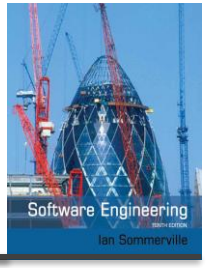


Chapter 15 – Software Reuse

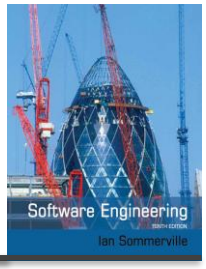
Troy University Resources

Topics covered



- ✧ The reuse landscape
- ✧ Application frameworks
- ✧ Software product lines
- ✧ Application system reuse

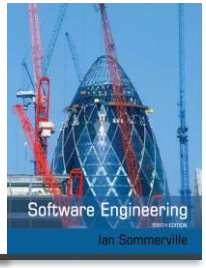
Software reuse



- 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 10 years.

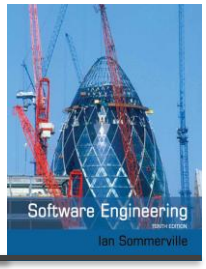
Troy University Resources

Reuse-based software engineering



- 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.

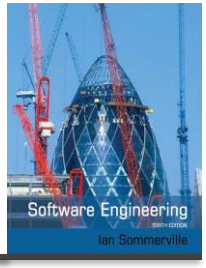
Benefits of software reuse



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 design and implementation faults should have been found and fixed.

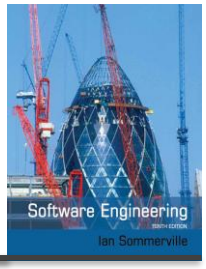
Troy University Resources

Benefits of software reuse



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 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.

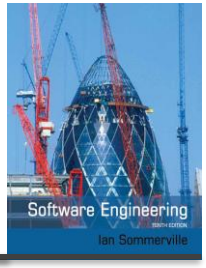
Problems with reuse



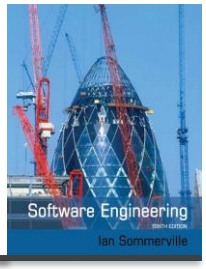
Problem	Explanation
Creating, maintaining, and using a component library	Populating a reusable component library and ensuring the software developers can use this library can be expensive. Development processes have to be adapted to ensure that the library is used.
Finding, understanding, and adapting 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 costs	If the source code of a reused software system or component is not available then maintenance costs may be higher because the reused elements of the system may become increasingly incompatible with system changes.

Troy University Resources

Problems with reuse



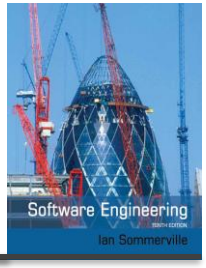
Problem	Explanation
Lack of tool support	Some software tools do not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system. The software process assumed by these tools may not take reuse into account. This is particularly true for tools that support embedded systems engineering, less so for object-oriented development tools.
Not-invented-here syndrome	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 challenging than reusing other people's software.



The reuse landscape

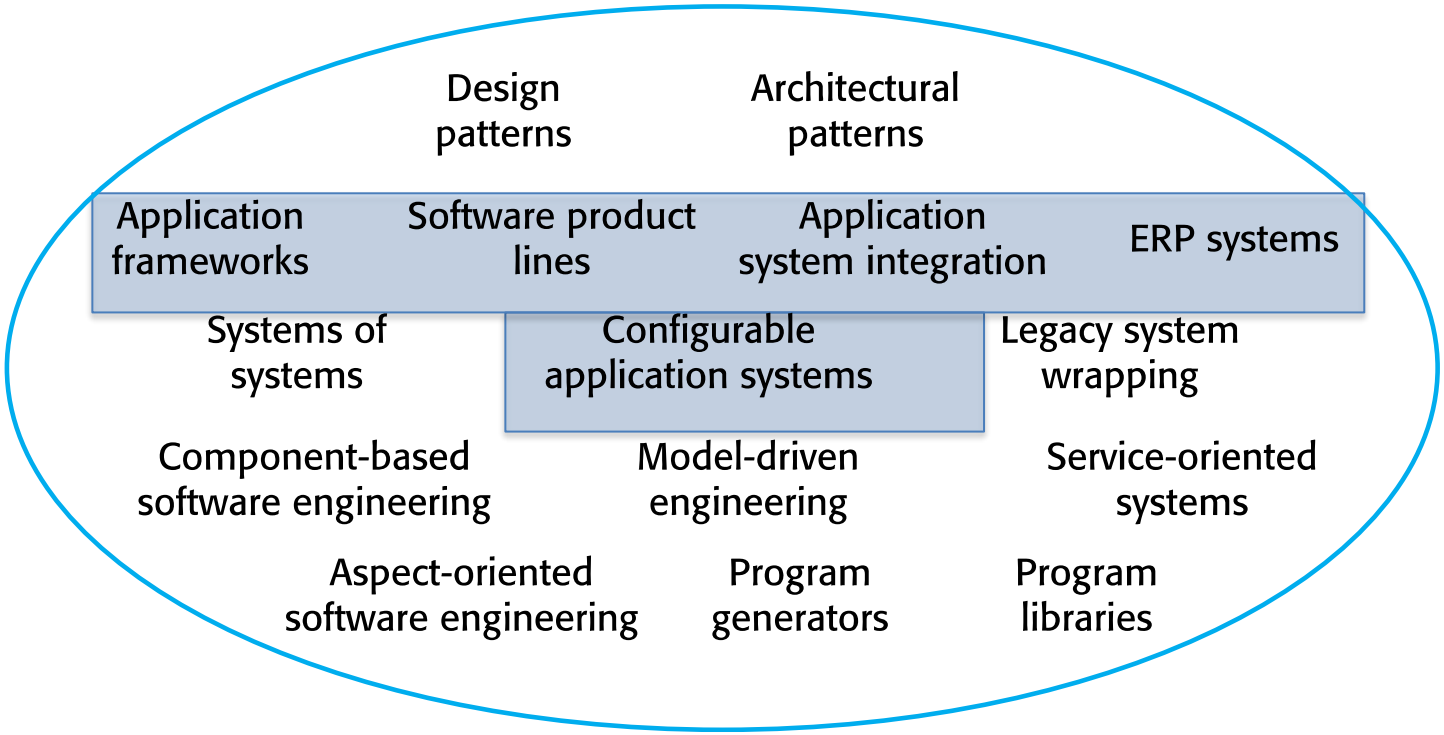
Troy University Resources

The reuse landscape



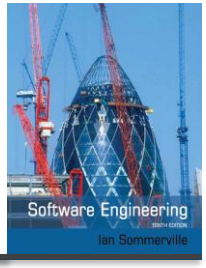
- Although reuse is often simply thought of as the reuse of system components, there are many different approaches to reuse that may be used.
- Reuse is possible at a range of levels from simple functions to complete application systems.
- The reuse landscape covers the range of possible reuse techniques.

