Software Reuse

With materials adapted from Software Engineering, 8th Ed. by Ian Sommerville

Topics covered

- Software reuse
- Reuse landscape
- Design patterns
- Generator based reuse
- Application frameworks
- Application system reuse

Software reuse

- Software reuse refers to reusing work products in multiple systems
  - Reusing source code
  - Reusing components
  - Reusing designs, architectures
  - Reusing ideas
  - Etc.
- Goals of software reuse
  - Lower cost
  - Reduce delivery time
  - Higher quality
  - Etc.

Levels of reuse

- Application system reuse
  - Entire system reused either by
    - incorporating it without change into other systems
    - developing application families
- Component reuse
  - Component reused in another component of the same system or other systems
- Object and function reuse
  - Well-defined object or function reused in multiple systems

Reuse benefits 1

- Increased dependability
  - Reusable components are more dependable
- Reduced process risk
  - Likely less uncertainty to reuse the existing reliable components
- Standard compliance
  - Reusable components follow certain standards
  - Improving dependability, reliability, usability

Reuse benefits 2

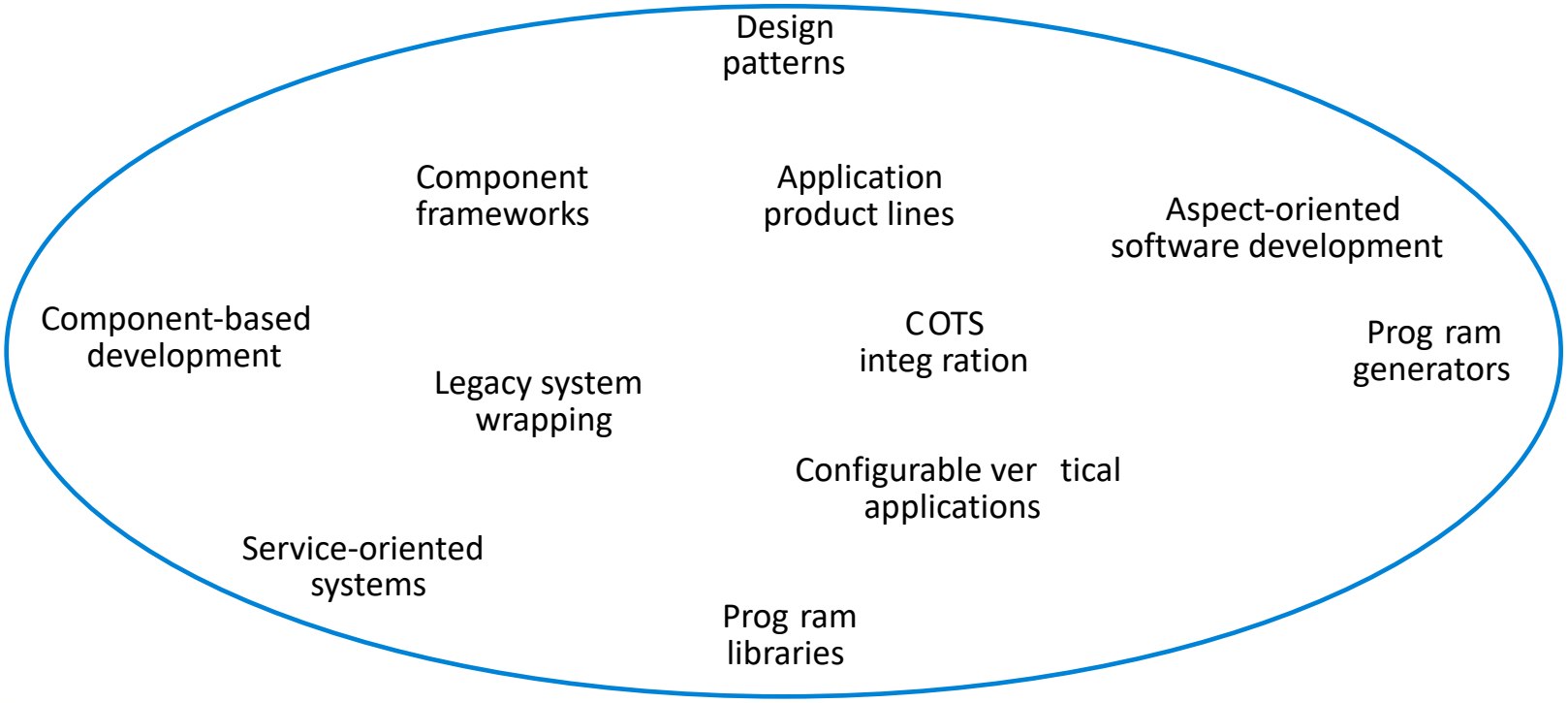
- Accelerated development
  - Reduce schedule and time to market
- Effective use of personnel and expertise
  - Use right skills for right work
  - For example, if we buy a payment system, we don't need people with deep online banking knowledge
- Reduce overall software costs
  - Cheaper to buy existing components than developing the same ones

