Software Reuse and Component-Based SE

ITSE422

Lecture #2: Software product lines & COTS product reuse

Main References

- Ian Sommerville, *Software Engineering*, 8th edition, chapter18, 19 (Software Reuse & *Components and component models*)
- Ivica Crnkovic, Magnus Larsson. *Building reliable component based software systems*, Artech House, 2002.
- Roger S. Pressman, Software Engineering: A Practitioner's Approach, Eighth Edition, McGraw-Hill Higher Education, 2015

Topics covered

- Software product lines
- COTS product reuse

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 requirements.

Application frameworks and product lines

- Application frameworks rely on object-oriented features such as polymorphism to implement extensions. Product lines need not be object-oriented (e.g. embedded software for a mobile phone)
- Application frameworks focus on providing technical rather than domain-specific support. Product lines embed domain and platform information.
- Product lines often control applications for equipment.
- Software product lines are made up of a family of applications, usually owned by the same organization.

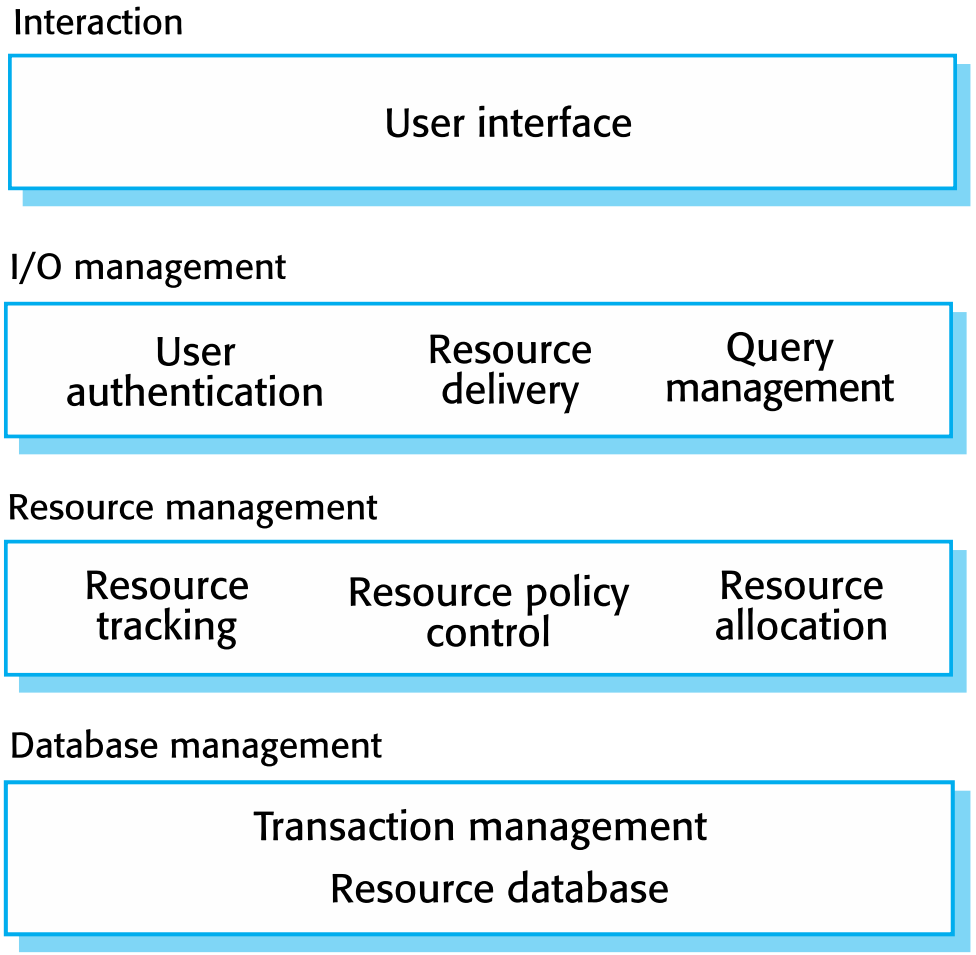
Product line specialisation

- Platform specialization
  - Different versions of the application are developed for different platforms. (ex: Windows, Solaris and Linux platforms).
- Environment specialization
  - Different versions of the application are created to handle different operating environments e.g. different types of communication equipment. (ex: emergency services).
- Functional specialization
  - Different versions of the application are created for customers with different requirements. (ex: library automation system).
- Process specialization
  - Different versions of the application are created to support different business processes. (ex: an ordering system may be adapted to cope with a centralised ordering process in one company and a distributed process in another.).

