



## Modernizing the Full Value of Your Core Applications

Organizations with a portfolio of legacy IT applications face a confusing array of modernization options. Those with applications written in common languages like COBOL or Natural may be considering the use of off-the-shelf tools that convert source code to Java or VB.Net. The promise of fast, ready-to-go, fully automated conversion can be alluring but there are hidden consequences and risk factors. This paper examines the pros and cons of this type of modernization approach and presents an alternative that protects the full value of your IT investment.

### Your Modernization Needs

A key factor in selecting a modernization approach is to understand what you hope to achieve for your organization. Often the focus is on the technology change, since this is obviously a significant hurdle. The battle cry may be, "We need to get off the mainframe and onto a more cost-effective platform." Or perhaps the focus is on the impending retirement of your legacy programmers, so a change to .Net or Java will open up a larger pool of IT resources. Code conversion tools can help to enable such a technology change.

By asking a few important questions, however, you are likely to determine that your needs go far beyond a technological shift. Consider the business functions supported by the legacy application(s) in question:

- Do your business needs in these areas change and evolve over time?
- Have your IT staff members been kept busy maintaining this functionality in the past? Is there often a backlog of update requests?
- Is your ability to respond to new business opportunities dependent on how rapidly IT changes can be implemented?
- Will this need to be nimble be important, perhaps even strategic, for many years to come?

When considering core IT applications, most organizations answer a resounding "yes, of course" to these questions. Typically 85% of the costs associated with an application are in maintaining the application. This means the new versions of your core applications produced by your modernization program must be highly maintainable. This is crucially important for your enterprise.

Therein lies one of the key limitations of off-the-shelf tools: the maintainability of the resultant code. Such tools tend to produce code that resemble the legacy programs in structure and flow, and fail to utilize the advantages offered by the modern programming environment.

By necessity, such tools are based on the definitions of the legacy and target programming languages. They cannot take advantage of any characteristics that are specific to your



applications, such as naming conventions, where specific types of functionality tend to be located in the programs, or coding patterns used to accomplish certain common tasks. The tools must work for any program written in the legacy language, which means the tools incorporate a variety of simplifying assumptions.

The most common complaint of those faced with maintaining the modernized applications is that the code "looks generated." Names created for new variables or program modules are artificial rather than meaningful. Perhaps most importantly, the new programs tend to retain the organization of the legacy code. Often one can easily tell the source language by looking at the new programs, so that Java produced from COBOL looks like "JOBOL."

This is significant when modernizing to an object-oriented (OO) environment like Java or .Net. Modern OO programming practice dictates a certain structure for the source code, organizing the code into numerous classes that all have predictable characteristics and interact in prescribed ways. These characteristics are well understood by OO programmers and make it easier for them to read, understand, search through, document, and make changes to your applications. Program updates are performed using this consistent programming style.

Now consider a Java application that has been 100% generated from Natural using an off-the-shelf tool. A large percentage of the generated classes will simply be based on components found in the Natural code. Such classes do not have the characteristics expected by OO programmers. This makes it more difficult to interpret the code, and to predict how one part of the application will interact with and impact other components. Program comprehension has a huge impact on the time and effort required to maintain the applications, which means significantly longer timeframes to fix problems or implement changes to generated code.

Furthermore, this places your OO programmers in a quandary when adding new classes during maintenance. Should they add properly-designed classes to the application? Ideally yes, but then how will these classes interact with the generated classes that don't have proper characteristics? Again, the effort required is larger than it needs to be, which results in higher maintenance costs.

And here is why this should concern you greatly: this is not a one-time event. This extra time and effort represents ongoing costs that will be multiplied over the next ten or fifteen years during the lifetime of the new application. This will represent a continual degradation of the ability of the application to stay up-to-date with the needs of your business units, which likely means multiple instances where your business units will be later than they need to be in offering new and improved products or services. This reduces the potential value of your modernized applications for supporting the enterprise.

Off-the-shelf tools may offer a ready-made solution to the need for technology change, but beware of the ongoing costs that such a modernization approach can entail.



## To Your Standards

Experience shows that virtually every organization considering a modernization program also has other needs that expose the limitations of off-the-shelf code conversion tools. Here is a sampling of typical requests:

- The screens should conform to our user interface standards.
- The programs should conform to our corporate coding standards.
- Currently each one of our programs has security rules embedded within it. These rules control which users can access which functions. The conversion process should extract all of this logic and have it reside within a single module in the modernized application.
- We have analyzed the use cases for this application and decided upon a list of functions to be made available as web services in a service-oriented architecture (SOA). The modernized application should be organized accordingly.

