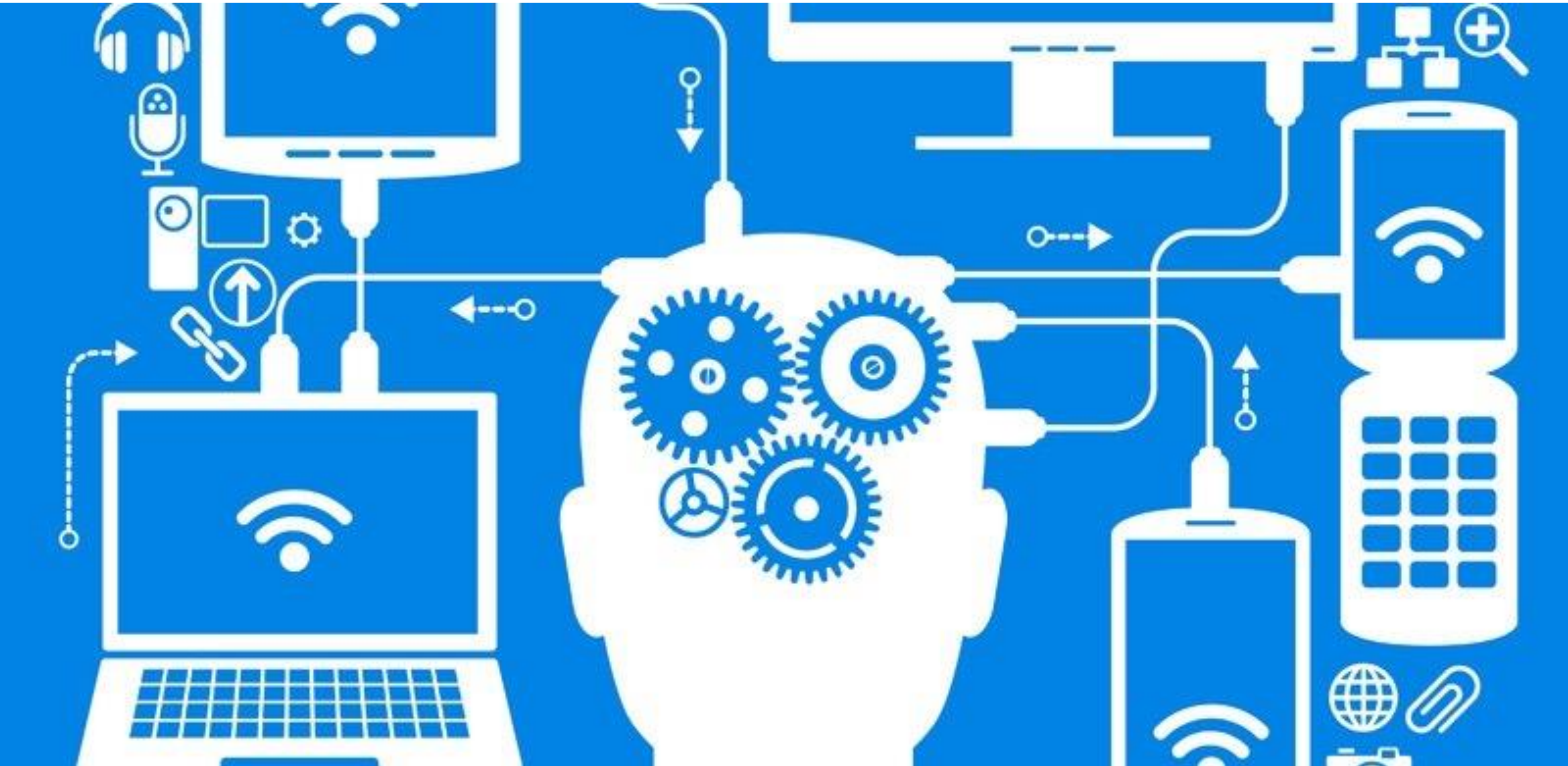
Introduction to Software Engineering

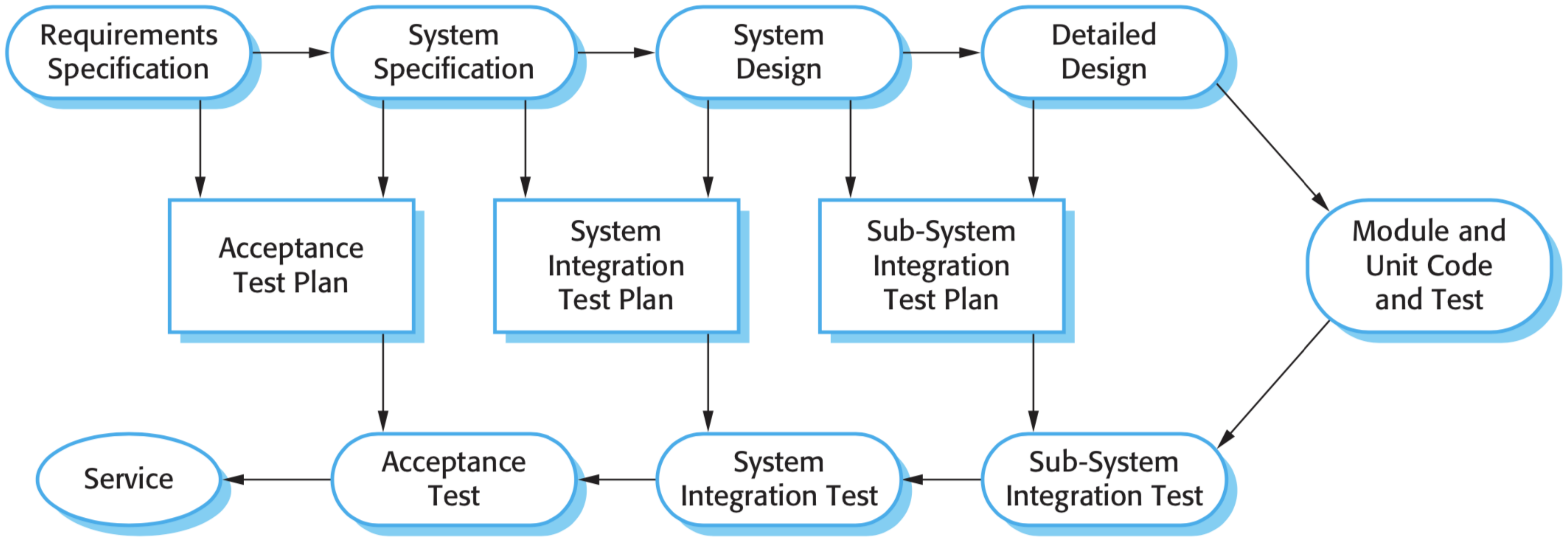
Elishai Ezra Tsur
Neuro & Biomorphic Engineering LAB

Dan Zilberstein

JERUSALEM COLLEGE OF TECHNOLOGY
LEV ACADEMIC CENTER



Software Verification & Validation


Requirements Specification
System Specification
System Design
Detailed Design
Acceptance Test Plan
System Integration Test Plan
Sub-System Integration Test Plan
Module and Unit Code and Test
Service
Acceptance Test
System Integration Test
Sub-System Integration Test

Advanced concepts in Software Engineering: Software Reuse

Application system reuse. The whole of an application system may be reused by incorporating it without changing into other systems or by configuring the application for different customers.

Component reuse. Components of an application, ranging in size from subsystems to single objects, may be reused. For example, a pattern-matching system developed as part of a text-processing system may be reused in a database management system.

Object and function reuse. Software components that implement a single function, such as a mathematical function, or an object class may be reused. This form of reuse, based around standard libraries, has been common for the past 40 years.

Advanced concepts in Software Engineering:
Software Reuse Benefits

Increased dependability

Reused software, which has been tried and tested in working systems, should be more dependable than new software. Its **design and implementation faults** should have been found and fixed.

Advanced concepts in Software Engineering: Software Reuse Benefits

Reduced process risk

The cost of existing software is already known, whereas the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in **project cost estimation**. This is particularly true when relatively large software components such as subsystems are reused.

Advanced concepts in Software Engineering: Software Reuse Benefits

Effective use of specialists

Instead of doing the same work over and over again, application specialists can develop reusable software that **encapsulates their knowledge**.

Advanced concepts in Software Engineering: Software Reuse Benefits

Standards compliance

Some standards, such as user interface standards, can be implemented as a set of reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of **standard user interfaces** improves dependability because users make fewer mistakes when presented with a familiar interface.

