LogicLibrary™

Demystifying SOA, Software Reuse and Metadata Catalogs –

A Roundtable Interview with Industry Experts

Introduction

This paper discusses the relationships among service-oriented architecture (SOA), software reuse and metadata catalogs. Their connection is explored through an interview with three industry experts: Dr. Jeffrey Poulin, Eric Marks and Brent Carlson, whose backgrounds are described below. Because of the roundtable format, questions were presented to the group instead of to a particular speaker, and there was no prescribed order for the experts' responses. The purpose of the conversation was to demystify and clarify SOA, reuse and metadata catalogs, so that they are easier to understand and to implement.

Brent Carlson

Brent Carlson is vice president of technology and co-founder at LogicLibrary, Inc., where he drives the development and delivery of LogicLibrary's products. Carlson is a 17-year veteran of IBM, where he served as lead architect for the WebSphere Business Components project and held numerous leadership roles on the "IBM SanFrancisco Project." He is the co-author of two books: *San Francisco Design Patterns: Blueprints for Business Software* (with James Carey and Tim Graser) and *Framework Process Patterns: Lessons Learned Developing Application Frameworks* (with James Carey).

Recently named to *InfoWorld*'s prestigious list of the "Top 25 CTOs" in business, Carlson is also a frequent presenter at industry conferences, including recent talks at Web Services EDGE 2005, Software Architecture Summit 2005, Rational Software Developers Conference 2005 and Enterprise Architect Summit 2005. Carlson holds 16 software patents, with eight more currently under evaluation.

Eric Marks

Eric Marks is president and CEO of AgilePath Corporation, a Web-services solutions firm based in Massachusetts. Marks focuses on driving executive insight, business planning, and the execution of Web services and SOAs to achieve business results and IT productivity. A software and technology services veteran with 17 years of experience, Marks has worked with PricewaterhouseCoopers, Cambridge Technology Partners, Novell, Electronic Data Systems, StreamServe, Ontos, Square D Company and Noblestar Systems. Marks is a frequent speaker and author on various information technology topics. His latest book, *Executive's Guide to Web Services* (Wiley & Sons), was released in March 2003.

Dr. Jeffrey Poulin

Dr. Jeffrey Poulin has over 14 years of experience as the technical lead on large systems development and integration projects. He currently serves as Systems Architect at Lockheed Martin Systems Integration. Dr. Poulin's landmark work on the business case for software reuse forms the basis for the reuse measurements and return on investment (ROI) models used by industry and government organizations worldwide. In addition, he has authored over 70 technical publications, including *Measuring Software Reuse: Principles, Practices, and Economic Models*, a book published by Addison-Wesley.

SECTION 1: Service-Oriented Architecture and Software Reuse

Question Index:

Q: How do you define service-oriented architecture?

A: At the highest level, an SOA enables the delivery of share-able functions that can be reused in multiple applications for significant technical and business benefit. To achieve share-ability, the functions must be properly sized, message-oriented and discoverable.

Marks: When I talk about service-oriented architecture, it boils down to a very simple concept: a computer architecture based on having shared, reusable services available to users. That's as plain as I can make the description of a service-oriented architecture.

Carlson: A definition I use in my technical presentations is that an SOA is an architecture that provides for the definition and delivery of core application functions through a series of coarse-grained services that are meant to be assembled through a message-oriented infrastructure.

There are some key points worth highlighting in this definition. First, you want to expose core application functions; you're no longer building monolithic applications. You're trying to derive pieces of functionality that can be assembled into multiple applications, which is a key principle that a service-oriented architecture needs to enable and encourage.

The services in an SOA need to be coarse-grained, meaning they do enough work to pay for the overhead of invoking them. Most service-oriented architectures are implemented in SOAP (Simple Object Access Protocol) and XML (Extensible Markup Language), which can be a verbose way to communicate. For example, it's inefficient to update a "customer ID" function with a SOAP call to determine when all of the items in an order will be ready. It is more productive to create a service that queries the status of an order based on delivery and shipping information in the system. This is a perfectly good definition and use of a service in an SOA.

And, last but not least, what SOA is really trying to do is enable message-oriented application development or assembly, which is the "Holy Grail" that tools vendors are going after.

Marks: It's critical for business users to have a mechanism for discovering the services available to combine in a composite application or complex, business process workflows. It's also important for the technical community to be able to find out what services and components are available for their purposes.

Q: Why is SOA so important from a business perspective?

A: An SOA is a powerful framework for responding quickly to change at a variety of levels – from application development and business-process revisions to mergers and acquisitions.

