

Why Aren't Developers Buying Into Third-Party Components?

BY SUSAN MESSEHEIMER

The arguments in favor of using third-party components—rather than building them from scratch—appear to be compelling. So why has the uptake among developers been slower than projected?

THE PROS

Vendors and happy developers alike point to several advantages of using third-party components.

The costs of third-party components themselves are small, and the expense of evaluating a third-party component and integrating it into your application is typically much less than the cost of building the same functionality yourself. Indeed, build-it-yourself costs can run high enough to doom a project. According to Mike Sax, president and founder of component provider Sax.net, “Any project with a limited budget and short time frame is a no-brainer for component use.”

“All the logic says that components should be used more frequently. But we’re not dealing with logic here; we’re dealing with psychology,” said Dan Appleman, CEO of component vendor Desaware. “Many developers just want to build it themselves, regardless of the economies of scale.”

This attitude gets support from what amounts to a fallacy: that software development doesn’t cost much. “Software is seen as being cheap to develop, especially compared to the cost of hardware development,” Appleman noted. “Of course, the true cost of software development is hidden.” And often a lot higher than it first appears.

Clearly, it’s almost always quicker to deploy a third-party component than to build the same capability from scratch. “We find that people are less likely to use components when there’s no real time pressure to complete a project,”



said Sax. “Sometimes it just seems more attractive to create your own thing because it will do exactly what you need—and for the developer, it’s often more satisfying to have built it. Of course, from a business perspective, there are few advantages to taking the time and resources to build in-house.”

Chris Tryon, manager of programming at New Jersey-based engineering firm Hatch Mott MacDonald, asked, “Why build a fancy grid, which may take weeks, when I can buy one in an hour, plug it in and use it in two? My time is the most important factor.”

Doing testing and quality assurance on new code can be expensive. Often you can try one or even a couple of third-party components—which have already been tested if they’re from a rep-

utable vendor—for less than it would cost to build and test your own.

After all, out-of-the-box components are typically more mature than those developed in-house—component makers are always upgrading their product, which is difficult to do internally. So the technology you’re using on your project is generally more mature, too. What’s more, third-party commercial component vendors document their products, which tends not to happen as often with internally developed code.

“There are virtually no real-world situations within a corporate environment where reusable components would not be appropriate,” said James McGovern, enterprise architect at The Hartford Financial Services Group. “They make the most sense for major industry—ver-

tical-specific applications such as claims, underwriting and policy administration.”

THE CONS

But the arguments against using third-party components are compelling as well.

For one, third-party components may not have the features you need, thereby restricting the functionality of your app, perhaps in critical ways. Or they may include features you don’t need, adding complexity (for example, greater support requirements) and an unnecessary learning curve—and therefore cost—to your deployment even if you don’t use them.

“It’s true that a third-party component—or any external code—will not always have the exact same programming

► continued on page 24

Why Aren't More Developers Buying Into

← continued from page 23

style as internally developed software," said Sax. "That's a small trade-off for the huge amounts of time that are saved and the fact that you get a head start with functional, well-tested code."

According to Appleman, the question to ask is, "Can the additional complexity and configuration be worth the significantly added cost of developing your own code from scratch? This has to be evalu-

ated for each case."

Another problem is that developers lose flexibility when they take an off-the-shelf component. "Many components assume that you program one way and one way only and

expect you to use the component their way and not the way you've developed your product," said Leo Mrozek, president of Illinois-based Mrozek & Associates, which designs and hosts Web sites. According to

Mrozek, most components are aimed at junior developers rather than at serious corporate programmers.

"Vendors often sell components to managers under the banner of increasing developer productivity—which can be especially useful in shops that employ primarily entry-level talent or are very lean," said Hartford's McGovern. "The value proposition changes accordingly for those shops that hire and retain top-notch IT talent."

David Yack, president of Colorado Technology Consultants, agreed. "A lack of extensions, which users need so they can configure or customize components' capabilities, keeps people from leveraging components more. Ironically, this flexibility is what would provide greater lock-in to a particular vendor."

