5 myth-busting reasons for choosing an automatic migration vs. a manual rewrite

There are still many myths out there surrounding software migrations. For starters, there's the perception that *automation is just more trouble than it's worth*. Though this may be true for really small and simple applications, where the set up may not be worth the time, most conversion projects are large enough that there are easily enough economies of scale. In these cases, the initial setup might take several days, but then the migration tools can convert thousands of lines of legacy code in just a fraction of the time that it would take to do so manually.

Performance expectations lead us to another false impression: *software migration can be completely automated*. While it is true that there are amazing technical solutions available that automate most of the work, there are still some aspects that require human intervention. The most obvious is in the project management and analysis areas, but there are also some code characteristics that simply cannot be automatically upgraded. In many cases it's just a matter of the cost of the implementation not making the automation feasible, but in others a native translation may not be possible due to the differences between the source and target platforms.

Finally, it appears that some fall for the preconceived belief that *it's just plain better to start from scratch*. It might be the case when you are dealing with a somehow disposable system with useless, outdated functionality and no value at all. Otherwise, to concur with this idea is to simply devalue all of the effort and thought that was put into developing the application, willing to risk years of business knowledge embedded in these systems. The truth is, a rewrite from scratch implies a much more difficult task, even though some may claim that it is balanced out by the fact that you can significantly reduce the total amount of code and fix some of the imperfections that were present in the source application. But that's something you can also perform with an automated migration approach, and without incurring in all that risk, time and cost.

Due to these misconceptions, valuable resources are wasted in projects that sometimes just never get to see a successful ending. There's no doubt that any software renewal project isn't a simple, overnight task, but a well-planned automated migration can make the process comparatively painless. Besides, system upgrades are generally worth the effort. Of course they involve time and expenses, but it provides savings on the long run and a quick return on the investment, when your staff stops spending all that time maintaining legacy systems and developing workarounds to compensate for technology shortfalls.

So here are the main 5 myth-debunking reasons why an automatic migration is a far better software modernization approach than a manual rewrite, based not only on ArtinSoft's own experience in migration projects but on all the customer and industry analyst feedback and evidence gathered over the years.

Technology Availability: It's simple: many people just are not aware about the existence of automated migration solutions. Case in point, on a survey made by Microsoft earlier this year in the UK[1] almost 40% of the respondents answered "False" on the statement "*There are tools to automatically convert Visual Basic 6.0 applications to Visual Basic .NET*". And that's even when there is a free, though rather limited, migration utility that ships with Visual Studio, called the Microsoft Upgrade Wizard. So it was no surprise when more than 85% answered equally when asked about Visual Basic 6.0 to C# conversion tools.

But these tools DO exist, and have been helping companies and developers for years to leverage the investment embedded in legacy applications. Of course, as mentioned before, 100% automation is normally not possible, though some people still expect code translators to do all the work with no human interaction at all. There are normally manual adjustments required to achieve compiling running code in the target language, and even sometimes a real-world migration process is an iterative task where tweaking the source codebase beforehand is also anticipated and recommended. But technology has evolved to the point that there is hard to find a good reason now to carry a software transformation effort based entirely on manual labor. And fear of the unknown is just not a valid ground to discard potential time and money savings.

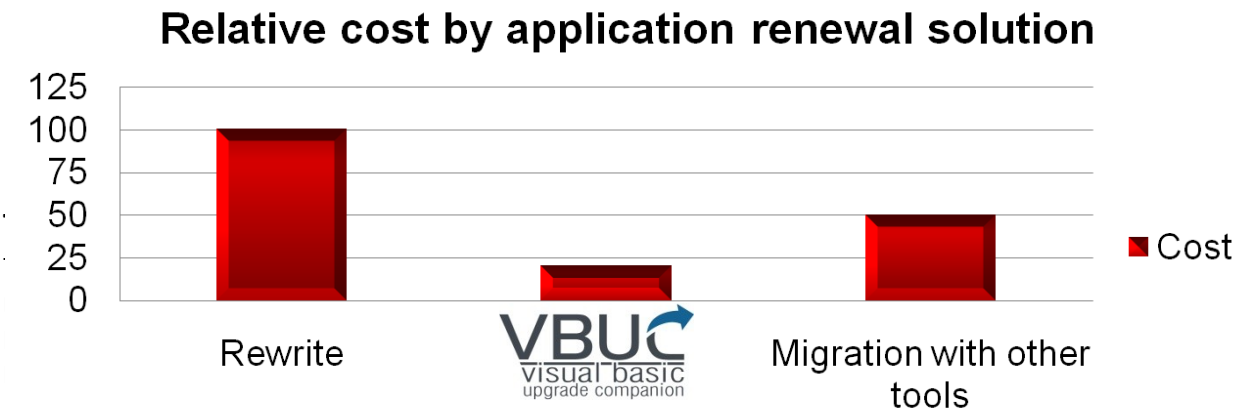
Code Quality: Automatic migration technologies today are able to generate code of exceptional quality, where you will not be able to tell if it was produced by a tool or by a vastly experienced human developer. So there's real value when you consider that they allow handling the process even when the people involved are not highly skilled in either the source or the target languages.

In the case of advanced VB6 to .NET migration products, extensive code analysis can be performed to detect patterns that can be upgraded to more .NET-like, native structures, making the output more readable and maintainable. This includes data type enhancements, grammar pattern transformations and detailed code improvements.

