

A Roundtable on Application Modernization, Part 2

[System iNEWS Magazine](#)

System iNEWS Staff

Fri, 12/01/2006 (All day)

In [Part 1](#) of the Roundtable discussion, (November 2006, article ID 20710 at *SystemiNetwork.com*). our panel began by talking about the obstacles that prevent shops from modernizing their applications (e.g., lack of time, staff, and resources; unable to convince management to allot funds; insufficient control of current software; no straightforward way to do it). The conversation then turned to discussing the need for shops to take a tactical approach to moving their apps to the web, as well as the pros and cons of using WebSphere, CGI, EGL, and other technologies to move forward. Part 1 ended on the thought that when we consider a modernization effort, we must distinguish between external and internal apps, and to show an ROI to management, we need to focus on modernizing external applications.

Part 2 picks up on that thread, then goes on to discuss what you can do to modernize yourself, and concludes with a discussion about where PHP fits into the modernization puzzle.

Again, Wayne Madden, *System iNEWS* editor in chief moderated the panel. The panel consisted of senior technical editors Mel Beckman, Paul Conte, Sharon Hoffman, and Michael Otey. The panel also included technical editors Don Denoncourt, Nahid Jilovec, Scott Klement, Bryan Meyers, Dan Riehl, and Carson Soule and special guest contributor Carsten Flensburg.

Carson: We have to be careful here because there's a tendency to follow IBM's definition of modernization, which means buy WebSphere and put a graphical front end on it, and even if you don't buy WebSphere, put a graphical front end on it. What we're saying is that a lot of this is competitively differentiating types of code. Much of this is not a package. Most System i customers who I run into have homegrown applications that really make a difference to their company, and modernizing those applications means more than just putting a graphical front end on them. Often, these customers need database upgrades and other things that really do enhance their applications' value at the company.

Bryan: If you're going to try to convince me to spend five years and five million dollars to upgrade my applications, I don't think the path is clear. If I'm going to spend five million bucks, it better be with a strategy that will survive at least another five years beyond the first five. I don't want someone 10 years from now saying, "Oh, you're using that? Well, you're going to have to modernize it."

Don: I'm considered the Java WebSphere guy. To get Java to work, you have to set up and use complex frameworks, and there are many frameworks to choose from — and not necessarily the ones that IBM recommends. You need a Java expert to at least configure and set up your application infrastructure to be successful, and it can be very successful after that point. A developer with a modicum of training can be relatively proficient in extending that example architecture. But that aside, a lot of shops that don't have that developer on board are now looking at a variety of languages. PHP is now available for the i5, Ruby on Rails will soon be available (as soon as there's a Java version of Rails), and EGL is another alternative that Paul mentioned. All of us are actively looking at these different languages. I also want to ask Mike whether .NET is as complex as Java for frameworks.

Michael: I think there are fewer of the prebuilt applications around for .NET than there are for Java. As an essential language framework where you're providing things like mathematical classes and graphical interface classes, .NET has a comparable level of sophistication and capability. I personally do not find it as complex, but there's a major difference in the complexity of the development tools. The WebSphere tools are way too hard to work with. Microsoft's Visual Studio's interfaces are more intuitive. An average person can pick up on this and develop using Visual Studio tools. So there's a difference there.

Don: Do you think a lot of your clients should also consider PHP, EGL, and Ruby?

Michael: No, I wouldn't recommend developing in any of those. I would go with ASP.NET, but that's my preference. To get to a more general point, we talked about application modernization, and sometimes application modernization is out of the IT guy's hands. It's a management decision whether or not you're going to modernize those applications. Maybe the idea is to modernize yourself. Learn a new skill — one of the graphical development environments or something basic like SQL — and see how that skill might fit into a certain aspect of your applications. Learn it, and maybe you can make your new skill work in a given business problem as a way to move forward.

