Mobilize.NET

How To Select An **Automatic VB to .NET Migration Product**

YOUR APP. NEW. AGAIN.

Software migration products are complex technologies that need to strike the right balance between automation and quality of the generated code.

At one end of the migration product range there are compilers, which take source code and translate it 100% automatically to a different language; a language that it is not fit for "human consumption" and cannot be easily maintained.

On the other side, there is the full manual upgrade, in which an engineer optimizes code quality at a very high migration cost.

In the case of Visual Basic 6 migrations, customers have several choices depending on which optimization is required. This article shows the criteria that should be analyzed when considering the acquisition of a VB6 migration product.

Code Quality: is it really .NET?

The .NET code generated by the migration product should allow the resulting application to evolve and scale up without any limitations.

The end result should be, as much as possible, native .NET code, in order to take advantage of the platform features. This way, a trained .NET programmer won't find any difficulty in understanding and maintaining the generated code.

The migration philosophy behind the Visual Basic Upgrade Companion (VBUC) is to produce native .NET code and take advantage of the all the technologies available in the .NET Framework.

Here are some examples of features where the VBUC generates high quality, .NET code:

- Database access technologies are converted to ADO.NET.
- Common ActiveX controls used in VB6 are replaced by .NET native components.
- Traditional VB6 error handling is replaced to structured .NET Error Handling (Try … Catch)
- Type inference: The Visual Basic Upgrade Companion performs an extensive data analysis to infer the most appropriate data types for variables, parameters, and return values, avoiding the use of generic data types like Variant and Object.
- Code Refactoring: The Visual Basic Upgrade Companion performs special refactoring over the resulting .NET code to make it more natural to a .NET developer.
- The migrated application can be moved to the web or cloud. This is possible through Mobilize.Net's WebMAP solution

Unlike the VBUC, other tools in the market generate code that looks like the original VB6 source code with most of its keywords replaced by the constructions contained in a runtime support library.

The maintenance over the resulting code is complex, and it looks and feels more like VB6 than .NET. As a result, the migrated application also faces lots of maintainability challenges, like the presence of proprietary graphical controls that can't be mixed with .NET intrinsic controls.

Use of Proprietary Runtimes Increases Long Term Risk

A simple way of building a highly automated migration product is to rely on an extensive run-time library to reduce the gap between Visual Basic 6.0 features and .NET features.

That is, at its core, re-implementing VB6 in .NET. However, this approach will release you from one legacy environment only to lock you in with a proprietary runtime supported by a smaller company.

The migration philosophy behind the Visual Basic Upgrade Companion is to produce native .NET code with no dependency on the legacy platform or any third-party runtime, so that customers can effectively take control of the evolutionary path of the migrated applications without any restrictions.

To ensure this, the VBUC employs complex transformations and heuristics in order to generate VB.NET or C# code with the direct support of the .NET framework. Only when the difference in functionality is too high due to functionality differences or code readability and maintenance issues, the VBUC may make use of a small set of helper classes that are distributed, along with their source code, with the migrated code, creating no dependency on Mobilize.Net.

Using a large runtime might be an option to speed up the migration process, and can lower the initial cost of the VB to .NET migration (though in the long term it will always be more expensive to maintain a hybrid application on .NET), but if you decide to go with this approach you need to make sure that:

- **Does not limit the future scalability of the migrated application:**

  Most VB to .NET migration tools generate one class per each VB6 control, so that every single control used in the migrated application is an instance to a class located in the proprietary library.

  All the buttons, control arrays, text boxes, everything, is referenced to that library's classes. Also the generated forms inherit from classes located in the proprietary runtime and not the native .NET Form.

  Adding new featires on this type of code will definitely compromise the final application's maintainability. For example, dropping these runtime controls onto a regular VB.NET form causes either a design-time or a runtime error.

Another issue with proprietary runtimes is the incompatibility with most Visual Studio tools.

For example, if you use the Test Wizard included in Visual Studio to create unit tests for a VB.NET project that was generated using one of these runtimes, you will get a Visual Studio exception.

Provides full source code and documentation, so you don't have to rely on the vendor:

If you decide to use a runtime approach, make sure that you can get the complete source code and documentation of this runtime, along with the rights to modify it without any limitations, so that your release cycles don't have to depend on the vendor's, and the risks associated with it.

Some vendors provide open-source versions of their runtimes which are useful, but open-source software opens other complications especially with the open-source licensing.

