

Reusable Components

Chapter 20

In this Lecture you will Learn:

- Why reuse is regarded as important
- Some of the problems with reuse
- How to plan system development to achieve reuse of components
- The different techniques for obtaining and managing reuse
- How to design a reusable component

Why Reuse?

- Reuse allows programmers to save time and effort by reusing other developers' code
- Reuse applies to procedural and functional languages as well (e.g. Fortran and C Libraries)
- The growth of Visual Basic can be attributed to the easy availability of reusable controls offering all kinds of functionality
- Object-oriented languages have been promoted as enabling reuse (e.g. Java)

Economic Argument for Reuse

- Hewlett Packard achieved
 - ◆ Improved productivity
 - ◆ Faster time to market
 - ◆ Fewer defects
 - ◆ Return on investment of 215% on one project, 410% on another

Quality Argument for Reuse

- Reusing a component that has been developed and tested elsewhere saves time and money in testing and quality assurance
 - ◆ IBM maintains a library of zero-defect components in Ada, PL/CX and C++

Why hasn't O-O Delivered?

- Inappropriate choice of projects for reuse
- Planning for reuse too late
- Level of coupling between classes in an O-O design
- Lack of standards for reusable components

Choice of Projects

- Jacobson et al. (1997) identify four kinds of software businesses suitable to develop reusable components. In all of these they talk of the organization developing a Reuse-driven Software Engineering Business (RSEB)
 - ◆ Large organizations with a portfolio of projects and Information System infrastructure
 - ◆ Hardware developers using embedded software
 - ◆ Consultancy companies, particularly those that work in vertical markets
 - ◆ Large software product developers where components can be reused across a product range

Organizational Structure

- Jacobson et al., (1997) based on experience at HP, describe organizations as typically going through 6 stages of development of a reuse culture.
 - ◆ No code reuse
 - ◆ Informal code reuse
 - ◆ Black-box code reuse
 - ◆ Managed workproduct reuse
 - ◆ Architected reuse
 - ◆ Domain-specific reuse-driven organization

Organizational Structure

- Allen and Frost (1998) argue that organizations do not have the right approach to or mindset for reuse
- This requires
 - ◆ Organizational change
 - ◆ Software tools to manage repositories of reusable components

Appropriate Unit of Reuse

- Components rather than classes are the unit of reuse
- Classes are often too closely linked to other classes
- Components are groups of classes that deliver some higher-level functionality

Definitions of Components

- Jacobson et al. (1997) state that
 - ◆ A component is a type class or any other work product that has been specifically engineered to be reusable.

Components as Sub-systems

- Sub-systems that provide more functionality than a single class
- It should also be possible to reuse intermediate products from the life cycle
 - ◆ Use cases
 - ◆ Analysis models
 - ◆ Design models
 - ◆ Test models

Component Standards

- Recent factors that promote reuse
 - ◆ CORBA (Common Object Request Broker Architecture) standard for interoperability of components
 - ◆ Java as a language with mechanisms to package reusable components (and standards like Enterprise Java Beans)
 - ◆ The growth of the WWW as a means of distributing reusable components

Component Standards

- New standards
 - ◆ SOAP (Simple Object Access Protocol) based on XML (eXtensible Markup Language)
 - ◆ Web Services mechanism to find services on the Internet using UDDI
 - ◆ UDDI (Universal Description, Discovery and Integration) directory service
 - ◆ .NET and C#

A Sample of Languages and Development Environments with Mechanism for Reuse

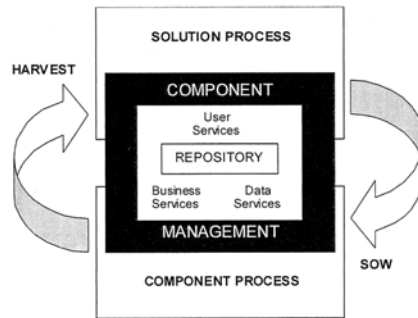
Language or development environment	Mechanism for component reuse
Borland Delphi	Object Pascal units compiled into .dll files—Dynamic Link Libraries
Microsoft Visual Basic	.vbz files—Visual Basic Extensions .ocx files
Microsoft Windows	.ole files—Object Linking and Embedding DDE—Dynamic Data Exchange .dll files—Dynamic Link Libraries COM—Common Object Model DCOM—Distributed Common Object Model
CORBA	.idl files—Interface Definition Language IOP—Inter-ORB Protocol
Java	.jar files—Java Archive packages JavaBeans
Microsoft .NET	MSIL—Microsoft Intermediate Language CLR—Common Language Runtime WSDL—Web Service Description Language