Reuse problems

- Increased maintenance costs
  - Reused elements may be difficult to change
  - Reused elements do not fit into new changes
- Challenges in finding, understanding, evaluating, adapting reusable components
- Hidden risks in reusable components
  - Safety, privacy, security concerns of reusable components
  - Performance
  - Reliability
- Support uncertainty

Reuse landscape 1

- Many different approaches applied for reuse
- Reuse ranges from simple functions to complete systems



Reuse landscape 2

- Design patterns
- Component-based development
  - Integrating components according component-based models
- Application frameworks
  - e.g., .net framework, J2EE
- Legacy system wrapping
- Service-oriented development
  - Shared services, e.g., Web services

Reuse landscape 3

- Software product lines
  - Adapting product in different ways for different customers
- COTS integration
  - Built on integrating existing components
- Configurable vertical applications
  - Product configured for different customers
- Program libraries
- Program generators
- Aspect-oriented software development

Topics covered

- Reuse landscape
- Design patterns and architectural styles
- Generator based reuse
- Application frameworks
- Application system reuse

Design patterns and arch. styles

- A design pattern is a way of reusing **abstract knowledge** about a problem and its solution
  - E.g., polymorphism, messaging
  - E.g., singleton
- A pattern is a description of the problem and the essence of its solution
- An architectural style is a high-level pattern for architecture

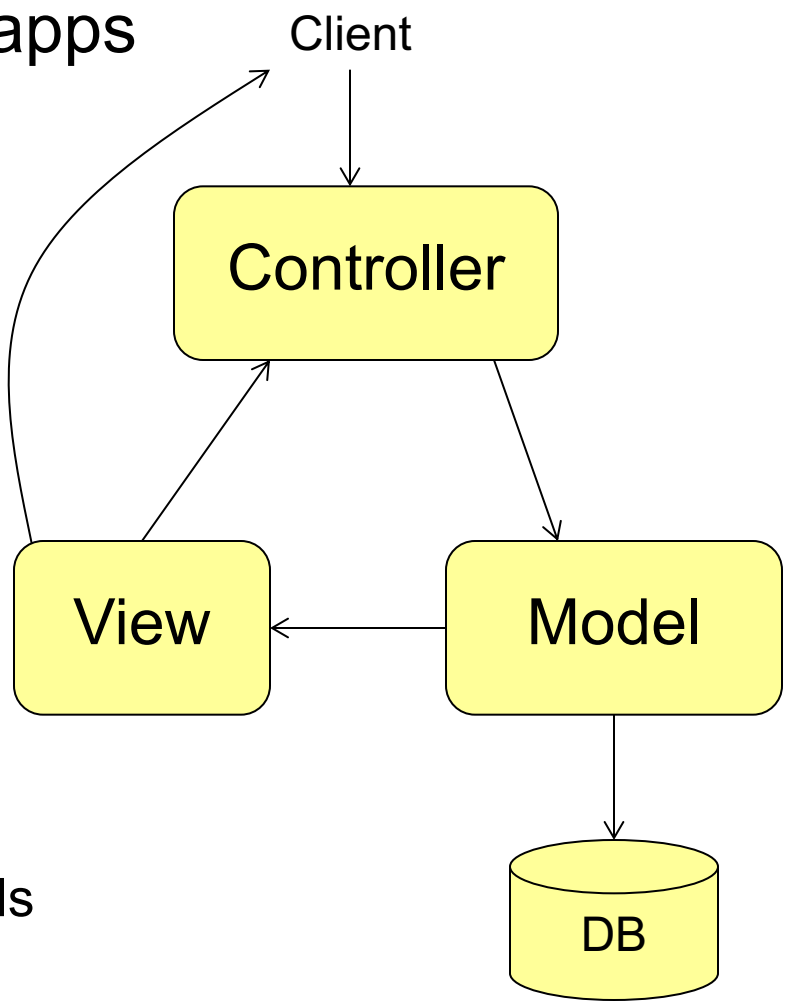
Design patterns and arch. styles

- Design patterns
  - Singleton
  - Observer
  - Builder
  - Bridge
  - Strategy
  - Factory Method
  - Composite
- Architectural styles
  - Client-server
  - MVC
  - N-tier
  - Pipes and filters
  - Repository
  - Broadcasting
  - Microservices

