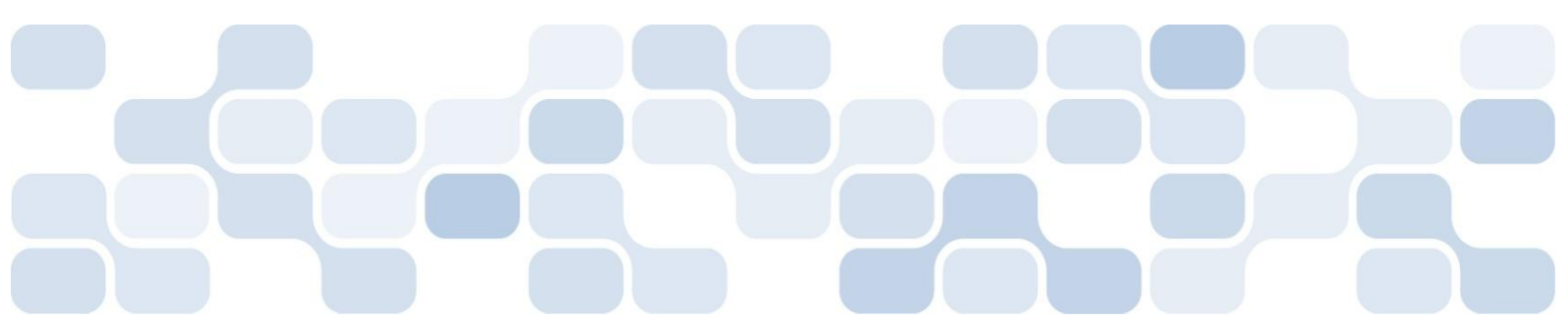




**Business Planning Guide
for Visual Basic 6.0 Applications**
White Paper

April 2008

For the latest information, please see www.microsoft.com/vstudio



The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, IntelliSense, Microsoft Dynamics, SharePoint, Visual Basic, Visual Studio, Windows, Windows Server and Windows Vista are trademarks of the Microsoft group of companies

All other trademarks are property of their respective owners.

Introduction.....	1
Background.....	2
Strategies for Supporting Visual Basic 6.0 Applications	8
Support-Strategy Decision Framework.....	15
Resources for Project Teams.....	20
Summary of Support for Versions of Visual Basic	26
Conclusion	27

INTRODUCTION

This white paper is intended to help business decision makers, chief information officers, chief technical officers, chief executive officers, and project sponsors plan for the future support and direction of bespoke systems that are based on the Microsoft Visual Basic 6.0 platform.

The Visual Basic 6.0 platform was released in 1998, and has been the foundation for millions of custom applications. Now, after four subsequent releases of Visual Basic, Microsoft has finished putting in place a range of support, migration, and replacement options for Visual Basic 6.0 applications, to give business decision makers and project teams a complete set of strategies for moving applications forward, based on individual business requirements, capabilities, and cost drivers.

This white paper is divided into three major areas:

1. Background and history of Visual Basic, including a benefit-level comparison of Visual Basic 6.0 and Visual Basic .NET for readers who have limited exposure to .NET
2. Strategies and guidance for when to choose what strategy, to help businesses plan for the future road maps of their Visual Basic 6.0 applications
3. Resources for teams who use Visual Basic 6.0 applications

BACKGROUND

Since the launch of Visual Basic 1.0 in 1991, Visual Basic has grown to become the world's most popular programming-platform language—growing in capability over almost two decades and 10 releases. More than six million developers worldwide use Visual Basic to create custom applications that drive global business forward.

Versions 1.0 through 6.0 of Visual Basic focused on enabling developers to create Microsoft Windows applications and COM infrastructures quicker and easier than on any other development platform. In 2002, Microsoft released Visual Basic 2002 based on the .NET platform, enabling a new generation of capabilities: Windows applications and services, mobile and device applications, scalable Internet and intranet applications, and XML Web service-enabled applications. The .NET platform also gave Visual Basic the ability to better meet the demands of the enterprise software-development life cycle than the previous COM-based versions—providing first-class support for accessibility, object-oriented development, interoperability, connectivity, standards compliance, domain modeling, and automated test capabilities.

Microsoft has released four versions of Visual Basic that are based on the .NET platform and are referred to often as *Visual Basic .NET*, steadily adding productivity capabilities to Visual Basic. Microsoft released the current version of Visual Basic 2008 in November 2007.

Every programming language grows and changes, according to customer requirements and industry drivers. Visual Basic 2002 through Visual Basic 2008 introduced more enhancements than every previous version of Visual Basic combined. However, the huge leap forward that was required to meet customer demands also had the result of breaking code compatibility with Visual Basic 6.0. Because applications that are written in Visual Basic 6.0 cannot just be recompiled in Visual Basic 2008, and because new Visual Basic 2008 capabilities create compelling new opportunities, many organizations approach the retirement of Visual Basic 6.0 applications with more planning than just moving them to the next version of the platform.

From the launch of Visual Basic version 2002 to the launch of Visual Basic version 2008, Microsoft has steadily added software tools, training services, and technical support to give organizations the most flexibility for moving applications, operations, and project teams forward from Visual Basic 6.0. This commitment to helping the Visual Basic community move forward from Visual Basic 6.0 has resulted in the widest support offering, and signifies the commitment of Microsoft to the future growth of Visual Basic as the premier tool for rapid application development.