Advanced concepts in Software Engineering: Software Reuse Benefits

Accelerated development

Bringing a system to **market as early as possible** is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced.

Advanced concepts in Software Engineering: Software Reuse Problems

Increased maintenance costs – if the source code of a reused software system or component is not available, then maintenance costs may be higher.

Lack of tool support – some software tools do not support development with reuse. This is particularly true for tools that support embedded systems engineering, less so for object-oriented development tools.

Not-invented-here syndrome – **s**ome software engineers prefer to rewrite components because they believe they can improve on them.

Creating, maintaining, and using a component library – populating a reusable component library and ensuring the software developers can use this library can be expensive.

Finding, understanding, and adapting reusable components – software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment.

Advanced concepts in Software Engineering: Software Reuse

Approaches: application frameworks

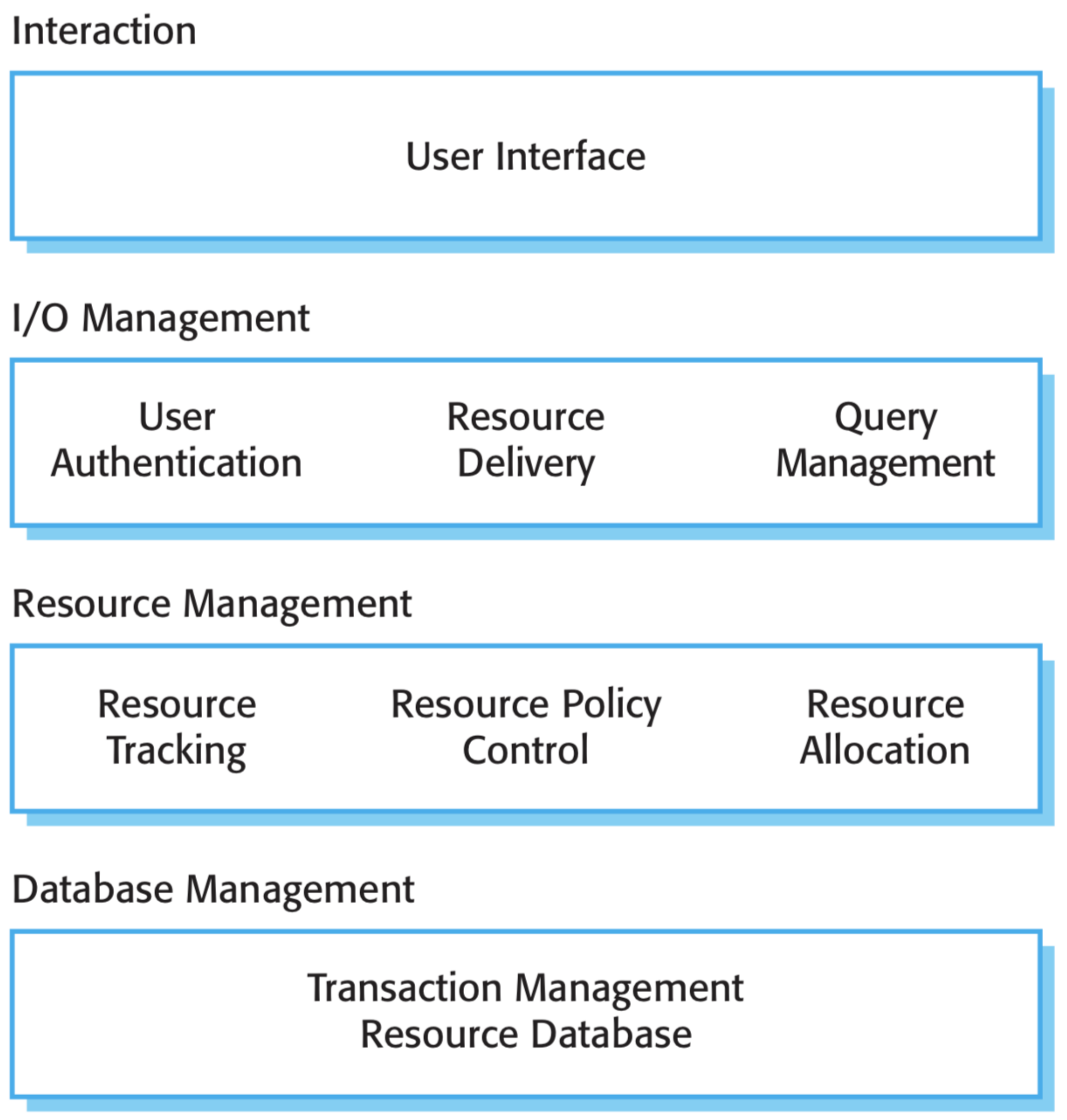
Collections of abstract and concrete classes are adapted and extended to create application systems.

