

Chapter 4

Software Reuse

Don't reinvent the wheel, Do something smart!

By Yohannes S.

Contents

Introduction

Units of reuse, Benefits, problems of reuse

The reuse landscape

Approaches that support software reuse,
Reuse planning factors

Application Frameworks

Framework Definition, Web application
frameworks, Extending frameworks

Software product lines

Base systems for SPL, Application
Frameworks vs PLs

Application System Reuse

ERP

Configurable application systems

Introduction

In most engineering disciplines, systems are designed by composing existing components that have been used in other systems.

Software engineering has been more focused on original development

but it is now recognised that to achieve better software, more quickly and at lower cost, we need a design process that is based on systematic software reuse

There has been a major switch to reuse-based development over the past 15 years

Standards, such as **web service** standards, have made it

easier to develop general services and reuse

Introduction...

The software units that are reused may be

System reuse

Complete systems, which may include several application programs may be reused.

Application reuse

An application may be reused either by incorporating it without change into other or by developing application families.

Component reuse

Components of an application from sub-systems to single objects may be reused.

Object and function reuse

Small-scale software components that implement a single well-defined object or function may be reused.

'concept reuse' - rather than reuse a software

Introduction...

Benefits of software reuse

Benefit	Explanation
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.
Effective use of specialists	Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge.
Increased dependability	Reused software, which has been tried and tested in working systems, should be more dependable than new software. Its

Benefit	Explanation
Lower development costs	Development costs are proportional to the size of the software being developed. Reusing software means that fewer lines of code have to be written.
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.
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

Introduction...

Problems with reuse

Problem	Explanation
Creating, maintaining, & using a component library	Populating a reusable component library and ensuring the software developers can use this library can be expensive. Development processes have to be adapted to ensure that the library is used.
Finding, understanding, & adapting reusable components	Software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment. Engineers must be reasonably confident of finding a component in the library before they include a component search as part of their normal development process.
Increased maintenance	If the source code of a reused software system or component is not available then

Introduction...

Problem	Explanation
Lack of tool support	Some software tools do not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system. The software process assumed by these tools may not take reuse into account. This is particularly true for tools that support embedded systems engineering, less so for object-oriented development tools.
Not-invented-here syndrom	Some software engineers prefer to rewrite components because they believe they can improve on them. This is partly to do with trust and partly to do with the fact that writing original software is seen as more