Wayne: I asked a question at one of our road shows that I thought got good traction and good response. That is: What is it that's the real shift for people when they want to modernize themselves? What is it that really shifts in them? It's not simply "I learned a language." It's not just conceptual things that they have to learn first. We know it's desire and commitment. Anyone can say that they don't have enough time. It's the age-old excuse when you don't want to do something, you don't find it valuable, or you don't consider it a priority. But if you're going to modernize yourself, there's a point where you say, "Oh, I know what I'm going to do now," and the rest is downhill. What's the downhill part for the average person in our market?

Don: I think Mike hit on that; you have to become, in Roger Pence's parlance, "a programmer's programmer." That's a big point.

Wayne: So, what's the shift to get there? What's going to shift in your thinking, your understanding, your knowledge, or your skills where you say, "Okay, I get it now."

Carson: My theory is that it may not be an "aha" moment, but instead "grow or die."

Nahid: I'm certainly in agreement with the thought that you should modernize yourself, but you must avoid the trap of learning a language and never using it. You need to have the ability to use it, so how do you overcome that?

Michael: No application is ever finished; there are always parts that people are working on to extend it to solve different business problems. Maybe one of the skills you learn is somehow applicable to one of these problems, and you use that skill to extend the application. I do integration work along those lines all the time.

Wayne: And you can lease space for \$9.95 a month to have your own website and play if you want to.

Mel: That's exactly what I was going to say. With all of the programming technologies that you have today, you can almost run them on your tie tack. They're easy to run on low-end platforms.

Carson: I actually like Mike's answer better. If you learn SQL, even if you're just writing RPG III, you can use SQL there. You can advance into RPG IV and learn procedures. You can learn new APIs and a zillion things you can use every day, and that goes all the way to extending applications to a browser. The opportunity is there.

Michael: The potential in SQL is something that many people don't realize. When you learn SQL, you can use it in RPG, but it goes beyond that. You can use it in PHP, in Java, in ASP, and in ADO.NET. SQL goes across languages much more than people realize.

Scott: I'd like to go back to what Nahid said earlier: People need to take this language and use it. Several time-share services are available if you have a System i-specific technology that you can't run on your home computer. For a couple of bucks a month, you can rent space on a time-share or a public space on the Internet; you can log on and use the time-share's i5 on your own time. There are even free time-share servers, though access is usually pretty heavily restricted on those.

Nahid: And the end goal is what? What are you going to create?

Scott: You could create an open-source application, or you could put up your own website and do some fun stuff with your site.

Paul: In my experience, modernizing skills, specifically in the System i program community, is a watershed piece of knowledge. And interestingly enough, these skills have been around for a long time but haven't permeated the community as much as possible — and that's especially true for modular programming. The ability to understand how to structure applications in a way that provides insulation from change, that promotes stability in terms of interfaces, that enables the reuse of code, and that factors out common parts of the

design into artifacts like procedures or modules are skills that people can learn and practice in ILE RPG as a start.

Scott: You can practice them in RPG II if you want.

Paul: This is true. But in my experience of teaching courses in this community, people don't learn as well if you try to give them an abstract concept and ask them to implement it in something that doesn't have a natural way to express it with its syntax and structure. That's one of the reasons it's valuable to learn a contemporary language like Java. Even if you don't get into the whole J2EE thing, you can learn the fundamentals of, say, Java or C sharp, which directly reflect some of these core principles in the language. It's then easier to bring those back into RPG, whichever variation it is, or into PHP or other types of languages.

But this is something that also has immediate value. No matter how you look at going forward, it's not just about [creating a graphical] interface — it's about being able to get more value out of the business code as well. And if we're talking about service-oriented architectures or the ability to set up business processes rather than transaction programs only, those essentially can be most effectively done if applications are well factored and highly modularized, regardless of what the language is. Here's an easy way to take your own pulse on this: If you're programming a lot in RPG and not using subprocedures, you haven't crossed that watershed yet. Probably 80 percent of our readers can use that measure. And then there are places to go beyond simply making that transition.

