

Migrating Visual Basic to .Net An Approach Specification

Table of Contents

Why to migrate from Visual Basic to .Net.....	3
Methodology.....	3
Application Migration	4
1. Code Migration.....	5
2. Language Selection (C# or Vb.Net)	5
3. COM Components Migration.....	6
4. Presentation Layer Migration	6
Approach 1 - Thick Client / Rich Client / Desktop Application.....	6
Approach 2 - Thin Client / Web Application	7
Approach 3 - Smart Client Application	7
Approach 4 - Rich Internet Application	8
5. Business Logic Layer Migration	8
6. Data Access Layer Migration	8
Data Migration.....	9
Approach 1 – MySQL	9
Approach 2 – SQL Server 2005.....	9
Approach 3 – Oracle 10g.....	10
Summary	10

Why to migrate from Visual Basic to .Net

There are many advantages of .Net over Visual Basic:

1. Full object-oriented capabilities
2. Better performance
3. An incredible IDE with so many features like code snippets, Easy Class creation with UI etc. makes faster, better and cheaper development
4. Interoperability – Easy Cross-language integration
5. Language Independence - One way to do things, no matter what .NET language you are using
6. Easy web service development with Web Service tools
7. Overcome DLL Hell problem
8. No need of registration of components
9. Rich support of Base classes with advanced functionality like Threading, Remoting and Reflection etc.
10. Easy Application deployment and maintenance – Simple Copy and Paste
11. Side-by-Side Code Execution – Multiple versions of same assembly can reside on one machine
12. Advanced security support
13. Better Administration and management
14. Master Page support for Asp.net development
15. Data Access with ADO.NET – advantages over ADO like minimized open connections, in memory data representation etc.

Following is the only disadvantage of .Net over Visual Basic:

1. .Net framework installation on deployment machine

Methodology

We execute following high-level steps to migrate Visual Basic application to .Net:

1. Application analysis
Analyzing existing application is beneficial to understand functionality and its architecture. Out come of this step is to determine appropriate migration strategy.
2. Select migration strategy
Depending upon the client requirement for migration time line and cost, any one of the following migration strategies can be selected:
 - a. Complete Migration Strategy - All components of an application are migrated to .NET

Advantages:

- Migrated applications can be advanced with new functionality that uses .NET technologies and give better performance
- Migrated applications can be integrated into new .NET solutions, making them usable in more scenarios
- Fully-migrated applications can be used on new hardware and new versions of Windows

Disadvantages:

- An application is available only after complete migration is done

- b. Staged Migration Strategy - Allow to migrate some parts of your application first and then start with other parts.

Advantages:

- More control over the progress and the cost of the migrated project
- Minimized risk, because it returns the application to a stable production-quality state after each stage
- Migrated components can often immediately benefit from the performance and scalability features of .NET without having to wait for the entire application to be migrated
- Migration of the low priority or costly portions of an application can be postponed indefinitely, resulting in a partial migration

Disadvantages:

- Performance improvements will not be as noticeable as in a full migration because of added overhead required for COM interoperability
- Applications with a dependency on COM components are harder to deploy and maintain than applications that are fully migrated to .NET because COM objects need to be registered and have versioning issues that .NET objects do not have

3. Design target application architecture
After understanding existing application and its architecture, design .Net application architecture.
4. Migration of VB Application to .NET
 - a. Migrate Visual Basic Source Code in to .Net (Vb.Net or C#)
Using Microsoft Code Conversion tool, convert VB code in to .Net.
 - b. Fix migration issues
Solve all the issues generated in the upgrade report generated by Microsoft Code Conversion tool during migration.
 - c. Development
As per the architecture defined and migration strategy, develop the components or write the code to migrate functionalities to .NET where direct conversion is not available.
5. Unit testing & Integration Testing
Testing is required to validate functional equivalence between existing Visual Basic and migrated .Net application.
6. Deployment
Deploy the migrated .Net application as per the deployment architecture.