The reuse landscape



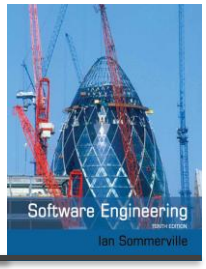
Troy University Resources

Approaches that support 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. Described in Chapters 6, 11 and 17.
Aspect-oriented software development	Shared components are woven into an application at different places when the program is compiled. Described in web chapter 31.
Component-based software engineering	Systems are developed by integrating components (collections of objects) that conform to component-model standards. Described in Chapter 16.

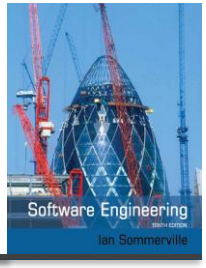
Approaches that support software reuse



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. Described in Chapter 7.
ERP systems	Large-scale systems that encapsulate generic business functionality and rules are configured for an organization.
Legacy system wrapping	Legacy systems (Chapter 9) are 'wrapped' by defining a set of interfaces and providing access to these legacy systems through these interfaces.
Model-driven engineering	Software is represented as domain models and implementation independent models and code is generated from these models. Described in Chapter 5.

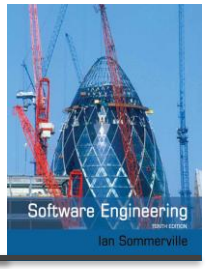
Troy University Resources

Approaches that support software reuse



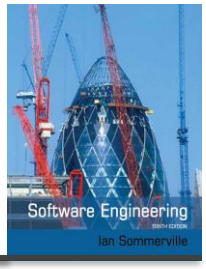
Approach	Description
Program generators	A generator system embeds knowledge of a type of application and is used to generate systems in that domain from a user-supplied system model.
Program libraries	Class and function libraries that implement commonly used abstractions are available for reuse.
Service-oriented systems	Systems are developed by linking shared services, which may be externally provided. Described in Chapter 18.
Software product lines	An application type is generalized around a common architecture so that it can be adapted for different customers.
Systems of systems	Two or more distributed systems are integrated to create a new system. Described in Chapter 20.

Reuse planning factors



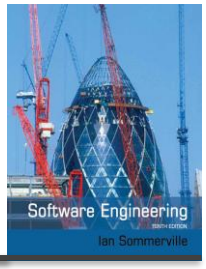
- The development schedule for the software.
- The expected software lifetime.
- The background, skills and experience of the development team.
- The criticality of the software and its non-functional requirements.
- The application domain.
- The execution platform for the software.

Troy University Resources



Application frameworks

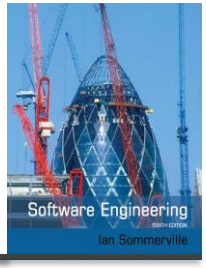
Framework definition



- *"..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."*

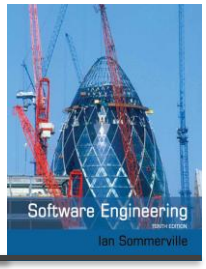
Troy University Resources

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.