With the release of Visual Basic 2008, Microsoft is encouraging organizations to take advantage of these options and plan for the replacement of Visual Basic 6.0 applications—taking advantage of the new capabilities and opportunities that Visual Basic 2008 gives the developer community.

What Does Breaking Compatibility Mean?

Visual Basic 2002 through Visual Basic 2008 break compatibility with Visual Basic 6.0. What exactly is the effect of this? For Visual Basic, it means two things:

4. Except for the simplest of projects, an application that is developed in Visual Basic 6.0 cannot just be recompiled within a later version of Visual Basic without some rework.
5. With the new capabilities of the .NET platform, Visual Basic applications are capable of meeting more business requirements than ever before. Many businesses use the opportunity of moving applications from Visual Basic 6.0 to Visual Basic 2008 as the basis for rethinking and expanding their applications to add more business value.

Why Did Visual Basic 2002-Visual Basic 2008 Break Compatibility with Visual Basic 6.0?

While it might seem counterintuitive, compatibility was broken because customers demanded functionality that Visual Basic 6.0 just could not deliver. Several key requirements from customers made breaking compatibility the right choice for Visual Basic and its community:

Internet adoption—Visual Basic 6.0 was designed for desktop Windows applications that are used by a small number of simultaneous users in a single, physical location. The rapid adoption of the Internet by global businesses—together with an immediate need for cost-effective, scalable, and reliable Web-based applications that are accessible by tens of thousands of simultaneous users—meant a shift from the stateful, COM-based architecture of Visual Basic 6.0 to a more scalable, secure, and fault-tolerant platform.

Demand for interoperability—Visual Basic 6.0 was not designed to interoperate with other languages; its proprietary data formats had to be reformed to meet industry standards that enabled greater business connectivity via the Internet. Organizations began to look towards system interoperability with partners and vendors as essential to growth. This meant that application designs had to embrace standards-based "loose coupling": the ability to interoperate with other systems, based on an interoperability contract, and irrespective of operating system, programming language, or location.

Need for accessibility and security—As business software applications have grown in strategic importance, the software industry has matured and placed greater emphasis on meeting accessibility regulations, localization requirements, and trustworthy computing. These capabilities required a shift in platform architecture that could not just be replumbed into Visual Basic 6.0.

Feature requests—A number of long-overdue customer requests, such as inheritance and object-oriented programming support, structured exception handling, advanced graphics support, better versioning capabilities, robust deployment mechanisms, and an end to "DLL hell" required Microsoft to make essential changes to the architecture of the Visual Basic development system.

Because of its rearchitecting for the .NET platform, Visual Basic is reenergized for development of future applications for business. The language combines its natural readability and ease of use with the full power and features of the .NET platform. As the language moves forward, Microsoft keeps the future versions of Visual Basic faithful to the original Visual Basic goal: prioritizing business productivity and rapid application development.

Since Visual Basic 2002, Microsoft has added new capabilities to each subsequent version—ensuring that learning and skills from development in earlier versions of Visual Basic can be reapplied and can act as a foundation for building applications of the future.

Microsoft is committed to growing and enhancing Visual Basic continually, supporting and creating new opportunities for companies and developers who use Visual Basic to build and support critical applications. When Visual Basic 2002 broke compatibility with Visual Basic 6.0, Microsoft signaled that its commitment to the future of Visual Basic was stronger than ever.

What Changed from Visual Basic 6.0 to Visual Basic 2002 Through Visual Basic 2008?

When looking at the changes from Visual Basic 6.0 to later versions, it is important to understand the differences between Visual Basic 6.0 and Visual Basic 2008 (the current shipping version).

There are two categories of changes from Visual Basic 6.0 to Visual Basic 2008: new capabilities for developing applications, and breaking changes.

New Capabilities

The limited Web programming solution of Visual Basic 6.0, called "webclasses," had limits in terms of scalability, security, and designer support. Visual Basic 2008 and ASP.NET far exceed Visual Basic 6.0 in the rapid application development of applications with richer capabilities, higher performance, and better scalability.

Visual Basic 2008 extends the reach of development from Windows applications to Windows services, mobile devices, console applications, and 32-bit and 64-bit applications.

Visual Basic has substantially improved flexible and scalable connectivity, compared to the proprietary APIs of Visual Basic 6.0, with the ability to interoperate with applications and services that are written, not only with Microsoft technologies such as C# and Visual Basic 6.0, but also with non-

Microsoft platforms such as Java on Linux. While Visual Basic 6.0 had solid integration with Microsoft Office client products, Visual Basic 2008 extends these capabilities much further, with support not only for Microsoft Office client products, but also for Microsoft Office server products, Microsoft SQL Server, the Microsoft Office SharePoint family of products, Microsoft CRM, and almost every enterprise product that is produced by Microsoft.

The security features of Visual Basic 6.0 were limited to what application developers built themselves. Visual Basic 2008 introduces ready-to-use, robust security features that have support for trust levels, role-based security, encryption, and multiple mechanisms for authentication.

Visual Basic 6.0 had limited support for object-oriented development, and delivered only a basic set of reusable library components to aid developers in building applications. Visual Basic 2008 has full object-oriented capabilities, standards support, and access to the complete .NET framework of library components for building enterprise-ready, modern applications quickly, securely, and with a greater level of robustness.