Application Migration

Considering the nature of application, features and components used; it can be classified in one of the following three categories in terms of migration:

1. Low Complexity
 - a. Dynamic-link libraries (DLLs)
 - b. Application with only Forms or MDI Forms
2. Medium Complexity

- a. Data binding using ActiveX Data Objects (ADO)
 - b. Graphics
 - c. Menus
 - d. Windows API
 - e. User controls
3. High Complexity
 - a. ActiveX Dynamic HTML (DHTML) applications
 - b. ActiveX documents
 - c. COM/DCOM/COM+ Components
 - d. Legacy one-tier Microsoft Access database with DAO or Remote Data Objects (RDO) data binding
 - e. Dynamic Data Exchange (DDE)
 - f. Object Linking and Embedding (OLE)

Following sections describe approaches and efforts required for migrating VB application to .Net

1. Code Migration

Visual Studio IDE has a facility to bring up a wizard needed for migrating VB programs to Vb.Net. The detailed steps about converting Visual Basic code in to Visual Basic.Net is listed at <http://www.codeproject.com/vb/net/Migration.asp>.

Development Effort: Since it is anticipated that everything in VB may not go over to VB.NET, an upgrade report is also produced at the end of migration. It also shows all the files and any facing issues in each of these files. Development should be required to fix all migration issues. The code needs to be updated to comply with .Net Coding standard as per

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconNamespaceNamingGuidelines.asp>

2. Language Selection (C# or Vb.Net)

The choice between C# and VB.NET is largely one of subjective preference. Both have access to the same framework libraries. Many people prefer C# instead of VB.Net. There are many third party tools available to convert VB.Net code to C# code.

The following are actual differences between VB.Net and C#:

VB.NET Features:

1. Support for optional parameters – Used for COM interoperability
2. Support for late binding with Option Strict off - type safety at compile time goes out of the window, but legacy libraries which don't have strongly typed interfaces become easier to use
3. Support for named indexers
4. Simpler event handling, where a method can declare that it handles an event, rather than the handler having to be set up in code
5. VB.NET parts of Visual Studio .NET compile your code in the background. While this is considered an advantage for small projects, people creating very large projects have found that the IDE slows down considerably as the project gets larger

C# Features:

1. Language support for unsigned types
2. The using statement, which makes unmanaged resource disposal simple
3. Unsafe code - This allows pointer arithmetic etc, and can improve performance in some situations

3. COM Components Migration

Using the .NET wrapper around a COM interface can take advantage of the new .NET features while using existing code. But at the same time it has some of the disadvantages like:

1. Performance will not be as good compared to code written entirely in .Net
2. DLL Hell problem
3. Imported COM library will be large in size than actual size
4. There is also no way to hide an interface that is meant only for use within the COM component
5. Because of data type mismatch between COM and .Net, generates complex, error-prone and tedious marshaling code
6. If COM throws an error then .Net will not get detailed specification about an error

Development Effort: Looking at all the disadvantages mentioned above, if Visual Basic application uses COM components like ActiveX Control, ActiveX DLL etc. in any of the layer like Presentation, Business or Data Access then it is preferred either to develop these components in to .Net or write .Net code in to appropriate layer.

If any third party component is being used as a COM then the replacement needs to be found in .Net or code needs to be developed

4. Presentation Layer Migration

Approach 1 - Thick Client / Rich Client / Desktop Application

Proffered when the business logic is complex and requires more time to process, requires heavy data entry and/or frequent navigation across multiple windows.

Advantages:

1. Rich User interface
2. Offline capabilities – Need not be connected on a Network
3. High Developer Productivity
4. Responsive & Flexible
5. Better Performance
6. Efficient use of local resources
7. Easy integration with local applications and APIs

Disadvantages:

1. Tough to Update – Each client location needs modifications
2. Tough to Deploy – Deployment had to be done at all client locations
3. Supports only on windows platform

Examples:

1. Animation CAD Software

Development Effort: If migration issues are very less for a UI and code, it can be reused, still modification is required to make sure it uses .Net Coding standard and Code and UI should not have any COM dependency.

If migration issue report is complex then it is better to develop UI and code for the same.

Approach 2 - Thin Client / Web Application

A thin client is often the most appropriate if you need to make an externally facing application available to a diverse external audience.

Advantages:

1. Easy to update, Deploy and Manage – Single location update
2. Can be executed from any location
3. Since data is retrieved and used online, raw data can be secured
4. Servers can be dedicated for more resources for faster processing

Disadvantages:

1. Network dependency
2. Use more bandwidth for sending data
3. Less responsive
4. Complex to develop
5. Rich UI is not supported and common application features such as drag-and-drop, undo-redo, and context-sensitive help may be unavailable

Examples:

1. Virtual Earth
2. Google Maps

Development Effort: As this is web application, UI needs to be created and accordingly code will be developed in .Net.