Carson: Just one more note — you can use Java on the i5 without having to use WebSphere. If you want to learn some Java language, you can write a called routine in Java on the i5, and there are plenty of places where doing so makes sense, such as accessing sockets, using JDBC to access SQL Server and other databases, and so on. So there's lots of low-cost ways to sharpen your skills.

Scott: JDBC is a good one. Using HSSF to create Excel spreadsheets is another good use for Java. I don't know about sockets; you can easily create those with the native APIs. A great place to start sinking into learning a new skill is to write some little utilities that don't necessarily need any screens, and to get some of the great free tools that are available for Java as a back end.

Sharon: Paul and Bryan made a great point: A big stumbling block is the way that people think about code in a traditional RPG environment. We think about an application, and it doesn't have a lot of moving pieces. All of these environments, and the Java frameworks are particularly intimidating in this way, have multiple moving pieces, so if you learn Java or HTML to name two core skills in that particular environment, you still can't build anything. It's as if someone gave you the ingredients but not the recipe. And the recipes are not straightforward.

One of the ways around that a little bit is to use some of the code generation tools. My experience with these tools is that they write ugly code, and they're not easy to learn from. What they generate has so much in it that you can't focus in on how to build something. And so you have to go back to the idea of building something small using all of those pieces. But a lot of this isn't about programming languages, it's about program architecture. For a long time, we had the luxury of not having to deal with program architecture, and some of us maybe never thought about programs that way.

Bryan: I think that's exactly true. So many of us for so long have dealt with applications in large terms — you know, this is a two-million-line application — and we need to start thinking in smaller and smaller terms. To think of a 20-line procedure that you can use a hundred times, rather than how you're going to fix this two-million-line application and modernize it.

Carsten: My background is in business, and what I've been missing in this discussion is that everything starts with the business requirement and the business strategy. And of course, that should lead to an IT strategy, and you should always look three years ahead. It might not be where you end up at the end of those three years, but you need to have some direction for what you're doing. And that should be matched closely with the business strategy.

For me, that's an important argument, and something that my company has been spending a lot of time on, partly because we've been acquired by a large global company that works this way. The company wants to know where we want our business to move before it puts up the money for us to do anything in IT. When we present spending proposals, we must explain precisely how our proposal matches the IT strategy that has been

decided for the company. If it does match, there's no problem. If it doesn't, we won't proceed. It's a simple calculation for us. That also constrains what we can do technically. We've discussed a lot of technical topics and issues here illustrating the many options. When we know exactly what we must deliver, we can start talking about the technology that will take us there. For various business requirements, that can result in very different types of applications and technical solutions.

Wayne: That's a good point, because businesses that operate that way see the vision, the strategy, and where they want to go, and they put the pressure on IT to come up with a strategy to deliver. There's a mix of that, but we're primarily in a small-to-medium business market, with some large shops, and many businesses operate much more informally. They actually ask IT, "What do you think we can do to make our business better?"

Carsten: There's a problem if the business asks IT where it should move; it's putting the cart in front of the horse, so to speak. That should be addressed first.

Wayne: As a final topic, what are your opinions on IBM moving to or adding PHP? What will be the key implementation factors to make sure it works so people can use it? PHP is currently the self-declared winner of the web development world, period. And at a 60 percent pace, PHP is outpacing any other choice for web development.

Paul: It's simply performance, performance, performance.

Carson: Well, let's look at what PHP doesn't have to do. It doesn't have to perform well — WebSphere proved that. It doesn't have to be easy to adopt — everything on the System i proves that. And it doesn't have to be a complete or full implementation early on. Furthermore, IBM and Rochester don't have to support it because CGI proves that. So, the bar is really, really low. And it seems to me that we don't want to change that. Only one thing needs to happen, and that's for Zend to write a decent database to plug in. If its database interface is full and clean, the community will grab PHP and run with it.

Wayne: Here's the issue: Mel, I've heard that IBM has given zero resources to Zend and telling it to show up with a DB2 interface, and Zend won't have it. IBM's message is "just load Linux, Apache, My SQL, and PHP." I can buy any box for that — that's not PHP on the i5.

