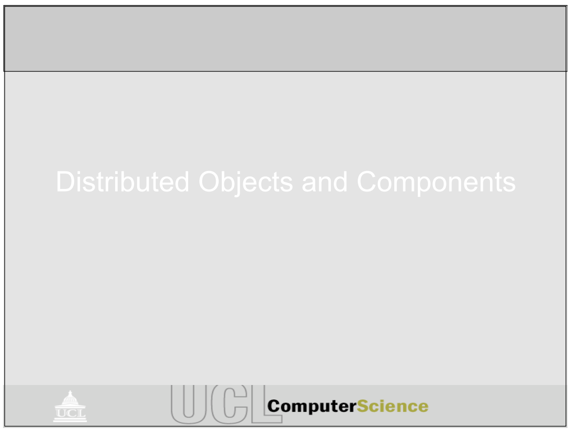
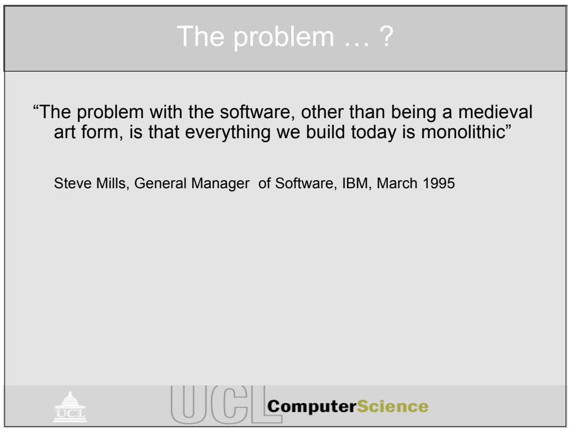


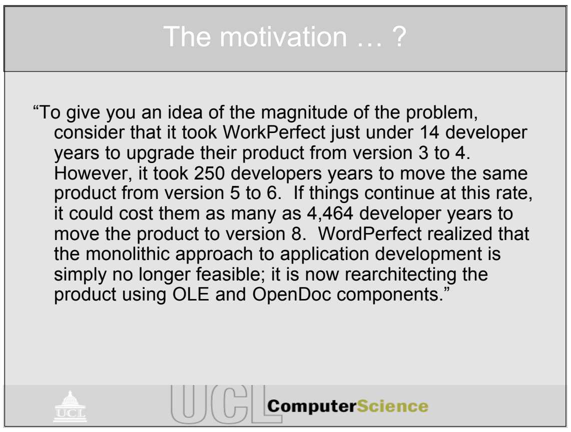
Distributed Objects and Components



The problem … ?

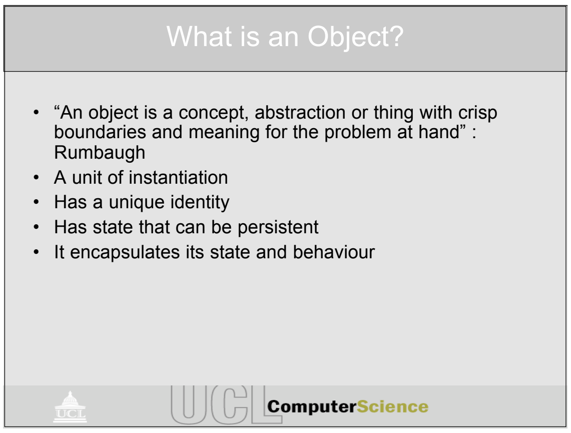
"The problem with the software, other than being a medieval art form, is that everything we build today is monolithic"

Steve Mills, General Manager of Software, IBM, March 1995



The motivation … ?