Approach 3 - Smart Client Application

Smart client applications can be designed to combine benefits of a rich client and a thin client application. The smart client simply acts as a front end to your business logic. The core business logic resides in web services. It is designed to run on a broad spectrum of client devices including desktop PCs, Tablet PCs, and handheld mobile devices such as Pocket PCs and Smart phones.

Smart client architecture is often the most suitable for an internal application that needs to integrate with or coordinate with other client-side applications or hardware, or that is required to work offline or provide specific high-performance functionality through a responsive user interface.

Advantages:

1. Rich user interface
2. Easy to Deploy – Single location update
3. Easy to Update and manage - Automatically update without user action with Click Once Deployment strategy
4. Work offline so no network dependency

Disadvantages:

1. Restricted access to system resources
2. The EXE gets downloaded at client side. So, takes more time to load the first time
3. Supports only on windows platform
4. Required .Net framework on the client machine
5. Since MSIL EXE gets downloaded at client there is risk of client "de-compiling" your code
6. Unreliable offline functionality
7. Configuration files (app.config) can not be used easily as in case of pure desktop application
8. Extra headache of data synchronization when client gets online from offline

Examples:

1. Microsoft Money
2. Outlook® messaging and collaboration client
3. Google Earth
4. World Wind (NASA)

Development Effort: UI and code need to be developed as per the smart client architecture. Composite Application Block (CAB) is a platform provided by Microsoft with inbuilt smart client architecture to develop smart client application which frees developer to worry about business logic than its architecture.

Approach 4 - Rich Internet Application

Rich Internet applications (RIA) are Web applications that have the features and functionality of traditional desktop applications. RIAs typically transfer the processing necessary for the user interface to the Web client but keep the bulk of the data (i.e., maintaining the state of the program, the data etc) back on the application server.

Advantages:

1. Run in a web browser and do not require installation
2. Typically much more responsive than a standard web applications
3. Richer UI and support for common application features such as drag-and-drop, undo-redo, using a slider to change data, calculations performed only by the client and which do not need to be sent back to the server, for example, a mortgage calculator
4. Has a look and feel of a desktop application
5. Asynchronous Communication with the server which feels more responsive
6. Efficient Use of Network

Disadvantages:

1. Hard to develop
2. Restricted access to system resources
3. JavaScript is required to be enabled
4. Script download time for client side intelligence can increase
5. Loss of visibility to search engines - Search engines may not be able to index the text content of the application
6. Dependency on network connection

Examples:

1. Outlook Web Access - The webmail client used in Microsoft's Exchange Server
2. Gmail - Webmail client for Google's email service
3. Yahoo! Mail (beta) - A webmail client for Yahoo!'s email service

Development Effort: UI should be created and code should be developed. UI should be done with .NET 3.0 Framework's Windows Presentation Foundation (WPF) using XAML or Microsoft Silverlight.

5. Business Logic Layer Migration

Development Effort: Migration code can be reused to understand business logic But code needs to be developed in .Net due to architecture change, or complying with .Net coding standard.

6. Data Access Layer Migration

The Visual Studio conversion wizard can move about 90% of your ADO code so it can run in a .NET application using the COM interop layer. COM interop puts a wrapper around the COM

classes and tries to translate what COM expects to what .NET expects and back. Unfortunately, ADO does a number of things that COM interop will never understand. This means that you'll find that some code simply does not work and you will never come to know about the exact problem. The following are the disadvantages of using ADO in .NET through migration code generated:

1. Disadvantages listed in [Section 3. COM Components Migration](#)
2. Migration issues causes lots of complex development and redevelopment
3. Dependency on the MDAC stack
4. As code will still be implemented using ADO, it does not provide any advantage of ADO.NET architecture which mainly benefit with its performance

Development Effort: Code should be developed in .Net because of migration issues and COM interop. Even using ADO with .Net is unsupported by Microsoft.

If the selected data source (Oracle, SQL Server, MS Access) is supported by Data Access Application Block from Microsoft then it should be used to develop data access layer more faster and efficient way. All the database operations and its complexity to give the best result with maximum utilization of resource is integrated with Data Access Application Block. Using Data Access Application block, developers are not required to worry about designing internals of data access layer.