Model-View-Controller (MVC)

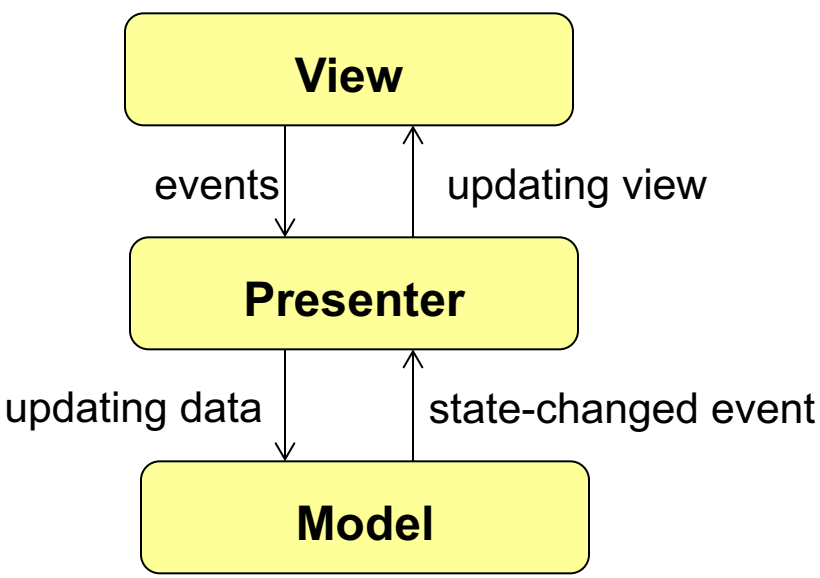
- Originally introduced for UI apps
- Now popular for Web apps

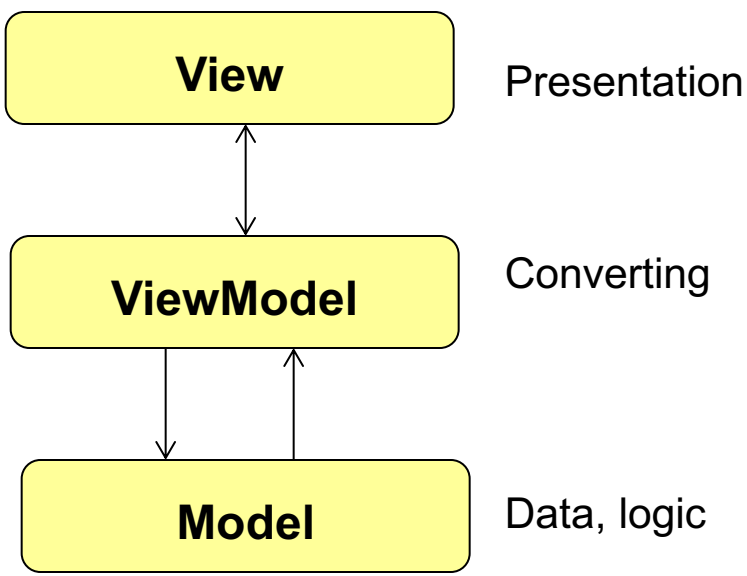
- Components
  - Model
    - Data, logic processing
  - View
    - Presentation
  - Controller
    - Accepting commands, inputs
    - Converting, passing commands



Model-View-Controller (MVC) – 2

- Many variations
  - Presentation – Abstraction – Control (PAC)
  - Model – View – Adapter
  - Model – View – Presenter
  - Model – View – ViewModel





Topics covered

- Reuse landscape
- Design patterns
- Generator based reuse
- Application frameworks
- Application system reuse