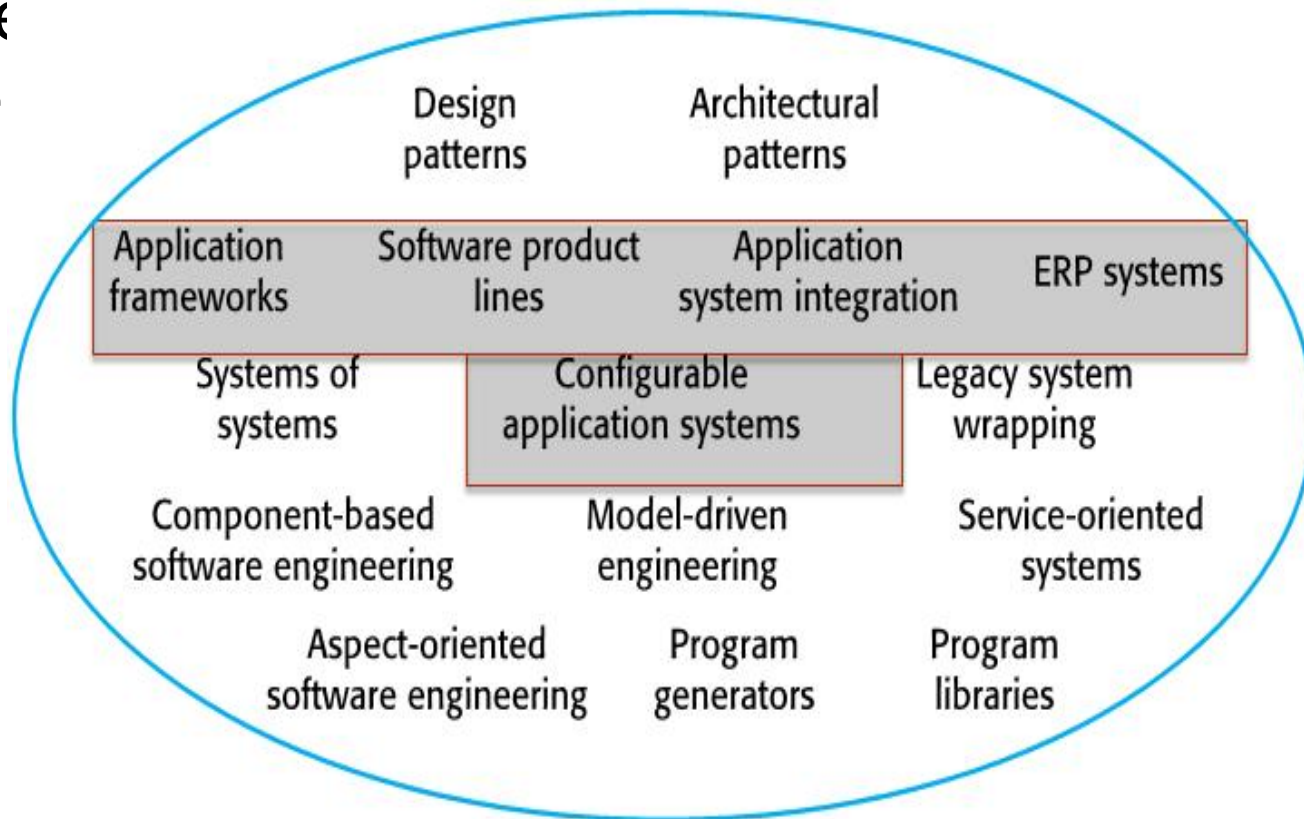
The reuse landscape

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

possible



software reuse

Approach	Description
Application frameworks	Collections of abstract and concrete classes are adapted and extended to create application systems.
Application system integration	Two or more application systems are integrated to provide extended functionality
Architectural patterns	Standard software architectures that support common types of application system are used as the basis of applications.
Aspect-oriented software development	Shared components are woven into an application at different places when the program is compiled.

software reuse...

Approach	Description
Configurable application systems	Domain-specific systems are designed so that they can be configured to the needs of specific system customers.
Design patterns	Generic abstractions that occur across applications are represented as design patterns showing abstract and concrete objects and interactions.
ERP systems	Large-scale systems that encapsulate generic business functionality and rules are configured for an organization.
Legacy system wrapping	Legacy systems are 'wrapped' by defining a set of interfaces and providing access to these legacy systems through these interfaces.
Model-	Software is represented as domain

software reuse...

Approach	Description
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 libraries	Class and function libraries that implement commonly used abstractions are available for reuse.
Service-oriented systems	Systems are developed by linking shared services, which may be externally provided.
Software product lines	An application type is generalized around a common architecture so that it can be adapted for different customers.
Systems of	Two or more distributed systems are

Reuse planning factors

Given this array of techniques for reuse, the key question is *“which is the most appropriate technique to use in a particular situation ?”*

Obviously, this depends on the requirements for the system being developed, the technology and reusable assets available, and

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.

Application Frameworks

Object-oriented development suggested that one of the key benefits of using an object-oriented approach was that objects could be reused in different systems.

However, experience has shown that objects are often **too small** and **are specialized** for a particular application.

It has now become clear that object-oriented reuse is best supported in an object-oriented development process through larger-grain abstractions called **frameworks**

So, what is framework?

“..an integrated set of software artefacts

Application Frameworks...

Frameworks are moderately large entities that can be reused.

They are somewhere between system and component reuse.

Frameworks are a sub-system design made up of a collection of abstract and concrete classes and the interfaces between them.

The sub-system is implemented by adding components to fill in parts of the design and by instantiating the abstract classes in the framework.

Frameworks provide support for generic features that are likely to be used in all applications of a similar type.

Frameworks support design reuse in that they

Application Frameworks...

Web application frameworks

Support the construction of dynamic websites as a front-end for web applications.

Examples: Joomla, Drupal, Wordpress, ...

WAFs are now available for all of the commonly used web programming languages e.g. Java, Python, Ruby, etc.