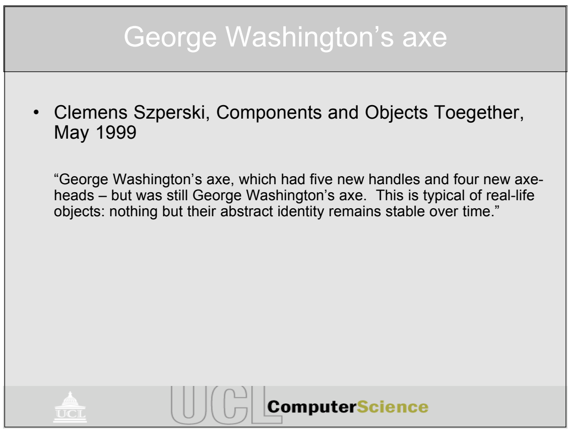
"To give you an idea of the magnitude of the problem, consider that it took WorkPerfect just under 14 developer years to upgrade their product from version 3 to 4. However, it took 250 developers years to move the same product from version 5 to 6. If things continue at this rate, it could cost them as many as 4,464 developer years to move the product to version 8. WordPerfect realized that the monolithic approach to application development is simply no longer feasible; it is now rearchitecting the product using OLE and OpenDoc components."



What is an Object?

- “An object is a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand” : Rumbaugh
- A unit of instantiation
- Has a unique identity
- Has state that can be persistent
- It encapsulates its state and behaviour

ComputerScience

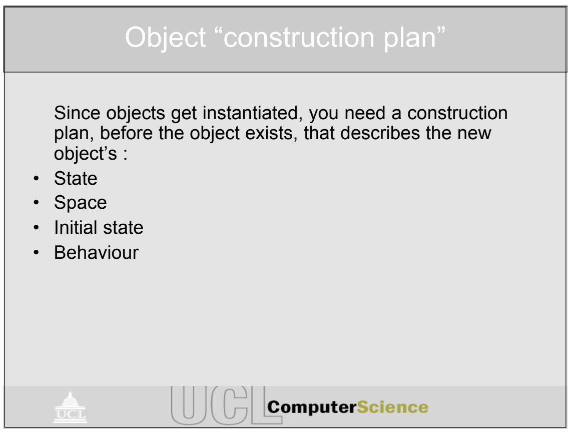


George Washington’s axe

- Clemens Szperski, Components and Objects Toegether, May 1999

  “George Washington’s axe, which had five new handles and four new axe-heads – but was still George Washington’s axe. This is typical of real-life objects: nothing but their abstract identity remains stable over time.”

ComputerScience

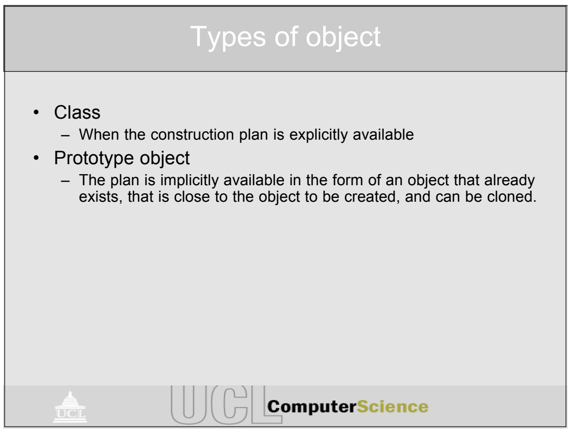


Object “construction plan”

Since objects get instantiated, you need a construction plan, before the object exists, that describes the new object’s :

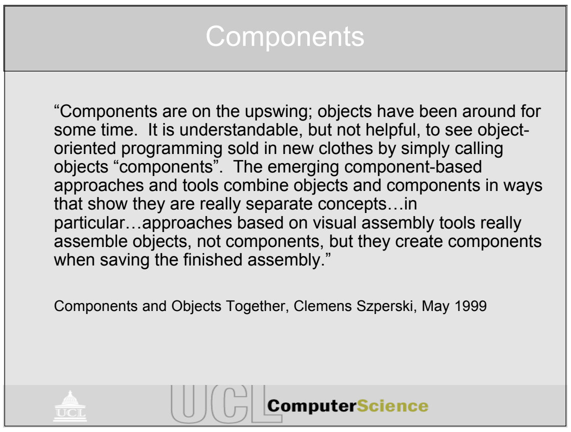
- State
- Space
- Initial state
- Behaviour