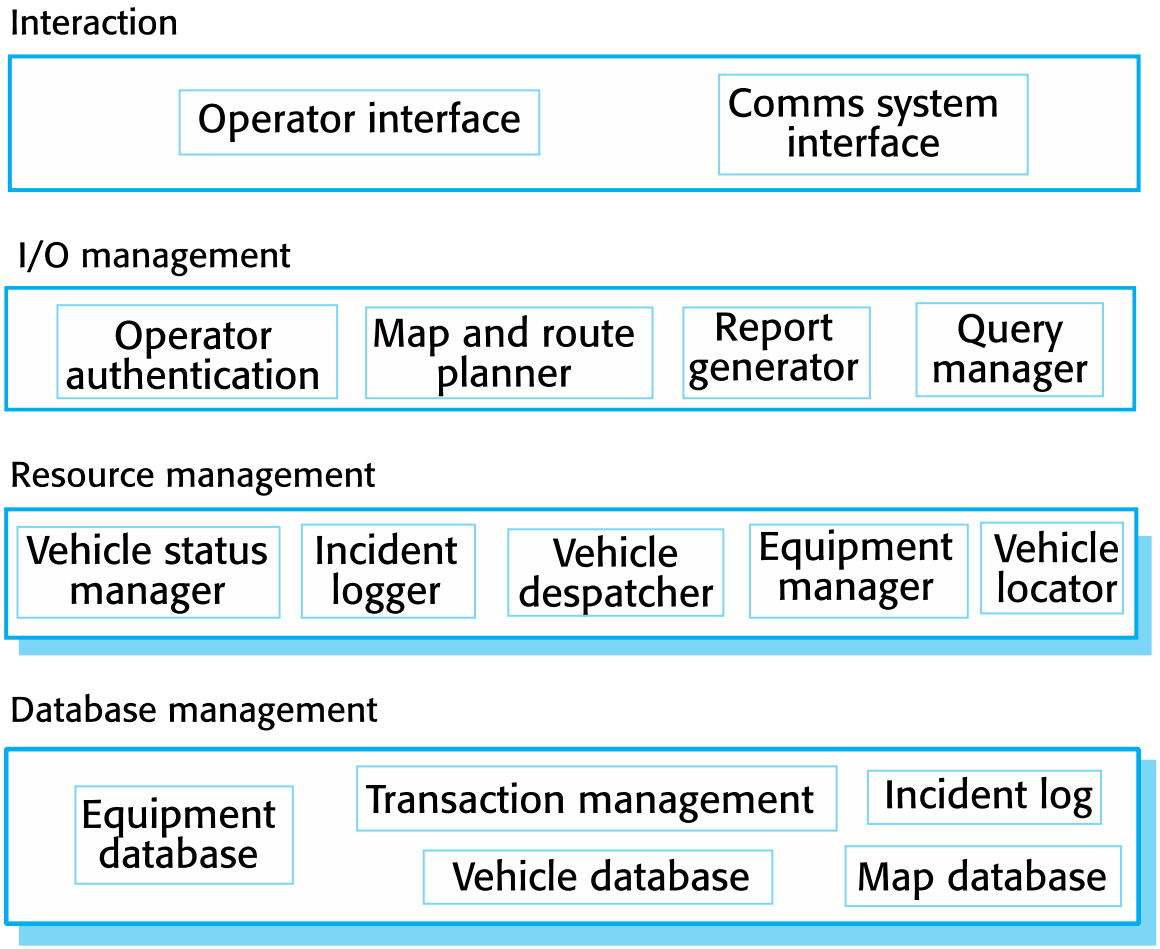
Product line architectures

- Architectures must be structured in such a way to separate different sub-systems and to allow them to be modified.
- The architecture should also separate entities and their descriptions and the higher levels in the system access entities through descriptions rather than directly.

