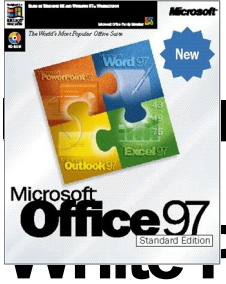

Office 97

White Paper

Published: October 1996

For the latest information, see http://www.microsoft.com/office/

Microsoft Component Strategy

The use of component software and the promised benefits they will provide end users and developers is one of the hottest topics in the software industry today. However, discussing a company's component strategy is complicated by the multitude of definitions of what components are. To date, components have been defined and discussed by industry publications, developers and customers in a number of ways, including the following:

- Microsoft Foundation Classes (MFC) libraries and custom dynamic-link libraries (DLLs) that provide prewritten code used to build applications
- Controls used by developers to build applications (e.g., WordArt is a component of Word)
- Specific-purpose "plug-in" pieces of code that work within a browser (e.g., a Java Applet that calculates a user's interest rate on a car loan)
- Documents that can plug into a container, such as a Microsoft® Excel spreadsheet within the Office Binder
- "Light" developer versions of applications that could be used in the context of a container such as Lotus Notes®.

A Consistent Definition

As you can see, there are numerous kinds of components, but fundamentally all are reusable pieces of code that accomplish a focused task. And, in the most popular usage, the components must work within the context of a container. These components range from those that are very well integrated within an application, such as the Office DLL, to those that are less integrated but more "pluggable" within applications, such as OLE servers. The following figure shows how some different types of components might fall along this spectrum.

Pluggability

- OLE Servers
- Java Applets
- ActiveX
- Controls, VBX

Integration

- Add-ins
- Visual Basic®, Applications Edition
- Graphics Filters
- Converters
- Spelling and Grammar
- DLLs
- C++ Classes
- Office Art
- Office Assistant
- Memory Manager

Microsoft Approach

Office applications use a number of components across this spectrum. Not only is Office built with integrated components such as an Office DLL or OLE servers (e.g., the shared spelling and grammar checker), but Office also acts as a container for pluggable components such as ActiveX™ Controls. Microsoft's component strategy is as follows:

- To develop Office using a number of components while maintaining a high level of consistency and integration throughout the products
- To distribute Office as a set of consistent, well-integrated tools for the end user rather than as a number of loosely integrated pieces
- To provide flexible options that allow users and IS managers to determine which components of Office reside on a user's desktop and which reside on the server
- To provide for customization of Office by enabling Microsoft and third-party components to plug into Office documents or solutions using Visual Basic for Applications®
- To allow Office documents to act as components themselves within the Microsoft Internet Explorer container

This document is designed to guide you through Microsoft's approach to building with components, distributing Office as a set of integrated tools, and providing an open forum for third party and Microsoft plug-ins.

Building Office with Components

Microsoft has been using components to develop applications for some time. File Open, WordArt, Equation Editor and Mapping are all examples of this strategy. In fact, over 50 percent of Office 97 has been built using either components or shared code.

For example, Office has a variety of reusable components, functions and DLLs that are used by programmers within Microsoft but are not necessarily documented or available to outside programmers. In addition to these integrated, proprietary pieces, Office uses components that are more pluggable, such as the spelling and grammar checker and Office Art, to build applications. This strategy not only supplies Office users with consistent functionality across products, it also speeds application development.

Selected Office 97 Components

Web FindFast	NEW	Office Runtime	BETTER
Binder	BETTER	AutoCorrect	BETTER
ClipArt Gallery	BETTER	OLE Props, Notes/FX	BETTER
Graphics Filters	BETTER	Command Bars	NEW
Converters (HTML)	BETTER	Answer Wizard	NEW
Setup/Admin	BETTER	Office Art	NEW
Chart/Graph	BETTER	Data Access Objects	BETTER
Query	BETTER	VBA	NEW
File Management	BETTER	Hyperlinks	NEW
ODMA	NEW	Grammar	BETTER

Although building with components provides consistency for users and faster development schedules, there are certain trade-offs. For example, speed of task execution might sometimes be sacrificed due to the overhead involved when a component must communicate with the container. In addition, components designed to work with multiple containers run the risk of being too generic to be useful. Microsoft developers weigh these trade-offs and decide whether using components or native code is the most appropriate, ensuring that speed of development does not supersede execution speed or usability.