Data Migration

Approach 1 – MySQL

The MySQL® database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use.

MySQL Server was designed from the start to work with medium size databases (10-100 million rows, or about 100 MB per table) on small computer systems. It is preferred choice when applications built on Linux, Apache, MySQL, PHP / Perl / Python.

Advantages:

1. Requires less hardware resources compared to SQL Server 2005 and Oracle
2. Free under the terms of the GNU General Public License
3. Supports all known platforms like Linux, Windows, OS/X, HP-UX, AIX, Netware etc
4. Migration toolkit supports mostly all databases like Oracle, SQL server, Access etc.

Disadvantages:

1. Database limit is 4 GB

Approach 2 – SQL Server 2005

SQL Server 2005 is a comprehensive database software platform providing enterprise-class data management and integrated business intelligence (BI) tools as shown below.

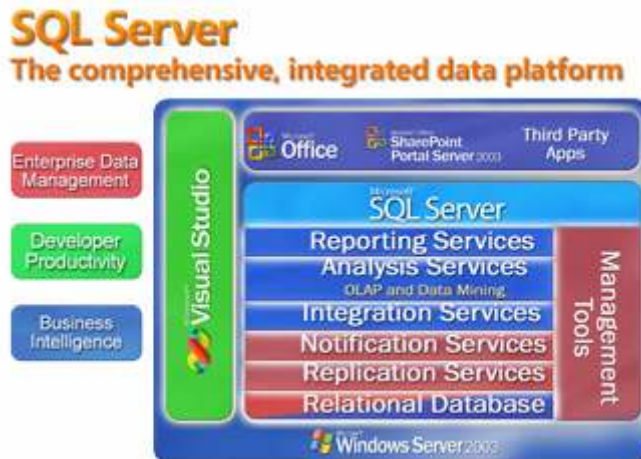


Figure 1. SQL Server 2005 Core Components

Advantages:

1. Transact-SQL is more powerful language than MySQL dialect
2. Database size limit is in terms of terabytes(1,048,516 terabytes)
3. Superior ease of use and seamlessness in developing service-oriented architecture (SOA) applications
4. SQL Server business intelligence (BI) features are integrated with Visual Studio, unlike Oracle BI features
5. Easy, GUI based Import/Export wizard to migrate any database in to SQL Server

Disadvantages:

1. Works only on windows based platform

Reference:

1. Detailed comparison between SQL Server 2005 and Oracle 10g in each of the area is available at <http://www.microsoft.com/sql/prodinfo/compare/oracle/default.mspx>

Approach 3 – Oracle 10g

Oracle is not only the world's most reliable and performant database, but also part of a complete software infrastructure for enterprise computing

Advantages:

1. Supports all known platforms
2. Oracle Migration Workbench is a tool that simplifies the process of migrating databases like SQL Server, Access, MySQL etc. to the Oracle platform

Disadvantages:

1. Enterprise Editions has 4GB database limit while all other editions are available without any database limit

Summary

Even if migration tool is available to convert Visual Basic code in to .Net, still some of the complex UI will have to be created or some UI needs to be modified to get maximum benefits of .NET.

Business logic and Data Access Layer will have to be developed in .Net.

If any COM components are used then COM component code will to be re-written in .Net to get maximum benefits of .NET and avoid all the problems of COM inter-operability.

Application Selection:

Features	Rich Client	Thin Client	Smart Client	Rich Internet Application
Rich User Interface	X		X	X
Easy Deployment and Maintenance		X	X	X
Responsive	X		X	X
Web Service and offline/online support			X	
Network Dependency		X		X
Windows Dependency on Client	X		X	
Server Configuration only can increase performance		X		X
Better Performance	X		X	X
Integration with local applications and APIs	X			
Restricted access to local system resources		NA	X	X
Fully supported visibility to search engines		X		

Database Selection:

Feature	MySQL	Oracle	SQL Server
Small to medium Data Storage (up to 4 GB)	X		
Large Data Storage		X	X
Small User Group	X		
Large User Group		X	X
Better Performing when integrated with Microsoft Visual Studio .Net Platform as per the features required which are listed at, http://www.microsoft.com/sql/prodinfo/compare/oracle/ss2005oracle10gnetdev.mspx			X
Database Migration Wizard/Tools	X	X	X