Yet not all developers feel thus constrained. "The controls I have used [LeadTools and Infragistics' UltraGrid] have more flexibility than I could ever hope to master or use," said Hatch Mott MacDonald's Tryon. "Even with all this flexibility, I have found them simple to use; I'm able to add functionality with a line or two of code."

Then there are intellectual property liability issues. "Every piece of sample code on the Web that is in some way useful and is free has some kind of copyright or other message that must remain with the code," said Mrozek. "Many of my clients simply will not use this code, and many have coding standards against it. Components add another layer, as there could be copyright or patent info embedded in the component that the developer might be unaware of."

And although using a third-party component doesn't grant the vendor a copyright over the work that uses it, Gary Cornell, publisher of Apress, which is focused on the needs of programming professionals, admitted, "One reason component use is not as strong as it might be is simply that developers have to get preapproval for a project that uses third-party components. No approval is needed if you stay in-house."

Licensing issues also come into play with components.

It's amazing what you can do with the right tool!



con-sum-mate *adj.* (kon-suh-mit, kon-so-mot)
1. Complete or perfect in every respect
2. Supremely accomplished or skilled
3. Complete

*Pictured Spread 3 for Windows Forms.
Also available for COM and ASP.NET
development.

FarPoint Spread

- Drop-in objects, shapes, and graphics
- Calc engine with 273 built-in formulas
- Cross-sheet referencing
- Hierarchical views
- Enhanced printing options
- Column filtering
- The most extensive feature set available
- No runtime or royalty fees
- Free maintenance downloads
- Unlimited email support
- Unlimited access to FarPoint support forum
- 30-days of free phone support

A project isn't quite right when you don't use the appropriate tool. The same is true for application development. Without the appropriate tool your application isn't quite right. Why waste the time with components that have less functionality? Spread is all you'll need to get the job done right the first time. Spread is the **consummate spread/grid** development component. Log on to www.fpoint.com to **download a 30-day free trial** and check out our other great products.

FarPoint

www.fpoint.com
or 1-800-645-5913

Providing flexible, powerful and
reliable solutions for your
development needs.

Component Use?

"Many of the negative attitudes about third-party component use can be traced to people confusing commercial components with freeware—and they are most definitely not the same thing," said Cornell.

Most commercial component vendors, for instance, avoid using General Public License (GPL) or Lesser General Public License (LGPL) agreements, since the legally viral nature of GPLs and LGPLs obscures who owns what when components are tweaked or integrated with other code.

Sax.net's Sax acknowledged developers' confusion in distinguishing between commercial components from accountable vendors that come with a clear license agreement and (more or less) free code found on the Internet. "Professional, commercial component vendors have to evangelize their commitment to quality and legal transparency," he said.

Colorado Technology's Yack would like to see some evolution in licensing models. "Product licenses that require one license for each developer are not always practical for projects that involve a large number of developers," he said.

One of the biggest drawbacks to the use of components for developers is that there usually is no access to the source code. What if your component vendor no longer supports the component(s) you use, or has gone out of business? This makes upgrading or rebuilding your component-dependent app difficult—though these risks can be ameliorated by getting the component vendor's source code. *If* you can get the source code.

Mrozek said he won't use components from vendors unwilling to make their source code available to their customers. Happily, as Sax and Desaware's Appleman pointed out, many component vendors make their source code available to developers.

Yack has found ways to insulate his apps from vendor meltdown: "There are different techniques for using components to ensure you have built a wrapper to allow easy replacement by another provider," he noted. "For example, if you are using a component where there are multiple providers, and licens-

ing can vary among them from time to time, wrapping a layer of abstraction around them could benefit the business later."

Compliance and security also create challenges for development teams looking to

use third-party components. The Sarbanes-Oxley Act requires all companies to have documented proof of the software, hardware and platforms upon which their systems run. "Compliance and documentation might be an issue for components from smaller or independent component devel-

opers," Mrozek said.

