The software reuse landscape

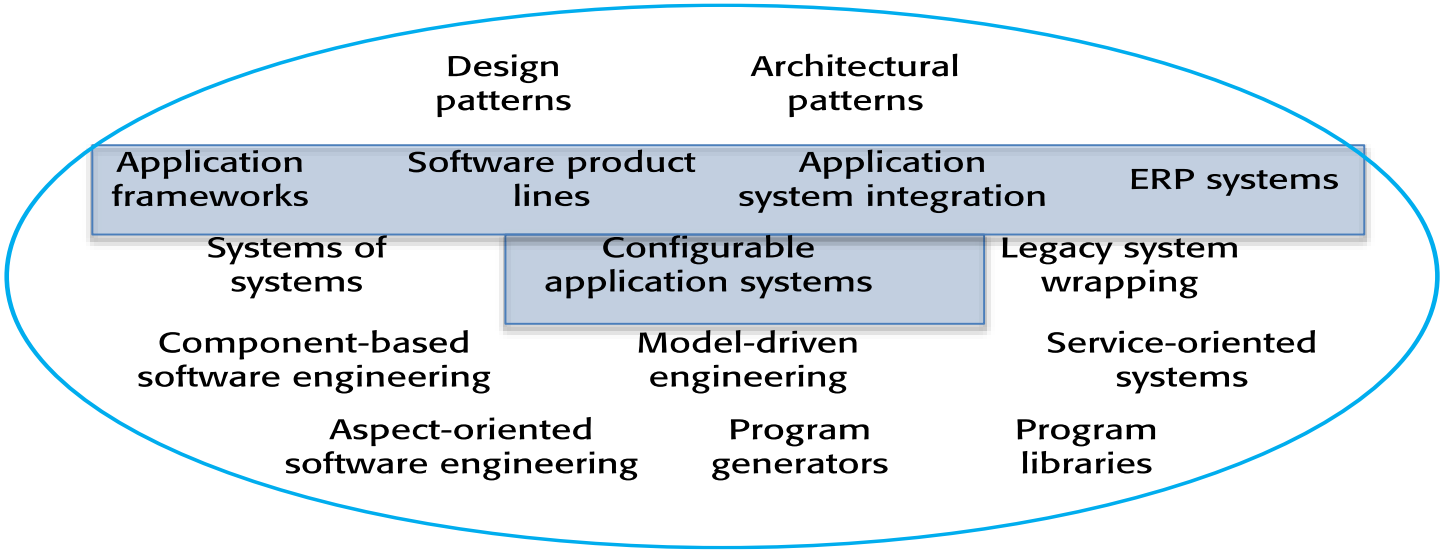
Software Engineering 10

Since the beginning of the 21st century, there has been a major switch to reuse-based software engineering

Although reuse is often simply thought of as the reuse of system components, there are many different approaches to reuse that may be used.

