DCOM: The Distributed Component Object Model

Rob Jaeger
Department of Computer Science

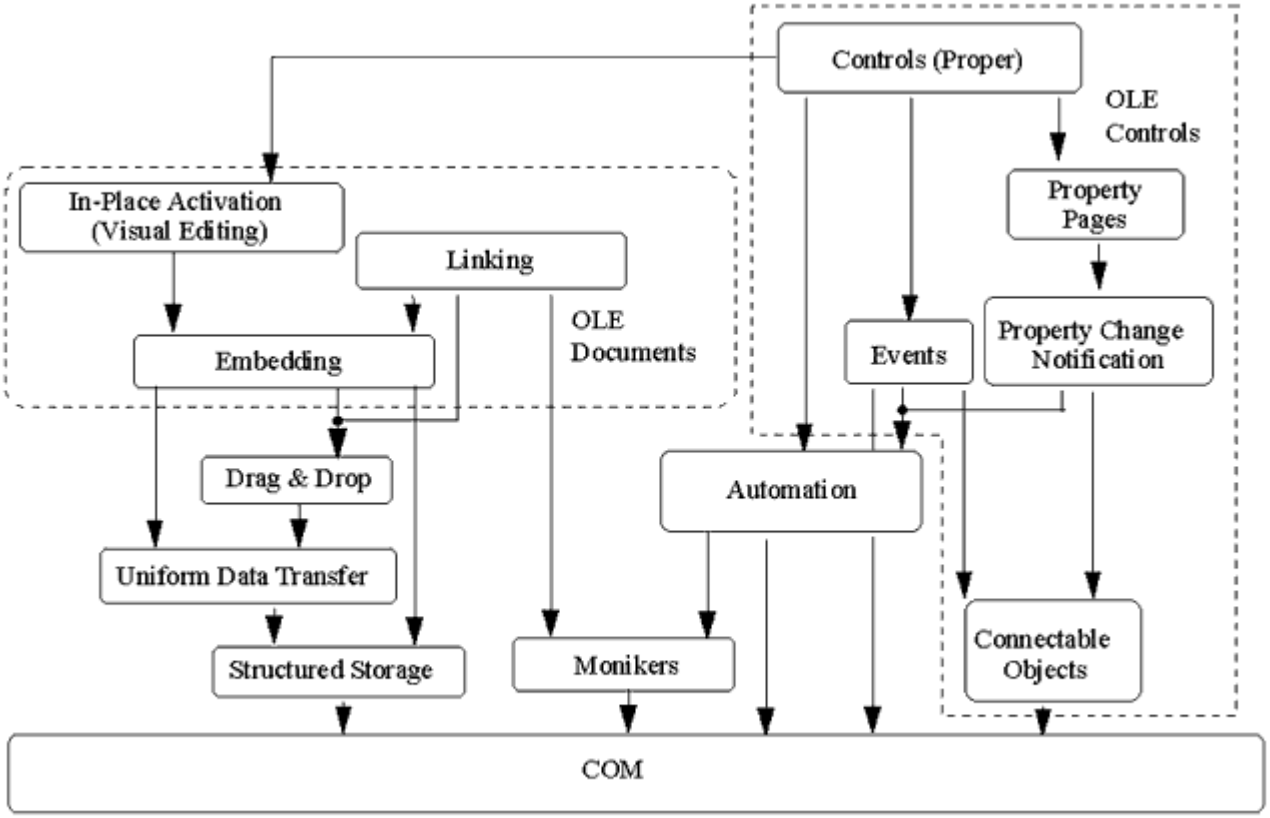
Purpose

- The Distributed Component Object Model (DCOM) is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner.
- DCOM is designed for use across multiple network transports, including Internet protocols such as HTTP.
- DCOM is based on the Open Software Foundation's DCE-RPC spec and will work with both Java applets and ActiveX components through its use of the Component Object Model (COM).
- Ex: Use Java to build a Web browser applet that calculates the value of a portfolio of securities, using DCOM to communicate stock values to the applet in real time over the Internet.

The Component Software Problem

- The most fundamental problem that COM solves is: How can a system be designed so that binary executables from different vendors, at different times, are able to interoperate? Solution lies in answers to these four questions:
  - Basic interoperability. How can developers create their own binary components, yet be assured that these binary components will interoperate with binary components built by different developers?
  - Versioning. How can one system component be upgraded without requiring all the system components to be upgraded?
  - Language independence. How can components written in different languages communicate?
  - Transparent cross-process interoperability. How can we give developers the flexibility to write components to run in-process or cross-process, and even cross-network, using one simple programming model?