Breaking Changes

Visual Basic 6.0 had a rich Windows forms development environment. Visual Basic 2008 also has a rich Windows forms development environment, influenced by the Visual Basic 6.0 forms environment, but it is not totally backwards-compatible. The data-binding model in particular has been improved to create a flexible and robust binding model that is based on ADO.NET, with better abilities for accessing data on servers and in remote, disconnected locations. Visual Basic 6.0 RDO and DAO data-binding was optimized for single-user same-machine database access, while ADODB was designed for tightly coupled database server access within a local-area network. Visual Basic 2008 and ADO.NET are optimized for multiuser, client-server access via the Internet or across wide-area networks.

Visual Basic 6.0 had its own proprietary model for variables and types. Visual Basic 2008 applies more rigidity on types and variable conversions, to better prevent runtime issues and better enable interoperability with other languages.

A Visual Basic 2008 application, like any .NET-based application, is "managed"—meaning that it is monitored by the .NET common language runtime, with garbage collection, version control, security infrastructure, and greater error-prevention capabilities than an "unmanaged" Visual Basic 6.0 Windows application. Visual Basic 2008 applications have subtle behavior differences from Visual Basic 6.0; the performance profile is different, runtime-eventing models are different, and objects are created and destroyed with a different life cycle.

What Stays the Same?

Every version of Visual Basic is designed for productivity. Most Visual Basic 6.0 developers adjust quickly and thrive in Visual Basic 2008; it has many of the same rapid application-development features as Visual Basic 6.0, such as edit-and-continue, IntelliSense, case-insensitivity, and so on. In addition, Visual Basic 2008 introduces new productivity features, such as background compilation and error reporting as-you-type, code templates, and My* classes for easy access to machine resources.

Programming and business logic is very similar. Most business logic in Visual Basic 6.0 can be used in Visual Basic 2008, with only minor changes. Visual Basic 2008 includes upgrade tools to help move code forward, and compatibility classes that give .NET-platform library classes a familiar "VB6" interface.

Is Visual Basic 2008 a Good Choice for Future Developments?

Before Microsoft released the .NET platform, every different application type relied on a different programming paradigm: VBScript for Web applications; Visual J++, Visual C++, and Visual Basic for Windows applications; Transact SQL for SQL database programming; and VBA for Microsoft Office customizations. Every different product shipped with its own proprietary customization toolkit. Each of these toolkits had proprietary library components, variable types, and development environment. Project teams would build specialist knowledge, working on one solution that could not be reused easily on another solution type. With the introduction and rapid adoption of the .NET platform and the Visual Studio .NET development environment, Microsoft standardized the customization model for every product that was to be built on the .NET platform. For project teams this means better transfer of skills and more consistency when creating different solution types.

The two first-class .NET languages are Visual Basic and Visual C#, and both can be used to create stand-alone applications or solutions for Microsoft SQL Server, Windows SharePoint Server, Microsoft Office Server System, Microsoft Office clients, Microsoft CRM, and more. The .NET platform has become an accepted standard for developing applications and integrating them across the enterprise.

Both Visual C# and Visual Basic are natural choices for creating enterprise solutions. Both languages are built on the .NET platform; support the same variables and types; use the rich programming components of the .NET framework; and have similar performance, scalability, and interoperability capabilities. Choosing between Visual C# and Visual Basic becomes a matter of personal preference; project teams that have prior experience in Visual Basic or VBA are more productive with Visual Basic; project teams that have prior experience in C++ or Java are more productive in Visual C#. Visual Basic 2008 has more features for rapid application development,

whereas Visual C# 2008 has more features for fine-tuning code. Both languages fully support Microsoft Team System testing, source control, and visual-design tools, and Microsoft is committed to growing both languages in future versions of Visual Studio and the .NET platform.

For most Visual Basic developers, Visual Basic 2008 is the natural choice for future development; it supports a familiar language, development-environment capabilities, and easy-to-learn productivity features that made previous versions of Visual Basic incredibly popular for creating business applications.

Visual Basic versions 1.0 through 6.0 were sometimes seen as being easy to learn and get started with, but lacking in high-end capabilities; this was sometimes called the "VB glass ceiling." Visual Basic 2008 has no such limitations, and enterprises such as Monsanto and the Washington State Department of Licensing have built their custom business applications on Visual Basic and the .NET platform. Visual Basic 2008 breaks through the Visual Basic 6.0 glass ceiling with an enterprise-ready architecture that is based on the .NET platform—designed from the ground up, to support the requirements of the world's largest businesses.

Third-party support for the .NET platform is enormous. There is a huge community of component vendors that are marketing specialized components to support application development. Almost all of these components support both Visual Basic and Visual C#. Developers and project teams have embraced the .NET platform—finding it more productive and powerful than any other programming platform. There are vendors who specialize in creating new Visual Basic 2008 software and moving existing Visual Basic 6.0 applications forward to Visual Basic 2008.

For organizations that have an existing investment in Visual Basic 6.0 applications, Visual Basic 2008 is an even more compelling choice. Microsoft has training, tools, and interoperability support for moving forward to Visual Basic 2008, and interoperating between applications that are written in Visual Basic 6.0 and Visual Basic 2008.

Is Visual Basic 2008 a good choice for future developments? Absolutely. Visual Basic 2008 is designed for Visual Basic programmers and enterprise-class development.

The plans of Microsoft for the future of Visual Basic center on continually improving productivity, adding rapid application-development capabilities, and ensuring that Visual Basic fully supports and utilizes the power of the .NET platform. Breaking compatibility with Visual Basic 6.0 helped future-proof Visual Basic; the .NET-platform architecture means that features such as 64-bit processor support could be added easily to Visual Basic 2008 without breaking compatibility again. Microsoft is committed to growing Visual Basic in future versions, ensuring a powerful programming environment with the shortest learning curve and quickest time-to-value.

