Estimate migration costs for VB6 conversion software

Every day we receive requests and queries from VB6 developers and companies who are interested in migrating their code to .NET. Migration a complex application to .NET is a serious matter, therefore it's no surprise that virtually all our prospects plan to compare our software with the similar tools from the competition.

We take migration seriously and we take competition even more seriously. As a matter of fact, we stepped into this market niche only when we were sure we could offer the best VB6 software that could be possibly written. And that's why we continued to improve our VB Migration Partner until it offered support for virtually all major VB6 features, including outdated features such as "classic" drag-and-drop and DDE. (You might be surprised to learn how many VB6 apps rely on these almost-forgotten features.)

Understandably, many customers are concerned about the price of the software they are buying. This is good for us, because we know that our price list is slightly lower than comparable products. However, we believe that the crucial question should be different.

Should the total cost of migration be a better selection criteria when selecting a migration tool?

All the real-world migration experiences and case studies show that the largest part of migration costs is related to the manual labor necessary before and after the mere action of passing the VB6 code through the conversion software. In real-world scenarios, the cost of the software is a very small fraction of the total costs, often in the range 10-20% if not lower. Therefore, when comparing migration tools, the most "correct" selection criterion should be based on the manual labor that each tool saves you and your team.

In this document, we explain why VB Migration Partner compares favourably in this respect to other similar products.

1. Preliminary revision of the original VB6 project

Prior to VB Migration Partner's debug, all VB6 conversion tool on the market – including the free Upgrade Wizard tool formerly included in Visual Studio – supported only a subset of VB6 keywords and features. For this reason, a number of "preliminary changes" had to be applied to the original VB6 project, to ensure that it could be converted correctly to VB.NET. In many cases these fixes introduced minor issues in the VB6 project, which had to be fixed until the migration could finally begin. Such preliminary step is seldom required when using VB Migration Partner, given its full support to virtually all major VB6 features.

For example, if your VB6 app uses GoSub, On Goto, or On Gosub statements, using a conversion tool other than VB Migration Partner requires that you spend at least a few minutes for each of these keywords. The problem can be more serious if your VB6 code relies on other, less common features - such as Dynamic Data Exchange (DDE), ADO data sources or data consumers - because you'd be forced to re-write large portions of your code. Implementing a reliable communication infrastructure based on .NET remoting or WCF may take one week or two if you aren't a .NET guru, or it takes ten seconds flat if you use VB Migration Partner.

Graphics is another field where VB Migration Partner excels, with its support for Line and Circle methods, double-buffering (the AutoRedraw property), and user-defined system coordinates (ScaleMode property). The Code Sample section on our web site can prove that you can convert a graphic-intensive VB6 application in a matter of seconds. Only VB Migration Partner supports these graphic features; other less powerful tools require that you manually implement all graphic statements.

2. Functional equivalence

VB Migration Partner can handle even the most subtle differences between VB6 and VB.NET, including true auto-instancing (As New) semantics, by-reference ParamArray parameters, correct disposal of database connections and other COM objects, thread-safe behavior for multithreaded components, 100%-compatible file I/O statements, and the many methods that behave slightly differently in VB6 and VB.NET. These features can dramatically cut the time required for testing the migrated application; most of them aren't supported by other migration tools.

3. Late binding

VB Migration Partner comes with a library of .NET controls that expose the same properties and methods as the original VB6 controls. This detail ensures that your code works fine after the conversion to .NET, **even if the control is accessed in late-bound mode**. If you use another conversion tool you might need to manually rewrite each and every statement that uses late binding. If your VB6 code uses many Variant, Object, and Control variables, this task alone can keep you busy for days.

4. Rich user interface

In theory all conversion tools support all VB6 user interface features. In practice, however, tools from our competitors require that you manually fix each individual form to account for minor differences between VB6 and .NET controls. An indirect evidence of their limits is that none of our competitors offers examples of real-world VB6 forms and the corresponding (migrated) .NET forms, perhaps because they would be forced to mention all the manual fixes that are necessary to have a working .NET application.

Also, none of our competitors matches VB Migration Partner's support for advanced UI features, such as drag-and-drop (both OLE and "classic" flavors), popup menus, graphic statements, dynamically created controls (the Controls.Add method), and 100%-compatible object models for complex controls such as TreeView, ListView, Toolbar, and StatusBar.

5. Third-party ActiveX controls

VB Migration Partner correctly converts all the 60+ controls included in VB6 toolbox and a simplified approach for handling 3rd-party controls. Such an approach is based on wrapper classes that inherit from a .NET control but expose the object model of the original ActiveX control. The benefit of this approach is that the effort required to migrate a given control is roughly proportional to the number of properties, methods, and events that your app actually use: for example, if your app uses just 20 properties and methods of the original control, you have to implement only 20 members of the wrapper class.