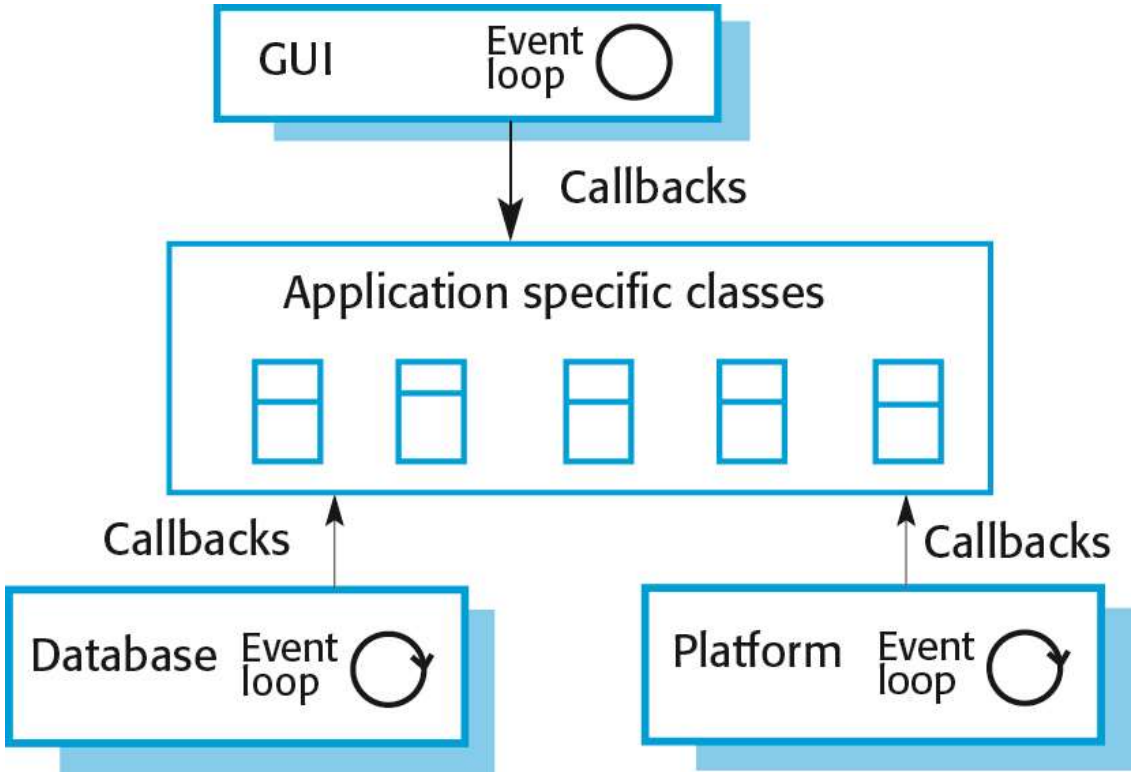
Reuse is possible at a range of levels from simple functions to complete application systems.

The reuse landscape



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

Application frameworks



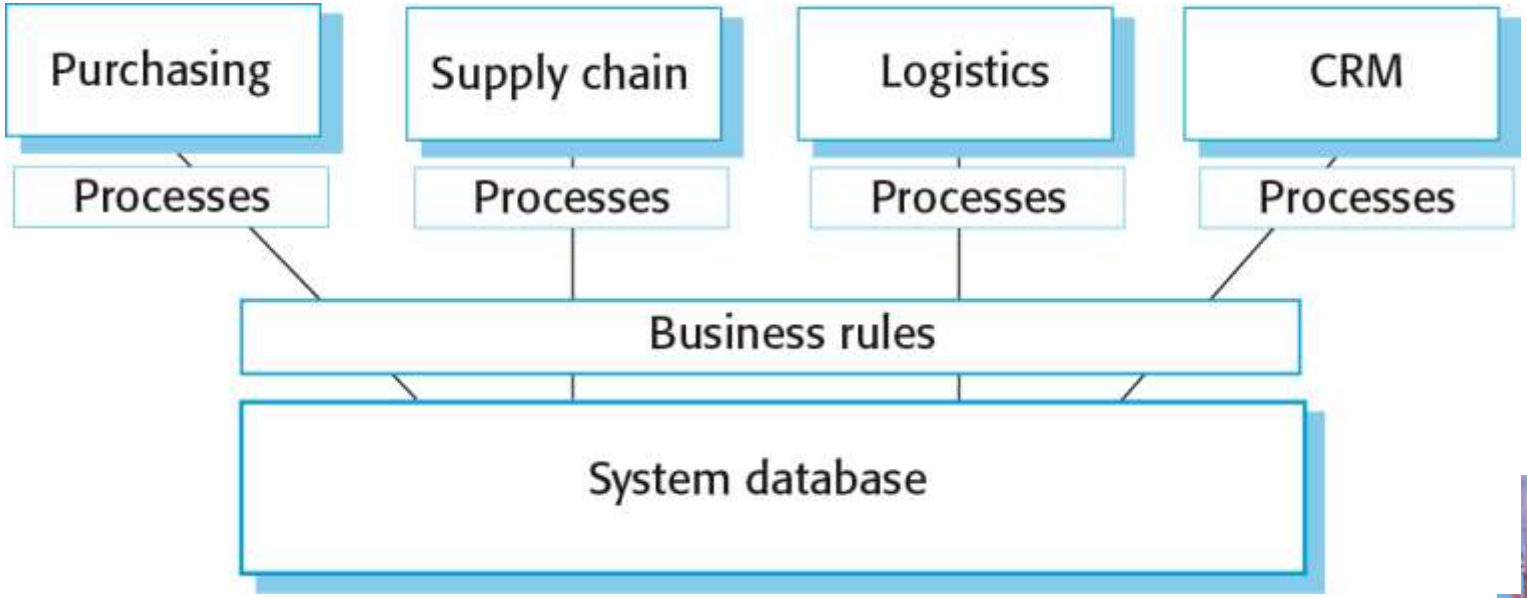
Application system integration:
Two or more application systems are integrated to provide extended functionality.