- **Doesn't have royalties of any kind associated with it:**

  Don't pay fees for maintenance for the runtime, or royalties for the distribution of the applications in the future. Even when there are claims it will be free, there are always risks, like another company purchasing the original vendor, or simply going out of business

Extensible and Customizable

A migration product must be extensible and customizable because you're going to need the flexibility of adding functionality:

- Having control over the generated .NET code (for example, migrating the error handling using On Error ... Goto or using Try ... Catch)
- Replacing third party controls with .NET native components
- Implementing specific programming patterns used by your organization
- Increasing automation by changing the behavior of the tool to perform repetitive changes automatically

The Visual Basic Upgrade Companion provides the following mechanisms to customize the generated code:

Migration Profiles:
the user has control over which features and transformations to use on a specific migration through the concept of Migration Profiles.

These profiles improve the quality of the generated code, providing precise control over the transformations.

For example, migrating a specific third party component to a native .NET equivalent, but leaving another one through COM Interop. The Visual Basic Upgrade Companion provides 11 Code Conversion Rules, and supports over 44 different ActiveX libraries.

Custom Maps:
the Visual Basic Upgrade Companion includes a "mapping" mechanism that allows defining the transformation of one element of a library used in the VB6 code to a member of an assembly in .NET. This allows the customer to identify and implement timesaving transformations, and when combined with the implementation of an adapter for the .NET component, can speed up the migration process dramatically.

Customizations:
Mobilize.Net's VBUC development team can also include additional rules in the VBUC that require more context than a simple mapping, such as error handling patterns, architectural modifications, and others.

These customizations are useful to:

Modify the generated code to fulfill specific customer needs (e.g. company coding standards, application architecture changes, ActiveX replacement, etc.)

Increase automation to:

- Reduce manual intervention to fix the resulting code
- Reduce compilation and runtime errors of the migrated code

Code Comments as Customizable Mechanism and Re-migration Capability

Other migration tools in the market use an approach where the user has to insert commented lines (something similar to compiler directives) in the original Visual Basic 6.0 source code.

These commented lines are used by the migration tool to identify where the resulting code needs to be changed in some manner.

Years ago, Mobilize.Net used this approach in earlier migration tools and identified the following issues:

1. .Requires substantial effort

This approach (inserting comment lines into the original source code) can be used in very small migration projects without creating any unnecessary overhead to the overall migration process, but when the migration larger scale, the amount of manual insertions of the source code comments will raise the cost of the migration project significantly.

Some transformations may require a lot of comment insertions in the original Visual Basic 6.0 source code.

For example, if you want to convert all the VB6 error handling mechanism (On Error ... Goto) to the .NET structured error handing (Try ... Catch) you need to insert a comment for each transformation.

On a large project, this means that you may have to insert thousands of comments, and a comment into the original Visual Basic 6.0 code requires a similar effort that a manual change in the migrated code.

On the other hand, the VBUC applies this transformation simply by switching on the On Error Handling feature in the Migration Profile.

2. Logistics Problems for Re-migration Capability

Another common use for inserting comment lines into the original source code is to provide a re-migration alternative.

This means that the development team of the application can continue making maintenance changes into the VB6 source code while the migration team is upgrading the application to .NET.

However, this strategy requires that the migration team use the same base code that the development team is working on.

This requirement is not always feasible in large companies, especially because of information security issues.

Another problem is that the migration team will be working with an unstable version of the source code, increasing the testing cost of the upgrade project.

In order to provide re-migration capability and avoid these issues, the VBUC team uses a proven, well-structured continuous migration methodology.

Automation

The level of automation is important because this feature will have a direct impact on the cost of your migration project.

There are some migration tools out there with a very low level of automation, and others that perform better but will generate low quality code and require the use of a huge runtime.

The key is to aim for a high level of automation without sacrificing quality of the generated .NET native code.

To evaluate the level of automation and the quality of the code generated by the Visual Basic Upgrade Companion you can get a trial version and test it on your own code, and if you need some assistance during the evaluation don't hesitate to contact us.