As it is practiced in our class…

Advanced concepts in Software Engineering: Software Reuse

Approaches: Software product lines

An application type is generalized around a common architecture so that it can be adapted for different customers.

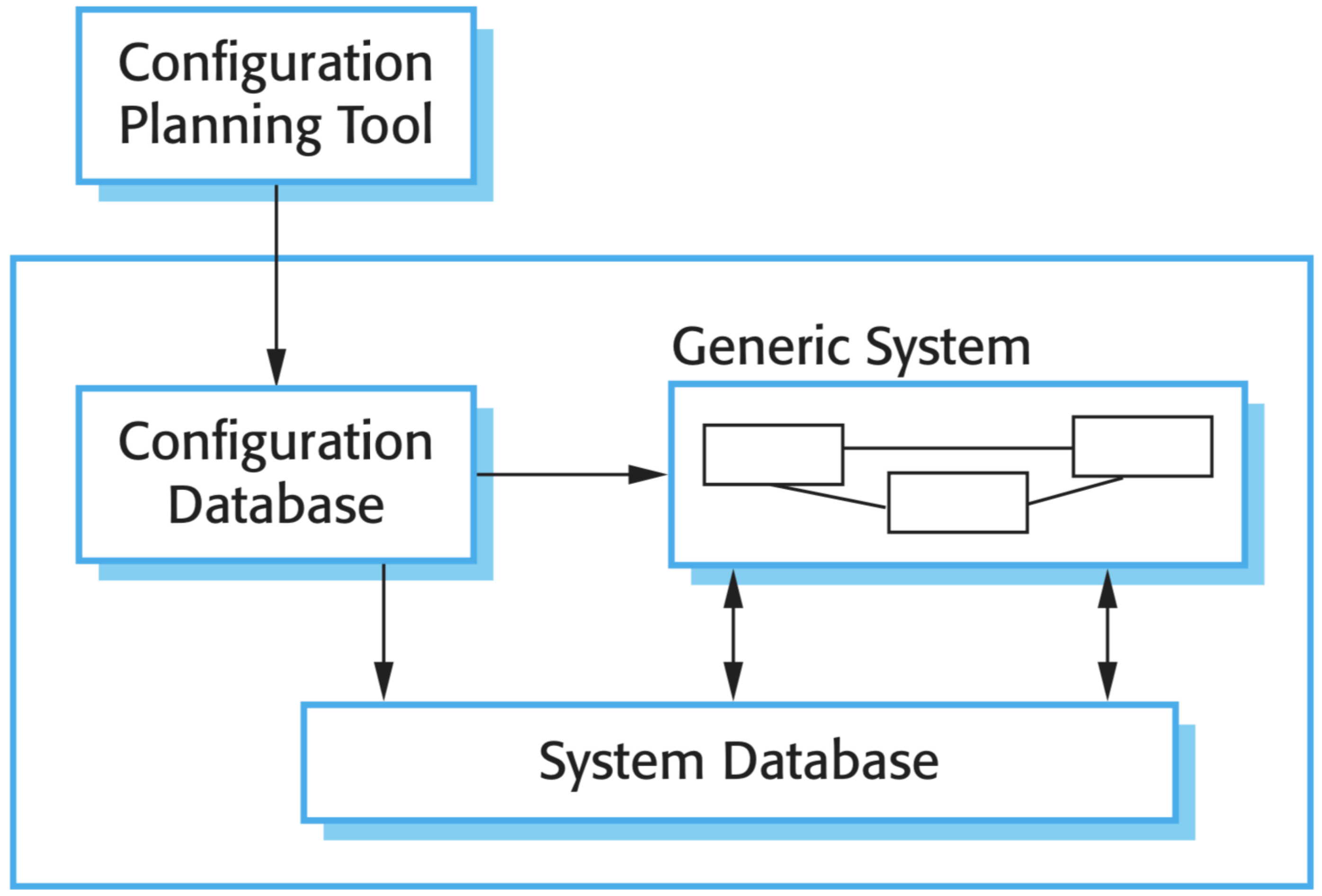


$\longrightarrow$ Generic System

Advanced concepts in Software Engineering: Software Reuse

Approaches: Software product lines

An application type is generalized around a common architecture so that it can be adapted for different customers.