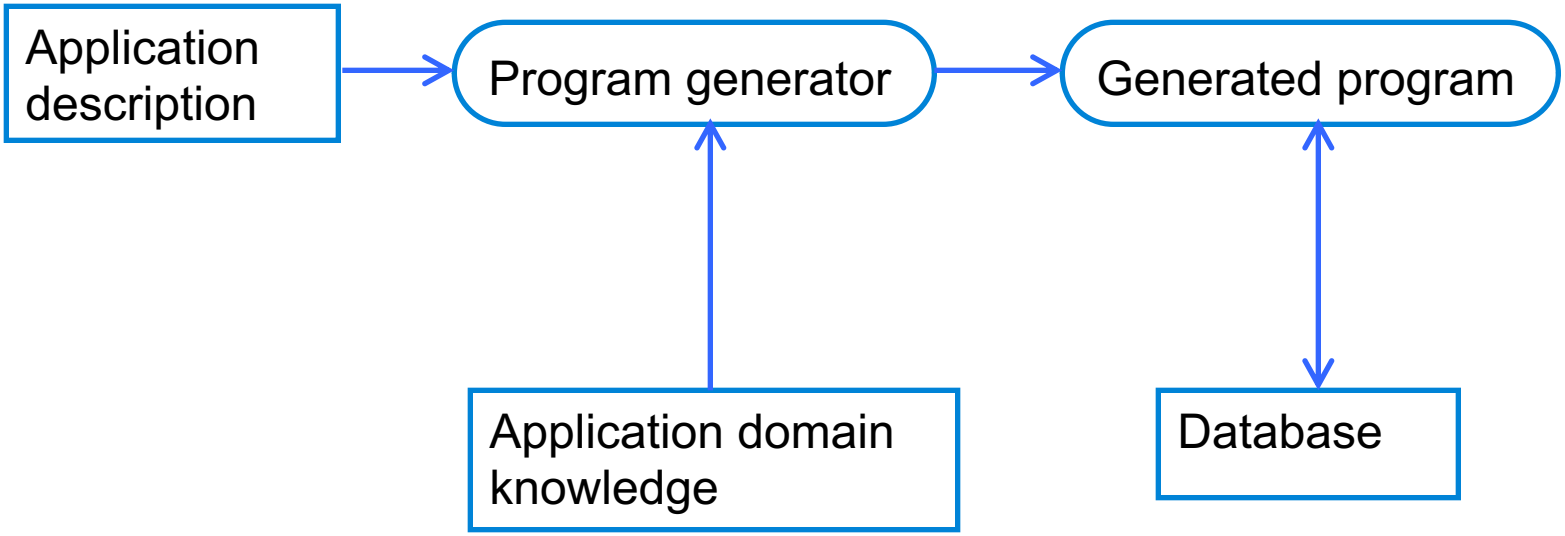
Generator-based reuse

- Program generators involve the reuse of standard patterns and algorithms
  - e.g., Visual Studio code generator for UI, Rational Rose
- Code is embedded in the generator and parameterized by user commands
  - A program is then automatically generated

Types of program generator

- Types of program generator
  - Application generators for business data processing
    - e.g., generating SQL script for DB queries
  - Code generators in CASE tools
    - e.g., HTML generator, Visual studio
- Generator-based reuse is very cost-effective
  - but its applicability is limited to a relatively small number of applications

Reuse through program generation



Topics covered

- Reuse landscape
- Design patterns
- Generator based reuse
- Application frameworks
- Application system reuse

Application frameworks

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

- Frameworks are entities that can be reused
- Examples
  - Desktop-based: .net, WPF, MFC, Eclipse
  - Web: jQuery, dojo, AngularJS, React, Vue, Ember
  - Mobile: React Native, Xamarin, PhoneGap, Flutter

Framework classes

- System infrastructure frameworks
  - Support the development of system infrastructures such as communications, user interfaces and compilers
  - Example, .net, WPF, Eclipse
- Middleware integration frameworks
  - Standards and classes that support component communication and information exchange
  - E.g, COM+, CORBA, EJB
- Enterprise application frameworks
  - Support the development of specific types of application
  - E.g., frameworks for financial systems

Topics covered

- Reuse landscape
- Design patterns
- Generator based reuse
- Application frameworks
- Application system reuse