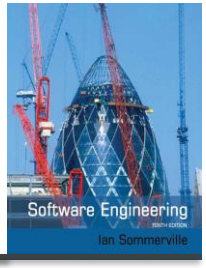
Web application frameworks



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

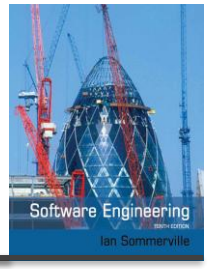
Troy University Resources

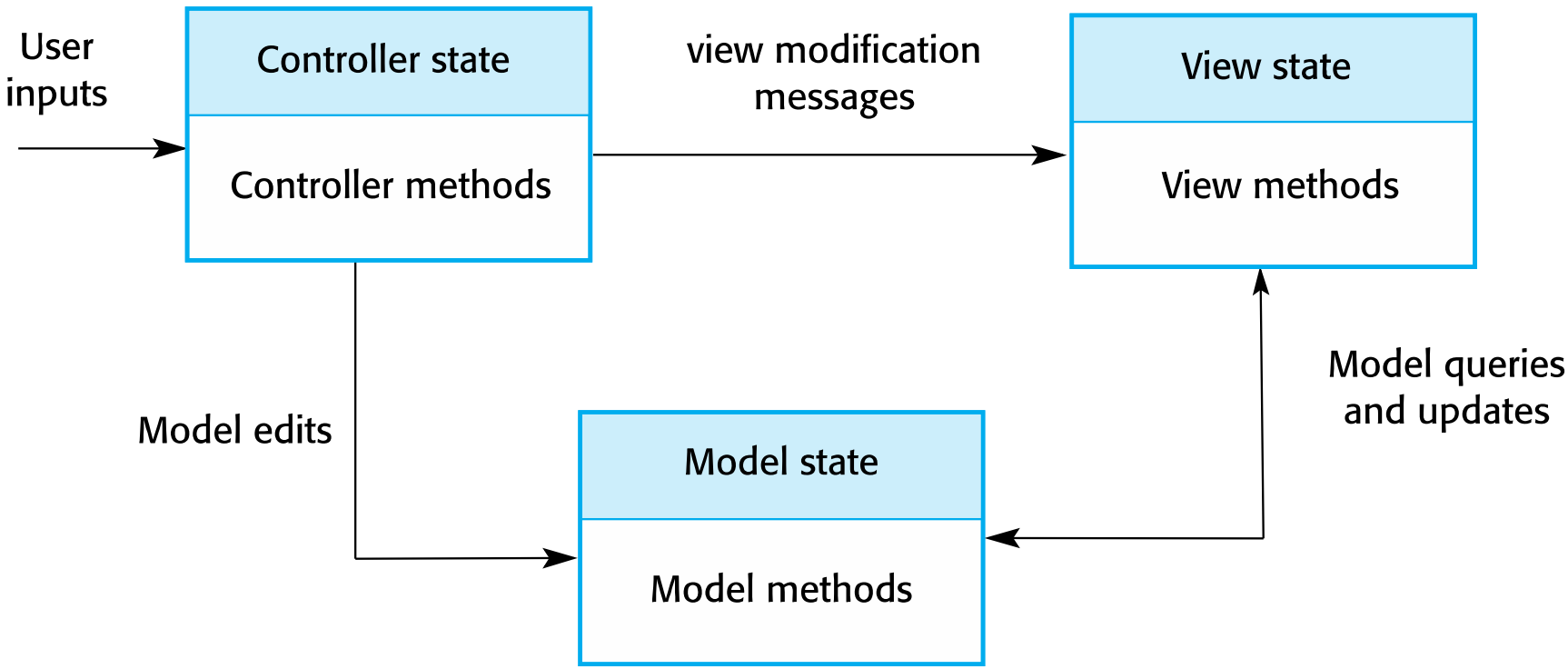
Model-view controller



- 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 (as discussed in Chapter 7).

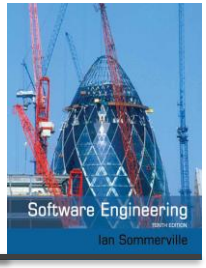
The Model-View-Controller pattern





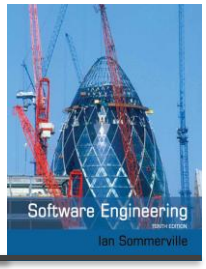
Troy University Resources

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.
- *User interaction*
  - Most web frameworks now provide AJAX support (Holdener, 2008), which allows more interactive web pages to be created.

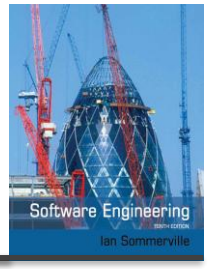
Extending frameworks

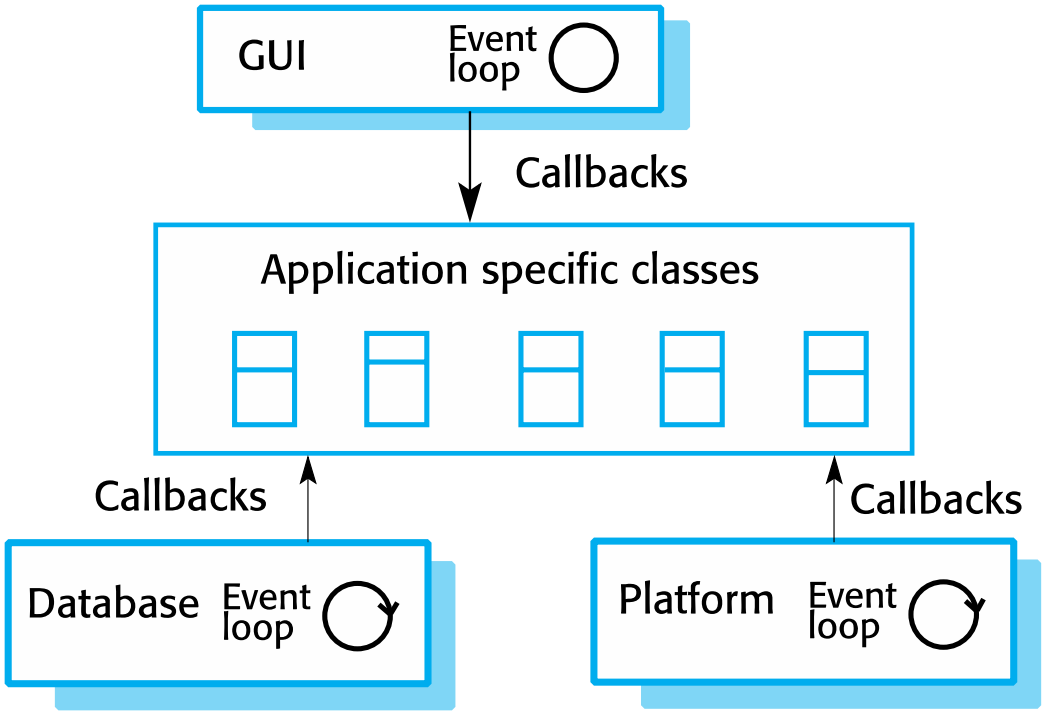


- Frameworks are generic and are extended to create a more specific application or sub-system. They provide a skeleton architecture for the system.
- Extending the framework involves
  - Adding concrete classes that inherit operations from abstract classes in the framework;
  - Adding methods that are called in response to events that are recognised by the framework.
- Problem with frameworks is their complexity which means that it takes a long time to use them effectively.

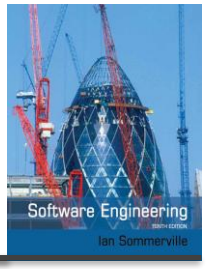
Troy University Resources

Inversion of control in frameworks



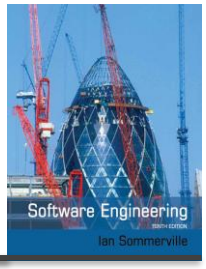


Framework classes



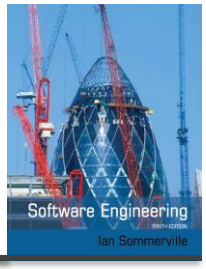
- System infrastructure frameworks
  - Support the development of system infrastructures such as communications, user interfaces and compilers.
- Middleware integration frameworks
  - Standards and classes that support component communication and information exchange.
- Enterprise application frameworks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