Configurable application systems: Domain-specific systems are designed so that they can be configured to the needs of specific system customers.

Systems of systems:
Two or more independently-owned, distributed systems are integrated to create a new system.

ERP systems:
Large-scale systems that encapsulate generic business functionality and rules are configured for an organization.

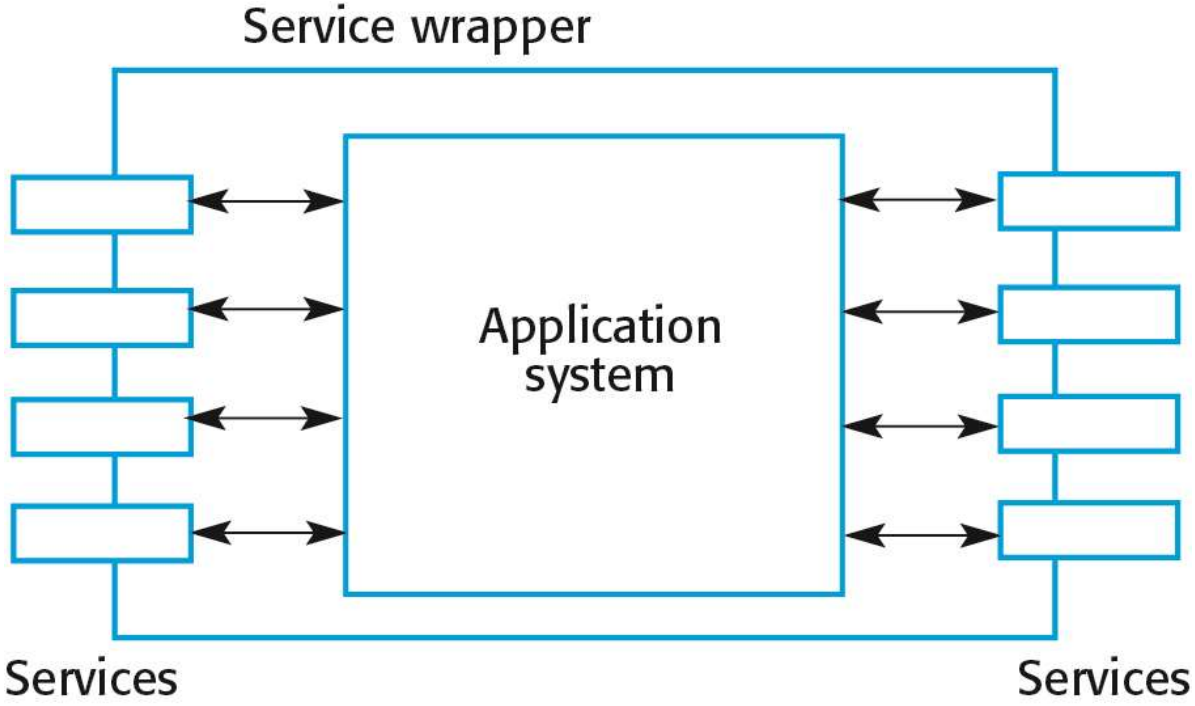
ERP systems


Purchasing
Supply chain
Logistics
CRM
Processes
Processes
Processes
Processes
Business rules
System database


Software Engineering

Legacy system reuse:
Legacy systems (Chapter 9) are ‘wrapped’ by defining a set of interfaces and providing access to these legacy systems through these interfaces.