Application system reuse

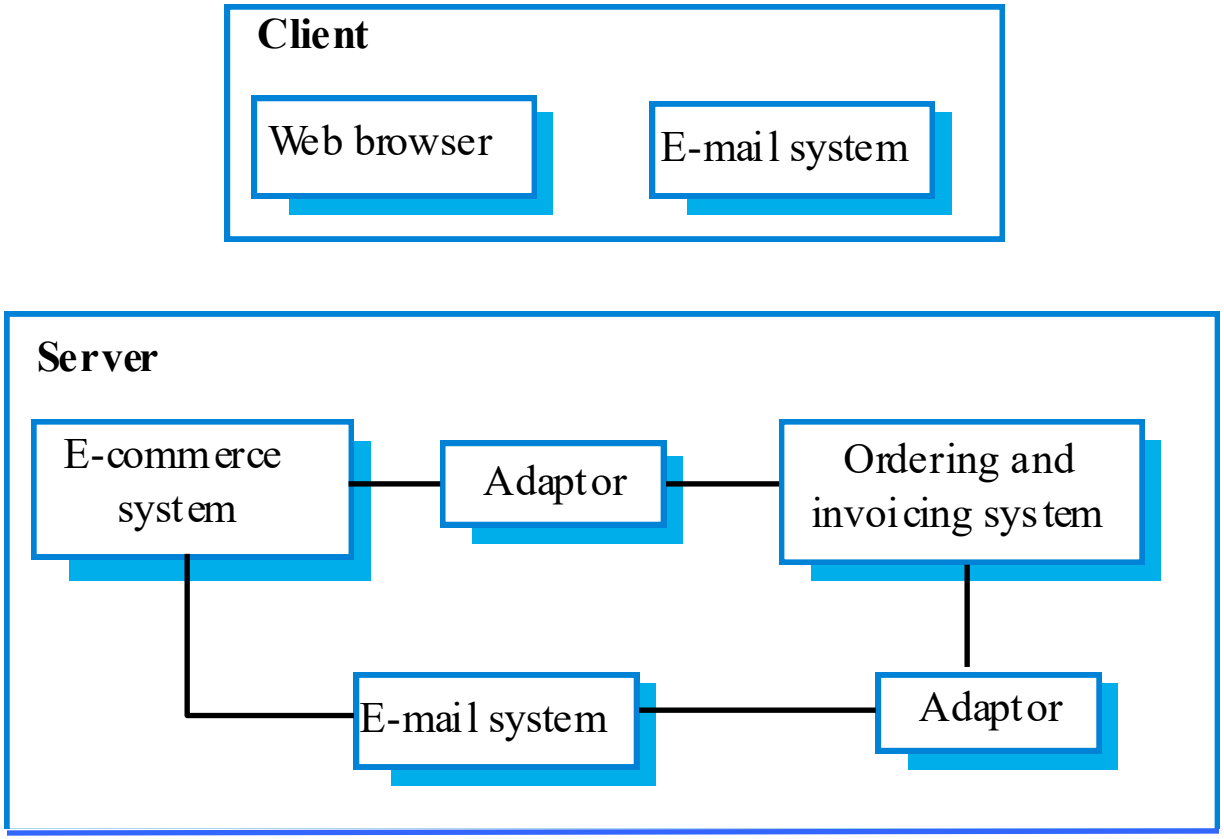
- Involves the reuse of entire application systems either by
  - configuring a system for an environment
  - integrating two or more systems to create a new application

- Two approaches covered here
  - COTS product integration
  - Product line development