STRATEGIES FOR SUPPORTING VISUAL BASIC 6.0 APPLICATIONS

This section covers the available strategies for existing investments in Visual Basic 6.0 applications, and includes a framework for deciding on the strategy to employ.

Assessment

Before looking at strategies for supporting Visual Basic 6.0 applications, the first step is to perform an assessment of your existing applications.

- How many Visual Basic 6.0 applications are deployed in your organization?
- What is the state of each? Is it in production, or has it been retired?
- How mission-critical is each application?
- Does each application still meet the requirements of the business?
- For each application, are there people in place who know how to support, change, and enhance it? Is its technical documentation up-to-date?

After the assessment, prioritize the order in which your organization will look at each application, and use the following guide to form a strategy for each.

Strategies

There are four different strategies for Visual Basic 6.0 applications:

- Leave
- Interop
- Migrate
- Replace

Leave

What it means Leave the application in Visual Basic 6.0, and keep supporting it.

Useful when The application is mature and does not need frequent updates. Updating the application means finding appropriate resources and continuing to invest in the application when another strategy might be more appropriate.

While this is a cost-effective strategy, leaving an application in Visual Basic 6.0 must be seen as limited in long-term applicability; businesses change and generate new requirements for existing applications, but Visual Basic 6.0 might be incapable of meeting many of the new business requirements that, in contrast, come standard in the .NET platform, such as accessibility and trust-level security.

The critical responsibility with leaving an application in Visual Basic 6.0 is ensuring that the application is kept operational. There are three important parts to this:

Development expertise—An organization must ensure availability of a project team that can support the application, apply critical fixes, and make ongoing changes to the application.

Runtime environment—As organizations roll out Windows Vista and future operating systems to their desktops, they must ensure that Visual Basic 6.0 applications will continue to function in the new operating systems. Microsoft has announced runtime support for Visual Basic 6.0 applications until 2017, meaning operating systems—including Windows Vista and Windows 2008—will continue to support the running of Visual Basic 6.0 applications. This applies to the Visual Basic runtime and components; for applications that employ third-party components, an organization must check with the vendors of these components to ensure that they will continue to be supported.

Development environment—For enhancements to Visual Basic 6.0 applications, developers use the Visual Basic 6.0 integrated development environment (IDE) to change and recompile the application. As with the runtime environment, organizations must ensure that they continue to provide workstations that support the Visual Basic 6.0 IDE. Microsoft has announced Visual Basic 6.0 IDE support for Windows Vista, and extended support finishes in April 2008.

In practical terms, this means that project teams should continue to use Windows Vista or Windows XP for making changes to Visual Basic 6.0 applications. As with runtime support, organizations must check with third-party component vendors to ensure that any third-party components also are supported.

Leaving an application in Visual Basic 6.0 often is seen as an interim step, suitable for mature applications that have limited lifetime expectancy. Ultimately, organizations will have to choose between retiring Visual Basic 6.0 applications and moving them to the current version of Visual Basic, or using virtualization technologies to keep the application running beyond the support life cycles of the operating system and other software upon which the application depends. Leaving an application in Visual Basic 6.0 for the short term allows an organization to balance the investment in moving applications forward with availability of funding and resources.

Interop

What it means Interop (short for *interoperability*) involves making enhancements to a Visual Basic 6.0 application by using Visual Basic 2008, or gradually replacing Visual Basic 6.0 components with Visual Basic 2008 components. The result is a gradual rewrite with a long period in which the organization supports a hybrid Visual Basic 6.0/Visual Basic 2008 application.

Useful when Organizations have a Visual Basic 6.0 application that undergoes regular modification, meaning that the organization cannot afford to "freeze" the application for the amount of time that a full upgrade or rewrite will require.

Visual Basic 2008 has a number of capabilities that make interop very compelling: the ability to reference Visual Basic 6.0 business objects from Visual Basic 2008; the ability to perform debugging from Visual Basic 2008 into Visual Basic 6.0; and the Interop Forms toolkit, which enables Visual Basic 2008 forms and controls to be used from within Visual Basic 6.0 applications.

This strategy can be cost-effective, because an organization can add value immediately to an existing application with new Visual Basic 2008 capabilities. The strategy is most useful when the existing Visual Basic 6.0 application is well-structured and written in a componentized architecture—enabling new functionality to be added and old components to be replaced easily.

As with leaving an application in Visual Basic 6.0, interop should be seen as an interim step, because the components of the application that are kept in Visual Basic 6.0 still face the same support issues as leaving the entire application in Visual Basic 6.0. Interop is most used for extending the lifetime of existing applications temporarily or moving the components of an application gradually to Visual Basic 2008.

Migrate

What it means Visual Studio 2008 automated upgrade tools are used to migrate a Visual Basic 6.0 application forward to Visual Basic 2008.

Useful when The application is written in a componentized architecture that can be migrated forward piece by piece, and the application/business is stable enough that the development team does not have to deliver frequent changes to the application.

Microsoft has released a number of utilities to help organizations automatically migrate a Visual Basic 6.0 application forward to Visual Basic 2008. The upgrade is usually a three-part process:

1. Prepare the application in Visual Basic 6.0.
2. Upgrade by using the automated tool.
3. Perform some rework to finish moving the application forward.