Marks: What's compelling about SOA is the broad range of strategic and tactical benefits that are enabled by its service orientation. On the strategic-benefit side, there's agility and time to market for business initiatives, new products and new services. A lot of clients we're working with are interested in applying SOA to mergers and acquisitions. It also enables easier, faster integration with business partners and customers.

On the tactical side, more specifically related to the IT community, there are benefits in terms of software reuse, efficiency and lower cost for integration across the enterprise. Implementing an SOA may help enable the **zero-integration enterprise**. In a nutshell, an SOA is important from a business perspective because it allows an organization to run its business and IT operations more flexibly than ever before.

In the **zero-integration enterprise**, IT and other aspects of the organization are explicitly tuned for rapid change absorption, leading to dramatic reductions in their integration costs.

Carlson: The main objective of an SOA is to give businesses flexibility in rapidly building out applications or automating business processes to support changes in the organization.

There are a lot of core functions in the business that don't necessarily change, like accounting practices and such. But, to be competitive or gain a cost-advantage, a company must be able to rapidly define new processes and implement them in their IT application infrastructure. This kind of flexibility and responsiveness is a large part of what's driving SOA, and organizations are starting to achieve it.

Q: What are the major challenges associated with delivering a successful SOA initiative?

A: Typically, the challenges involve establishing proper technical and business guidance that encourage adoption of an SOA. These include establishing a common SOA architecture, selecting services with the right scope to support the architecture, and providing both guidance to and governance over the users of the architecture. Sometimes, the challenges are more fundamental; an organization needs to first create the business-process model that will drive the SOA.

Poulin: The first major challenge is establishing a standard architecture that allows everyone to build applications that look and feel the same, reducing your development and training costs and making the programs easier to use across the enterprise. When the architecture is in place, you gain the flexibility of being able to easily move components in and out of applications when you need to adapt to change.

The second major challenge is effectively communicating what the architecture looks like and the services, components and assets that fit into it. This communication involves processes, procedures, tools and, finally, the governance necessary to ensure that everybody is doing what you planned for when you established the enterprise architecture.

Carlson: That governance, or aligning the service development effort against the architecture, is a key piece of what IT organizations need to do. It's also one of the major challenges in implementing an SOA. Technically, services must be implemented correctly, support security policies and provide the necessary scalability in terms of response time and throughput. But, you also need to make sure that the services' functional capabilities are very consistent with the business architecture.

Perhaps the biggest challenge in getting an SOA right is defining an appropriate level of granularity for a service's functional capabilities, so it's not just replicating existing application APIs and adding yet another layer of technology on top of an existing program. Finding the balance between specificity and flexibility is a tough nut to crack. You need appropriate architectural governance. And, the architect needs to interact deeply with the business analyst to extract the true functional needs from the business processes, instead of letting the analyst repeat the existing business flows and process that they're used to from existing IT solutions. These business requirements need to be translated into well-designed services that can actually be implemented.

Marks: Based on our field experience with many clients, there are three challenges we see time and time again. The first one is that executives are confused about where and how to get started with their SOA. We often spend a lot of time helping them define a process and a business architecture that supports their services objectives and business goals – before we even get to the reference architecture for SOA and a technical architecture. The second challenge is how to identify service opportunities across the enterprise. Most clients don't understand the differences among a service, a **Web service** and different types of services in a **service portfolio**. This confusion is a common semantic issue that we must address. The third challenge is in spending enough time on the organizational side of implementing an SOA, the governance model that will be used to manage it and the enforcement of the associated technical and business policies.

A **Web service** is a service with additional properties, such that 1) its messaging protocol conforms to SOAP, 2) it is typically transmitted over HTTP (although other bindings are allowed), and 3) its service interface is almost always specified by a WSDL document (which also describes where the service is physically deployed).

A **service portfolio** is the collection of business services in their enterprise, which is created as an organization develops the services it needs in its SOA.

Q: How is reuse related to SOA?

A: It delivers the "S" in SOA – specifically, share-able services that can be reused quickly for fast response to change.

Poulin: One of the cornerstones of SOA is having well-defined services. To deliver the technical and business benefits of reuse, you want to deploy your service once, then invoke it from multiple locations for use in multiple applications and contexts. Not all services can be used in more than one context, but if you do a good job of identifying proper services – not too specific and not too broad – they'll be used across your enterprise by many applications.

Marks: One of the first criteria to use in deciding whether to build the service is: how reusable is it across the enterprise? How much value does it have for multiple constituencies? That is really where software reuse and services reuse play into implementing an SOA.