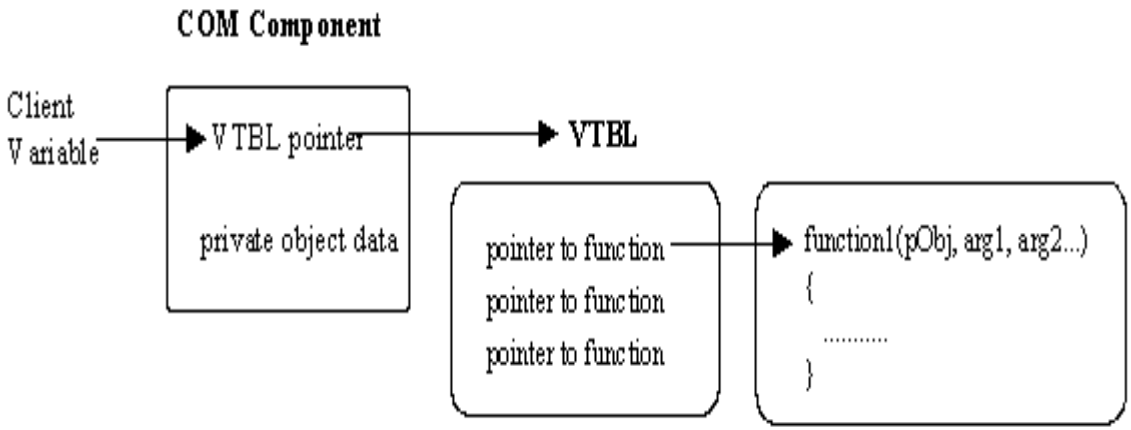
COM: Foundation for OLE



COM/DCOM Object

- Supports one or more interfaces
  - grouped collection of related methods
- A CLOCK example with 3 interfaces
  - iClock - setting & reading time
  - iAlarm - alarm methods
  - iTimer - stop watch methods
- COM objects communicate through Interface Pointers

Virtual Function Tables



Communication through Interfaces

- OLE's drag-and-drop support
  - All of the functionality that a component must implement to be a drop target is collected into the IDropTarget interface;
  - All the drag source functionality is in the IDragSource interface.

DCOM Specification

- Objects/Interfaces defined using MS IDL having unique identifiers
- Specifies a binary structure for the interface between the client and the object which compiler must produce for interoperability

Interfaces

- Logically immutable - cannot change/add/delete methods
  - avoids version incompatibility potential
  - inheritance used to enhance functionality

Client access to COM objects

- COM objects run inside a server
- Server can contain multiple COM objects
- Access Methods
  - in-process - client links to a library/ function call
  - Local Object Proxy - RPC to communicate with process on same machine
  - Remote Object Proxy - DCE RPC network communication
    - DCOM is mechanism that supports remote server access

Marshalling data between processes

- format and bundling data for sharing
- client proxy/ server stub
  - client calls server interfaces through the proxy
  - stub unmarshalls the parameters and makes call inside server
  - server stub replies with return values passed from proxy to client
- IDL file used to create proxy and stub
- Interfaces/API functions that expose OS services, naming, events, etc..

DCOM Reality

- Microsoft Proprietary
  - DCE-RPC when talking to platforms other than MS, otherwise MS-RPC
  - Dynamic RPC ports a problem for firewalls -- NT can restrict DCOM ports
  - Doesn't use DCE security options
    - On NT 4, the only authentication provided by default is NTLM, which uses RC4 for packet encryption. Key length varies from 40 bit for export to 128 bit for the North American version.
  - No shared memory functionality
  - Only NT-DFS

Corba and DCOM

- DCOM is a MS specific distribution solution. CORBA > 20 different vendors.
- CORBA products support Microsoft and non Microsoft operating systems.
- CORBA can bridge between Microsoft desk tops and UNIX servers.
- Distributed applications can be developed using both CORBA and DCOM.
- DCOM is well suited to front-end application development.

DCOM and Corba

- If entire distributed application runs under Microsoft platforms, DCOM might be a good choice.
- The OMG has defined a COM<->CORBA interworking specification which standardizes bridging.
- CORBA has existed since 1990. Commercial implementations have been available since 1992. DCOM was made available in beta form in 1996.

Similar Architectures but ...

- DCOM supports objects with multiple interfaces.
- Every CORBA interface inherits from CORBA::Object, the constructor of which implicitly performs common tasks as object registration, object reference generation, skeleton instantiation, etc.
- In DCOM, such tasks are either explicitly performed by server programs or handled dynamically by DCOM run-time system.

Similar Architectures but ...

- DCOM's wire protocol is strongly tied to RPC, but CORBA's is not.
- DCOM specification contains many details that CORBA considers to be implementation issues and thus CORBA does not specify
- http://www.cis.ohio-state.edu/~demirbas/CORBA/corba-dcom.html