The quality of the migrated application depends on the quality and structure of the original Visual Basic 6.0 application. Middle-tier components with business logic usually migrate easier than forms-based applications that rely on user-interface tweaks and unique characteristics of COM. Some elements of a Visual Basic 6.0 application, such as RDO and DAO data-binding and non-zero-based array logic, cannot be migrated automatically. Because Visual Basic 2008 applications utilize a different type of data-binding and different standards for variable types, this code usually has to be reworked.

The upgrade tool can be used to migrate an application gradually—starting with the business logic, and finishing with the user-interface layer. Many organizations find this approach useful for applications that are built in a componentized architecture.

One consideration with migration is that a poorly maintained application, or one in which the architecture is outdated, will result in a migrated application that has the same limitations.

Another consideration is that the migrated application still might have dependencies on COM components, which might have to be replaced over time as the

application is maintained.

It is difficult to predict ahead of time what percentage of the application can be upgraded automatically and what percentage will need rework. Many organizations conduct a feasibility study by using the automated assessment tools of Microsoft, combined with a trial migration to prepare a better estimate. Some organizations also might choose to migrate portions of the application, allowing them to move applications forward gradually, and using interop between the Visual Basic 6.0 components and migrated Visual Basic 2008 components to deliver a hybrid application to users.

In addition to the Microsoft migration tools, some organizations might choose to hire an outside consultant to run a migration. Consultants who have experience with migrating Visual Basic 6.0 applications often can handle a migration much more quickly, because they are intimately familiar with the ins and outs of the process.

There are also a number of small independent software vendors that market products that can automate migration. These vendors often migrate only the Visual Basic 6.0 syntax and rely on interoperability libraries to adapt the .NET Framework to the structure of the old Visual Basic 6.0 interfaces. This can be a very attractive option, but it should be balanced with the reality that those interoperability libraries might not be compatible with future versions of the .NET Framework.

Replace

What it means Replace the Visual Basic 6.0 application with an off-the-shelf solution or by rewriting in Visual Basic 2008.

Useful when The Visual Basic 6.0 application is no longer suitable to meet business requirements.

When a Visual Basic 6.0 application has reached its end-of-life—meaning that it no longer meets business requirements, or the once unique functionality that is required is now offered by independent applications vendors—there is an opportunity to retire and replace the application, instead of rebuilding or migrating it.

Replacement means either buying an off-the-shelf prebuilt replacement application or developing a new custom application.

When available, replacing with an off-the-shelf product is usually very cost-effective, whereas developing a new application is usually the most expensive option, but one that allows the business to restate its requirements and commission an application that creates the best fit moving forward.

Build-versus-buy is always a complex decision. However, keep in mind that many Visual Basic 6.0 applications were built to serve financial, accounting, HR, and other back-office needs that, except for the largest of enterprises, were not well-served by the software industry in the 1990s and early 2000s, when Visual Basic versions 1.0 through 6.0 were widely used. Today, many enterprise application vendors offer flexible, customizable, and powerful applications that can more than replace the functionality of custom-built applications. Many of these vendors, including the Microsoft Dynamics line, also offer subscription-based, on-demand services—hosting and managing the applications for you.

SUPPORT-STRATEGY DECISION FRAMEWORK

When you plan the support strategy for a Visual Basic 6.0 application, you should consider six common drivers. You can use the Visual Basic 6.0 application's applicability against each driver to help reach a decision on the appropriate strategy, or use it as a basis for discussion with the project team to better drive understanding within the team.

User requirements	
<p>Does the existing solution still meet user requirements?</p> <ul style="list-style-type: none"> • Explicit feature requirements • Implicit feature requirements: security, interoperability, deployment, accessibility, usability, dependability, and performance • Can the solution meet the growth requirements of the business? • Scalability • Geo-location <p>Solutions that still meet most requirements are candidates for leaving in Visual Basic 6.0, migrating, or extending with interop. Solutions that have fallen behind in the meeting of requirements are stronger candidates for replacement.</p>	
Leave	Still meets all user requirements
Interop	Meets almost all user requirements
Migrate	Meets some user requirements
Replace	Does not meet user requirements

Supportability	
<p>Is the existing solution easily supportable?</p> <ul style="list-style-type: none"> • Quality of the solution implementation • Skills availability of in-house and vendor production teams to support the application • Support arrangements with vendors of third-party controls and components • Adherence to development standards • Capability of operations staff to support and maintain the solution <p>Easily supportable solutions are candidates for leaving in Visual Basic 6.0. Solutions that have support constraints are strong candidates for replacement.</p>	
Leave	Good support in place for Visual Basic 6.0
Interop	Good support in place for Visual Basic 6.0 and newer versions
Migrate	Good support in place for Visual Basic 6.0 and newer versions
Replace	Good support in place for newer versions of Visual Basic

Cost

How sensitive must the strategy be to cost?

- Production and procurement costs
- Annualized licensing and support costs

In the short term, leaving in Visual Basic 6.0 or interop are the most cost-effective strategies. In the longer term, migration or replacement are more cost-effective, due to benefits from platform standardization, lower support costs, and licensing consolidation.

Leave	Short-term cost-effective
Interop	Short-term cost-effective
Migrate	Long-term cost-effective
Replace	Long-term cost-effective

Time-to-market

How quickly does the business require the application?