Interaction model is based on the Model-View-Controller composite pattern

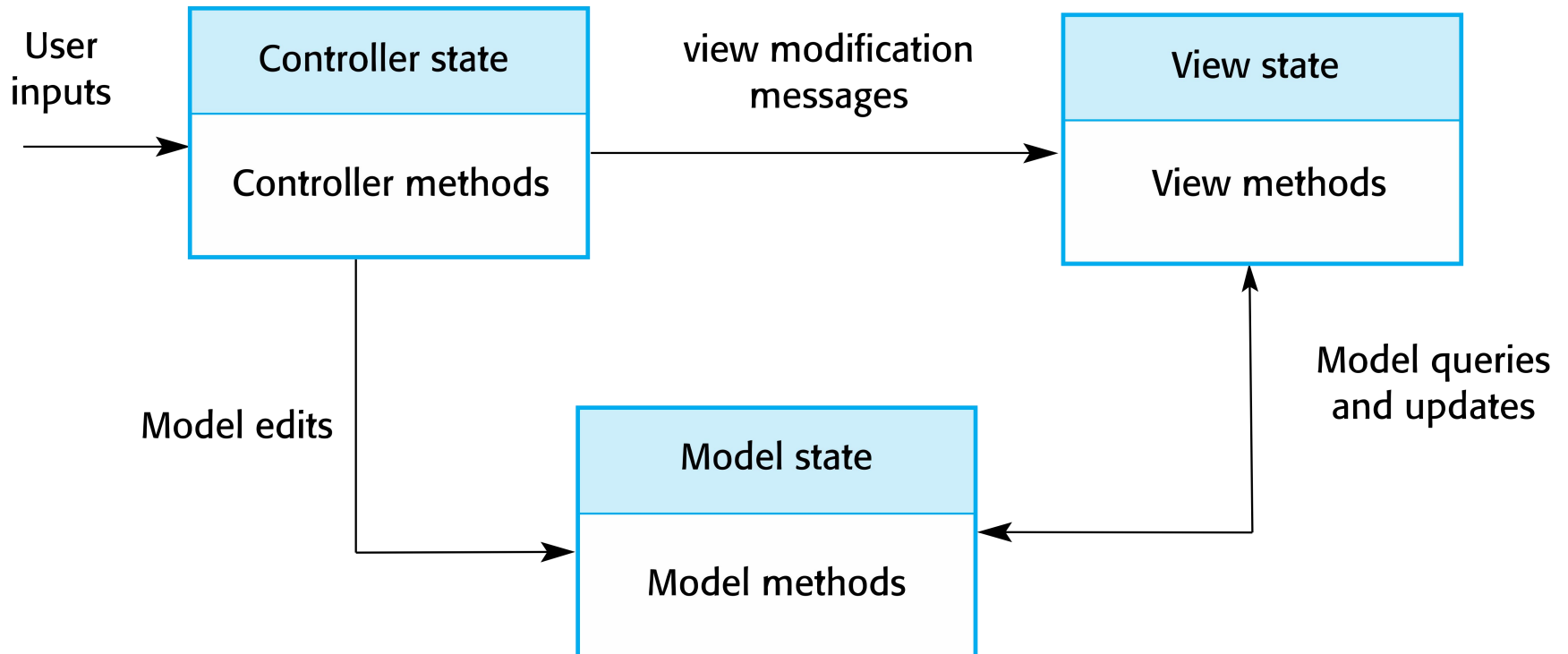
Model-view controller

System infrastructure framework for GUI design.

Allows for multiple presentations of an object and separate interactions with these presentations.

MVC framework involves the instantiation of a number of patterns

Application Frameworks...



The Model-View-Controller pattern

Application Frameworks...

WAF features

Security - WAFs may include classes to help implement user authentication (login) and access.

Dynamic web pages - Classes are provided to help you define web page templates and to populate these dynamically from the system database.

Database support - The framework may provide classes that provide an abstract interface to different databases.

Session management - Classes to create and manage sessions (a number of interactions with the system by a user) are usually part of a

WAF

Application Frameworks...

Extending frameworks

Frameworks are generic and are extended to create a more specific application or sub-system.

They provide a skeleton architecture for the system.