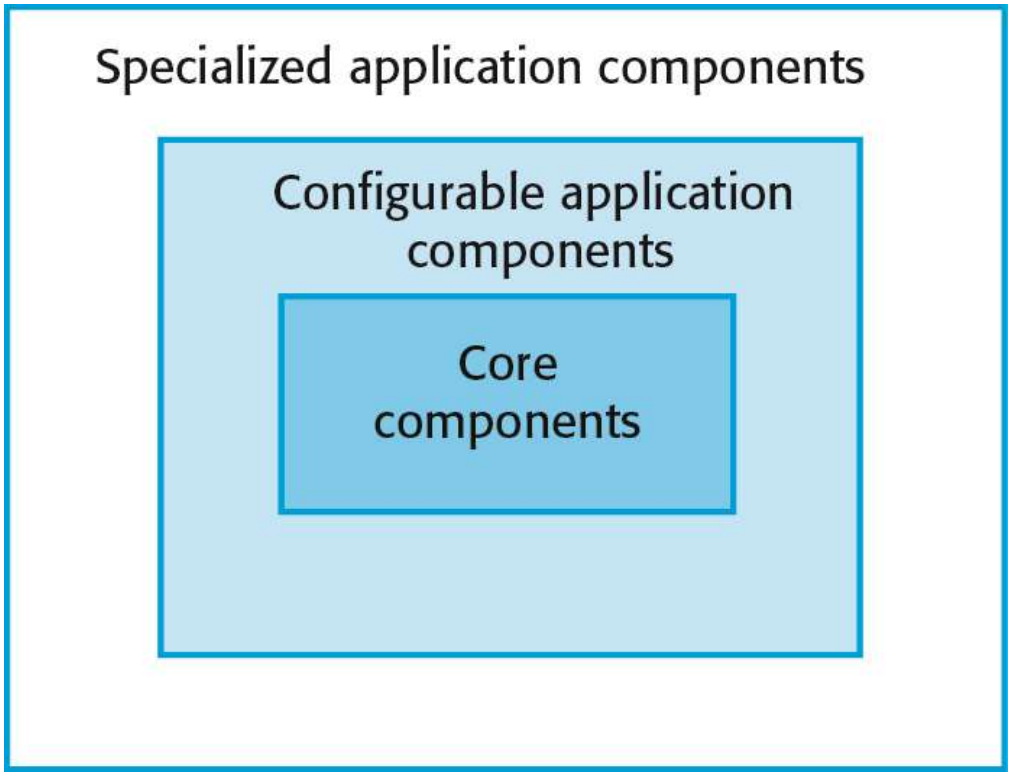
Legacy system wrapping



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

.

