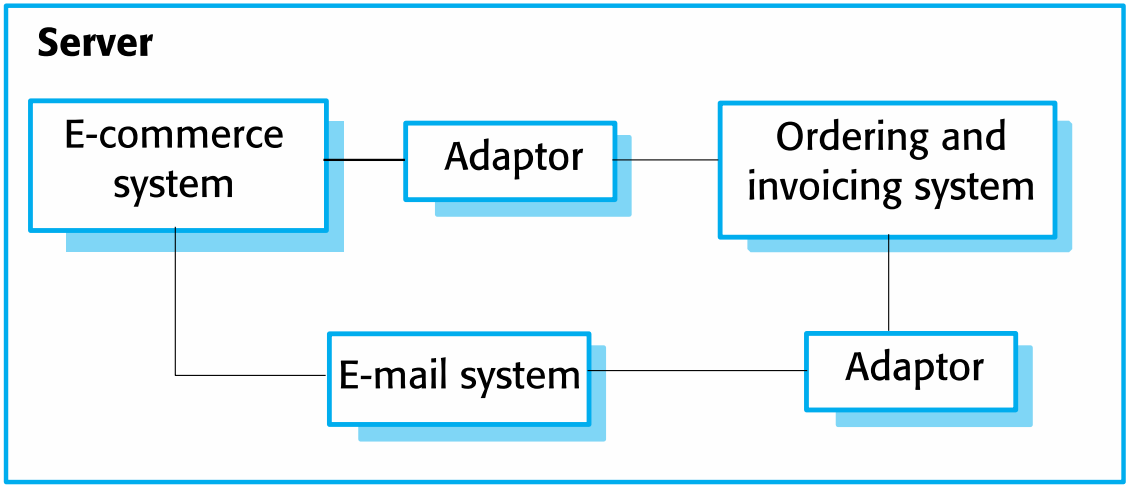
15.4.2 Integrated application systems

- **Integrated application systems** are applications that include two or more application system products and/or legacy application systems

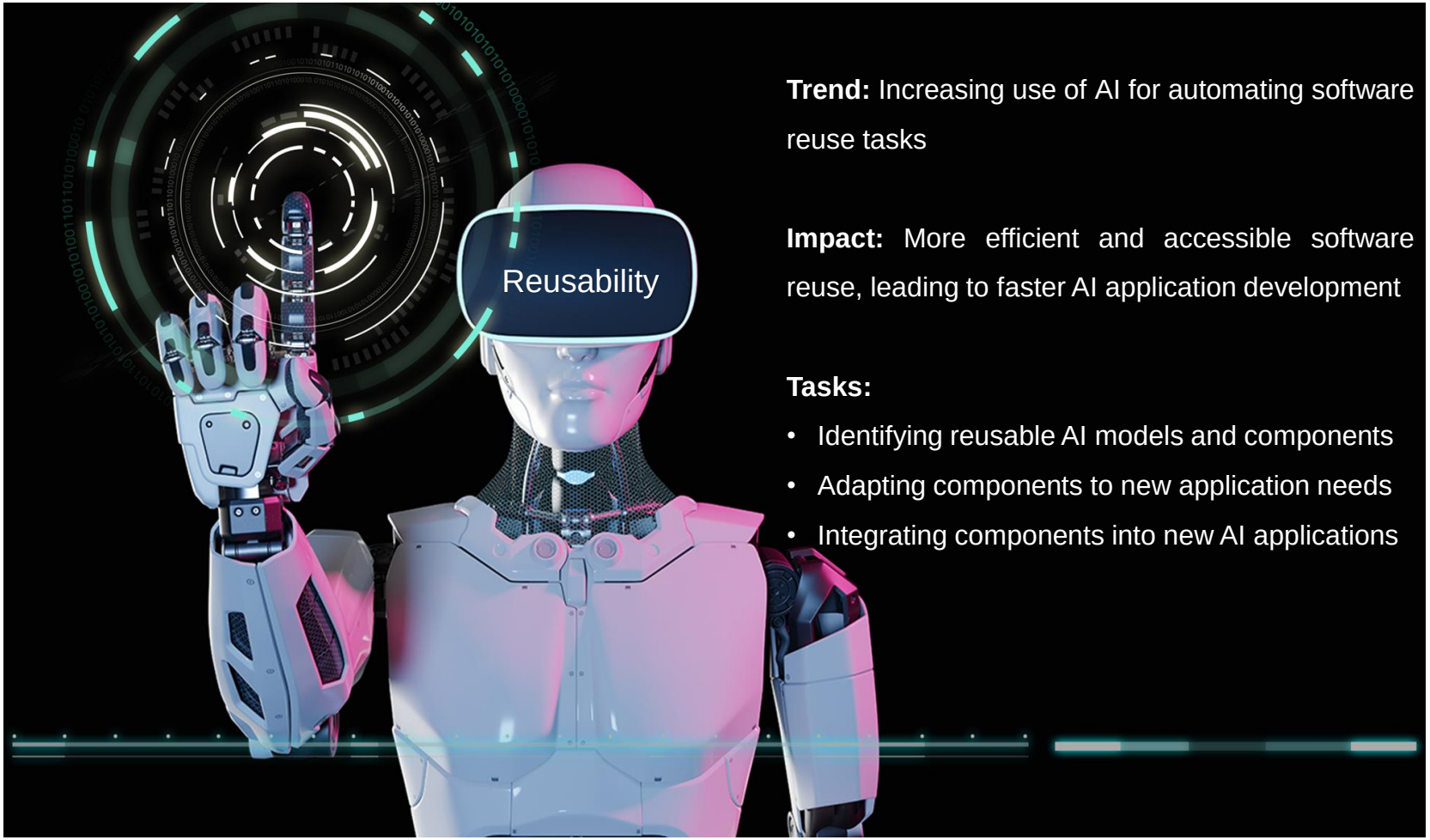
Desing focus:
- Functionalities
- Data exchanged
- Features selection
- Service wrapper