The architecture of a resource allocation system



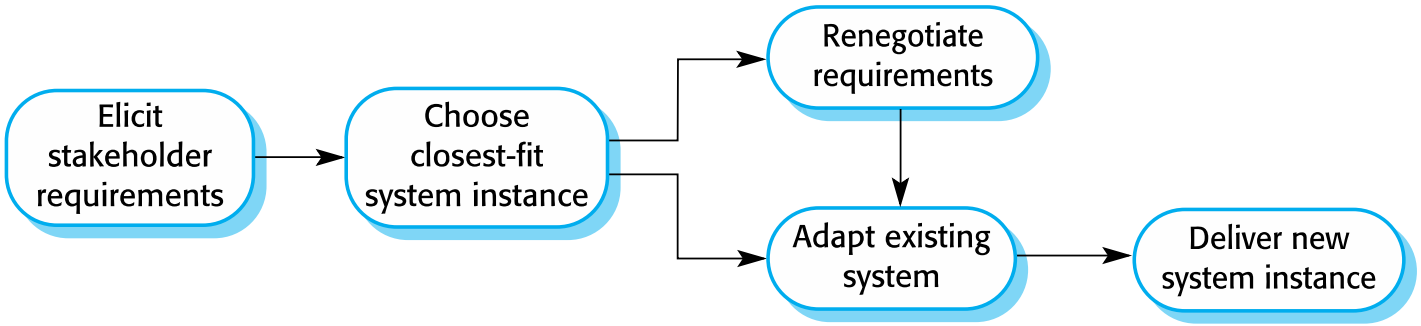
The product line architecture of a vehicle dispatcher



Vehicle dispatching

- A specialised resource management system where the aim is to allocate resources (vehicles) to handle incidents.
- Adaptations include:
  - At the UI level, there are components for operator display and communications;
  - At the I/O management level, there are components that handle authentication, reporting and route planning;
  - At the resource management level, there are components for vehicle location and despatch, managing vehicle status and incident logging;
  - The database includes equipment, vehicle and map databases.