Extending the framework involves

Adding concrete classes that inherit operations from abstract classes in the framework

Adding methods that are called in response to events (=callbacks) that are recognised by the framework.

Problem with frameworks is their complexity

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.

A software product line is a set of applications with a common architecture and shared components, with each application specialized to reflect different requirements.

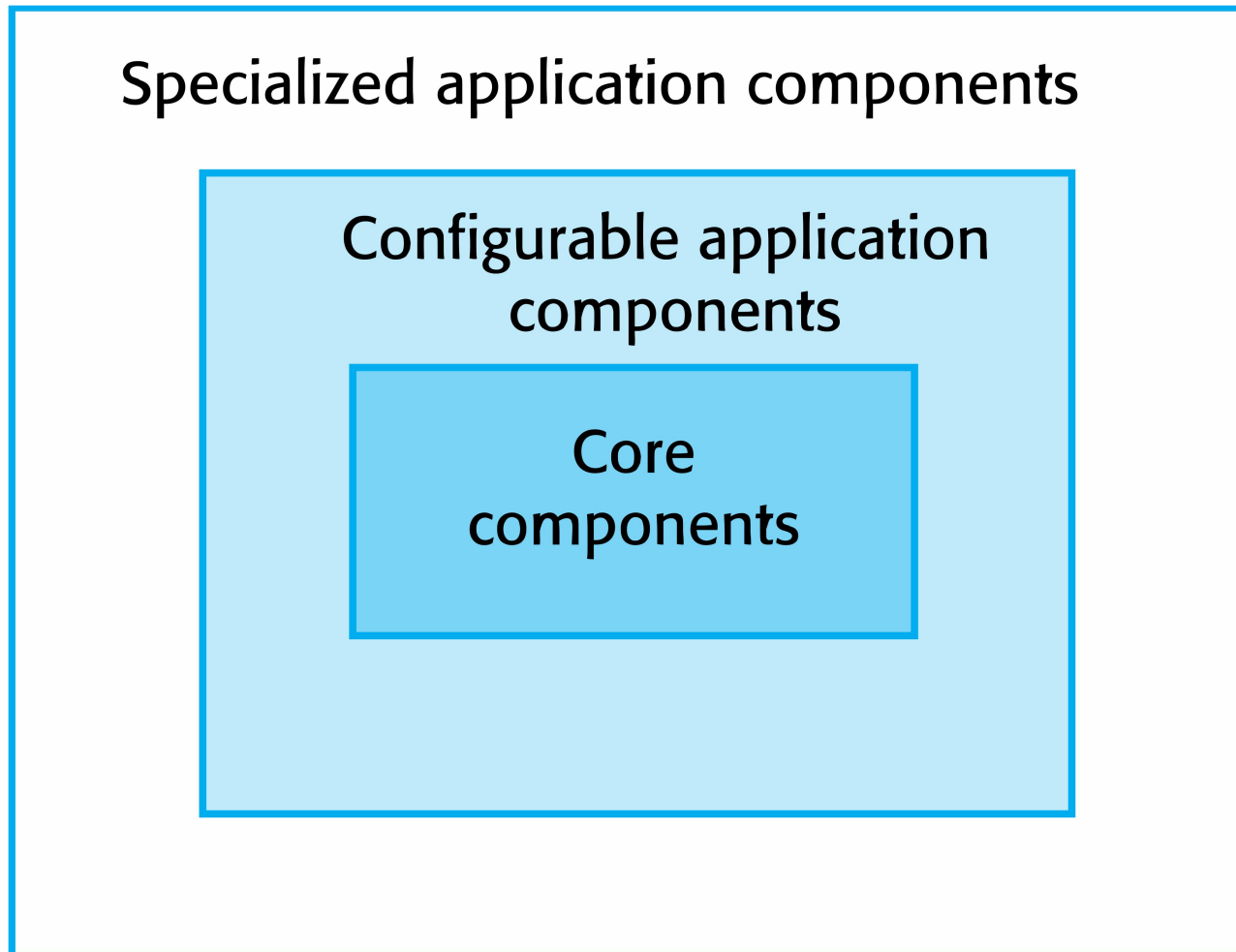
Adaptation may involve:

- Component and system configuration;
- Adding new components to the system;
- Selecting from a library of existing components;

Modifying components to meet new

Software product lines...