Q: Isn't reuse the "Holy Grail" of software development? Can reuse really be achieved?

A: Reuse is happening now more than ever, which is being driven by four things: a strategic architecture, underlying technology consolidation, explicit reuse tools and business-user behavior.

Poulin: The term, "Holy Grail," implies something that will never be achieved, and I thoroughly disagree with that implication. I've been on and led many effective reuse projects, and I have worked with lots of different companies that are doing software reuse very successfully. Impressive levels of reuse (and corresponding benefits) can be achieved if you take on the challenges we discussed earlier. Even with a modest reuse effort, achieving levels of 15 percent to 20 percent reuse on a large development project is very achievable. I've done it many times.

Carlson: Speaking as a technologist, Holy Grail is a real misnomer because we've been doing reuse all along. In fact, the level of reuse that happens without people even realizing it continues to grow over time. If you think about it, the initial set of C subroutines that somebody wrote to handle lower-level functionality is reuse; people just didn't recognize it as such. There is a huge amount of reuse that occurs at a technical level when you move into a modern component-based infrastructure and an architecture with a service-oriented capability layered on top of it. For example, I don't have to write transaction controls anymore because these capabilities are given to me in J2EE and .NET. I don't have to deal with directory naming issues because those are addressed through a set of APIs in the infrastructures so I can concentrate on higher-value activities.

Technology consolidation also facilitates reuse by constraining the set of things we have to work with. And, reuse tools have evolved away from improvised methods like word-of-mouth, over the cubicle wall and slap-a-spreadsheet-on-a-fileserver where you hope people figure out what's going on. Professionally developed systems, explicitly designed for reuse, such as LogicLibrary's Logidex, allow you to better govern, manage, distribute and track service usage and asset reuse.

SOA takes reuse to the next level – into the business arena – and employs it to deliver significant business flexibility.

Poulin: You asked, "Haven't people tried reuse before and had it fail?" Look at client-server applications that have been around for a very long time and implemented with remote procedure calls (RPCs), CORBA (Common Object Request Broker Architecture), etc.

They're all essentially componentized architectures and message-pointed communications, which is exactly what we're talking about with SOA. Service-oriented technology is a big step forward in making reuse really achievable with less effort. The big difference now is the consolidation of underlying technologies. Rather than having lots of incompatible technologies that compete and make it difficult to share components, we're standardizing on J2EE and .NET.

Marks: The barriers to reuse have really fallen away with the rise of standards for services and enabling component platforms. But, what's really compelling now is that there's a business community that can reuse business services in addition to the IT community that's sharing components and services. Suddenly, there's a massive user base of people reusing assets, which creates a culture of reuse that is reinforced with policies, tools and processes. That's what makes reuse really achievable now.

SECTION 2: Metadata and the Metadata Catalog

Question Index:

- **What is metadata? Why is it important to the software development lifecycle, reuse and SOA?**
- **Exactly what are the differences between data-level metadata and software-development-level metadata?**
- **How will a metadata catalog enable software reuse and, therefore, my SOA initiatives?**
- **If the development team is only using a few Web services, wouldn't a metadata catalog be overkill? What would they gain from using a catalog?**
- **Can't I just use a portal program like Microsoft's SharePoint (or some other home-grown collaboration software) and mandate the use of best practices and common components to achieve the same results?**
- **What about UDDI? Doesn't it provide the same benefit as a metadata catalog?**
- **Doesn't IBM, Microsoft or another major vendor offer metadata-catalog functionality? Shouldn't I use theirs since I have other software from them already?**
- **Wouldn't it be easier to manage all the types of metadata with an all-in-one system than a best-of-breed solution specific to one kind of metadata?**

Q: What is metadata? Why is it important to the software development lifecycle, reuse and SOA?

A: In the case of reuse, metadata is essential for explaining the asset from all aspects, so the asset is easier to find, reuse and ensure its compliance with an SOA.

Carlson: Simply speaking, metadata is data about data. It describes the information that you're trying to manage. When we're talking about software reuse, metadata refers to information about software-driven assets. Asset metadata describes the work products of an asset, known as its artifacts. For example, metadata will tell you where an asset's artifacts live, which is typically in a point-tool like a version repository, control repository, requirements management tool, defect tracking tool, etc. The metadata references this information.

Of course, you also want to understand how an asset relates to other assets and be able to manage those asset relationships as well. Metadata tells me about previous versions of the service and whether I need other components to deploy it.