Support /Reputation

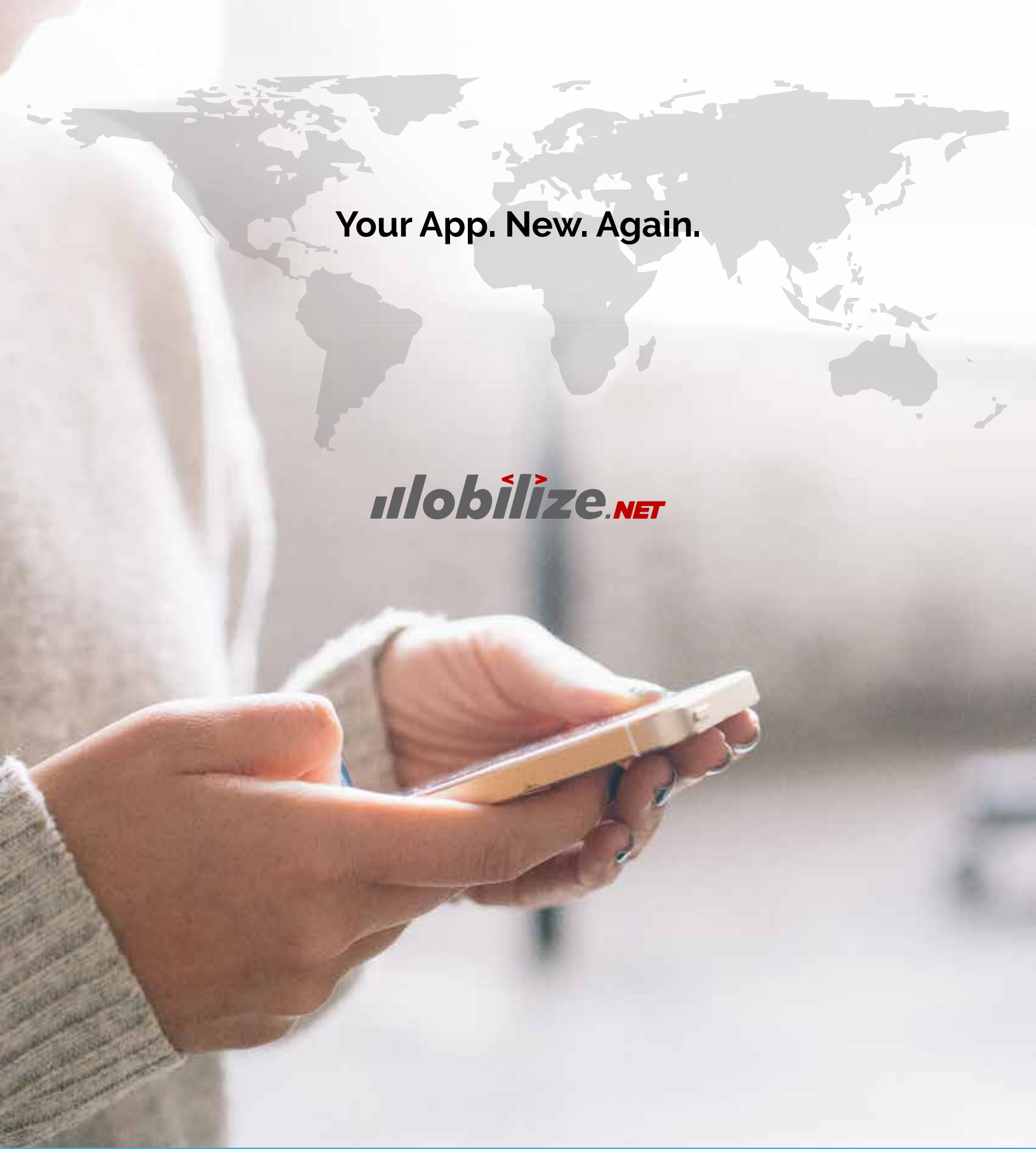
It is also important to check on the level of expertise, available support and credentials of the vendor.

Mobilize.Net has been in the software migration business for more than 15 years, and has a proven track record of dozens of satisfied customers worldwide, including lots of Fortune 100 companies, representing tens of millions of lines of code successfully migrated to .NET.

All this is reflected in the design of our products, where our R&D department frequently releases new versions incorporating the experience gathered from real world migration projects and the requirements requested by actual customers, rather than relying on theoretic research that commonly yields useless, "we-did-it-first" features.

And besides its powerful migration products,

Mobilize.Net provides extensive documentation and a full range of comprehensive, expert consulting services and dedicated support that will ensure that your overall experience will be as smooth as possible.



Call today.

+1-425-609-8458 info@mobilize.net www.mobilize.net