Troy University Resources



Software product lines

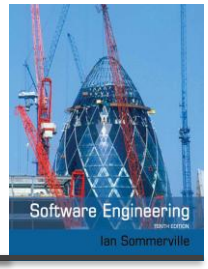
Software product lines

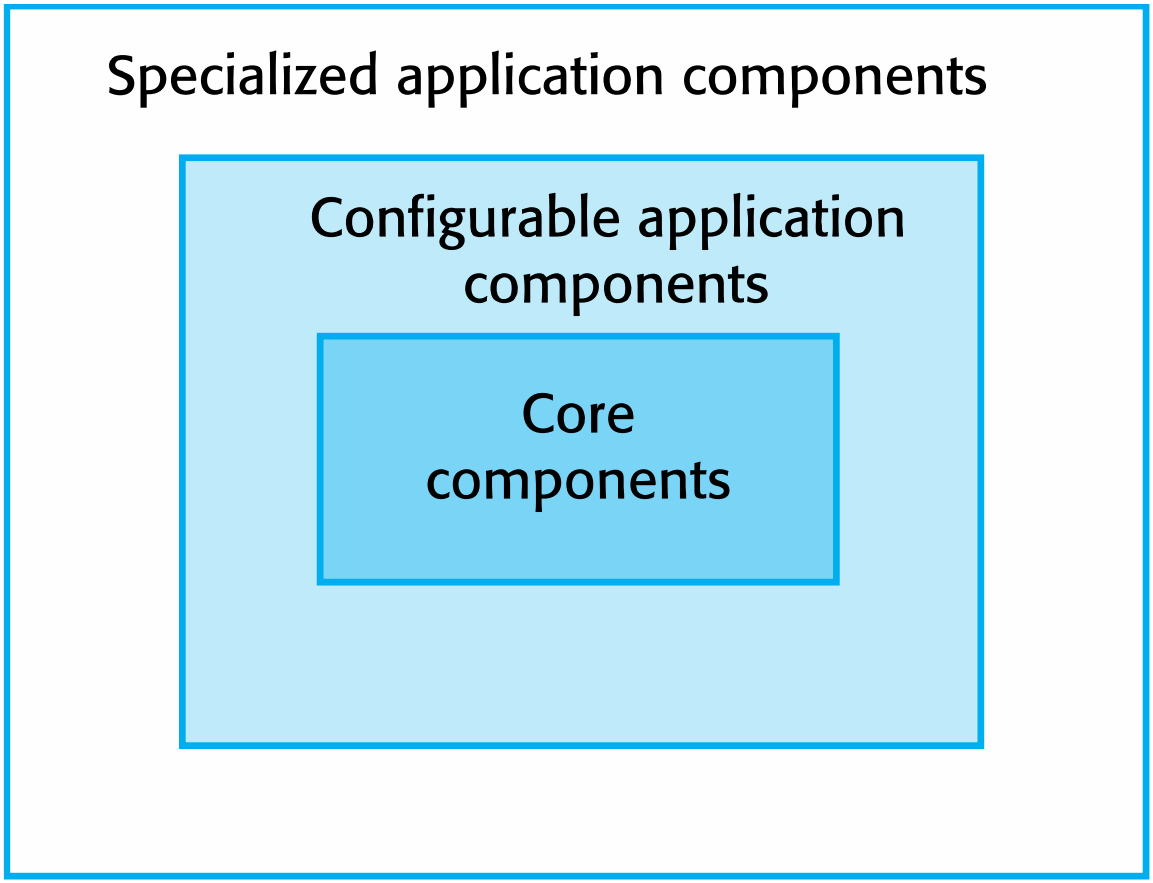


- Software product lines or application families are applications with generic functionality that can be adapted and configured for use in a specific context.
- A software product line is a set of applications with a common architecture and shared components, with each application specialized to reflect 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.

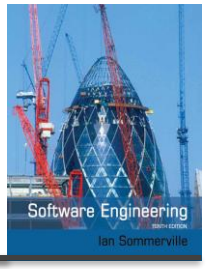
Troy University Resources

Base systems for a software product line





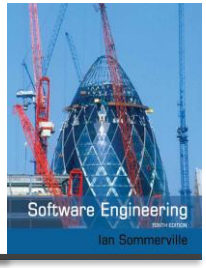
Base applications



- Core components that 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.

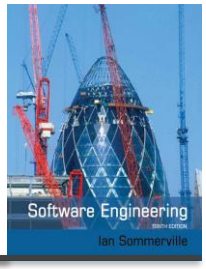
Troy University Resources

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.

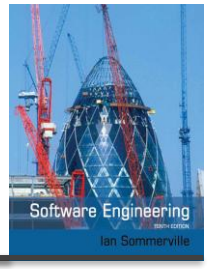
Product line architectures

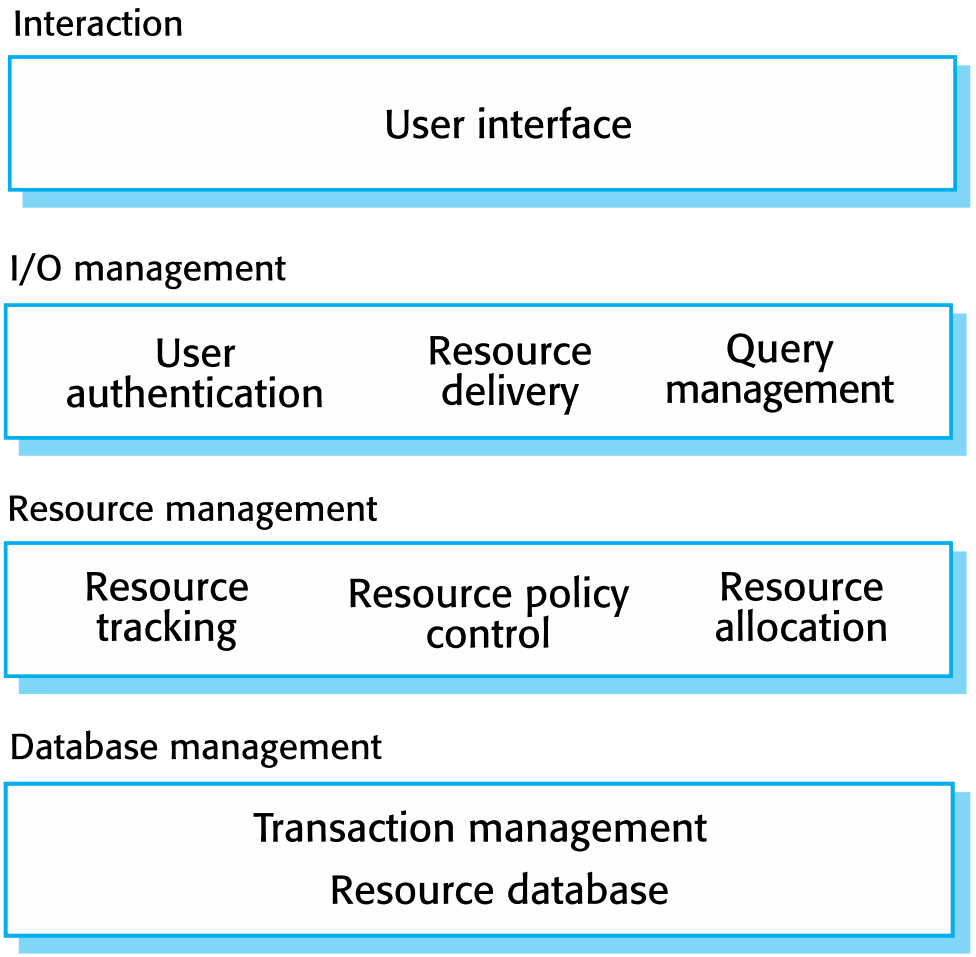


- Architectures must be structured in such a way to separate different sub-systems and to allow them to be modified.
- The architecture should also separate entities and their descriptions and the higher levels in the system access entities through descriptions rather than directly.