Product instance development



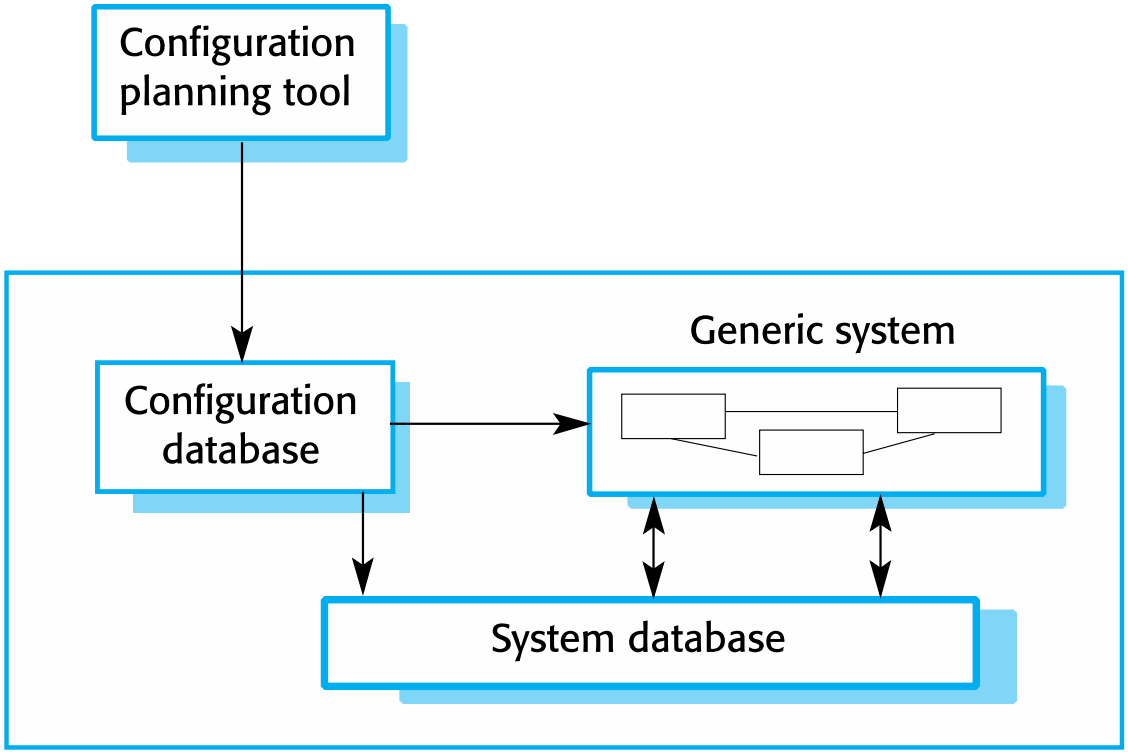
Product instance development

- 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

Product line configuration

- Design time configuration
  - The product line is adapted and changed according to the requirements of particular customers.
- Deployment time configuration
  - The product line is configured by embedding knowledge of the customer's requirements and business processes. The software source code itself is not changed.

Deployment-time configuration



Levels of deployment time configuration

- Component selection, where you select the modules in a system that provide the required functionality.
- Workflow and rule definition, where you define workflows (how information is processed, stage-by-stage) and validation rules that should apply to information entered by users or generated by the system.
- Parameter definition, where you specify the values of specific system parameters that reflect the instance of the application that you are creating

COTS product reuse

- A commercial-off-the-shelf (COTS) product is a software system that can be adapted for different customers without changing the source code of the system.
- COTS systems have generic features and so can be used/reused in different environments.
- COTS 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 of patient.

Benefits of COTS 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, as I discuss below.
- Businesses can focus on their core activity without having to devote a lot of resources to IT systems development.
- As operating platforms evolve, technology updates may be simplified as these are the responsibility of the COTS product vendor rather than the customer.

Problems of COTS 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 support and evolution.

COTS-solution and COTS-integrated systems

COTS-solution systems	COTS-integrated systems
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

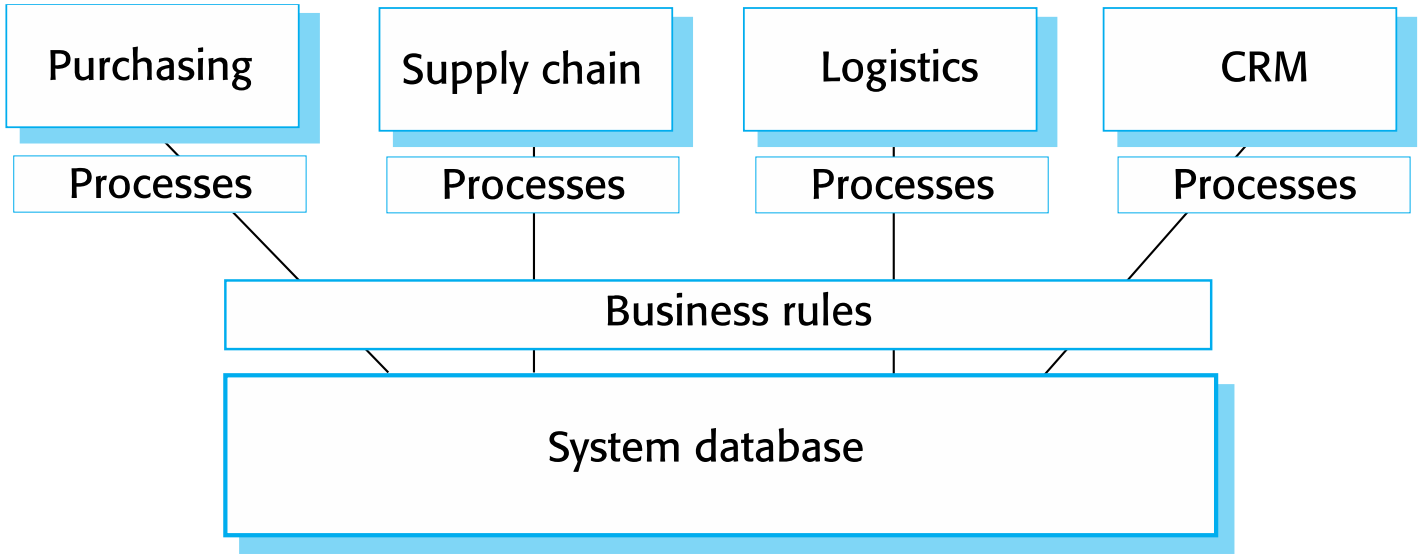
COTS solution systems

- COTS-solution 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, a COTS-solution system may be produced for dentists that handles appointments, dental records, patient recall, etc.
- Domain-specific COTS-solution 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.