Be carful:
- No control
- Inter-operability
- System evolution
- Vendor support



Software Reuse in AI era



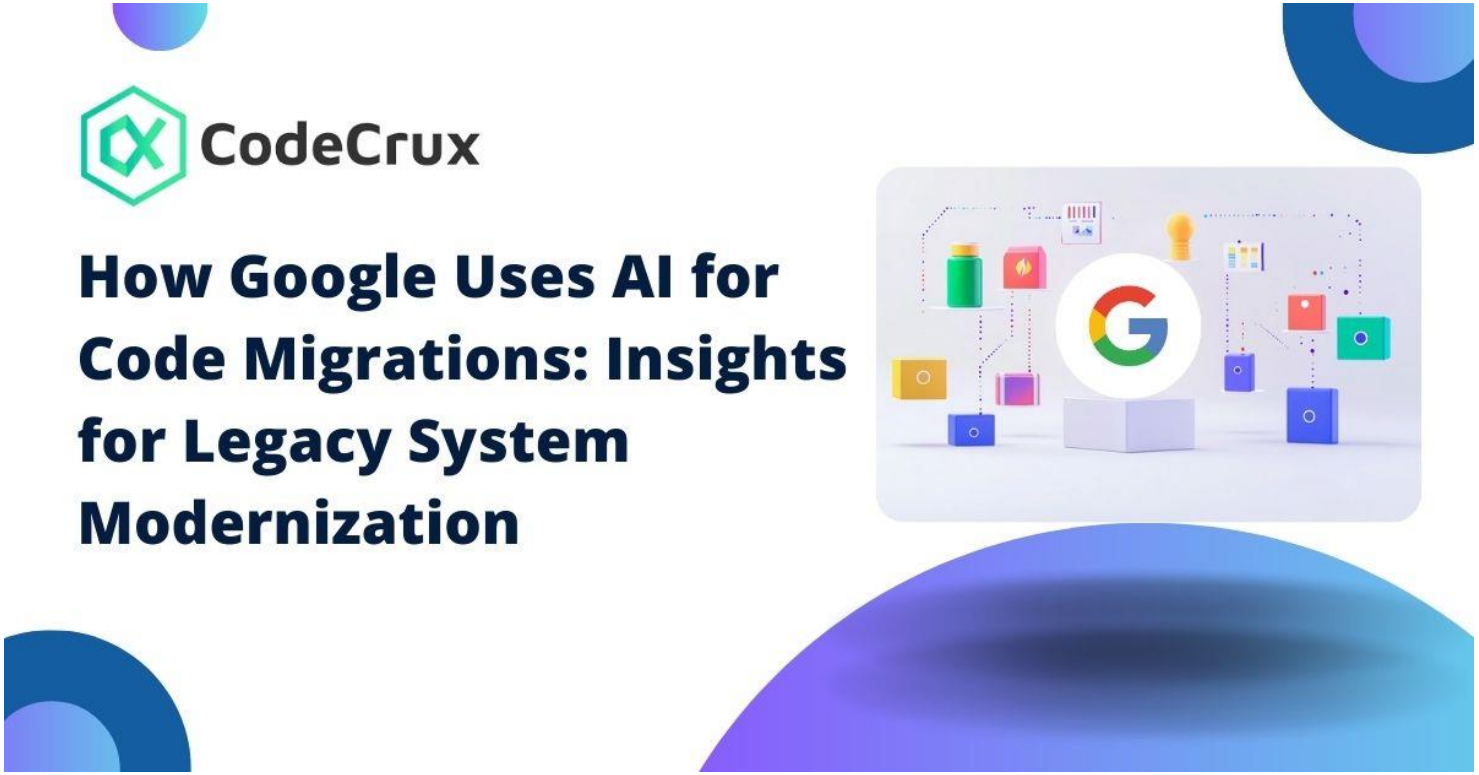
Trend: Increasing use of AI for automating software reuse tasks

Impact: More efficient and accessible software reuse, leading to faster AI application development

Tasks:

- Identifying reusable AI models and components
- Adapting components to new application needs
- Integrating components into new AI applications

Ex; Google uses AI to identify and reuse code snippets across its vast codebase



Aligning with industry trends

- **Model zoos:** Sharing and reusing pre-trained AI models
- **Transfer learning:** Adapting existing AI models to new tasks
- **Low-code/no-code AI development:** Enabling rapid development of AI applications using reusable components
- **AI marketplaces:** Facilitating the exchange and reuse of AI models and components
- **Cloud-based AI platforms:** Providing access to reusable AI components and development tools

To master today's lesson

- No homework this time