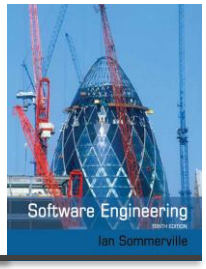
Troy University Resources

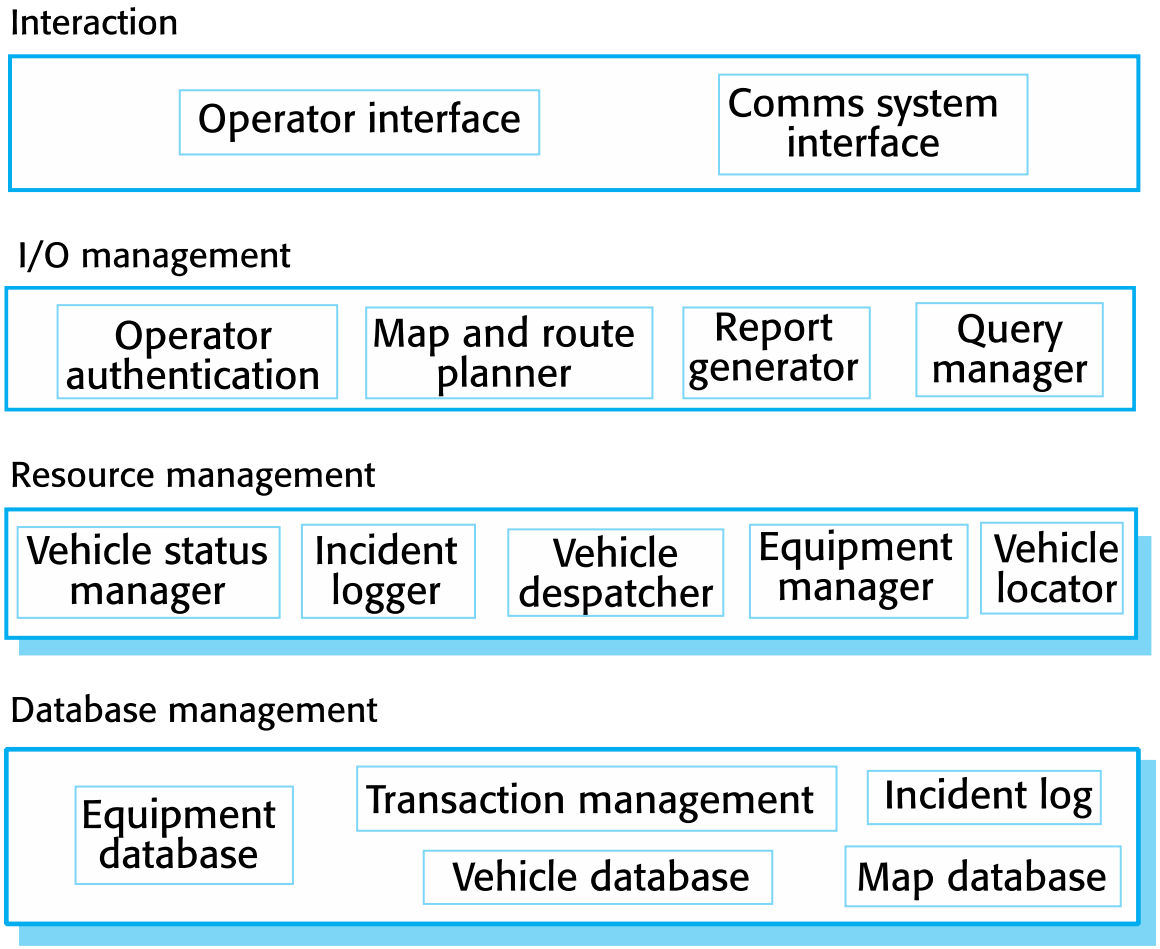
The architecture of a resource allocation system





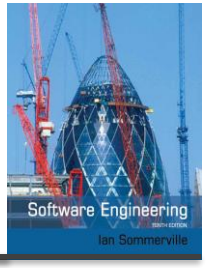
The product line architecture of a vehicle dIspatcher





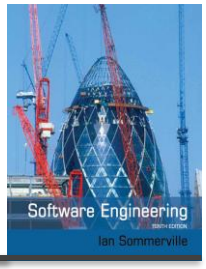
Troy University Resources

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.

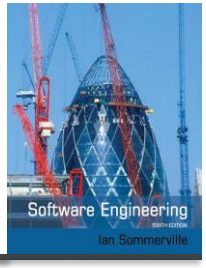
Product line specialisation

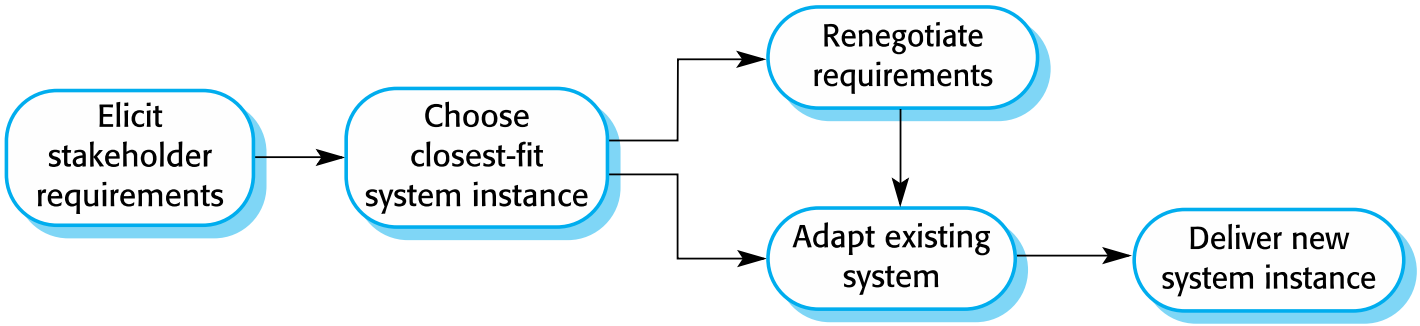


- Platform specialization
  - Different versions of the application are developed for different platforms.
- Environment specialization
  - Different versions of the application are created to handle different operating environments e.g. different types of communication equipment.
- Functional specialization
  - Different versions of the application are created for customers with different requirements.
- Process specialization
  - Different versions of the application are created to support different business processes.