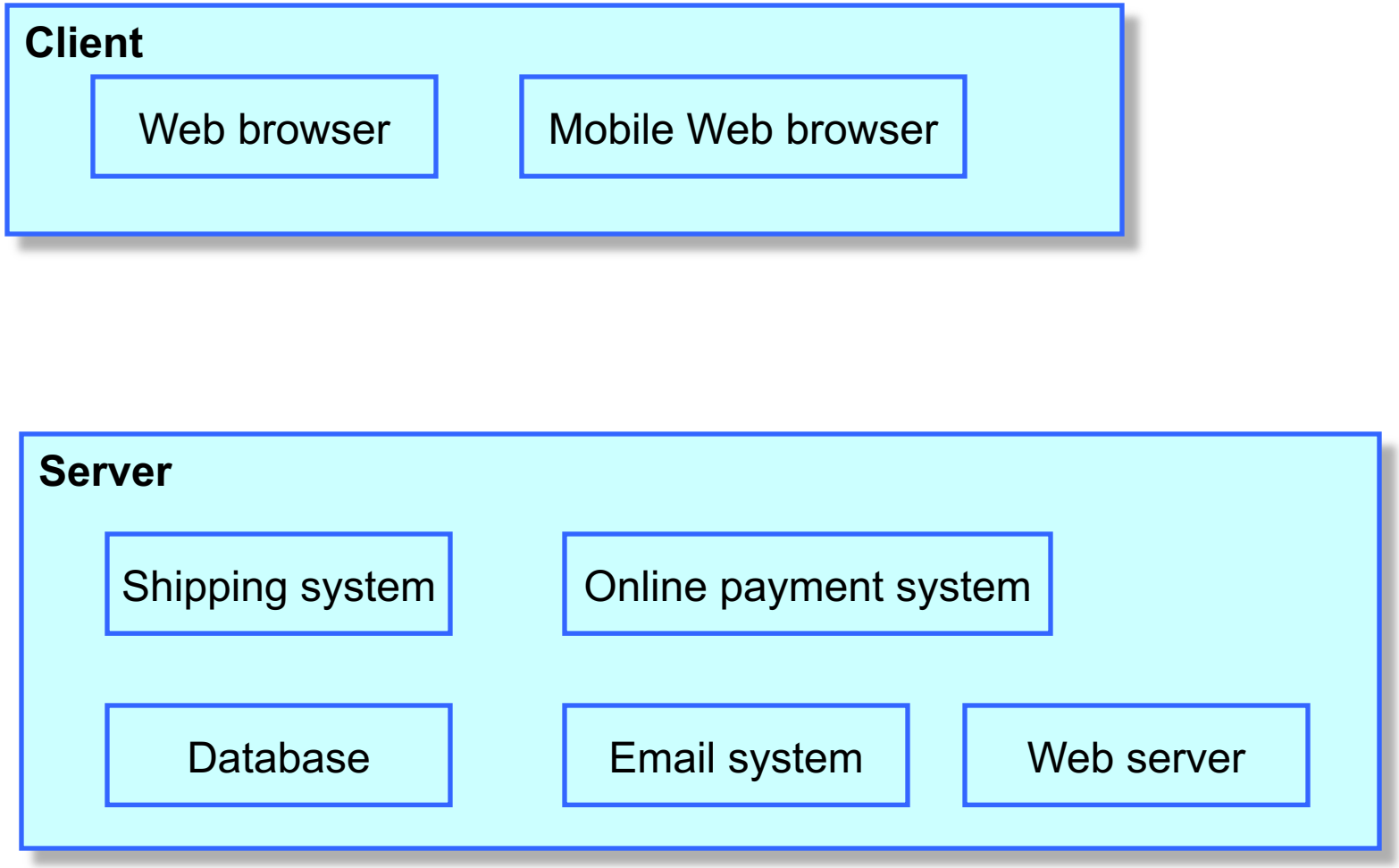
COTS product reuse

- COTS – Component Off-The-Shelf
- COTS is usually a system that offers an API
- Key benefits
  - faster application development
  - lower development costs

E-procurement system



Hailua.com.vn



COTS products reused

- On the client, standard e-mail and web browsing programs are used
- On the server, an e-commerce platform integrated with an existing ordering system
  - This involves writing an adaptor so that they can exchange data (glue code)
  - An e-mail system is also integrated to generate e-mail for clients

COTS integration problems

- Lack of control over functionality and performance

- Problems with COTS inter-operability
  - Possible incompatibility between components
- No control over system evolution
  - COTS vendors control over evolution
- Support from COTS vendors
  - COTS vendors may not offer support over the lifetime of the product

Software product lines

- Software product lines are applications with generic functionality that can be adapted and configured for use in a specific context

- Example
  - MS Office
  - MS Windows
  - SAP ERP (Enterprise Resource Planning), Oracle PeopleSoft