Product lines



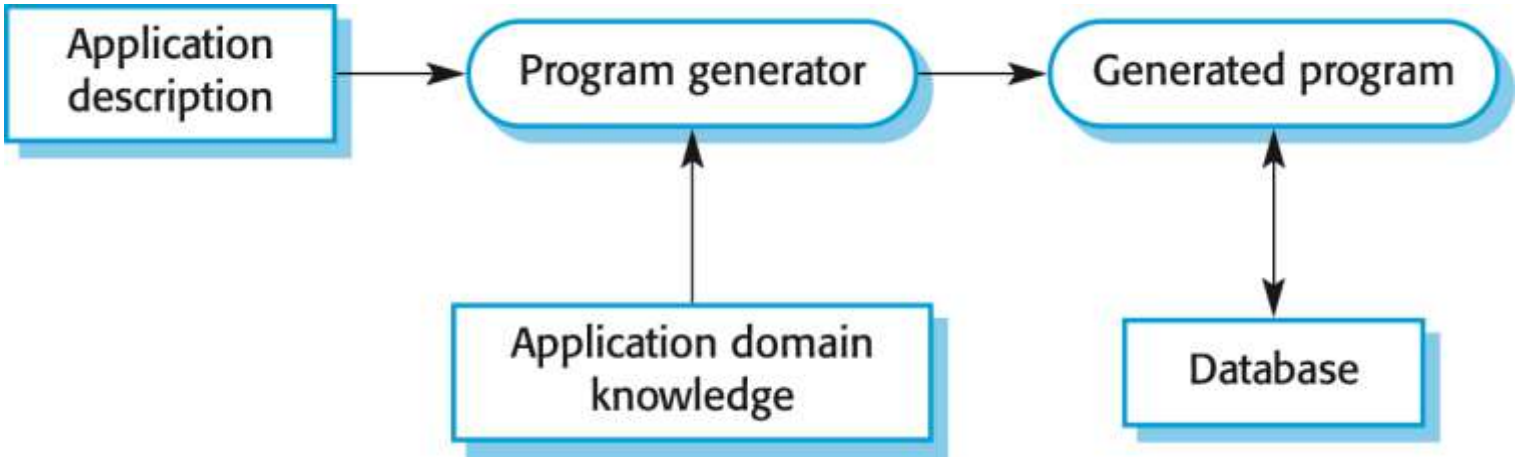
Component-based software engineering: Systems are developed by integrating components (collections of objects) that conform to component-model standards.

Program libraries:
Class and function libraries that implement commonly used abstractions are available for reuse.

Service-oriented development:
Systems are developed by linking shared services, which may be externally provided.

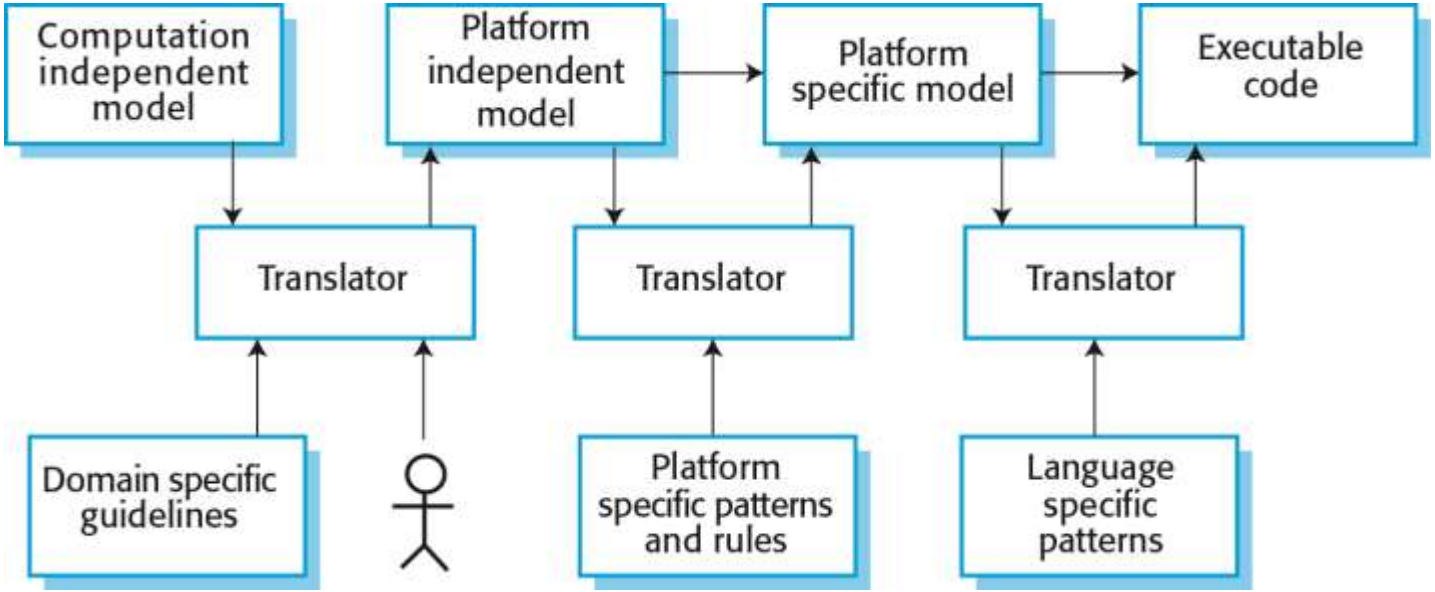
Program generators:
A generator system embeds knowledge of a type of application and is used to generate systems in that domain from a user-supplied system model.