Poulin: Metadata is essential for helping you find an asset and showing you how to use it.

Carlson: To this point, metadata also makes an asset searchable and reportable through classifiers. Classifiers are typed bits of information, such as this was written in Java; it is meant to support account management in our business domain; and it took us three-person months to build this service, which helps calculate ROI.

Metadata is the only way to understand what assets you have, how they relate to your environment, and how to manage them.

Q: Exactly what are the differences between data-level metadata and software-development-level metadata?

A: They describe data at different points in the "asset chain" and require different management tools. However, they are complementary because of their chain connection, with one link in the chain leading to the other. Work with data element (aka information model) metadata tools often produces business constructs, or entities, that are used when developing software development assets, such as components or services.

Carlson: Software development assets are on a continuum that ranges from very large knowledge assets, like corporate architectures, development processes and design patterns down to data-entity information, which is information about the data an organization is managing. All these things need to be managed; it's just that different tools are appropriate for different assets on the continuum. For example, Logidex focuses on coarse-grained assets that are produced from data on the more granular end of the continuum, such as data metadata and element metadata. There are data-element repositories that do a great job of managing data-element metadata. Their primary focus is helping the organization define a **canonical data model**. The entities generated by a data-element repository are perfect candidates for a software-asset metadata catalog like Logidex. They are all valid assets that I would like to expose to my developers through a metadata catalog.

Marks: In the field today, there's a lot of work being done to build out canonical data models within an enterprise to support their services and asset-reuse efforts. You need to manage all of these kinds metadata with the appropriate solutions, but they need to be federated to support the larger, greater good of an organization.

"Essentially, a **canonical data model** is a data model independent of any application. For example, a canonical model might use a standard format for dates, such as MM/DD/YYYY. Rather than transforming data from one application's format directly to another application's format, you transform the data from the various communicating applications to this common canonical model. You write new applications to use this common format and adapt legacy systems to the same format."
java.sun.com/blueprints/guidelines/designing_webservices/html/integration5.html

Q: How will a metadata catalog enable software reuse and, therefore, my SOA initiatives?

A: Assets housed in a metadata catalog are easier to discover, use and maintain. This supports reuse and my SOA by making the assets highly consumable.

Carlson: A metadata catalog allows an organization to proactively manage their metadata and make it available in a highly consumable manner. Otherwise, unfortunately, metadata is just gathered together in a static form, like a spreadsheet, which is not very manageable or distributable and tends to get stale. A good metadata catalog should enable an organization to do three things:

- Flexibly define the relevant metadata to be managed about a software development asset;
- Flexibly specify the process for publishing an asset into the catalog, along with associated governance and review mechanisms; and
- Make it easy for downstream consumers to find a flow of assets to build applications, business processes, workflows and the like.

Logidex helps organizations produce quality reusable assets through its configurable governance processes and templates and enables downstream application developers to discover those assets through its direct integration with integrated development environment (IDEs). We focused heavily on integration with IDEs because that's where developers live. If a tool forces developers out of their preferred environment, they'll be more resistant to doing the right thing and finding the assets that they should be using.

Marks: One of the main functions a metadata catalog fulfills in an SOA is discovery. If I'm discovering services or software assets through a metadata catalog, then I have visibility, management and control of them.

Poulin: Metadata about existing components makes it easy to find them and helps you determine if the component is suitable for use in your next application. Obviously, if you're not aware of pre-existing components, you won't be able to reuse them. A metadata catalog, coupled with other tool support, allows you to locate components based on your description of what you need.

Q: If the development team is only using a few Web services, wouldn't a metadata catalog be overkill? What would they gain from using a catalog?

A: A metadata catalog adds value regardless of the number of services in it, because it enforces reuse policies and behavior and increases asset consume-ability. If people are successfully reusing a service, they will inherently expect two things: that it will be maintained much like a software product and that there will be more services for reuse over time, both of which require catalog functionality.

Poulin: A traditional mistake is judging the success of a repository based on the number of components that are in it, which is really not the proper measure. The most successful repositories that I've seen are ones that have relatively few components – ranging from 50 to a couple hundred, say 200 to 350. The success of those repositories is based on the number of times that people use the services in them.

So, what do you gain from having a repository, even with a small number of Web services or components – or in all situations? You gain all of the other features that the tool provides to manage the assets, for example, configuration management (CM). CM is important because even with a small number of Web services, every time someone requires a change or needs to fix a bug, the tool's change management tracking can automatically communicate the changes to all the asset users so that everybody implements the changes properly before they have the same issue affect them.

