Reusability, Portability, and Interoperability

• “If reinventing the wheel were a criminal offense, many software professionals would be languishing in jail” [Schach 2001]

Reuse Concepts

• Portable
  – Easily modifying a product as a whole to run under another compiler, OS, or hardware architecture

• Reuse
  – Using components of one product to facilitate the development of another product
Reuse Concepts

• Reusable components
  – Code fragments
  – Modules
  – Design
  – Part of a manual
  – Set of test cases or test data
  – Duration
  – Cost estimates

Types of Reuse

• Accidental reuse (Opportunistic reuse)
  – Developers realize that components of a previously
devolved product can be reused in the new product

• Deliberate reuse (Systematic reuse)
  – Using software components specifically constructed
for future reuse
Types of Reuse

• Advantage of Deliberate reuse versus Accidental reuse
  – Components are designed and constructed specially for use in future developments
    – Easy to reuse
    – Safe to reuse
    – Robust
    – Well documented
    – Thoroughly tested
    – Makes maintenance easier

• Disadvantage of Deliberate reuse versus Accidental reuse
  – Developing reusable product can be
    • Expensive
    • Time consuming to
      – Design
      – Implement
      – Test
      – Document
    • No guarantee the component will be reused
Impediments to Reuse

• Software professionals would like to rewrite a module rather than reuse an existing module
  – I can make a better product
  – Not Invented Here (NIH) syndrome
  – How to solve this problem?
    • management can offer incentives for reuse

Impediments to Reuse

• Developers would be willing to reuse a module if they are assured it will not introduce faults into their product
  – Every software professional has seen bad code
  – How to solve this problem?
    • Exhaustively test the reusable modules before making them available for reuse
Impediments to Reuse

• Organizing and storing reusable modules are difficult
  – Developers must find modules faster than they can develop such modules
  – How to solve this problem?
    • Use CASE tools

Impediments to Reuse

• Reuse is expensive
  – Three costs are involved in reuse
    • Cost of defining and implementing reuse process
    • Cost of making it reusable
    • Cost of reusing it
Impediments to Reuse

• Legal issues of reuse
  – Legal issues arise with contracted software
    • The software belongs to the client
    • Reusing modules from one client for another constitutes copyright.
  – These legal issues do not arise for in-house development

• Using Commercial Off The Shelf (COTS) products
  – Source code is not distributed to the client
  – Products using COTS components have limited extensibility and modifiability
Object and Reuse

• Composit/structured design (C/SD) states:
  – Modules with functional cohesion are candidates for reuse
    • Modules with functional cohesion perform one task
    • Disadvantage:
      – The data must be reused with the module
        » If the data is not identical in the new project the data or the functionality of the module must change

Object and Reuse

• Modules with information cohesion are better candidates for reuse
  – A module that represents a real world entity
  – A module that contains data and the action that operates on the data
  – Object (instance of a class) is such a module
  – If Object-Oriented paradigm is utilized correctly, the resulting object has information cohesion
    • Results in reuse
Design Reuse

• An engineer realizes that a module or class can be reused from a previous project
• This type of reuse happens in an organization that produces similar applications
  – Banking
  – Spacecraft
• Libraries are available for scientific applications such as LAPACK++

Application Framework

• Incorporates the control logic of a design
• Reusing the application framework results in faster product development
  – Less design work
    • More of the design is reused
  – Portions reused are harder to design
Design Patterns

• A solution to a general design problem in the form of a set of interacting classes that have to be customized to create a specific design [Schach 2001]

Software Architecture

• The architecture of a software product may be:
  – Object-Oriented
  – Pipes and filters (Unix components)
  – Client-server
Reuse and Maintenance

• Traditionally reuse prompts shorter development time
• For every $1 spent on development $2 is spent on maintenance
• Another important reason for reuse is to reduce the time and cost of maintenance

Portability

• A product that is significantly less expensive to adapt to a new computer system than to write from scratch
  – The strategy is:
    • To ensure product as a whole can be adapted easily to run on a variety of different hardware-OS
    • Some of the cost can be recouped
    • A product written in JAVA is a such product
Portability

• Incompatibilities
  – Hardware
  – Operating system
  – Numerical software
  – Compiler

Interoperability

• Mutual cooperation of object code from different vendors, written in different languages and running on different platforms
  – Component Object Module (COM)
  – Common Object Request Broker Architecture (CORBA)