Articles & White Papers

Microsoft framework provides a method for building components that interact in distributed applications

OLE/COM FUNDAMENTALS from a BUSINESS PERSPECTIVE

b y m a x d o l g i c e r

Although COM was defined by a single vendor, the payback for corporate developers and suppliers in using the specification can be huge. For example, COM supports multiple interfaces. As a result, developers can create custom interfaces based on real business needs. Another example: COM is a binary standard that supports binary interoperability. Using COM as the "glue" or middleware to hold the components together, discrete business components developed by different organizations can be united to create functionality and solutions.

Component architecture

Building enterprise-class systems for today and tomorrow presents challenges to both corporate developers and software providers. Today's business development environments demand an ability to quickly assemble highly flexible, distributed applications tailored to specific requirements. Meanwhile, suppliers are faced with the challenge of creating pieces of software that can be used by corporate developers. To ease some of the challenges, professionals are turning to object technology as a way to reduce the complexity of both designing and implementing systems. These objects must be packaged as components to allow for construction of "built-to-order" information systems from off-the-shelf components.

Components are standalone objects that can plug-and-play across process boundaries, networks, languages and operating systems. In distributed systems, components are units of work and distribution. The business value proposition of components is the reduction of application complexity, development cost and time to market. In addition, they can also provide improved maintenance, reusability, platform independence and distribution of functionality.

However, components alone cannot provide an infrastructure for multiple software components created by different vendors and corporate developers to interact, collaborate and integrate. This problem can be solved by extending the classical object approach with a standard component infrastructure, which must make it easier for components to be more autonomous, self-managing and collaborative.

Characteristics that such a distributed business component framework must possess include the following:

Component Interoperability Model -- The framework must provide an environment in which components built to a standard by either corporate developers or software suppliers can interact and collaborate to provide business solutions. These components must be able to be instantiated and accessed independent of the physical location in the distributed system.

Rapid Business Application Development -- The framework must provide the ability to quickly snap together custom business applications from a pool of reusable components.

Persistency and data transfer -- The framework must provide consistent mechanisms to store component object state and transfer data between components in a component-independent manner.

Interface repository -- A mechanism is needed for storing and accessing information about a component.

Reuse of functionality -- There must be a way to reuse components across lines of business applications.

Integration -- The components need to be accessed from 4GL environments where the capabilities of the objects are discovered at runtime and the actions on the object can be scripted by the 4GL languages.

Dynamic -- The components will interact in a dynamic and constantly changing environment. There must be a mechanism for deploying and versioning components and for maintaining them in the runtime environment.

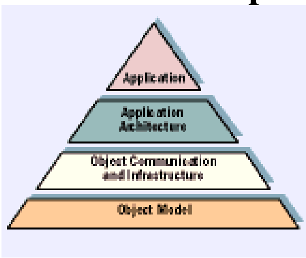
Enterprise computing

COM provides specification and packaging technology for building business components. In addition, it provides the basic component architecture that allows components to interact in a predictable manner. The architecture is sometimes referred to as the COM bus. Many businesses today use COM as a base foundation infrastructure for building new enterprise-class systems. The architecture, as well as the integration a COM-based implementation affords, is an attractive business value proposition. Business can also leverage the power of COM’s abstraction model, interface-driven design and reuse as a means of integrating existing code assets into the new applications.

Problems and solutions

The Component Object Model is the cornerstone of Microsoft technologies. The model provides the specifications and the key services needed to create component software that can benefit the business developer, the software supplier and, therefore, the computing industry. COM is unique because the architectural foundation is based on reusable design and reusable code. As a result, Microsoft can continue to bring out more and more COM and OLE technologies, such as ActiveX, OLE-DB and OLE Transactions (Viper), based on the basic architecture. COM can be described as an extensible systems object technology that can accommodate new and existing designs. Corporate developers and suppliers can also take advantage of the COM architecture to create exclusive business services and technologies. Thus, COM can provide the ultimate in extensibility to the enterprise.

The EnterpriseFramework Architecture



The business application architecture and the applications are layered on top of the COM infrastructure.

Fig. 1

Source: International Systems Group Inc.

Object purists are usually quick to point out that COM is not object-oriented but object-based. This is debatable, but there is no denying the essential business value provided by COM and its framework. COM provides the enabling technology for building and architecting the next generation of distributed and component-based applications. Leveraging the existing technical abstractions as well as expanding to business abstractions can help provide a framework for building enterprise-class applications. Thus using COM as a base technology provides an opportunity to create a business-driven framework that clearly reflects the functionality of a business.