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 by incorporating knowledge of business processes and rules.

The architecture of an ERP system



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 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.
- Setting parameters that define how the system is deployed on its underlying platform.

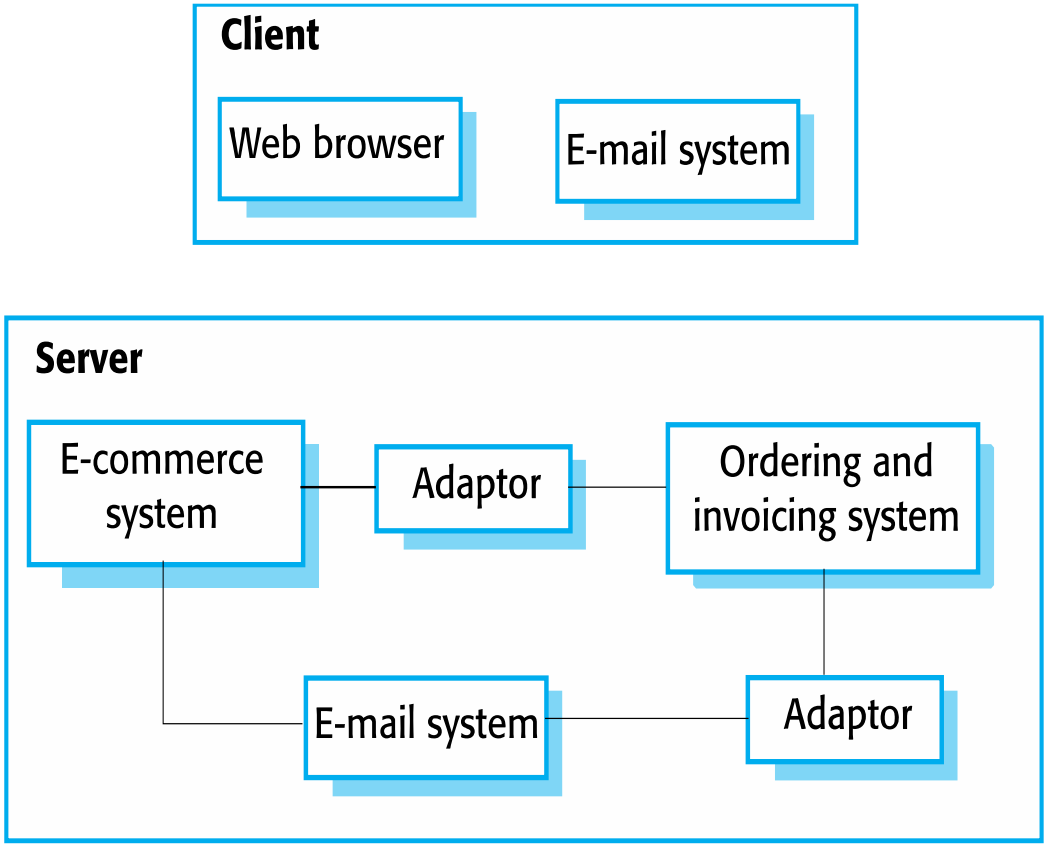
COTS integrated systems

- COTS-integrated systems are applications that include two or more COTS products and/or legacy application systems.
- You may use this approach when there is no single COTS system that meets all of your needs or when you wish to integrate a new COTS product with systems that you already use.

Design choices

- Which COTS products offer the most appropriate functionality?
  - Typically, there will be several COTS products available, which can be combined in different ways.
- How will data be exchanged?
  - Different products normally use unique data structures and formats. You have to write adaptors that convert from one representation to another.
- What features of a product will actually be used?
  - COTS products may include more functionality than you need and functionality may be duplicated across different products.