Carlson: A catalog also helps instill the "right way" to do things as you move forward into a major initiative like SOA. It puts the proper tools into people's hands. And, using the tool teaches them how to follow the process. This upfront training is important even if you have only one service that people are using. It's really valuable because, then, people don't even think about not doing the right thing.

A metadata catalog also enables downstream version management and fact analysis. And, it provides traceability about whether your services and their follow-on versions are being used immediately. If you don't have that traceability, then you've lost the ability to definitively move your architecture forward.

Marks: This is, again, a very interesting question, and I see it a lot. I've addressed it as a concept called, "the SOA network effect." The idea applies Metcalf's law to an SOA, which says that the value of an SOA is proportional to the square of the number of users *and* the number of services. Doing this math shows that a metadata catalog gives you a way to enable either more services or more users – and preferably both. If you've got a few services with many users, it's still equally valid to have a service repository in place for people to find them. On the other side, if it's a lot of services with relatively few users, a catalog gives you discovery and management control of the services available for reuse. At a certain point, putting a service in the catalog enables more services and more users to drive the accrual of the benefits from reuse.

Q: Can't I just use a portal program like Microsoft's SharePoint (or some other home-grown collaboration software) and mandate the use of best practices and common components to achieve the same results?

A: Collaboration software like SharePoint may not meet requirements, such as controlling the reuse of internal and *external* assets, and it could increase maintenance costs.

Carlson: The key issue is achieving the same results. You're not using the right tool for the job. Of course, you can force any tool in any situation. If you're really, really, really lucky and people work really, really hard, maybe you'll get some decent results. In any sort of practical situation, you need to apply a purpose-built tool to fully address the needs of the organization.

Poulin: You also need a proper tool to get the full benefits of reuse. You can't deny that informal reuse is good and, as we said earlier, we've been doing reuse since the 1960s on some level or another. The truth of it is, though, unless you've got a formal reuse tool to help control, manage and advance your business of developing software, you're not going to achieve the larger benefits of reuse.

For example, one of the largest costs in the software business is maintenance. With informal reuse processes and collaboration tools like a SharePoint, you have no way to manage the proliferation of copies of the software made by users of the tool. However, if your assets are controlled and communicated through a repository, you can have one copy that everybody uses. That is how you make a significant improvement in your long-term maintenance cost.

Marks: The main reason to use appropriate tools is that you need to enforce reuse policy and SOA standards internally and externally – internally, at design time for the developers, and externally, at run-time for services and assets from outside your organization that will be running within it.

Q: What about UDDI? Doesn't it provide the same benefit as a metadata catalog?

A: UDDI complements a metadata catalog instead of replacing it. Both are necessary for an SOA.

Carlson: UDDI is a valid part of an SOA. Some organizations are choosing to use it; others are not. Its value is really in the operational phase of development. I'm going to differentiate here between registry and repository because there's a clear difference. UDDI provides a registry for deployed services, indicating what things are available to execute against, much like the registry inside your Windows operating system shows the programs that are available to launch and some of their configuration settings.

What a UDDI does not provide is governance over design-time and development-time issues in production and consumption, which is one of the key purposes of a metadata repository. A metadata repository is there to manage the asset metadata that is produced and consumed as a result of service production. The services that are produced are then deployed into the run-time environment, which may include populating a UDDI registry.

Marks: UDDI registries have been extended to do other things beyond managing the pointers to various services available in an SOA. The main reason people are looking to the UDDI registry is that they tend to be integrated with Web-services management platforms for fail-over, backup service and ensuring a consistent operational, online run-time environment. You really need both a metadata catalog and a UDDI registry to implement SOA appropriately.

Carlson: Again, the key point is that a UDDI supports the operational policy and deployment environment whereas a repository lives in the development space. However, they tend to be bound together. In fact, Logidex can govern and automatically populate a UDDI registry at different points in the service development lifecycle.

Marks: That's a fantastic approach.

Q: Doesn't IBM, Microsoft or another major vendor offer metadata-catalog functionality? Shouldn't I just use theirs since I have other software from them already?

A: None of them offers a metadata catalog; they have instead partnered with LogicLibrary to provide this essential function. In fact, IBM has selected Logidex for use across its entire software division, and Logidex is the only third-party product hosted on the Microsoft Developer Network (MSDN).

