Overview

• What is a software component
• Modeling components
• Advantages of using components
• Component systems
• Disadvantages of using components
• Demonstration
What Is a Software Component?

• “A component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces.” – Booch, Rumbaugh, & Jacobson

• “A component is an object in a tuxedo. That is, a piece of software that is dressed to go out and interact with the world.” – Michael Feathers, Object Mentor Inc.
Properties of Components

- Highly reusable software module
- Specific purpose/responsibility
- Well-defined interfaces
- Loose coupling, and only via interfaces
- High cohesion
Components vs Classes

- Classes are not necessarily components
- A component could be implemented using a class or a set of classes
- Components don’t have to be implemented using classes
Modeling Components

```
<component>
Order
<provided interfaces>
OrderEntry
AccountPayable
<required interfaces>
Person
```
Modeling Components

Order

OrderEntry

AccountPayable

Order

Person
Modeling Components

[Diagram showing various components including Seminar Management, Student Administration, Facilities, Student, Seminar, Schedule, Security, Persistence, University DB, with connections and relationships indicated]
Modeling Components
Advantages

• High degree of reuse
• Helps enforce design before implementation
• Independent development/maintenance
• Interchangeability
• Allows software to be assembled
Software Assembly

- Building complex systems out of prebuilt components with no extra code required
- Allows complex systems to be constructed very rapidly
- Tools are available today that allow some component assembly to be done graphically
Component Generation

- Use code generators to generate components from high-level specifications

Examples

- User interfaces
- "Boilerplate" components
- Anything that is easier to model in a non-traditional programming language
Component Systems

- A standard way for components to interact with each other and the rest of the system
- Set of standard interfaces and/or conventions
- File format for packaging components
Component Systems

- JavaBeans
- Component Object Model (COM)
- .NET Assembly
- UNIX pipes
  - cat in.txt | sort | uniq | top -n5
Component Distribution

- Statically linked with executable
- Included with operating system/environment
- Included as part of a common framework
- Bundled with applications that use them
  - May have problems with conflicting versions overwriting each other on install (DLL Hell)
Disadvantages

- Overhead
- Security is only as good as component model
  - ActiveX components run at user’s security level
  - Java and .NET components can have restricted access granted depending on how they are obtained
- Components are tied to a particular component system
Component Description Problem

- It is difficult to thoroughly describe the operating characteristics of a component
  - Speed, capacity, and reliability
- Depends critically on usage
- It is even more difficult to describe the characteristics of several components in series
Component Description Problem

- Make generalizations about how component will be used
- Specify optimal usage patterns this component was designed for
- Include characteristic behavior under all conditions and with all input
Case Study: ESCOT

- Educational Software Components of Tomorrow (ESCOT)
- Library of “μApps” focused on teaching math to middle school students
- Used a process called Component-Oriented Rapid Development (CORD) to develop their math μApps
Case Study: ESCOT

- Met as a group to analyze and design µApp
- Created mock-up with paper and post-its to model how components would need to interact
- Chose JavaBeans as component system
- µApps would be distributed as applets
Case Study: ESCOT

- Distributed development effort with mostly independent teams
- Used Geometer’s Sketchpad as a component generator (JavaSketchpad)
- JavaScript was used to glue components together
Case Study: ESCOT

- Component-based design worked very well for distributed development effort
- Increasing levels of formality in design representations
- Prototyping is useful in resolving interoperability issues early
- Java platform idiosyncrasies increased testing time
Case Study: ESCOT

http://www.escot.org/
Demonstration
Questions?