Often highly secure systems and applications—including all of their parts and components—must be audited, which may not be possible if third-party components are present. "Will this component play well on the corporate desktop?" asked Mrozek. "Is there some-

thing embedded in there we don't know about that will bring up security concerns?"

THE UPSHOT

Despite performance trade-offs, licensing and intellectual property liability hassles, and compliance and security chal-

► continued on page 26

Easy to Use

ActiveReports features an easy-to-use, banded, fully integrated report designer with built-in wizards and converters, integrated toolbars, report and field explorer window, and detailed help instructions. The designer makes it easy to create the kinds of reports you need—from the most basic to the most complicated reports.

Download a Free Evaluation
www.datadynamics.com
 Standard: \$499
 Professional: \$1299

Easy to License

Licensing with ActiveReports for .NET is straightforward and easy to understand. There are no hidden costs, no extra licensing fees and no royalties charged for end users. Once you install the product after purchase, you are free to create and deploy your reports as needed.



Easy to Deploy

ActiveReports makes deploying your reports and end-user reporting capabilities easy. The reporting engine is provided as a single managed, strongnamed assembly. ActiveReports allows assemblies to be distributed using XCopy or placed in the Global Assembly Cache (GAC).

5870 Cleveland Avenue
 Columbus, OH 43231
 (614) 895-3142
 fax 899-2943

Why Aren't More Developers Buying In?

← continued from page 25

allenges, it looks like third-party components are becoming increasingly important.

Sax thinks today's users of third-party components "can be viewed as early adopters who are

interested in increasing efficiency, saving time, and gaining an advantage that their peers may not be aware of. Although components have been around for a while, it's quite possible that we're in the early part of a clas-

sic adoption bell curve."

Even Mrozek, who claims to be "in the camp of developers that tries to stay away from components at all costs," sees a solid future for them. "It will probably be another two to

three years before really productive and useful components are on the market that developers will want to use," he said.

He's not alone in that assessment. "I believe the reuse or leveraging of components

regardless of internal or third-party vendor is critical to efficient development of software," said Yack.

Components are especially helpful to developers who might not have a high degree of knowledge of a particular system or functionality. "The most efficient way for me to program is to reuse what I already know," said Tryon. "As a programmer, I need to know many different systems. It's hard to be an expert in each one. I use components because I'm not an expert in imaging software, but my clients demand that I be. The component developers have the ability to be experts in a specific field, thus increasing the productivity of everyone who uses their software, and I look like an imaging expert."

Aaron Marcus, president and principal designer/analyst at California-based Aaron Marcus and Associates, a user-interface and information visualization design consulting firm, offers an anecdote to explain the savings associated with reuse: "In the late 1980s, we worked with Kodak to study the value of reusing code in increasingly complex user interfaces. In the more complex UIs, if 20 percent of the code could be reused, then the company could recover development costs in four to five products."

Tryon agreed that third-party components are very cost-effective. "A programmer in my area billing \$120 an hour will never be able to create all the functionality of a component," he said. "Since components are continuously improved, I can add functionality without any coding. What could be better?"

THE RIGHT FIT

The trick is to pick the right component product, since quality can vary.

"An enterprise needs to determine whether the component's design integrates with existing systems," said McGovern. "Components should be selected from vendors based on terms such as quality, applicability for a particular purpose, access to source code, licensing restrictions (or lack of them), features and performance. These are things that ultimately help deliver valuable working software to your business." ■

Extraordinary data deserves extraordinary charting.

Advanced Charting, Advanced Results

Integrating advanced, award winning .NET Charting into your ASP.NET and Windows Forms applications is easier than you think.

To learn more, visit our website, download a full evaluation copy of Dundas Chart for .NET and experience our Data Visualization solutions for yourself.

www.dundas.com

Dundas
Chart
for .NET



Dundas
Data Visualization

Partnering
with
Visual Studio .NET
technology

Winner of the
'SDTimes 100' award
two years in a row.