- Speed to development and production
- Speed to make future changes
- Agility to react to platform changes

Time-to-market should be considered as the time that is necessary to finish the project and get the support strategy in place. The basis for reaching a decision might be influenced by cost, competing projects, importance of the application, and quality and availability of support for the current Visual Basic 6.0 application.

Leave	Shorter time-to-market
Interop	Shorter time-to-market
Migrate	Longer time-to-market
Replace	Longer time-to-market

Business growth

How capable is the Visual Basic 6.0 application to support future business growth?

- Life expectancy
- Interoperability and connectivity with suppliers, partners, and customers
- Ability to integrate with other business software: Office SharePoint, Office, non-Microsoft applications, and partner and vendor systems
- Scalability and performance requirements

The business-growth driver is one of the most important considerations; clearly moving an application to the .NET platform creates more opportunity to meet future business growth; but, at the same time, not every application will grow forever, and there are many short-term applications in production with limited life expectancies and little need for supporting growth.

Leave	Low need to meet business growth
Interop	Moderate need to meet business growth
Migrate	Moderate need to meet business growth
Replace	High need to meet business growth

Industry requirements

What external industry requirements does the application have to meet?

- Government compliance regulations
- Interfaces with partners and vendors
- Support for standards
- Parity and unique value proposition against competitors

Looking outside of the immediate business requirements, what compliance and regulatory requirements must the application meet? Some governments have regulatory requirements around accessibility and reporting, which Visual Studio 2008 is better placed to support than Visual Basic 6.0. Visual Studio 2008 has greater support for standards and application interoperability.

Leave	Industry requirements not relevant
Interop	Industry requirements moderately relevant
Migrate	Industry requirements moderately relevant
Replace	Industry requirements highly relevant

Evaluating Your Application by Using the Decision Framework

There are several approaches to evaluate the support strategy for your Visual Basic 6.0 application by using the decision framework. A simple

approach is to stack-rank each option 1-4 against each business driver. For example, in the following table, against the "User requirements" driver, Replace is the preferred option, followed by Interop, then Migrate and, in last place, Leave. After ranking each option, you can total the columns; the option that has the lowest total evaluates as the preferred option against each of the six drivers.

Example evaluation	Leave	Migrate	Interop	Replace
User requirements	4	3	2	1
Supportability	1	3	2	4
Cost	4	2	1	3
Time-to-market	3	2	1	4
Business growth	4	3	2	1
Industry requirements	1	3	2	4
Total	17	16	10	17

In the preceding example, interop—with the lowest total—has the best fit against each of the decision-framework drivers.

Estimating Cost and Return on Investment

Cost for each strategy varies on the type of application, the project team's strengths, and the future goals for the application. While there are no concrete rules, here are some general guidelines for considering each option in relative terms:

- Keeping an application in Visual Basic 6.0 without modification is the cheapest option. The cost is confined to operational support.
- Replacing an application by rewriting it is the most expensive option, and costs can be estimated with the same accuracy as any software-development project.
- The cost of migrating or using interop will vary by application, project team, and goals of the project. Many organizations initially perform a time-bound feasibility study by migrating an application component or adding interop features to a localized feature of the application in order to get a clearer idea of the costs, challenges and opportunities that are involved in performing a larger migration or interop project. Both migration and interop support this iterative approach.

Again, in relative terms, here are general guidelines for return on investment (ROI) for each strategy:

- There might be no new ROI for leaving an application in Visual Basic 6.0, if no new capabilities are added or new business requirements met. The return should be seen as short-term restricted by the support lifetime limitations of Visual Basic 6.0.
- The ROI for interop is derived from the benefits of adding new features cost-effectively. As with leaving an application in Visual

Basic 6.0, the return should be considered short-term restricted by the support lifetime limitations of Visual Basic 6.0.

- If the scope of a migration project includes removing COM components and rearchitecting the application to take advantage of the .NET-platform capabilities, the ROI can include discontinuation of Visual Basic 6.0 development support, operational savings from platform consolidation, and the benefits from new features.
- Replacing an application might offer a greater return than migration, because the application is architected from the ground up for the requirements of the business without the uncertainty of architectural constraints that are inherited from the Visual Basic 6.0 application. The return usually is based on the benefits of the new features, combined with discontinuation of Visual Basic 6.0 development and operational support.

RESOURCES FOR PROJECT TEAMS

This section discusses key points for each of the four strategies from a project team's perspective—what a team must do to implement each strategy. We also provide a list of Microsoft resources for each approach.

Leaving in Visual Basic 6.0

Leaving an application in Visual Basic 6.0 is a simple option for a project team; the team probably does not have to learn any new development techniques to keep an application running. The approach still needs research: Are third-party components supported, moving forward? Are the project and operation teams able to maintain functioning development and runtime environments?

Runtime support is in place for Windows Vista and Windows Server 2008. There is no planned runtime support beyond Windows Vista and Windows Server 2008. Windows 64-bit applications (Windows Vista and Windows Server 2008) are also supported as runtime platforms. Be aware that Visual Basic 6.0 applications execute in the WOW layer; it is not possible to mix and match 64-bit components with 32-bit components in the same process.

For third-party controls and components, support varies from vendor to vendor. As a general rule, more third-party controls are supported under Windows XP than Windows Vista; Windows Vista has many more security features than Windows XP, and some older COM components might not operate correctly, because the Windows Vista operating system has extra security around accessing and altering system resources. The WebBrowser control (TriEdit.ocx) is not supported in either Windows Vista or Windows 2008.