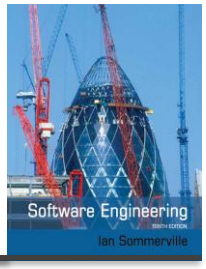
Troy University Resources

Product instance development





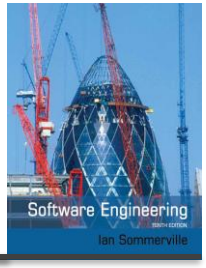
Product instance development



- 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

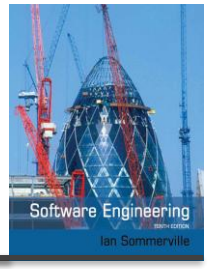
Troy University Resources

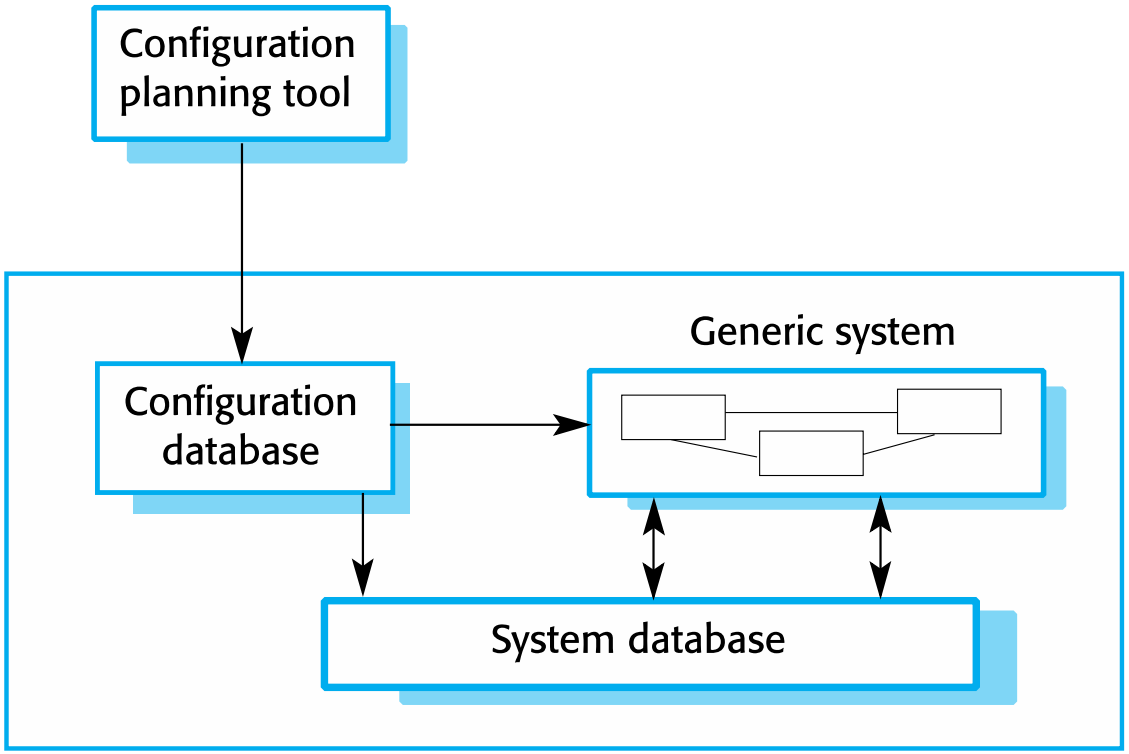
Product line configuration



- Design time configuration
  - The organization that is developing the software modifies a common product line core by developing, selecting or adapting components to create a new system for a customer.
- Deployment time configuration
  - A generic system is designed for configuration by a customer or consultants working with the customer. Knowledge of the customer's specific requirements and the system's operating environment is embedded in configuration data that are used by the generic system.

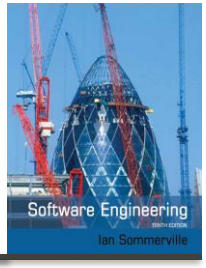
Deployment-time configuration



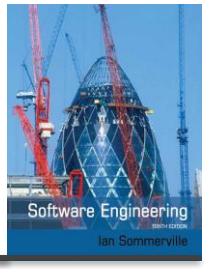


Troy University Resources

Levels of deployment time configuration



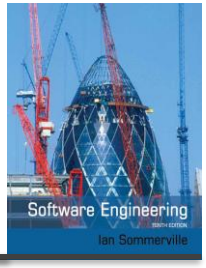
- Component selection, where you select the modules in a system that provide the required functionality.
- Workflow and rule definition, where you define workflows (how information is processed, stage-by-stage) and validation rules that should apply to information entered by users or generated by the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



Application system reuse

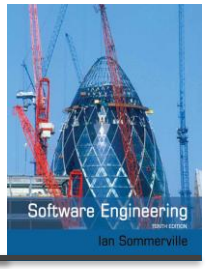
Troy University Resources

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 of patient.

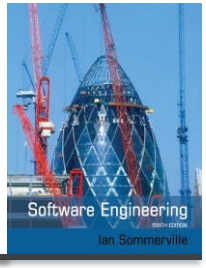
Benefits of application system reuse



- 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, 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.

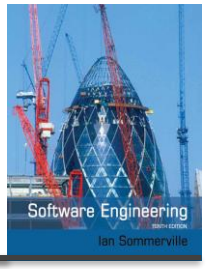
Troy University Resources

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 support and evolution.