Marks: It's a matter of focus. IBM and Microsoft have a lot of things going on, so I just don't see them being able to concentrate as well on the reuse, metadata-catalog space. An organization like LogicLibrary, with its specific emphasis on this area and associated metrics, is going to provide a far better outcome for a client. I really look for organizations that have the proper focus to support a client's effort, which will lead to a great outcome.

Carlson: IBM's recent acquisition of Ascential Software is focused around data warehousing, which relates to the data-element, metadata environment. This is certainly a useful function, but it's not competitive with Logidex. We are a strong partner of IBM. IBM recognizes LogicLibrary as such and even purchased Logidex for internal use within their software development organization. Likewise, Microsoft sees us as a strong partner that fills a white space in their application development tool suite. When Microsoft talks about white space, it is an area where they have explicitly chosen not to provide product as part of the Microsoft tool suite. They are very happy to have us working alongside them.

Q: Wouldn't it be easier to manage all the types of metadata with an all-in-one repository than a best-of-breed solution specific to one kind of metadata?

A: Different types of metadata require different repositories, just like different development activities require different tools. A properly designed, best-of-

breed tool will naturally gravitate towards another complementary tool to build necessary integration.

Carlson: There are always people who want to buy a single, "total" solution because they think it's going to be cheaper, or they wish there were one solution to all the world's problems. In reality, the types of activities in the different software development spaces vary widely and, therefore, tend to have specialty tools. So, it really makes sense to think about bringing in the right tool for the right job. If these tools are implemented and architected properly, they're going to support cross-integration. In fact, the tools vendors are going to seek out the best-of-breed solution in a complementary space and build integration.

Marks: An example of this is how Logidex is able to federate with UDDI registries. Functionally, there are good reasons for separate, best-of-breed reuse tools. In terms of management, it would be nice to have it all visible in a unified view of the various solutions.

Poulin: The primary thing I look for in a best-of-breed repository is ease-of-use, and the key to that is integration – tight integration with your software development environment. Logidex is unique, I think, in how tightly it integrates with IDEs, like Microsoft Visual Studio and IBM's WSAD (WebSphere Studio Application Developer) and Rational Application Developer. Anytime my developers have to leave their environment where they work everyday to look for reusable components in a standalone repository, they won't do it. That is the traditional reuse model and it simply does not work. But, if you make it easy for them, with the help and support that they need embedded right into their environment, they will.

SECTION 3: Governance and a Culture of Reuse

Question Index:

- **Should a reuse process be in place before implementing reuse technology?**
- **How should a metadata catalog support a reuse process?**
- **Can reuse technology provide governance over SOA projects, like improving developers' consumption and management of services, controlling the assets that are accessed and deployed in SOA initiatives, etc.?**
- **How do you effectively create a culture that embraces reuse technology and processes? With a carrot or a stick?**

Q: Should a reuse process be in place before implementing reuse technology?

A: Actually, there are a lot of advantages to piloting the process and technology together.

Carlson: It may actually be better to approach the process and technology together because it minimizes the chances of "analysis paralysis." If an organization tries to define the perfect process, they could be sitting on the dime for months or years before they execute anything; and when they try to execute, it's not going to be right anyway. The reality is that you can make a lot of progress doing some analysis at first, and then putting what you have into practice.

We strongly recommend that an organization define a reuse process and then pilot it with a project and a formal reuse tool. They will learn so much from the pilot that they can use to refine the process. They'll quickly get down to the right process after a couple iterations. And they'll be creating an environment that supports the process in the bargain.

Poulin: Obviously, you can do lots of reuse without a specific reuse process in place, but the process is what eliminates the proliferation of "cut-and-paste code." With a process, you're able to control what components you're reusing and reduce maintenance costs because you no longer have multiple copies of software out there. The best way to determine a reuse process is to pilot one and see what works for you.

Marks: We often develop both at the same time as we work with our clients. For example, we've been working with a number of organizations on "service mapping" to create a services chain that shows services opportunities across the value chain. This is essentially a top-down approach that creates a services view of a business' value chain. If there's not a component model in place for those services, then we flip it over and approach it bottom up, from a component point of view, to map components and transactions to the services. Then, you can actually make a decision about where it's most advantageous to start the reuse process. Where is the biggest bang for the buck? Is it working the component and software-asset reuse side of the equation first? Or, is it building the services layer, the SOA and Web services side first? Or, do you just go for the whole thing and do both at once? There's compelling value with any of these approaches. But, tackling it all together means you have to have a good process for getting it done.

Q: How should a metadata catalog support a reuse process?

A: A metadata catalog should provide automation for easy compliance with reuse, flexibility to fit a variety of cultures and metrics to show results from the process.