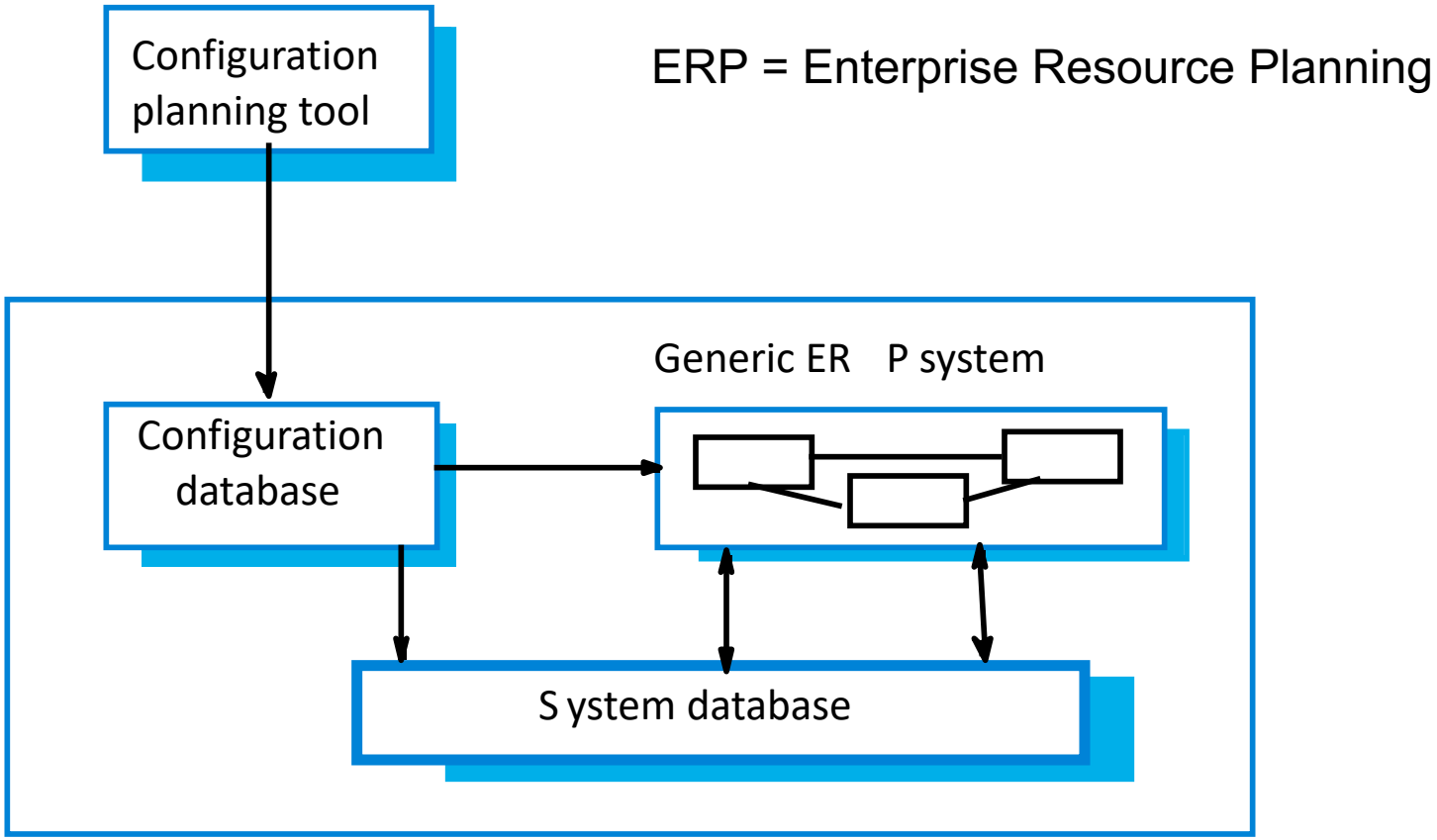
Product lines configuration

- Deployment time configuration
  - A generic system is configured by embedding customer's requirements and business processes
  - Software itself is not changed

- Design time configuration
  - A common generic code is adapted and changed according to the requirements of particular customers

ERP system

- Deployment time configuration



Key points

- Software reuse is a key focus of software engineering
- Advantages of reuse are lower costs, faster software development and lower risks
- Design patterns are high-level abstractions that document successful design solutions
- Program generators are also concerned with software reuse - the reusable concepts are embedded in a generator system

ERP system

- Enterprise Resource Planning (ERP) system is a generic system supporting common business processes e.g., ordering and invoicing, manufacturing, etc.

- Widely used in large companies
  - represent probably the most common form of software reuse
- generic core is adapted by including modules and by incorporating business processes and rules