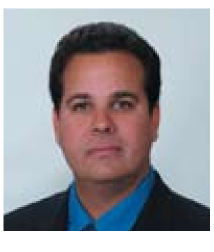
TOP 10 MISCONCEPTIONS ABOUT COMPONENT DEVELOPMENT

Components and software reuse are crucial for developers and architects. While most developers understand the value components bring to the table and use them effectively, very few know how to recognize reliable suppliers that will survive the test of time.With over 20 years of experience as a leading component vendor, Rene Garcia, co-founder and president of Software FX,Inc., provides a vendor agnostic analysis of misconceptions around components and software reuse in enterprise application development.



MISCONCEPTION #1:
Hey, we can develop that in-house!

Software development teams are usually conservative when it comes to the introduction of yet another tool into the development process. This fact leads to a natural resistance to use third-party components thinking this functionality can be easily implemented by their own development staff. The fact is that the software industry evolved into a set of specialized industries that are related, but where each industry has its own set of complexities and best practices. Simply put, thinking you could easily replicate the vast good will component vendors provide is not a smart assumption.

From an individual perspective, developers should invest in skills that increase their marketability. Components, just like OS and IDEs, fall into the "infrastructure development" category which requires a much different skill set as well as integration, performance and coding considerations that apply only to component building. This means, unless you are planning to develop and commercialize components, you will yield very few practical and strategic benefits from developing components in-house.

MISCONCEPTION #2:
Components will diminish our application's migration capability.

For starters, the components marketplace dynamics and the symbiosis that exists between larger vendors (e.g. Microsoft) and component vendors provides assurance you will have up-to-date components long before you make the decision of porting your applica-

tion to a new platform or framework. This simply means availability should not be a concern when using third-party components. Also, when it comes to portability, components allow developers to move from one system to another transparently and let developers focus their attention on more important aspects of application migration.

As a general guideline, you should select Component vendors who offer migration tools to ease the burden associated to moving an application to newer platforms and IDEs and prevent developers from breaking their original code.

MISCONCEPTION #3:
Components will slow down our application.

Too often, developers mistakenly pin down components as the primary cause of bottlenecks in their applications. Most commercially successful components are not only designed using best patterns and practices in mind as well as to conform to the platform they target (e.g.Web Controls take advantage of the server-bound nature of web applications) but also provide tune-up mechanisms and properties that allow developers and system administrators to easily boost performance under particular system configurations.

It is important to note that even if the proper components are chosen, if used improperly, the application will yield poor performance. For example, an ADO.NET object could cause a significant bottleneck if it is doing a complex or poorly designed SQL query. This does not mean that the object itself is slow, but that it is being used inefficiently.

From an individual perspective, developers should invest in skills that increase their marketability. Components, just like OS and IDEs, fall into the "infrastructure development" category which requires a much different skill set as well as integration, performance and coding considerations that apply only to component building.

MISCONCEPTION #4:
Components will make us dependent on smaller, vulnerable vendors.

In essence, the components marketplace provides support for easily removing one component and substituting it for an equivalent one. In addition, the component industry is a highly competitive marketplace based on features and price for many different categories in numerous distribution channels and across many geographies. ComponentSource, an online outlet for third-party software components, demonstrates the breadth and depth of the marketplace by listing over 1300 components in more than 100 categories and by over 280 vendors.

In addition, most component vendors offer special non-publicized services to support large deals and that will greatly reduce your dependability such as consulting, source code escrow, priority support and special pricing and OEM licensing.

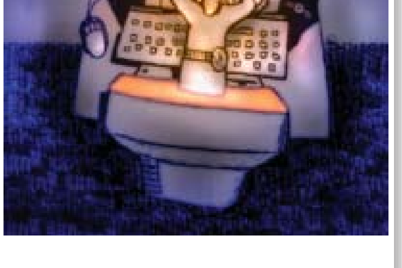
The bottom line is that while you will find your typical component vendor to be a small, privately-owned and self-funded organization, it is likely your preferred vendor has been in business (operating profitably) for over a decade. This serves as a testament of consistency and resiliency through numerous technology and economic cycles. This means that the notion of your component provider disappearing or going out of business is simply a misconception.

MISCONCEPTION #5:
Components provide no architectural value.

Architecture represents a large set of problems solved and a set of skills needed to construct applications. Components allow you to start new development by leveraging proven solutions and skills you can carve away big chunks of time, money and risk, even without writing a single line of code.

Measuring your component's architectural strength and quality is a little bit more challenging as components are normally provided in pre-compiled object form. You can use the following guidelines to infer your component's architectural value:

- The most important thing about well-architected software is that it can withstand change. Inquire about your vendor's ability to serve other platforms and IDEs.
- How well your component interacts and coexists with other components in your application. As a general rule of thumb, best-of-breed components from different teams (companies) should be able to coexist and provide a great UI experience.
- Vendors should be open to provide references about its customers and its development team. You should be able to get access to the product's architect and developers so they can address any of your concerns.
- Be careful of those vendors who only cite pricing differences as their only advantage; when buying components, always buy quality. It pays!
- Inquire about the extensibility mechanisms a component provides beyond what it is offered in the box.
- Be careful of those vendors who offer to change their code and compile special versions just for you. Branching is not a common practice among best-of-breed component vendors.