Carlson: A repository should, in fact, make it easy to implement governance processes and provide as much automation as possible so that it's easy to do the right thing in all respects.

Marks: To successfully drive a process, a repository must cleanly integrate with existing development behaviors and cultures, lifecycle processes for software development and release, QA and production. The repository's got to help a process become easier and less intrusive to the developers' normal, day-to-day behaviors while still enforcing reuse policy and good design.

Poulin: At the end of the day, the executive or development manager that invested in these processes, procedures and tools wants to know what the payback was. The repository needs to provide metrics that allow you to build the business case and say, "See, we are getting a lot of value out of what we're doing here."

Q: Can reuse technology provide governance over SOA projects, like improving developers' consumption and management of services, controlling the assets that are accessed and deployed in SOA initiatives, etc.?

A: SOA begets reuse, and reuse technology supports governance by automating and enforcing the reuse policy of an SOA.

Marks: I think the answer is yes, but in a different order. I think the SOA governance results in a policy for asset reuse not vice versa. Nonetheless, asset reuse is part of an explicit overall SOA governance model. It's one of the critical, core policies that must be enforced in governing an overall SOA.

Poulin: Any large software organization that claims to have control over its development processes must do it through a series of well-defined inspections and reviews. This starts early in the project, at the requirements level, and continues at the architecture level, the high-level design level, and so on.

At each of these steps here, you're prescribing to your development teams how you want applications to be built. You are also ensuring that they're built the way that you want – to incorporate reusable assets.

If your organization doesn't have this kind of governance or review process in place, and a new reuse initiative helps you create one, then you're winning on both counts.

Q: How do you effectively create a culture that embraces reuse technology and processes? With a carrot or a stick?

A: Both.

Poulin: It's a little bit of both and depending on who you're talking to. My background is strongly in the area of measurement, which can be both a carrot and a stick. If you're measuring someone and they're doing well, they'd consider it a carrot. And if they're not doing well, they'd consider measurement a stick.

Ideally, to have a successful program, you provide a way that everybody can succeed by doing what you want them to do. For example, one technique that we used to encourage the reuse program was to tax the development team some small amount of their budget, say ten percent, in order to fund the tools and the processes in the common components. The managers aren't usually very happy about that, but they realize the only way they can get their job done is to use the components that you're providing for them. The way to succeed is clear. As they perform, you're able to encourage them with the "measurement carrot" by measuring exactly how well they're doing against their reduced budget. Taking a small tax of ten percent off our development organization, when you set a reuse goal of 15 percent to 20 percent, is a very good trade. As I said earlier, this range of reuse is very achievable, so development is paying for something that's likely to succeed.

Marks: You need a carrot out there that results in benefits that people can experience from complying with reuse. And, you need a stick that says, "If you don't do reuse, or you do bad reuse, there's a downside." You also need to back this up with proper solutions that facilitate a culture and process of reuse. If you can automate the enforcement of a reuse policy with appropriate tools, backed by the carrot and the stick, and support it with education, training, and mentoring, then you have a recipe for success.

Carlson: Every organization is going to respond differently to a new process based on its culture, politics, etc. I do think you need a mix of positive and negative, but it has to be adjusted accordingly.

SECTION 4: Getting Started with Reuse

Question Index:

- **What are the key elements to getting a reuse project off the ground? Who needs to be involved?**
- **I have hundreds, if not thousands, of software development assets and artifacts. Won't getting started be a long process? How can I get value right away?**

Q: What are the key elements to getting a reuse project off the ground? Who needs to be involved?

A: The project should be tied to: (1) a high-priority issue, (2) a cross-functional team, (3) executive advocates and (4) participants who evangelize reuse results.

Marks: It's clearly a cross-functional team. I like to see it start with a core team that is backed by executive sponsorship at the "C level." Ideally, the team has architects, developers, and business analysts with visibility into the business context. It's also good to have a business owner that's driving the project. You've got to have this kind of a core team model to get results quickly. You can plan for the core team to dissolve over time, after which, hopefully, you've got a robust process that continues to live onward.

Poulin: The involvement really needs to be at all levels. Starting at the top, you need business support from executives. You'll also need support from your architects to provide the technical perspective and define what the application will look like. They also specify the reusable components and how they'll be reused and interact.

One thing we haven't touched on yet is the part-time commitment from individuals from each of your development teams to play a regular role in the repository and the reuse process. It doesn't have to be a full-time role, just enough involvement so that they can go back to their projects and communicate about the reuse effort to the rest of the developers.