COM provides a platform for architecting and constructing a business object framework. COM also provides specifications for building the components to operate with the bus and with each other. The payback for the corporate developer and software suppliers is an ability to create custom interfaces driven from business needs. Those business interfaces can be used to construct business components. The COM object-based framework is the "glue" or middleware that holds the components together to provide the business functionality. The business application architecture is layered. (See **Fig. 1**.)

Business components

Businesses are discovering that overall business assets are part of the definition of component interfaces, not necessarily the implementation. The interfaces and the interaction model can provide a higher level of reuse than that of implementation and can capture the essential business functionality. At a low level it certainly appears that implementation reuse can provide a value proposition to the enterprise. To a certain extent, this is true -- if objects are defined correctly and robustly inheriting the implementation from the base class does yield reuse benefits. Typically, implementation is misused to the point of making the class hierarchies quite fragile indeed. Complex inheritance hierarchies lead to unnatural and sometimes complex code dependencies between classes.

Accessing Visual Basic and C++COM Objects

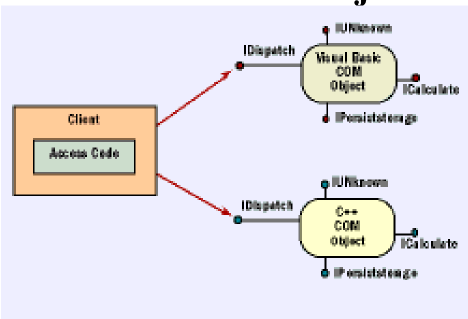


Fig. 2

COM has the ability to provide true binary compatibility between components; for example, the same code on a client can be used to access a COM object built using either Visual Basic or C++.

Source:International Systems Group Inc.

One of the greatest strengths of COM is its ability to provide true binary compatibility between components. Because there are no language dependencies between components, software suppliers and corporate developers are free to build COM objects in any language, which allows clients to access the functionality of the methods of an interface without regard to the language of the implementation. Therefore, COM at its most basic level is a binary standard for creating interoperable components. In order to adhere to the COM standard, an object must define the layout of its interface in strict accordance to the specifications.

Currently C++ is the language of choice for implementation because its vtable is laid out exactly like the COM vtable. However, the upcoming release of VisualBasic 5.0 will offer an ability to create COM objects in Visual Basic.

This capability will be a tremendous asset for corporate developers because it will allow the rapid development of COM components in a familiar programming environment. In addition, businesses can reap a late life-cycle performance tuning benefit because these components could be replaced by optimized C++ implementations later. (See **Fig. 2.**)

The interface is the thing

In a sense, the whole COM infrastructure can be viewed as an abstracted service set providing interfaces for technical services. The same paradigm can be extended to business applications by leveraging the power of COM's abstraction model to provide business services through the definition of custom interfaces. The model encourages the creation of business-driven interfaces that can be implemented by components that plug into the COM framework. These interfaces can be used to capture essential business qualities and functionality.

The interfaces are at the root of the COM's functionality. From a business perspective, a COM interface is viewed as a set of business related function calls, also known as methods or member functions. This interface serves as a contract between a client and the COM object. It serves to define the business functionality independent from implementation. COM clients interact with each other and with the COM system by calling interface member functions.

The interfaces that a COM object exposes describe the behavior or capabilities of the object. They are service interfaces used for a particular functionality. From an object-oriented (OO) perspective, interfaces can be considered "types." An object that supports an interface is said to be of that type. If a COM object implements the ICalculation interface, it can be referred to as a calculation object.

COM interfaces are also immutable. That is, once they are defined and published the interfaces cannot be changed because these interfaces are considered contracts. This can appear pretty restrictive. For example, what if a business functionality needs to be enhanced? How can the business functionality of an interface be extended if the interface cannot be changed? Think of the possible ramifications of such an activity. If the CalculateYield method on the ICalculation interface was changed, all clients that called that method would also have to be changed. This ripple effect of code changes cannot be allowed to occur inbusiness systems.

COM guarantees that existing functionality will not break by enforcing versioning at the interface level. In order to extend the functionality of an existing interface, a new interface must be created. To continue the above example, the enhanced calculation service interface ICalculation2 is created. This guarantees any clients that did not need the enhanced calculation functionality services would still continue to function as before, but new clients who were aware of the existence of the new interface could now leverage the new functionality offered by the interface.