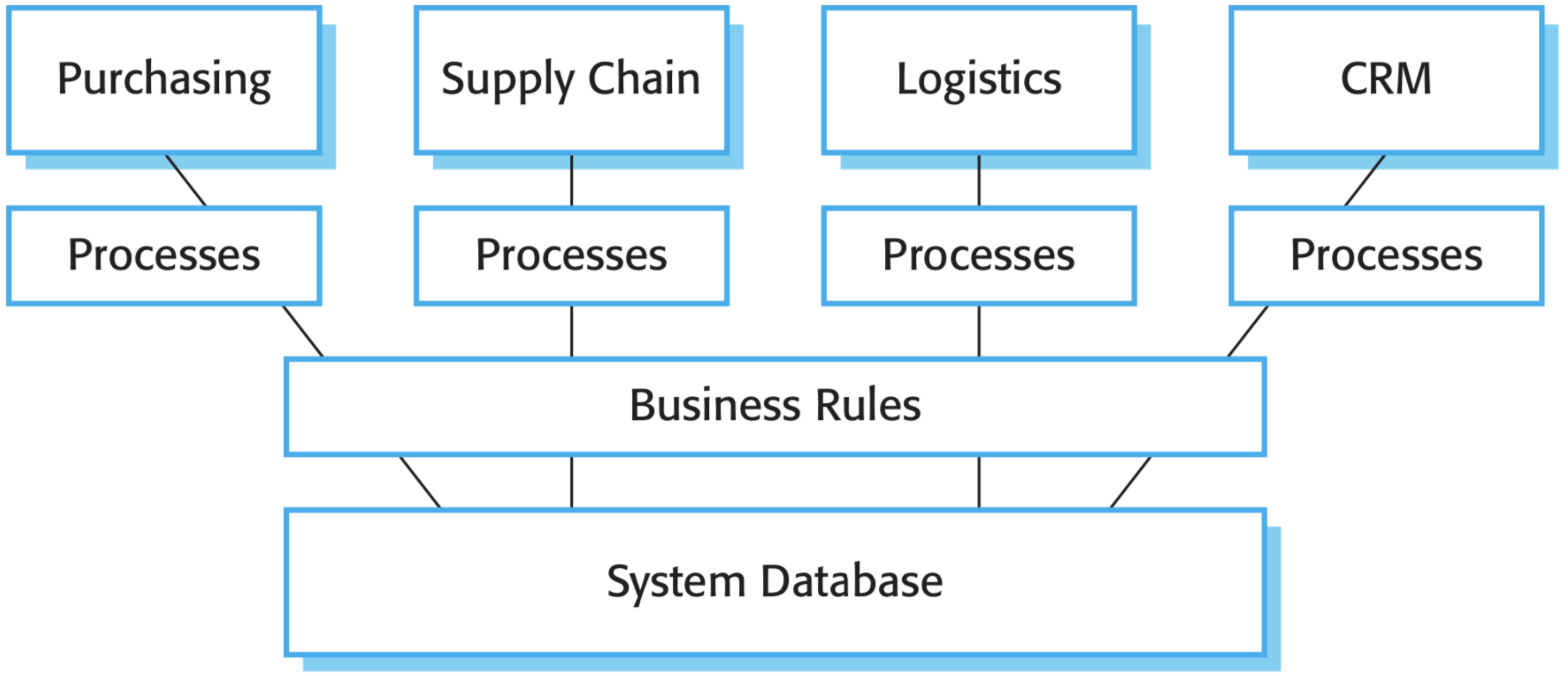


Advanced concepts in Software Engineering: Software Reuse

Approaches: commercial-off-the-shelf (COTS) product

Aa software system that can be adapted to the needs of different customers without changing the source code of the system:

For example, an **Enterprise Resource Planning (ERP)** system may support all of the manufacturing, ordering, and customer relationship management activities in a large company.



customer relationship management