Mel: Well, first, you can already interface to PHP using ODBC as a worst-case scenario.

Wayne: You can get SQL Server drivers and MySQL drivers for PHP. It's the real drivers, high performance, that Rochester writes, and what I've heard is that they haven't dedicated one to PHP. If IBM doesn't deliver a well-performing interface from PHP to DB2, I think that's garbage. Rochester isn't lifting one finger or one programming hour to help Zend — IBM just sent Zend off and said do it.

Mel: I don't see a huge technological hurdle to Zend doing this itself, because you can do it in C on the i5 and get a very tight binding, as tight as RPG is.

Scott: I completely agree with Wayne that that database piece needs to be there, as well as other integration with the rest of system and the ability to run CL commands and call RPG programs. Whether Zend or IBM does it doesn't matter to me. All of that is maybe going to be important for PHP to do well on the i5.

The other thing that annoys me about PHP is that I can't get a free DB2 driver that I can run PHP on to access the i5. I have to write some kind of strange interface myself that goes to the JDBC driver because IBM makes ODBC drivers only for Windows and Linux.

Wayne: Here's my problem: We have a chance to be in an advocate position that we haven't had in a few years. This is a huge issue. PHP could save the System i's bacon for IBM. Turn this around where it's a server of choice for people doing serious web development. So we need to have a voice in this.

Carson: It's more important to be integrated than to be fast. Mel, your comment about being able to get to the database, and it's a nontrivial interface, but it certainly can be done extends to the other integration items we would want. As Scott mentioned, we want the ability to call stored procedures and to get to data queues, message queues, and all the things that make an i5 unique. Couldn't a third-party vendor pull together an extension that you could compile into your PHP in a relatively straightforward manner?

Mel: Well, there's no PHP compiler, but you could build it as a module. But when people are writing PHP,

they're probably not going to use i5 artifacts that will tie them to that architecture if they can avoid it.

Carson: I don't agree at all.

Wayne: I think they will because it's better than the other things they have choices for.

Carson: Rochester's not going to win with generic PHP and generic Apache up against the rest of the world. It's not going to happen.

Mel: What main attributes of the i5 beyond the database would be the main draw for a PHP programmer?

Scott: Your existing business logic and your existing RPG, CL, or Cobol programs.

Mel: But you can invoke those now with PHP.

Carson: All we're saying is that integration has to be really clean, because it's hard to write a really nice front end to an RPG back end. When I talk to people, they like the way RPG and SQL (RPG VI particularly) integrate over DB2 to drive their business. They're not getting rid of the System i because it doesn't work well. The problem is that CGI is clunky and it's got issues, and WebSphere is clunky and it's got issues. How the heck do I solve this whole front-end problem without all the issues we spent the last hour talking about? And PHP holds the promise that it might be the solution, and if it is, it's gigantic.

Wayne: Well, it's gigantic in three ways. First is the existing customers. Second, even though IBM says it's modernized 2,000 applications, I haven't seen one yet, and you don't see many from ISVs. If the modernized apps were there, it would help the ISVs transform their applications much faster and preserve their investments. Third, we need to ask: Why would a brand-new customer who has a great idea for some industry and wants to develop an incredible web application want to use PHP? Why would he, instead of renting a \$200 - \$5,000 a year service, want to buy an i5? We need to identify those things that make the platform valuable, to turn the i5 around and make it a choice.

We need to publish a review on the PHP implementation when it hits the street this year, and that's what I'm striving to do. I want to send somebody to Rochester or bring [the PHP implementation] down to a machine, and then show up on the street with a review when it's released.

Carson: We're certainly interested in putting a lot of effort into that.

Wayne: Maybe the System iNetwork can write an open-source program to go with it.

Mel: We'll have some perspectives ready by the end of today.

Source URL: <http://iprodeveloper.com/application-development/roundtable-application-modernization-part-2>