COM interfaces are written in a high-level notational and descriptive language called the Microsoft Interface Definition Language (MIDL), a variation of Framingham, Mass.-based Open Group (OG) Distributed Computing Environment (DCE) IDL for describing business and technical interfaces. Microsoft's IDL allows interfaces to be specified independent of implementation and platform. The payback is that the same IDL can be used to implement interfaces on multiple platforms.

COM offers the ability to achieve a higher level of reuse than implementation, which is design. The designs are captured in the definition and specification of not only the Microsoft IDL's base interfaces, but also in any business-driven interface that can be created. The dynamics of the design are captured in the interaction between the objects that implement these interfaces. The compound documents portion of OLE is comprised of the compound document interfaces and the protocol that exists between them in order to carry out the functionality of the compound document specification.

This interface abstraction capability can be used to unify historically disparate interfaces. Microsoft is attempting to unify several different directory services, such as DCE CDS, Netware Directory Services (NDS) and NT Directory Services, through the ActiveX Directory Interface abstraction. The payback here is a uniform manipulation of directory service objects and schema in a heterogeneous environment through a suite of COM objects and interfaces. Another example is OLE-DB,which provides component-based access to data through a set of clearly defined interfaces.

Automation

Automation refers to the ability of a COM component to expose its behavior through a well-defined generic COM interface called IDispatch, which allows any application to expose its functionality, or methods, and its content, or indidvidual properties, to the Automation clients. Automation can deliver the programmability model to the programmer by decoupling interface dependencies between a COM client and the component itself. Automation allows COM objects to expose a programmable interface. This programmability allows a COM component to makes its services available to clients through a series of properties and methods. Automation provides the ultimate in interoperability on the Microsoft desktop. Companies that have standardized on the Microsoft desktop and Microsoft Office suite of productivity tools find that Automation gives the biggest bang for the buck. Microsoft is using Automation as the standard interaction with its component packaging technology, ActiveX, formerly known as OCX.

Wrapping new and existing business code as Automation interfaces allows corporations to push business functionality to the Microsoft desktop, providing an ability to access business functionality from applications like Microsoft Excel and Word. Today, business solutions can be created by the correct assembly of corporate components and applications. In addition, business developers reap additional benefits because all of the leading 4GL development environments support integration with these components. This allows the corporate software suppliers, the so-called component builder, a standard way to expose business functionality. Therefore, the component builder does not care which development tool the business developer or solution provider uses to snap together the business application.

The deep market penetration of Automation means that almost all development tools on the market today either provide or plan to provide some form of integration with Automation. The development environments that have the broadest market penetration are supporting Automation completely. The toolbars of development tools such as Visual Basic and PowerBuilder, natively provide access to the controls. Other 4GL vendors, such as Dynasty Technologies Inc., Lisle, Ill., recognize the strategic importance of Automation and promise to provide full support soon. Even Excel and Word use Automation natively for a scripting language. Now, corporate developers can add a tool like Excel to a programming toolbox by using its Automation interface. That capability allows users to utilize the functionality of Excel rather than writing custom code to increase productivity and reuse of corporate assets. Automation lets business developers use Excel to do their analysis graphically and provide charts of the results.

A component is only useful if there is a way to inquire about its capabilities at design time as well as at runtime. This information must be in a form that is useful to development environments as well as to component users. COM provides this capability through type libraries and help files. Basically, a type library contains all the information that completely describes a component and includes all the interfaces the component supports.

The description of the Automation interface is perhaps the most important and useful set of information from a business perspective. 4GL-based development environments such as Microsoft's Visual Basic, PowerBuilder from the Powersoft unit of Sybase Inc., Emeryville, Calif., and Delphi from Borland International, Scotts Valley, Calif., use such libraries to allow programmers to access to all the information in the development environment. This information defines the properties and methods of the Automation interface. All the methods including the parameter types and return values are described. A few lines of explanatory text is included to explain the purpose of a given method or property. More in depth information is provided through help files that ship with the component and are referenced from the type library. These 4GL environments also use the type libraries at runtime to provide the late binding to the specific methods that are representative of this scripting languages.

The business payoff

One of the most powerful features of a framework is its providing of an overall flow for application as well as business functionality by defining the basic workflow. A component-based framework takes it a step further through a set of well-defined interfaces that capture the processing flow. Components that implement these interfaces can plug into the framework to implement the desired functionality.