Registration-Free COM

One of the major sources of technical issues with Visual Basic 6.0 applications is DLL registration during deployment. To simplify deployment, Microsoft incorporated a new capability in Windows 2003, Windows XP, and Windows Vista that is named *registration-free COM*. As its name suggests, registration-free COM is a technique for installing and using components without making registry entries; the entries are included in a manifest file that is distributed with the application. While there are some limitations around using the technology in DLL add-ins, for many Visual Basic 6.0 executables, this simplifies the deployment process considerably and is a simple enhancement to make for Visual Basic 6.0 applications.

Virtualization and Terminal Services

Windows Server 2008 builds upon the existing impressive capabilities of Terminal Services, with new terminal-services features and capabilities. For Visual Basic 6.0 applications, this means the ability of using Terminal Services RemoteApp to host an application on the Windows Server and access it from client computers. The application executes on the server, but behaves and interacts with the local computer as though it were running on

the client. Multiple clients can access the same application on the server. This technique is useful for minimizing deployment difficulties and rapidly deploying a Visual Basic 6.0 client application.

More Resources

General resources	
Support Statement for Visual Basic 6.0 on Windows Vista and Windows Server 2008	http://msdn2.microsoft.com/en-us/vbrun/ms788708.aspx
Visual Basic 6.0 Resource Center	http://msdn2.microsoft.com/en-us/vbrun/default.aspx
Visual Basic 6.0 Downloads	http://msdn2.microsoft.com/en-us/vbrun/aa662927.aspx
Essential Training for Visual Basic 6.0 Developers	http://msdn2.microsoft.com/en-us/vbrun/cc297223.aspx

Simplifying deployment of Visual Basic 6.0 applications with registration-free COM	
Simplify App Deployment with ClickOnce and Registration-Free COM	http://msdn.microsoft.com/msdnmag/issues/05/04/RegFreeCOM/
Registration-Free Activation of COM Components: A Walkthrough	http://msdn2.microsoft.com/en-us/library/ms973913.aspx

Virtualization and Terminal Services	
Using Virtualization to Preserve a Visual Basic 6.0 Client-Server Application	http://msdn2.microsoft.com/en-us/library/cc522935.aspx
Visual Basic 6.0 Application Availability with Microsoft Virtualization	http://msdn2.microsoft.com/en-us/library/cc526339.aspx
Terminal Services in Windows Server 2008	http://www.microsoft.com/windowsserver2008/terminal-services/default.mspx
Terminal Services RemoteApp (TS RemoteApp)	http://technet2.microsoft.com/windowsserver2008/en/library/57995ee7-e204-45a4-bcee-5d1f4a51a09f1033.mspx?mfr=true

Virtualization and Terminal Services

Windows Server 2008 RC0 Terminal Services RemoteApp Step-by-Step Guide

<http://technet2.microsoft.com/windowsserver2008/en/library/61d24255-dad1-4fd2-b4a3-a91a22973def1033.mspx?mfr=true>

Interop

Interop is an incredibly cost-effective method for extending the life of Visual Basic 6.0 applications. Huge advances have been made in interop, since the initial release of Visual Basic 2002 to the release of Visual Basic 2008. It is now possible to create hybrid applications by using the Interop Forms toolkit—incorporating Visual Basic 2008 Windows forms and controls in Visual Basic 6.0 applications.

Additionally, it is easy to call methods in a Visual Basic 2008 DLL from a Visual Basic 6.0 application and vice versa. Every version of Visual Basic since Visual Basic 2002 has added steadily to interop capabilities, increasing in robustness and with the greater ability to trap and diagnose exceptions—resulting in faster development and better interop applications.

Visual Basic 2008 makes it easy to mark DLLs as "COM-visible," meaning that DLLs can be created easily with the express intention of accessing from Visual Basic 6.0. Although the Visual Basic 6.0 sections of the application retain the lifetime issues of pure Visual Basic 6.0 applications, these hybrid Visual Basic 6.0/Visual Basic 2008 applications offer more options to project teams that gradually are moving applications to Visual Basic 2008.

More Resources

Interop	
VB Fusion: Extend Your Visual Basic 6.0 Applications	http://msdn2.microsoft.com/en-us/vbrun/ms788241.aspx
Visual Basic Fusion: Best Practices to Use Visual Basic 6.0 and Visual Basic .NET Together	http://msdn2.microsoft.com/en-us/library/ms364069.aspx
Hybrid Visual Basic Series: Real-World Potential of Interop	http://msdn2.microsoft.com/en-us/library/cc300139(VS.80).aspx
Interop Forms Toolkit 2.0	http://msdn2.microsoft.com/en-us/vbasic/bb419144.aspx
Can I Interest You in 5000 Classes? (using .NET classes from Visual Basic 6.0)	http://msdn2.microsoft.com/en-us/vbrun/aa719105(VS.71).aspx

Interop	
Essential Training for Visual Basic 6.0 Developers: Introduction to Visual Basic Interop	http://msdn2.microsoft.com/en-us/vbrun/cc297223.aspx
"How Do I" Visual Basic Interop Video Series	http://msdn2.microsoft.com/en-us/vbasic/bb466226.aspx?wt.slv=topsectionsee#interop

Migrating

Migrating (or upgrading) a Visual Basic 6.0 application to Visual Basic 2008 gets the entire application to Visual Basic 2008. Typically, the migrated applications will need some rework to change code that the upgrade tool cannot migrate automatically, and to replace COM objects that the application still references.