The bottom line is that while you will find your typical component vendor to be a small, privately-owned and self-funded organization, it is likely your preferred vendor has been in business (operating profitably) for over a decade.

MISCONCEPTION #6:
Components and software reuse are technical decisions that should be made exclusively by developers.

Components not only yield architectural and technical benefits but business benefits as well. In the world of mergers, acquisitions, rapid business change and emerging technologies it is critical to be able to integrate applications. It seems we often need to build a new application out of parts of existing ones, or make new processes flow into existing processes.

Alan Zeichick, Editor in Chief of SD Times, the leading newspaper for Software Development Managers, provides clear guidance on this subject, *"Don't choose components based on the number of bullets on a feature list, and don't delegate the vendor-selection process to a coder. That's not how your company should choose a long-term partner, someone whose code you're going to insert into your own shipping products as if it is your own. Choose component vendors based on a strategic partnership — that is, companies that you want to be in business with. That's the way to leverage the value of component reuse within your organization.*

The component vendor is a company to whom you've outsourced part of your software development. What does that require? Trust. Stability. Credentials. References. A solid architecture. A solid business. Real support that you can count on next week, or next year," Zeichick concluded.

MISCONCEPTION #7:
That looks awesome! I want that.

Fancy-looking components may look good in a magazine ad but when is time to look at them in context and have them coexist with other controls and components they may look extremely disruptive. Look for components that can live in aesthetical harmony with the rest of the controls and components already in your application. In addition, you should be able to build your component aesthetical layer from the ground up, not the other way around. So, stay away from demoware components that sell you on impressive aesthetical looks as soon as you drop the component on the design surface, you will rarely need complicated aesthetics and you will find yourself looking to undo those changes for hours just to get to your starting point.

The component UI metaphors should be intuitive and innovative but not as much as to negatively impact your application's interaction layer. If your component provides a UI, makes sure it integrates well with other aesthetic aspects in your application, such as icon size and styles, colors, toolbars as well as mouse interactions. Ultimately, you should be able to replicate via API or deactivate any UI specific behaviors to benefit the aesthetical and functional integrity of your application.

MISCONCEPTION #8:
All components provide only generic functionality. So any will do…

Components allow you to start new development by leveraging proven solutions and skills you can carve away big chunks of time, money and risk, even without writing a single line of code.

The best components are those that are designed and built upon an extensible architecture that support new and domain specific functionality. Before you choose a component, inquire about the vendor's ability and practices to support special requirements from its customer base; you may be looking for a good-looking 3D pie chart now, but once you start developing, you will have special situations that will either need to be addressed by code or even by special customization and sub-products from the vendor.

There are many vendors with great extensible architectures. For example, Software FX, a Florida based component vendor, provides an Extensions Pack that complements the already powerful data visualization their Chart FX generic components with specific functionality for banking, statistical and business intelligence. Also, /n Software (www.nsoftware.com), a component vendor based in North Carolina, offers specific products for FEDEX, USPS, UPS, Amazon and FDMS beyond an already rich suite of generic communications components for web based and smart client applications. These are all great examples of vendors with specialized products targeting specific functionality beyond the generic and that will significantly increase developer productivity and provide a competitive advantage to your organization.

MISCONCEPTION #9:
All components integrate to the IDE [Visual Studio] the same way.

Of all the fronts a component vendor must work on, design-time experience is the one with the greatest impact in developer productivity. IDE integration could be the difference between sorting through an object's properties list for hours trying to understand a components API and integrating the component to your application swiftly. Although there is a standardized .NET component framework, the Visual Studio extensibility kit allows each vendor to exploit benefits in the environment differently. As a general rule of thumb, you should look for vendors that help you understand the changes to a particular property with either graphical designers or real-time changes to the controls on the design surface.

Finally, look for vendors that use modularization to make this IDE integration through independent designer assemblies that have no impact on deployment. The last thing you want is to unnecessarily and negatively impact your application's performance and stability at deployment time with code that is not needed at run-time.

MISCONCEPTION #10:
The long-elusive goal of reusable components is not achievable.

With so many vendors and so many products, the problem is not whether software reuse through third party components is possible or not. Vendor selection is undoubtedly the most significant decision you can make before developing any mission critical application. Despite the negative effects of doing a quick Google search, some developers keep downloading the first component that touts the right buzzwords, ending in a situation that is less than favorable. Break this habit and do your homework, you will stand a much better chance at connecting with the right vendor that you will keep for many years to come. ■