Base systems for a software product line



Software product lines...

Base systems for a software product line...

Core components

provide infrastructure support

These are not usually modified when developing a new instance of the product line.

Configurable components

that may be modified and configured to specialize them to a new application.

Sometimes, it is possible to reconfigure these components without changing their code by using a built-in component configuration language.

Specialized, domain-specific components

some or all of which may be replaced when a new instance of a product line is created.

Software product lines...

Application frameworks

Application frameworks rely on object-oriented features such as polymorphism to implement extensions.

Application frameworks focus on providing technical rather than domain-specific support.

Software product

Product lines

Product lines need not be object-oriented (e.g. embedded software for a mobile phone)

Product lines embed domain and platform information.

Product lines often control applications for equipment.

Software product lines are made up of

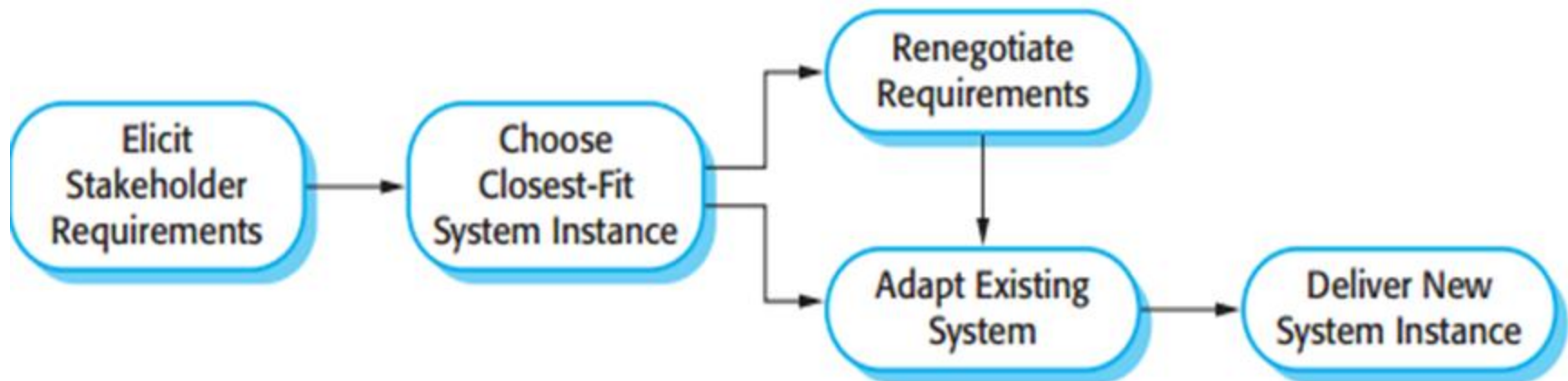
lines are made up of

Software product lines...

Product instance development

To create a specific version of a system, you may have to modify individual components.

For example for vehicle dispatcher system the police have a large number of vehicles but a small number of vehicle types, whereas the fire service has many types of specialized vehicles. Therefore, you may have to define a different vehicle database structure when implementing a system for these different services.



Software product lines...

Product instance development....

Elicit stakeholder requirements

Use existing family member as a prototype

Choose closest-fit family member

Find the family member that best meets the requirements

Re-negotiate requirements

Adapt requirements as necessary to capabilities of the software

Adapt existing system

Develop new modules and make changes for family member

Deliver new family member

Document key features for further member development

Application system reuse

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

Application systems have generic features and so can be used/reused in different environments.

Application system products are adapted by using built-in configuration mechanisms that allow the functionality of the system to be tailored to specific customer needs.

For example, in a hospital patient record system, separate input forms and output reports might be defined for different types

Application system reuse...

Benefits of application system reuse

As with other types of reuse, more rapid deployment of a reliable system may be possible.

It is possible to see what functionality is provided by the applications and so it is easier to judge whether or not they are likely to be suitable.

Some development risks are avoided by using existing software. However, this approach has its own risks.

Businesses can focus on their core activity without having to devote a lot of resources to IT systems development.

As operating platforms evolve, technology

Application system reuse...

Problems of application system reuse

Requirements usually have to be adapted to reflect the functionality and mode of operation of the COTS product.