Migrations can be done in a gradual manner—one component at a time, or as a single migration. Any architectural limitations of the original application will be carried through to the migrated application; the upgrade tool does not perform architectural refactoring.

The best practice is to assess the Visual Basic 6.0 application first, then to make any changes that will minimize rework in Visual Basic 2008, perform the migration and rework, and finally enhance the application with new capabilities of Visual Basic 2008.

After migrating, and replacing COM objects, the application will be 100 percent Visual Basic 2008, without the support lifetime limitations of Visual Basic 6.0.

More Resources

Preparing Visual Basic 6.0 applications	
Essential Training for Visual Basic 6.0 Developers: Upgrading a Visual Basic Application	http://msdn2.microsoft.com/en-us/vbrun/cc297223.aspx
Visual Basic 6.0 to Visual Basic .NET Upgrade Assessment Tool	http://www.microsoft.com/downloads/details.aspx?FamilyId=10C491A2-FC67-4509-BC10-60C5C039A272&displaylang=en
Code Advisor for Visual Basic 6.0	http://www.microsoft.com/downloads/details.aspx?FamilyId=A656371A-B5C0-4D40-B015-0CAA02634FAE&displaylang=en

Preparing Visual Basic 6.0 applications

Preparing Your Visual Basic 6.0 Applications for the Upgrade to Visual Basic .NET	http://msdn2.microsoft.com/en-us/library/aa260644.aspx
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

Migrating applications

Microsoft Patterns and Practices Guide to Upgrading Visual Basic 6.0 Applications to Visual Basic .NET and Visual Basic 2005	http://msdn2.microsoft.com/en-us/library/aa480541.aspx
Free book: <i>Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET</i>	http://msdn2.microsoft.com/en-us/vbrun/ms788236.aspx
Upgrade Resources for Visual Basic 6.0 Developers	http://msdn2.microsoft.com/en-us/vbrun/ms788239.aspx
Deconstructing the Visual Basic Upgrade Wizard	http://msdn2.microsoft.com/en-us/library/aa730876(VS.80).aspx
How to Use the Visual Basic Upgrade Wizard	http://support.microsoft.com/kb/317885

Replacing

Replacing an application can mean one of three options: buying an off-the-shelf replacement product, developing a replacement application, or extending another application to replace the Visual Basic 6.0 system.

This strategy has the most flexibility in terms of architecture, development standards, and requirements fit. Visual Basic 2008 is more powerful and capable than any previous version of Visual Basic, and project teams that have an existing history of Visual Basic development have a head start in terms of knowledge and skills that they can apply to replacing a Visual Basic 6.0 application with a new system in Visual Basic 2008.

Resources

Replacing

What's New (in Visual Basic 2008) for Visual Basic 6.0 Users	http://msdn2.microsoft.com/en-us/library/ms172617(VS.90).aspx
Developing Applications with Visual Basic (2008)	http://msdn2.microsoft.com/en-us/library/k6a6etxs(VS.90).aspx

Replacing

Visual Basic Case Studies	http://msdn2.microsoft.com/en-us/vbasic/ms789180.aspx
Microsoft Dynamics	http://www.microsoft.com/dynamics/

SUMMARY OF SUPPORT FOR VERSIONS OF VISUAL BASIC

Microsoft's support policy for Visual Basic follows the support life-cycle guidelines found at <http://support.microsoft.com/?LN=en-us&pr=lifecycle>.

In summary, Microsoft has a proven support model. Products go through three phases of support: mainstream, extended, and self-help.

During *mainstream* support, features might be added, design requests might be accommodated, hotfixes are distributed, complimentary phone and online support is provided, security updates are produced, and paid support is available.

During *extended* support, security updates are produced, some updates might be distributed, and paid hotfixes and paid support are available.

When a product reaches the *self-help* phase, Microsoft provides free access to online content, such as knowledge-base articles, online product information, and online-support webcasts.

Support Summary for Versions of Visual Basic

There have been 10 versions of Visual Basic. For planning purposes, the following table shows the current (at the time of writing) support phase for each version:

Name	Life-cycle Support Phase
Visual Basic 1.0 through Visual Basic 5.0	Self-help
Visual Basic 6.0	Extended support of Visual Basic IDE to April 2008 Extended support of runtime to 2017
Visual Basic 2002	Extended support to July 2009
Visual Basic 2003	Mainstream support to October 2008, extended support to August 2013
Visual Basic 2005	Mainstream support to March 2011, extended support to March 2016
Visual Basic 2008	Mainstream support

CONCLUSION

In this white paper, we discussed the four strategies that are available for supporting Visual Basic 6.0 applications. The technology now exists to realize each of the four strategies. In the decade between the release of Visual Basic 6.0 and Visual Basic 2008, Microsoft has produced, improved, and refined the interop, migration tools, support offerings, and training materials to help organizations take the next step in moving their applications from the COM-based infrastructure of Visual Basic 6.0 to the .NET platform—where technology can provide solutions to drive business forward in new directions.

Acknowledgements

This white paper was prepared by Ed Robinson, Microsoft Visual Basic Program Manager from 1999 to 2003, designer of many of Visual Basic's upgrade tools, co-author of *Upgrading Visual Basic 6.0 to Visual Basic.NET* and current CEO of ActionThis, Ltd; with assistance from Paul Yuknewicz, Lead Visual Basic Program Manager and designer of many of Visual Basic's new interoperability tools.