Program generation



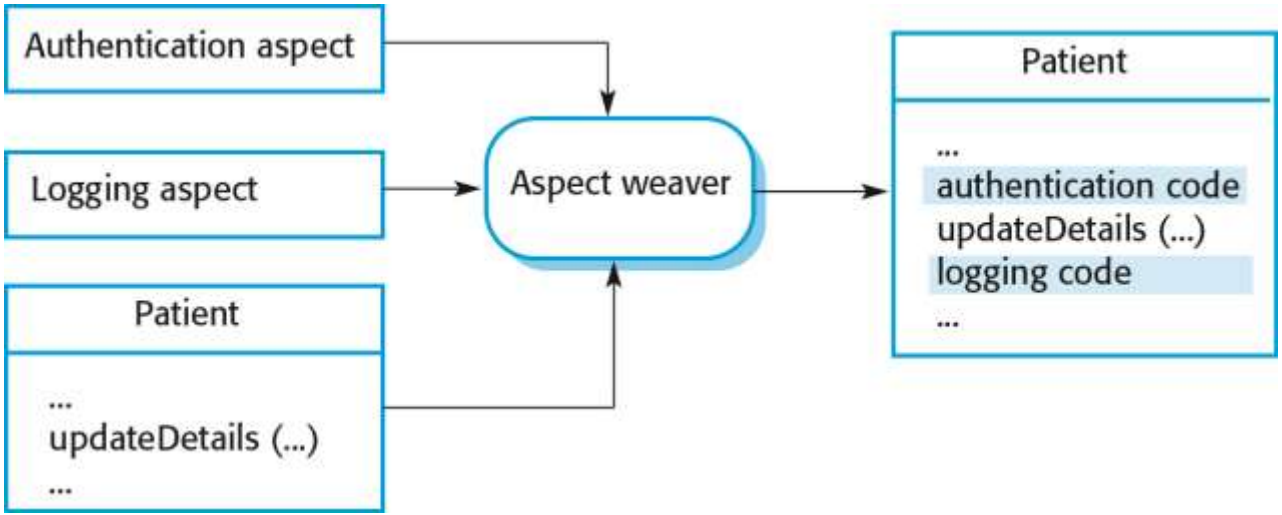
Model-driven engineering:
Software is represented as domain models and implementation independent models and code is generated from these models.

Model transformations



Aspect-oriented software development: Shared components are woven into an application at different places when the program is compiled.

Aspect weaving



Design patterns:
Generic abstractions that occur across applications are represented as design patterns showing abstract and concrete objects and interactions.

Architectural patterns:
Standard software architectures that support common types of application system are used as the basis of applications.

There is no ‘best approach’ to software reuse. The approach to be used depends on software available, skills and the organization itself.

Key factors include:

Development schedule, software lifetime, the development team, the criticality of the software, non-functional requirements, application domain, the software execution platform

Software reuse is a cost-effective approach to software development and there are a range of different ways that software can be reused.