ComputerScience



Types of object

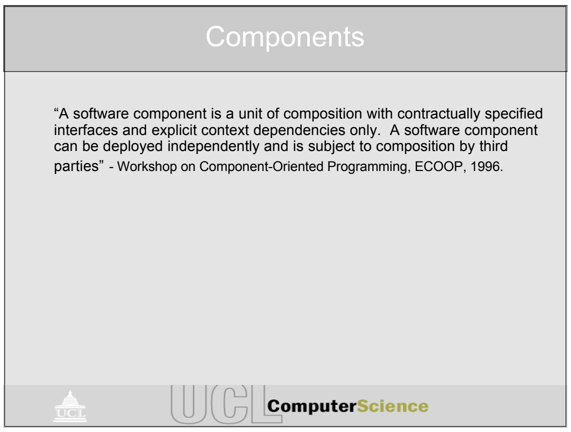
- Class
  - When the construction plan is explicitly available
- Prototype object
  - The plan is implicitly available in the form of an object that already exists, that is close to the object to be created, and can be cloned.



Components

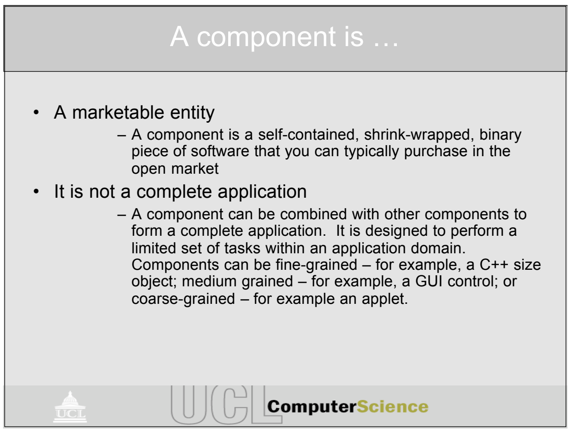
"Components are on the upswing; objects have been around for some time. It is understandable, but not helpful, to see object-oriented programming sold in new clothes by simply calling objects "components". The emerging component-based approaches and tools combine objects and components in ways that show they are really separate concepts…in particular…approaches based on visual assembly tools really assemble objects, not components, but they create components when saving the finished assembly."

Components and Objects Together, Clemens Szperski, May 1999



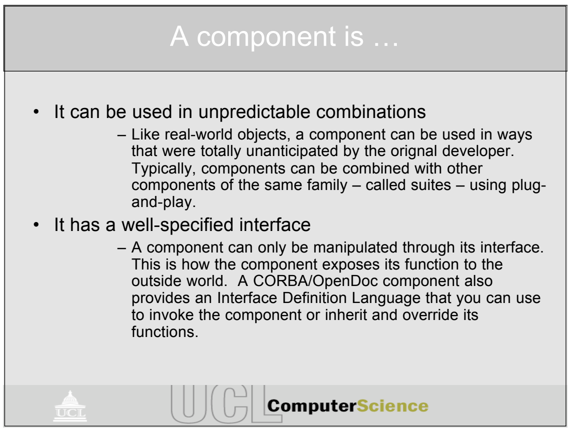
Components

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties" - Workshop on Component-Oriented Programming, ECOOP, 1996.



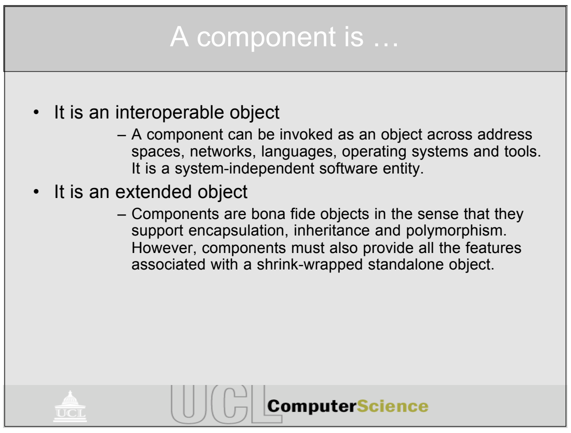
A component is …

- A marketable entity
  - A component is a self-contained, shrink-wrapped, binary piece of software that you can typically purchase in the open market
- It is not a complete application
  - A component can be combined with other components to form a complete application. It is designed to perform a limited set of tasks within an application domain. Components can be fine-grained – for example, a C++ size object; medium grained – for example, a GUI control; or coarse-grained – for example an applet.