The COTS product may be based on assumptions that are practically impossible to change.

Choosing the right COTS system for an enterprise can be a difficult process, especially as many COTS products are not well documented.

There may be a lack of local expertise to support systems development.

The COTS product vendor controls system

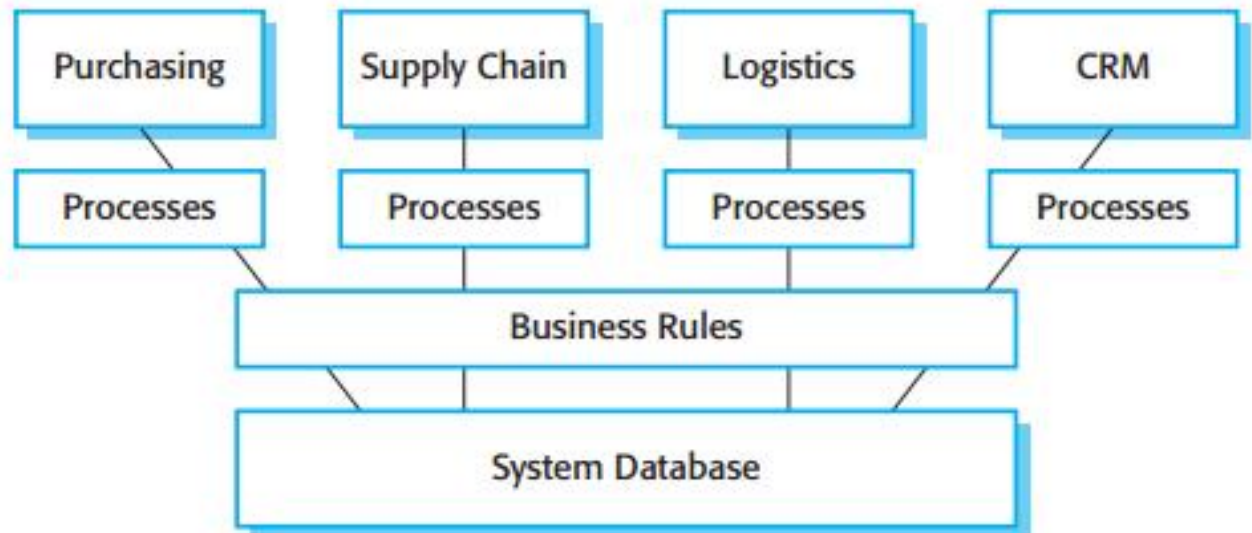
ERP systems

An Enterprise Resource Planning (ERP) system is a generic system that supports common business processes such as ordering and invoicing, manufacturing, etc.

These are very widely used in large companies - they represent probably the most common form of software reuse.

The generic core is adapted by including modules and business processes.

The architecture of an ERP system



ERP systems...

ERP architecture

A number of modules to support different business functions.

A defined set of business processes, associated with each module, which relate to activities in that module.

A common database that maintains information about all related business functions.

A set of business rules that apply to all data in the database.

ERP systems...

ERP Configuration

Selecting the required functionality from the system.

Establishing a data model that defines how the organization's data will be structured in the system database.

Defining business rules that apply to that data.

Defining the expected interactions with external systems.

Designing the input forms and the output reports generated by the system.

Designing new business processes that conform to the underlying process model supported by the system.

systems

Configurable application systems are generic application systems that may be designed to support a particular business type, business activity or, sometimes, a complete business enterprise.

For example, an application system may be produced for dentists that handles appointments, dental records, patient recall, etc.

Domain-specific systems, such as systems to support a business function (e.g. document management) provide functionality that is likely to be required by a range of potential users

systems...

Configurable Application Systems Vs Application System Integration

C o n f i g u r a b l e a p p l i c a t i o n s y s t e m s

Single product that provides the functionality required by a customer

Based around a generic solution and standardized processes

Development focus is on system configuration

System vendor is responsible for

maintenance

System vendor provides

A p p l i c a t i o n s y s t e m i n t e g r a t i o n

Several heterogeneous system products are integrated to provide customized functionality

Flexible solutions may be developed for customer processes

Development focus is on system integration

System owner is responsible for

maintenance

System owner provides