Both software suppliers and corporate component builders can leverage these frameworks as a vehicle for providing component suites. The suites are a set of semantically related components that work in concert to achieve more business processes, which allows solution builders to incorporate a significant amount of pre-canned functionality into new and existing custom business applications. Business can use the component suites as a path for legacy migration. Wrapping legacy applications and functionality in COM objects allows this functionality to be integrated into a new enterprise architecture and business process.

Corporate developers, suppliers and standards organizations are leveraging the technology and its interfaces to create new COM-based business component frameworks and vertical market solutions. The Agency Company Organization for Research and Development (ACORD) has created the OLIFE standard as a specification for building COM objects for the life insurance industry. One motivation for adopting COM technology for this standard was the widespread use of Microsoft Windows on the desktop coupled with a dramatic decrease in the cost of maintaining the standard by basing it on

COM.

Another example is the Health Level 7 (HL7) standards group, which provides a common data access model and communications structure so clinical data can be shared between health care information systems. The vendor supplied components used to build these systems insulate the underlying information system by exposing data and behavior through COM interfaces.

Portfolio risk analysis

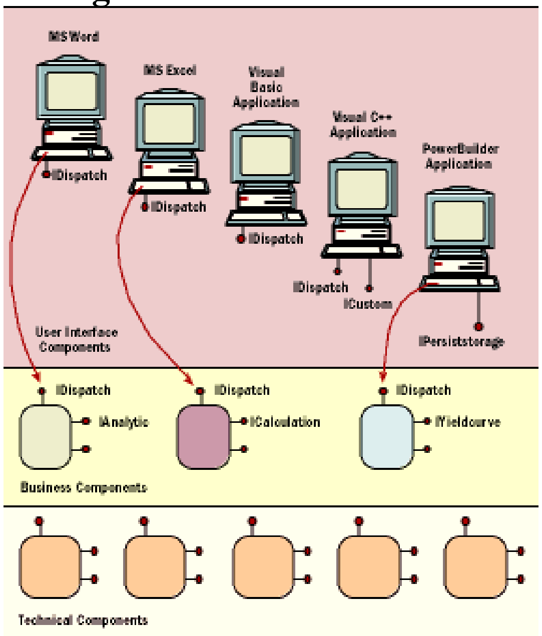
A typical business issue in the brokerage business is to find a way for traders and brokers to asses the risk of holding certain stocks in their portfolios. This process is referred to as Risk Management Analysis. Most brokerage firms have also standardized on the Microsoft desktop to help brokers manage portfolios and clients. Brokers and traders have become enamored with personal productivity tools like Microsoft Word and Excel. The subsequent business problem facing corporate developers is how to provide these users access to a corporate repository of analytic routines, yield curves and calculation engines necessary for executing risk analysis.

The Risk Management Architecture

In the risk management framework, the components that perform the actual business process implementation simply plug into the risk management "bus" and are viewed by the framework.

Fig. 3

Source: International Systems Group Inc.



Many firms have turned to COM and Automation as an answer to the risk analysis problem. By using a risk management analysis framework built on top of COM, they can provide users the functionality in an intuitive manner. The framework allows plug-and-play of various analytic, yield curve and calculation components into the overall business framework. Then the essential business process flow and the biggest corporate asset is captured in this framework. By using Automation, these capabilities can be extended to a plethora of development and productivity tools.

The essential business characteristics of the components are captured in both custom interfaces and Automation interfaces. The business process flow is captured in the protocol between them. The framework is then responsible for implementing the protocol by calling the correct methods on these components.

This separation allows corporate solution providers to focus on creating and delivering different analytic, yield curve and calculation components that meet new business needs. The new components are then transparently snapped into this framework, extending functionality with new analytic and yield curves.

The components that perform the actual business process implementation simply plug into the risk management "bus" and their methods are called by the framework. (See **Fig. 3**.)

This framework can also insulate the business developer from technical details such as language implementation and object communication and allow a focus on the business value proposition for the business -- the business application and their method.

This article answers the $64,000 question: How is COM essential to the construction of Enterprise Business systems? As shown above, COM accomplishes this by:

- allowing the creation of object-based systems from discrete components that adhere to a strict binary standard and provide interoperability at the interface level;
- providing a framework upon which to build these distributed business systems from components;
- providing extensibility by allowing new interfaces and components to be added in a plug-and-play manner;
- providing access to enterprise services through custom interfaces and components that implement these interfaces; and

- providing integration with other vendor products and business solutions that are built to the OLE/COM standard.

 1 Apple Hill, Suite 301, Natick, MA 01760
(508) 652-1100 FAX (508) 652-1220 e-mail:general@spgnet.com http://www.spgnet.com