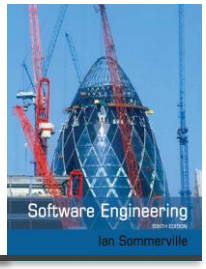
Configurable application systems



- 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.

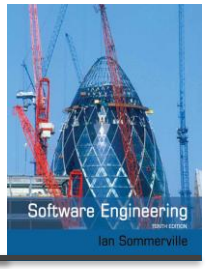
Troy University Resources

COTS-solution and COTS-integrated systems



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

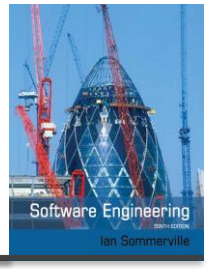
ERP systems

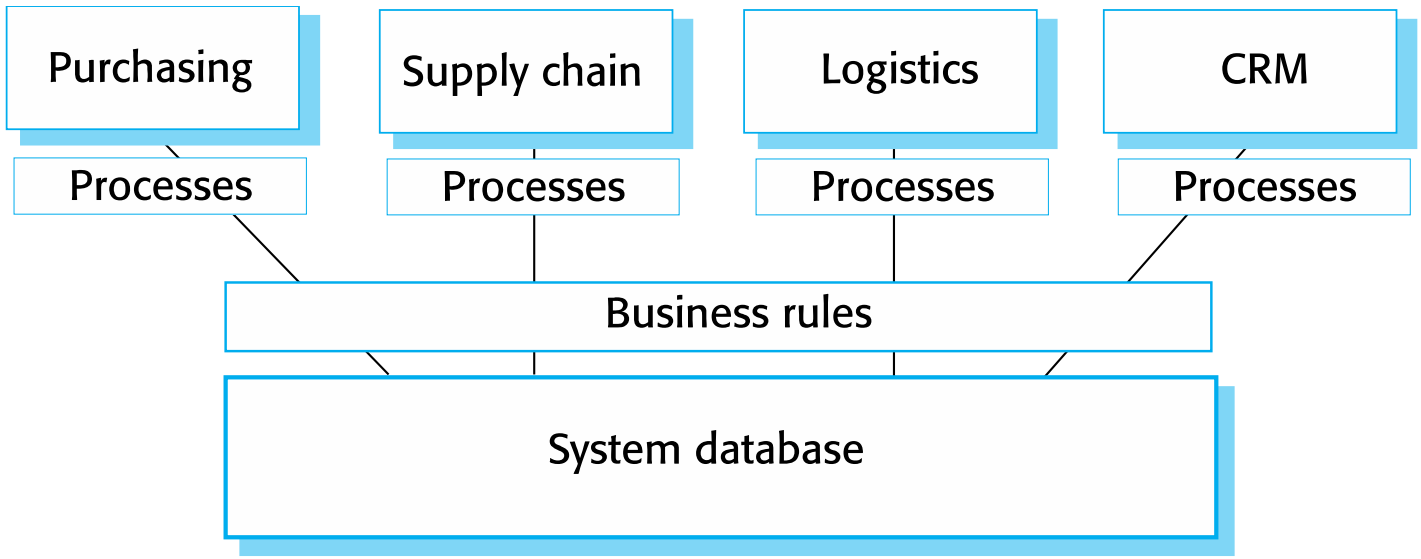


- An Enterprise Resource Planning (ERP) system is a generic system that supports common business processes such as ordering and invoicing, manufacturing, etc.
- These are very widely used in large companies - they represent probably the most common form of software reuse.
- The generic core is adapted by including modules and by incorporating knowledge of business processes and rules.

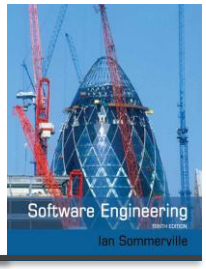
Troy University Resources

The architecture of an ERP system





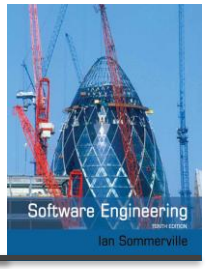
ERP architecture



- 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.

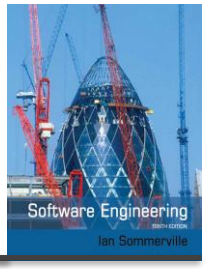
Troy University Resources

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.