A component is …

- It can be used in unpredictable combinations
  - Like real-world objects, a component can be used in ways that were totally unanticipated by the orignal developer. Typically, components can be combined with other components of the same family – called suites – using plug-and-play.
- It has a well-specified interface
  - A component can only be manipulated through its interface. This is how the component exposes its function to the outside world. A CORBA/OpenDoc component also provides an Interface Definition Language that you can use to invoke the component or inherit and override its functions.



A component is …

- It is an interoperable object
  - A component can be invoked as an object across address spaces, networks, languages, operating systems and tools. It is a system-independent software entity.
- It is an extended object
  - Components are bona fide objects in the sense that they support encapsulation, inheritance and polymorphism. However, components must also provide all the features associated with a shrink-wrapped standalone object.

What the technology means for the Client/Server view of computing

In client/server computing the client – typically a PC – provides the graphical interface, while the server provides access to shared resources – usually a database. Distributed objects and components will effectively revolutionize this view. Component-like objects allow us to create client/server systems by assembling "live blobs of intelligence and data" in an infinite number of lego-like arrangements.

This will have a huge impact on the Client/Server use of applications. Essentially the distinction between what is a client and server will blur, machines will be both at the same time. The large client/server applications will become subdivided into self-managing components that can play together and roam across networks and operating systems. Applications will no longer be split across client and server lines.

UCL ComputerScience

Deploying the component infrastructure

So, we know what objects and components are but how do we get to use them?

- COM
- COM+
- CORBA
- JAVA BEANS
- ENTERPRISE JAVA BEANS

UCL ComputerScience

COM

COM provides the component technology for Microsoft Windows Distributed interNet Applications (Windows DNA) architecture, which enables developers to integrate Web-based and client/server applications in a single, unified architecture. Using COM, developers can create distributed components that are written in any language and that can interact over any network.

UCL ComputerScience

COM +

COM+ will make it even easier for developers to create software components in any language using any tool. COM+ builds on the factors that have made today's COM the choice of developers worldwide, including the following:

(1) The richest integrated services, including transactions, security, message queuing and database access to support the broadest range of application scenarios.

(2) The widest choice of tools from multiple vendors using multiple development languages.

(3) The largest customer base for customizable applications and reusable components.

(4) Proven interoperability with users' and developers' existing investments. (Microsoft)

UCL **Computer**Science

CORBA

The *Common Object Request Broker Architecture (CORBA)*, developed by the Object Management Group (OMG) in 1990, enables invocations of methods on distributed objects residing anywhere on a network, just as if they were local objects.

A CORBA implementation employs Object Request Brokers (ORBs), located on both the client and the server, to create and manage client/server communications between objects.

UCL **Computer**Science

About ORBs

ORBS are the key to the CORBA distributed object architecture. They allow objects on the client side to make requests of objects on the server side without any prior knowledge of where those objects exist, what language they are in, or what operating system they are running on. To facilitate these requests and provide ORB interoperability, the CORBA 2.0 specification outlines a protocol named Internet Inter-ORB Protocol, which has quickly been embraced by industry leaders (IBM , Netscape Oracle). (EarthWeb)

UCL **Computer**Science

Java Beans

- *JavaBeans*, the platform-neutral component architecture for Java, has proven to be invaluable in the development of network-aware applications
- You can make any Java class into a bean just by changing the class to adhere to the JavaBeans specification. It's up to you to decide what you want to design as a bean and what you want to design as a Java class
- Regardless of its functionality, every bean should support the following characteristics and behavior: Persistence, Visual manipulation, Introspection, Events and Customization. (Netscape Communications)

UCL ComputerScience

Enterprise Java Beans

Enterprise JavaBeans takes the JavaBeans component architecture released in JDK 1.1 to the next level by providing an API optimized for building scalable business applications as reusable server components.