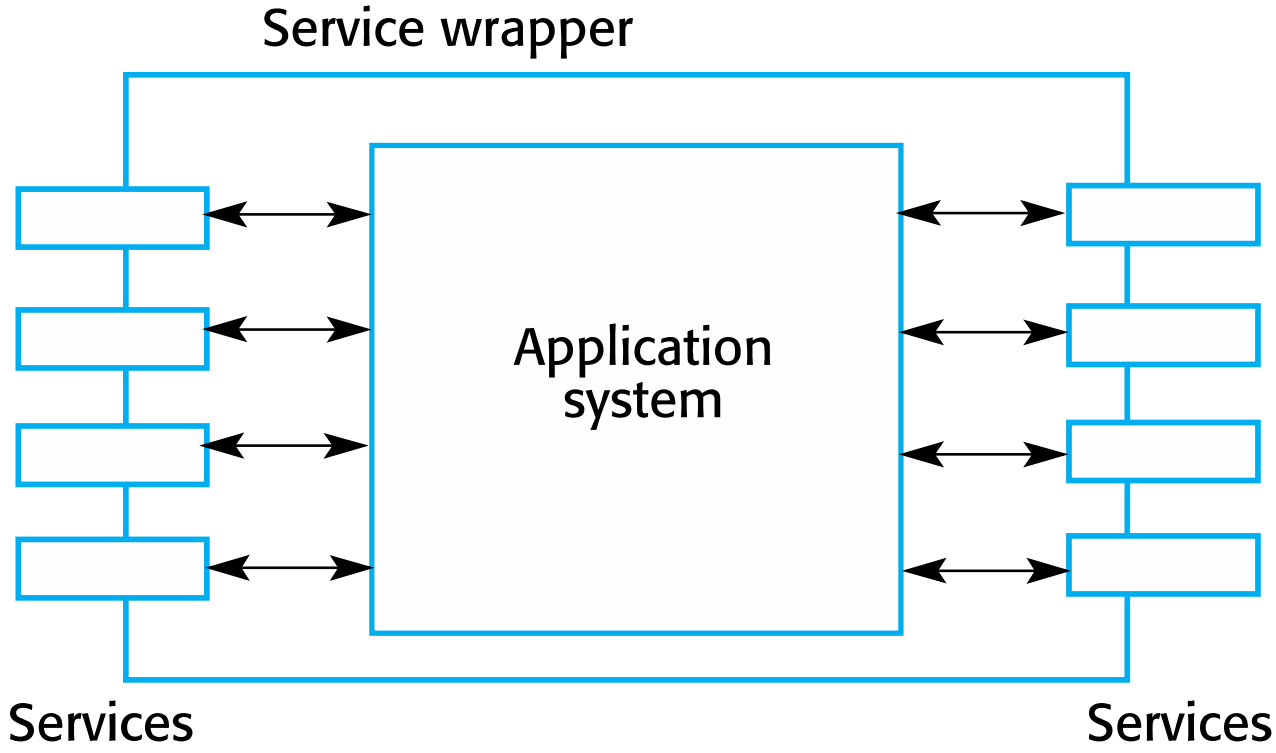
A COTS-integrated procurement system



Service-oriented COTS interfaces

- COTS integration can be simplified if a service-oriented approach is used.
- A service-oriented approach means allowing access to the application system's functionality through a standard service interface, with a service for each discrete unit of functionality.
- Some applications may offer a service interface but, sometimes, this service interface has to be implemented by the system integrator. You have to program a wrapper that hides the application and provides externally visible services.

Application wrapping



COTS system integration problems

- Lack of control over functionality and performance
  - COTS systems may be less effective than they appear
- Problems with COTS system inter-operability
  - Different COTS systems may make different assumptions that means integration is difficult
- No control over system evolution
  - COTS vendors not system users control evolution
- Support from COTS vendors
  - COTS vendors may not offer support over the lifetime of the product

Key points

- Software product lines are related applications that are developed from a common base. This generic system is adapted to meet specific requirements for functionality, target platform or operational configuration.
- COTS product reuse is concerned with the reuse of large-scale, off-the-shelf systems. These provide a lot of functionality and their reuse can radically reduce costs and development time. Systems may be developed by configuring a single, generic COTS product or by integrating two or more COTS products.
- Enterprise Resource Planning systems are examples of large-scale COTS reuse. You create an instance of an ERP system by configuring a generic system with information about the customer's business processes and rules.
- Potential problems with COTS-based reuse include lack of control over functionality and performance, lack of control over system evolution, the need for support from external vendors and difficulties in ensuring that systems can inter-operate.

- Questions?