Strategy for Reuse: SELECT

- The SELECT Perspective (Allen & Frost)
- **Solution Process** – Delivering services to meet business requirements while drawing on the component process in the search for reusable components
- **Component Process** – Developing reusable components in packages to deliver generic business services

The SELECT Perspective

- The analogy of sowing and harvesting
- A repository for component management is central to this approach, with facilities to store and index components and to browse and search for them
- Sowing reusable components and harvesting them in business solutions



M8748 © Peter Lo 2007

17

Strategy for Reuse: RSEB

- Reuse-driven Software Engineering Business (RSEB) (Jacobson et al.)
- Intermediate artefacts are reusable as well as finished code
- Requires an architectural process to design reusable models and code
- This requires a change to the way in which the software business operates

M8748 © Peter Lo 2007

18

Reuse-driven Software Engineering Business (RSEB)

- Developing a reuse business requires reengineering the organization
 - ◆ Requirements Capture Unit
 - ◆ Testing Unit
 - ◆ Component Engineering Unit
 - ◆ Architecture Unit
 - ◆ Component Support Unit

M8748 © Peter Lo 2007

19

Strategy for Reuse - Three Engineering Processes

- **Application Family Engineering (AFE)** – Produces layered architecture of application and components
- **Component System Engineering (CSE)** – Develops reusable components to support application development
- **Application System Engineering (ASE)** – Develops application systems designed to make use of components produced by CSE

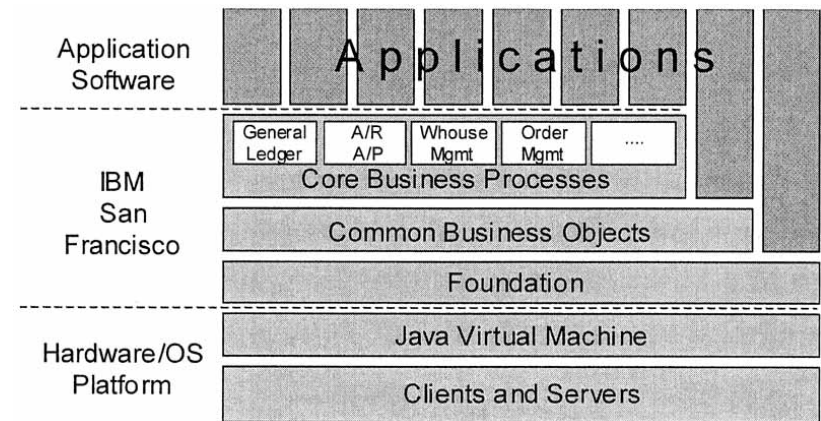
M8748 © Peter Lo 2007

20

Example for Reuse: Commercially Available Componentware

- VBX, OCX and COM Components
- Wide range of components using Microsoft architecture
- Controls for
 - ◆ Serial communication
 - ◆ Computer-aided design
 - ◆ Project management
 - ◆ Spreadsheets
 - ◆ Scientific charts
 - ◆ Barcode reading and printing

Example for Reuse: IBM San Francisco Project



Example for Reuse: IBM San Francisco Project

- Java distributed server-based components
- Layered architecture
 - ◆ Foundation Layer
 - ◆ Common Business Objects
 - ◆ Core Business Processes
 - ◆ General ledger
 - ◆ Accounts receivable and payable
 - ◆ Warehouse management
 - ◆ Order management

Facade Pattern

- To provide an interface to a set of interfaces in a sub-system.
- Using a Façade provides a higher-level interface that makes the sub-system easier to use because the operations are all operations of a single Façade class.
- The developer using the sub-system does not need to know the details of the internal structure of the sub-system.