In summary, components are integrated within Office applications when they help meet user needs for powerful, consistent, easy-to-use software.

The key here is that components are a means to an end and not an end in themselves — it is irrelevant to the user whether the functionality is built with a component or is native code. Users simply want the functionality and the ability to work efficiently.

Distributing Office as an Integrated Suite of Applications

Although Microsoft develops Office using a mix of components and integrated code, it will continue to distribute this collection as an integrated suite of applications. This decision is based on requests from customers for a single set of easy-to-deploy, integrated applications. IS managers have indicated that purchasing, deploying and managing numerous inconsistent applications on users' desktops would be exceedingly difficult and complex.

Although there has been much industry discussion about whether a collection of applets could replace the need for a suite of business productivity tools, today's components offer little functionality compared to the full Office suite. This limited functionality would make it difficult for users engaged in document creation to complete their jobs. In addition, the applets would need to evolve to provide the richness of functionality Office already has today, including support for the following customer priorities:

- **Integration.** Users and IS managers continually cite integration as one of the key reasons for using a suite. Customers want their applications to look and act alike, and this is best accomplished with an integrated development strategy. Using multiple components from multiple vendors makes it harder to integrate and simplify application usage.
- **Scalable features.** Customers may not want all their tools all the time, but they want access to them when needed. For example, 47 percent of Microsoft customers (including even the least experienced users) do a complete or custom install of Office rather than a typical install. In fact, only 3 percent choose minimum install. Customers want tools to be available when they need them, so they are willing to allocate the necessary hard drive space for the added convenience.
- **A unified development platform.** Office provides a unified platform for creating custom solutions. The consistency in the object model, the programming interface and the easy-to-use development environment has made Office one of the most popular development platforms, after the Windows® operating system.
- **Legacy file support.** Many customers find it essential for Microsoft to support their existing macros, solutions, archived documents, etc. In fact, this was the No. 1 concern Microsoft found in a recent upgrading customer study.

Although Office will not be distributed as a number of loosely integrated pieces, Microsoft is working to provide maximum deployment, installation and management flexibility for IS managers. Currently, administrators can determine which parts of the applications to install on users' machines, providing unique feature sets to different populations of users. In addition, they can make the appropriate trade-offs between network bandwidth and local storage by specifying that part of an application will run on the desktop, while the rest of the tools are run from a network server. For example, an administrator can specify that 90 percent of Office run from the server.

In addition, Microsoft is working to find ways in future releases to load smaller working sets onto a user's machine. Already, Office 97 does a great deal to load features "on demand" rather than loading them all at once. For example, the first time you use a graph in a PowerPoint® presentation graphics program session, you experience a slight delay because that code was not unnecessarily loaded when the system was booted.

Today's "packaged" components provide little functionality compared to Office. Although they are generally about one-third to one-half the size (RAM + disk footprint) of their corresponding applications, the spreadsheet or word processing components cannot stand alone — they still need a container. For components such as those in Lotus Notes, this creates a cost in both RAM and disk space — in many cases, this cost can be substantially more than that in Office.

Adding Components to the Office Container

Microsoft has also worked to ensure that Office can act as a container for additional components to help users customize their office. Office documents can now host ActiveX Controls. For example, the PowerPoint Animation Publisher is a control that can be plugged into PowerPoint to allow users to create multimedia presentations for the Web. In addition, a second Office component — the PowerPoint Animation Player — can be plugged into Netscape™ Navigator or Microsoft Internet Explorer to allow users without PowerPoint to view the presentation. Microsoft will continue to produce new components for Office and encourage third-party vendors to build Office plug-ins as well.

In Summary

Developing Office with shared components ensures not only speed of development, but true integration and consistency among the applications. Distributing Office as an integrated suite with flexible installation options provides for easier management and distribution on user desktops. Finally, allowing components to plug into Office provides an easy way for administrators to further customize Office for the needs of end users — providing them an easy-to-use tool that meets their individual requirements.

#########