Other VB6 migration tools use a different approach, based on a mapping mechanism that transforms references to a member of the original ActiveX control into the closest member of a .NET control, either from Microsoft or another vendor. In general, the number of members that can be migrated in such a way is relatively low (typically 20-30%), or even lower with complex controls such as grids and chart components; therefore, you should be prepared to manually fix the code that references those controls. The effort required by these manual fixes is proportional to the number of members you have used ***AND*** to the number of occurrences of the control. If you have used the control many times in your forms, VB Migration Partner's approach can be far more convenient.

Also, only VB Migration Partner's approach based on wrapper classes ensure that the migrated code works well even if the control is accessed in late-bound mode.

6. ADODB to ADO.NET migration

Code Architects' ADOLibrary hides all the differences between the ADODB and ADO.NET object models, including subtle details such as parameterized queries and calls to stored procedures. It offers support for server-side forwardonly-readonly cursors and client-side cursors with optimistic batch updates, plus (limited) support for SQL Server's keysets. Thanks to all the supported features, the migration from ADODB is a breeze, and takes only a small fraction of the time required by a manual conversion.

Other VB6 migration tools may offer partial support migration from ADODB by means of code transformations. These tools don't support most of ADODB's features and don't offer full functional equivalence with the original code. In practice, in most cases you have to tweak the generated code until you manually fix all the issues, a process that can keep you busy for weeks and that requires in-depth knowledge of both ADODB and ADO.NET.

7. Staged migrations

When converting large N-tiered VB6 applications you usually want to start with the data and business components, because these are the layers where .NET offers the greatest advantages in terms of performance and scalability. Another good reason for doing so is that you can reuse the existing (VB6) user interface layer to test the behaviour of the (converted) data and business objects. This approach is often dubbed as *staged* or *phased migration*.

The prerequisite for a successful staged migration is that your conversion tool is able to generate .NET DLLs that are binary-compatible with the original VB6 DLL, so that all the clients of the original DLL can continue to work as before and you don't need to have two distinct versions of each VB6 app. This apparently minor detail is essential to avoid versioning issues during the migration process.

VB Migration Partner is the only conversion software that generates .NET DLLs that preserve binary compatibility with legacy COM components, and therefore it is the only product that fully supports staged migrations.

Such ability is important because it allows you to publish a "hybrid" VB6/.NET solution even before the migration has been completed. If time-to-market is essential for you, the choice should be obvious. *(Please notice that this feature is only available when converting to VB.NET.)*

8. The convert-test-fix methodology

A real-world migration process may take weeks or months, and you might need to extend or modify the original VB6 code before the migration is complete. All the tools from our competitors work on a *snapshot* of the VB6 code; therefore, once the migration is complete you have to manually modify the .NET code to be in-synch with the VB6 application. This process takes time, is error prone, and can be very expensive.

You don't incur in this added cost if you adopt our convert-test-fix methodology. Our customers can extend or maintain the VB6 code base while it is being migrated to .NET, without having to worry about keeping the two versions in-sync.

9. Code maintainability

Even if it doesn't impact the immediate cost of migration, you should also account for the long-term cost of maintaining the migrated code. The less readable the generated code is, the more expensive it will turn to modify and extend it in the future.

In another whitepaper we demonstrate that VB Migration Partner can deliver more readable and maintainable code than its competitors.

Of course, our competitors disagree with our conclusions and may claim that their code generates better code. For this reason, we prefer to provide actual code examples of the code our VB Migration Partner produces, something that our competitor omit from their literature.

10. Documentation, code samples, and support

A realistic estimation of total migration costs must include the time needed to become familiar with the migration software and its idiosyncrasies, peculiarities, and defects. In this respects, nothing beats VB Migration Partner's online manual and the vast knowledge base, code examples, and workarounds. The documentation available on VB Migration Partner is four times larger than the documentation available for its closest competitor.

The sheer amount of the available documentation isn't the only way to measure how well documented a product is. If the quality of documentation is important for you, please consider that the VB Migration Partner' Team includes top-selling book authors who have written one dozens of books (over 10,000 pages in total) for publishers such as Microsoft Press, and hundreds of technical articles for US and Italian technical magazines.

11. Licensing

We believe so much in our software that we offer lifetime warranty on VB Migration Partner's support library: if you are a registered user you have the right to download future versions of the library. (Please read our EULA for more details.) Compare our approach with the time-limited license that some of our competitors offer, and you'll have a clearer idea of which product can really save your time.