On the other hand, an automatic migration does not fundamentally alter the architecture of the original system, even if certain aspects like data access and some pieces of GUI architecture do change. But the output of a good migration tool, which needs to strike the right balance between automation and quality of the generated code, is ready for application enhancement and evolution. At least in the case of ArtinSoft's Visual Basic Upgrade Companion (VBUC), you will be left with native VB.NET or C# code, with no dependencies on third-party runtimes, plus the business knowledge that was present in the original application gets completely preserved, including all comments. This allows taking advantage of the new technologies available in the .NET Framework, and eases the subsequent development and maintenance efforts. The VBUC employs best programming practices, but it even has the ability to be extended and customized, which allows generating code that adapts to specific requirements, be it corporate policies or personal styles and preferences, increasing even further the percentage of automation. And even in the worst case scenario, where you still need to rewrite a certain piece of the application after the automated migration phase, the end result will always be a fraction of the total cost and time. This leads us precisely to the next points.

[1] Nelson, Eric (2009). *Results of the Visual Basic Survey: Part 3 Move from Visual Basic 6.0 to .NET.* Microsoft UK. http://geekswithblogs.net/iupdateable/archive/2009/02/17/results-of-the-visual-basic-survey-part-3-move-from.aspx

Overall Cost: When dealing with projects related to business applications, in the end it is all about money. So one cannot justify automatic migration based solely on technology itself, but on cost savings, and when we look at this variable there are several dimensions. The most obvious is the cost of the actual upgrade process, where it can be said based on experience that an automatic migration can be done at 20% of the cost of a manual rewrite. And most of that expenditure is due to testing and fine tuning of the application on the new platform. The graph below reflects the feedback from one of ArtinSoft's customers, who concluded that an automatic migration using the Visual Basic Upgrade Companion yielded 80% savings when compared to a rewrite (and 60% savings when compared to a migration using another tool).



Then there's also other savings with this approach that must be considered. Take for example the training of end users. Since the application that results from an automatic migration is 100% functionally equivalent to the original one, it is not necessary to retrain them. With a rewrite, chances are that the output is not going to be so familiar, unless you manage to follow an algorithmic approach like the one employed by sophisticated automatic migration technologies. In some cases, these retraining costs can be enormous. For example, another of our customers estimated that its system would have required a 6 weeks training time, for 3000 users.

All in all, using technology is viable as long as it helps reducing the overall effort. Even more in times of economic uncertainty, where it is crucial to save up on scarce, valuable resources. So minimizing the investment when evolving the infrastructure is definitely a must, and automation is certainly a good way of doing that.

Project Time: Since Time is Money, this has a close relationship with the preceding point. Normally, a legacy renewal project is a daunting task, but a migration tool can perform a very large part of the work automatically, in a fraction of the time and using much less resources. This can be better demonstrated by means of real-world examples.

Using the same instance of the customer above who compared application renewal solution costs, the following numbers just speak for themselves; a manual rewrite for this highly complex application was estimated to take 18 months, involving key personnel from various departments, while a migration with the VBUC required the allocation of much less resources for only 6 months.

		Automatic Migration	
	Manual Rewrite	**Other tool**	**VBUC**
Time (months)	18	18	6

One of our published case studies, featuring Banamex, one of the largest Mexican financial institutions and part of Citigroup, shows how they opted for an automatic migration with the VBUC because of the shorter migration lifecycle it offered, which in turn helped protect their market position. After an extensive analysis it came out as the most cost-effective solution, when compared to the other options they were evaluating in order to reduce the effort and investment.

Banamex considered rewriting 124 of their business critical VB6 applications, comprised by a total of more than 5,000,000 lines of code, and estimated that at least 175 and up to 185 developers would need to be involved in the project for a period of 6 years, in addition to the resources required to manage and coordinate the normal maintenance of these systems in the meantime. In contrast, using the VBUC allowed less than half that amount of resources to complete the job in just 12 months.

	Manual Rewrite	**VBUC**
Time (months)	72	12

Finally, there's another success story involving Vertex Financial Services in the UK, who migrated one 616,000 lines-of-code application from VB6 to C#. Being in such a competitive industry, they definitely needed to accelerate the time to market of their renewed application, in order to meet the next product release. And they accomplished that by using the Visual Basic Upgrade Companion instead of rewriting the application from scratch.

	Manual Rewrite	**VBUC**
Time (months)	22	9

Additionally, it must be noted that the above results were achieved using previous versions of the tool. The latest release is definitely much more powerful, allowing greater time-savings.

Risk: Finally, it is a well-known truth that, by nature, human activities are prone to mistakes. Add the fact that legacy applications and all the changes made to them during the years are not always well documented, and that the original developers sometimes are no longer available, and you have a very high risk of business knowledge loss when performing a manual rewrite. Plus there is always some level of uncertainty involved, which makes costs and project timelines difficult to keep under control.

An automatic migration process leverages capital investment made in business applications. That is, all customizations and business rules will be preserved in the migrated system. Plus, as explained before, this approach provides a much faster renewal lifecycle, and with minimal organizational disruption, which helps ensuring time to market that can provide organizations with a valuable competitive advantage.

_____

So it is a proven fact: using automated migration tools as part of an overall software renewal initiative is the most viable way to leverage the current investment in legacy applications and move them to the latest platforms. The technology available today in that field can produce outstanding results in terms of cost and time savings, risk avoidance and generated code quality.

Unfortunately, there are still many misconceptions surrounding automatic migration. But many have already discovered how this approach constitutes the most cost-effective alternative for legacy transformation projects indeed, and the word is starting to spread. Which is really positive, since myths can only exist as long as the truth remains unknown.