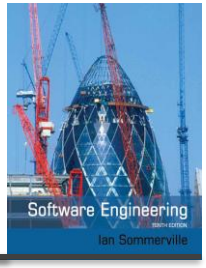
Integrated application systems



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

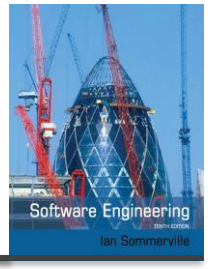
Troy University Resources

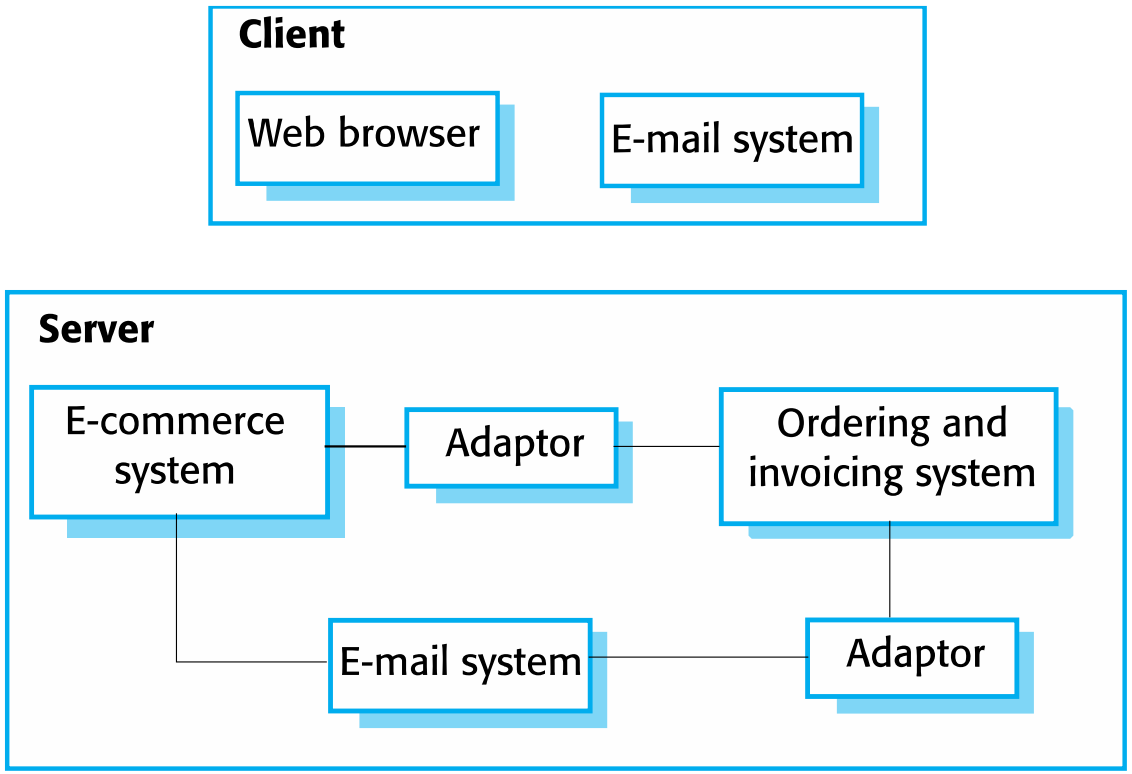
Design choices



- Which individual application systems offer the most appropriate functionality?
  - Typically, there will be several application system 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?
  - Individual application systems may include more functionality than you need and functionality may be duplicated across different products.

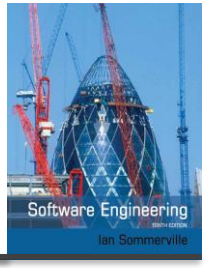
An integrated procurement system





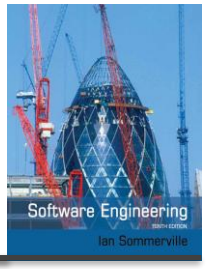
Troy University Resources

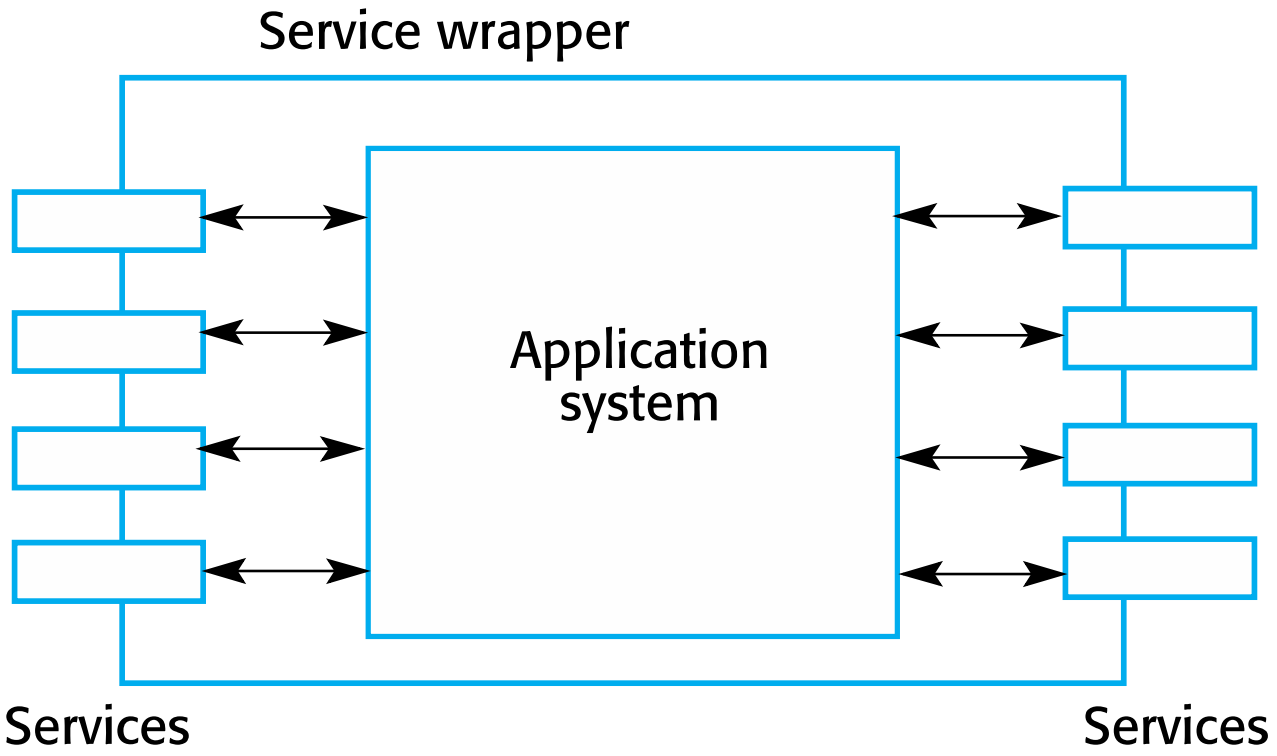
Service-oriented interfaces



- Application system 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.

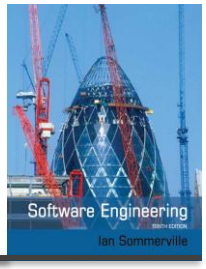
Application wrapping





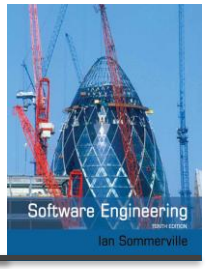
Troy University Resources

Application system integration problems



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

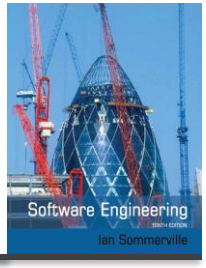
Key points



- There are many different ways to reuse software. These range from the reuse of classes and methods in libraries to the reuse of complete application systems.
- The advantages of software reuse are lower costs, faster software development and lower risks. System dependability is increased. Specialists can be used more effectively by concentrating their expertise on the design of reusable components.
- Application frameworks are collections of concrete and abstract objects that are designed for reuse through specialization and the addition of new objects. They usually incorporate good design practice through design patterns.

Troy University Resources

Key points



- Software product lines are related applications that are developed from one or more base applications. A generic system is adapted and specialized to meet specific requirements for functionality, target platform or operational configuration.
- Application system 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 application system or by integrating two or more application systems.
- Potential problems with application system 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.