These are representative of the organization-specific and application-specific needs that are part of most modernization projects. Tools that are based solely on the languages involved are unable to respond to this type of need.

As an example, one of Trinity Millennium Group's clients is a Federal Government agency. Trinity modernized one of their applications that has thousands of users worldwide. To eliminate the need for expensive and problematic re-training of all system users, the client required the new user interface to look and behave exactly the same as the legacy application, right down to the last keystroke, including tabbing between fields and using PF keys. Given that the user interface was migrating from Natural maps on the mainframe to ASP.Net on a web-based platform, no off-the-shelf tool could come close to satisfying this need.

Given the large quantity of screens involved, however, manual conversion would be costly and time-consuming. Automated conversion is highly desirable but must conform to the unique needs of this situation and must produce a well designed result. The key lies in the differences between a off-the-shelf tool versus automation that is refined and re-oriented as part of a given modernization project.

This is one of the key differentiators for Trinity's approach to modernization. Rather than applying predefined tools and hoping they come close to your needs, Trinity has developed a framework for conversion tool customization, including an extensive library of conversion solutions from previous projects. Our modernization specialists are able to quickly and cost-effectively re-orient these tools to satisfy the needs of a particular project.

For the Federal Government agency project mentioned above, our team developed the automated capability to convert Natural screen maps to Web-based screens. This includes the generation of VB.Net classes to provide an interface between the user interface and the rest of the new application. The new screens replicated the functionality of the old so faithfully that some users were reportedly unaware of the difference during initial testing.



Moreover, the generated classes are designed in appropriate OO fashion. This is possible in a case like this because we are able to examine the code for this particular application, understand the information content and structure of the specific components to be converted, map to a high quality result with all the characteristics desired by the client, and refine the automation accordingly. In comparison with off-the-shelf tools, we are able to achieve high quality results through automation because we are dealing with a smaller, more targeted problem domain.

Trinity is able to develop the automation you need quickly and cost-effectively because our code conversion framework is designed with one primary goal in mind – flexibility. We utilize a number of industry-standard and Trinity-specific notations to make it easy for our software engineers to specify the characteristics of your legacy applications, the desired target result, and the mapping between the two.

While not every organization would choose to replicate green screens in .Net, this is one example of how application-specific requirements can be met and still take full advantage of automated code conversion. An off-the-shelf code converter will provide a solution that was designed with no consideration of the specific needs of your organization. Accepting such a solution often means sacrificing those needs.

## **The Human Touch**

The screen conversion task mentioned above is an example where we were able to achieve 100% automated conversion. We have learned from experience, though, that not all tasks are best performed this way. Some application functionality can often be converted, for example, 80% with automation and then refined by hand. In general, the portions of legacy code that involve static definitions and predictable patterns are more amenable to automated conversion, while those involving complex inter-dependencies often benefit from conversion at least partially by hand. The desired technical characteristics of the new application also impact on the degree of automation that is optimal. We determine the split between automated and manual processing during our Transformation Planning phase, ensuring the proper balance between speed and high quality results. Trinity retains the human touch for the critical requirements analysis, OO design, re-factoring and code polishing tasks.

Trinity uses the term Automation-Enabled Modernization (AEM) to refer to this mix of automated and manual processing. By utilizing an AEM approach, Trinity is able to recognize and extract the business knowledge from your legacy applications. We do so by retaining the existing decision-making functionality embedded within your code, as well as the logic that supports your business processes. This ensures that important business knowledge is not lost during the application modernization process, and that the new application provides a familiar and predictable platform for moving forward with enhanced business function support.

# Trinity Millennium Group



## **Retaining Full Value**

The overall result is that Trinity is able to respond to your unique needs and to produce code that does not "look generated." The modernized applications we produce are truly maintainable by your organization, containing no proprietary components and designed to keep your long-term maintenance costs to a minimum. This provides a much more attractive overall ROI as compared with off-the-shelf code conversion tools.

We invite you to contact Trinity Millennium Group to learn how we can help you retain the full value of your legacy applications.

**877-615-1606**

**[www.tringroup.com](http://www.tringroup.com)**