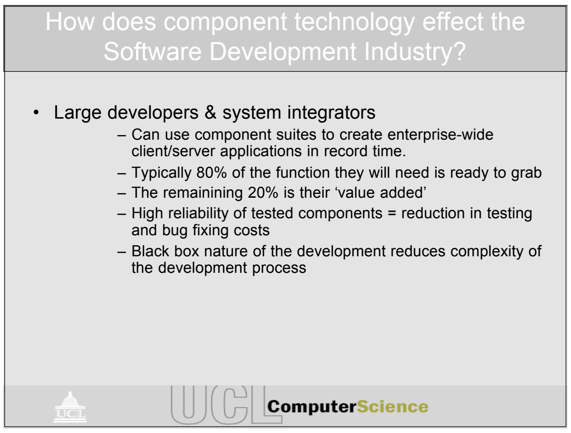
With Enterprise JavaBeans, developers can design and re-use small program elements to build powerful corporate applications. These 'componentized' applications can run manufacturing, financial, inventory management, and data processing on any system or platform that is Java-enabled. (Sun Microsystems)

UCL ComputerScience

How does component technology effect the Software Development Industry?

- Power Users
  - Can ‘easily’ assemble their own personalized applications using off-the-shelf components.
  - Scripts can be used to tie the parts together and customize their behaviour.
- Small developers & Indepenent Software Vendors
  - Components reduce expenses and lower entry barriers
  - Create individual components with the knowledger that they will integrate smoothly with existing software created by larger developers.
  - No need to reinvent functions – effectively they can ‘leap-frog’
  - Faster time to market

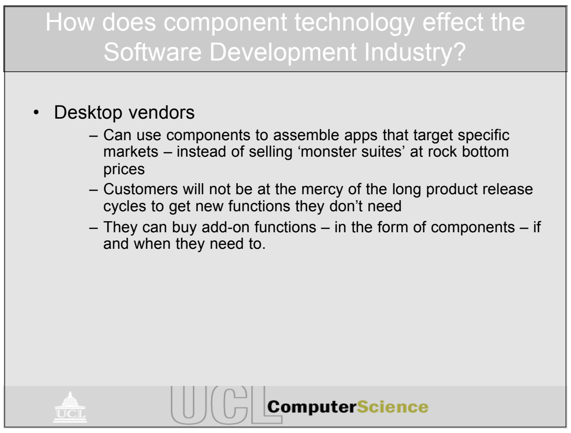
UCL ComputerScience



How does component technology effect the Software Development Industry?

- Large developers & system integrators
  - Can use component suites to create enterprise-wide client/server applications in record time.
  - Typically 80% of the function they will need is ready to grab
  - The remainining 20% is their 'value added'
  - High reliability of tested components = reduction in testing and bug fixing costs
  - Black box nature of the development reduces complexity of the development process

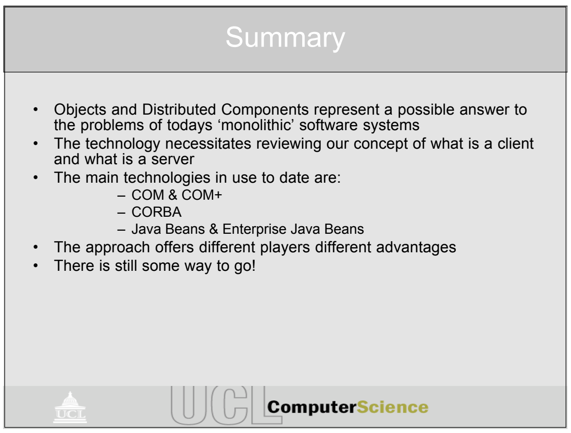
UCL **Computer**Science



How does component technology effect the Software Development Industry?

- Desktop vendors
  - Can use components to assemble apps that target specific markets – instead of selling 'monster suites' at rock bottom prices
  - Customers will not be at the mercy of the long product release cycles to get new functions they don't need
  - They can buy add-on functions – in the form of components – if and when they need to.

UCL **Computer**Science



Summary

- Objects and Distributed Components represent a possible answer to the problems of todays 'monolithic' software systems
- The technology necessitates reviewing our concept of what is a client and what is a server
- The main technologies in use to date are:
  - COM & COM+
  - CORBA
  - Java Beans & Enterprise Java Beans
- The approach offers different players different advantages
- There is still some way to go!

UCL **Computer**Science