SDTimes
2004
100

www.dundas.com
(416) 467-5100
(800) 463-1492

Development Solutions for Visual Intelligence™

.NET Tilts the Build-Vs.-Buy Balance

Microsoft's framework provides platform for new generation of business components

BY CAROL WEISZMANN

Microsoft's .NET Framework is changing how developers view third-party components. COM, .NET's predecessor, spawned a market for third-party components used in Visual Basic applications (usually written in C++, since VB is unsuitable for creating many kinds of components). A similarly successful third-party component market didn't emerge in the C++ world, largely because developers prefer writing code themselves.

But .NET is different for a couple of reasons. First, unlike Visual Basic 6 developers, those using VB.NET and C# can more easily develop their own components, which many prefer to do because it's so satisfying, even if it costs more. Second, the .NET Framework is so much richer in functionality than COM that, although it takes a while to learn, it already includes solutions that can obviate the need for either self-developed or third-party components.

Thus the choices with .NET expand beyond the build-your-own-versus-buy argument to include using .NET's existing capabilities. "The .NET Framework has really enabled component vendors to build a new generation

of business components that are used in mission-critical applications," said Mike Sax, president and founder of component builder Sax.net. ".NET components are much easier to cus-

tomize than the previous generation of COM/ActiveX components."

But beware: .NET is more complex than COM, and the newest version of a .NET component replaces its forebears

without any guarantee of the backward compatibility of newer runtime versions. So migrating your apps to a newer runtime version can mean upgrading all your components. ■

WHEN DEVELOPERS HESITATE

When an app is mission- or life-critical. "Components used in mission-critical or life-sustaining systems must be thoroughly vetted to a higher-quality standard than [is] typical of most component vendors," said David Yack, president of Colorado Technology Consultants. "Since you don't have access to source in all cases, it is not possible to perform independent verification."

When security really, really matters. "Use of components in security-sensitive systems can cause problems due to lack of ability to audit the implementation of the component," Yack observed.

When innovation is key. "Some of the components I use provide the

source code, but the vendor legally owns the rights to any modifications," said Chris Tryon, manager of programming at Hatch Mott MacDonald. So if you're building an innovative application aimed at providing a competitive edge, you might reject use of third-party components.

When your goal is performance and you're using a procedural language. "If one is writing infrastructure-related code using procedural languages to create device drivers, system utilities, etc.—and the goal is performance—it may be OK to simply meet the goals but be less rigorous in adhering to standards," said James McGovern, enterprise architect at The Hartford Financial Services Group.

When an application is very function-specific, not GUI-based, and small—such as software that takes only a few hours to write, according to Tryon.

When using components violates a contract. "Some contracts specifically don't allow use of third-party components or of code that is not part of the work product produced," said Yack.

When your organization isn't committed to them. "Component-based architectures shouldn't be thought of as a gradual thing," said McGovern. "You either embrace and practice it, or you don't. Components should reflect the spirit of an enterprise's business architecture."

—Susan Messenheimer



telerik.r.a.d.editor is the leading WYSIWYG rich-text editor for ASP.NET. The product can replace any <textbox> with an intuitive Word®-like editor, which enables even non-technical users to author and manage HTML content as easily as writing a document.

What's new in v4.0:

- "Section 508"-compliant content
- Enhanced interface and tools
- Four ways to strip Word formatting
- CSS-based skinning
- Improved HTML output
- New table management tools
- Code indentation in HTML mode
- Support for .NET v2.0
- Support for Windows XP SP2

r.a.d.editor is available as an individual component or as part of the r.a.d.controls suite for rapid ASP.NET development.

If everything in life could be that easy.

telerik
r.a.d.editor
the high-end WYSIWYG editor



The new r.a.d.editor v4.0 for ASP.NET. Flexible, robust, naturally intuitive.
Makes you wish you had something like it in real life.

www.telerik.com

Copyright © 2005 Telerik. All rights reserved. All product names are property of their respective owners. Microsoft is a registered trademark of Microsoft Corporation in the United States and other countries.