Marks: In all these cases, there's got to be an overarching imperative for embarking on a reuse initiative. It's either got to be a business or IT imperative that must be addressed for the organization to continue to thrive – that must be fixed or else. The project's internal champion should package it up as something that enables multiple benefits – business, IT and technical, so the project becomes strategic because of its scope of influence and how much it enables. This also means there's a higher likelihood of an immediate ROI.

Poulin: Start small. Select a few components and a small project. Prove the value. Even on a small project, the documented savings that you will be able to show are just overwhelmingly convincing to management as to the value of reuse in service-oriented architecture.

Marks: If you have a business context wrapped around the pilot, the success will sell itself, but if you get a technical result without business context, it won't get as much attention and won't serve as a catalyst for a more rapid rollout.

Q: I have hundreds, if not thousands, of software development assets and artifacts. Won't getting started be a long process? How can I get value right away?

A: Begin with a pilot project that focuses on a particular business process or domain and related assets. Do not try to "boil the ocean."

Poulin: Well, it might not be as hard as it sounds. Of the hundreds and thousands of assets in existence, only a small number of them are probably good candidates for reuse.

Target a small program that's just starting up and identify relevant new services that you want to make reusable – that you can invest in and succeed with. From there, you can grow into a larger number of components while still focusing on ones that have been used over and over again.

There are basically two ways to identify reusable components. The first way is to launch a formal study, called a "domain analysis," to determine what would be reusable. This tends to be time-consuming and require a lot of effort, but it is appropriate when you really don't have a whole lot of experience in the field. The second way is to observe several projects, after which you'll notice that there are certain things that are common to every solution you're developing; these are good candidates for reusable services. Go to your experienced developers, and they'll tell you where you should put your money to make a reusable software service, even if you have hundreds or thousands of assets.

Marks: Another way to cut this is by business domain and business-process domain. Tying back to the **service-mapping** notion, we've worked with organizations on selecting major business processes and drilling down into the component processes underneath them. This identifies the services and components that map to the component processes. On the other hand, you could do an IT-value-chain analysis and work bottom up. But, you still have to get to some kind of a problem domain. Some assets are viable candidates, and some are not. You've got to find the sweet spot for reuse, whether it's from a business-process perspective or from purely an internal, IT development perspective.

Service mapping creates a services chain that shows services opportunities across the value chain. This is essentially a services view of a business' value chain.

Carlson: It really comes down to the domain experts who know what's needed. On the technical side, the experts are the developers, team leaders. They're going to identify the technically-oriented assets that ought to be reused or built for reuse. On the functional side, it's the business analysts who are going to drive what should be built for reuse and the existing assets that should be harvested for reuse.

SECTION 5: Measuring Reuse

Question Index:

- **What metrics should I use to evaluate results and encourage adoption of reuse technology and processes?**
- **Let's drill down a little more on ROI. What are the elements of an ROI for a metadata management catalog?**
- **What's the bottom line on implementing reuse?**

Q: What metrics should I use to evaluate results and encourage adoption of reuse technology and processes?

A: Measure reuse based on participation in the process, cost savings, the value of reuse-driven business outcomes and ROI.

Poulin: When it comes to measurements and metrics, there are a couple of things I feel very strongly are essential to a successful program. Whatever metrics you use, keep them simple and easy to understand and ensure they are realistic reflections of what you're doing and its value to the organization. This is really a lot harder than it sounds. Unless the data that you're basing your metrics on are well defined, and the objectives are perceived as equitable, you'll have a hard time convincing people the value that you're presenting is true and realistic. Your reuse tool must support measurements and metrics so the results information is consistent and credible.

The first metric I use is simply a reuse percent, which is the portion of your project or application that comes from components in the repository. This indicates the level of participation from each development team. It does not tell you what the value is because ten percent reuse on a small project is not anywhere near the value of ten percent on a very large project. However, it gives you an idea who is participating and who isn't.

The second metric I use gives you a sense of value by calculating the total cost avoided in each project across the organization. As a result of doing reuse, how much money are you saving? Finally, you want to account for your reuse investments using the third metric, ROI.

The three measurements I mentioned are implemented in Logidex so you get well-defined, objective results throughout the life of your project.

Marks: I love that set of metrics. I would add the business outcomes of reuse to it. Measuring the reuse process itself is important, but I also care about things like faster time to market for my business project and a better ability to implement something for my customers. I've talked with folks about the SOA scorecards that we're working with, and reuse metrics are very important, but they have to be linked to process and business outcomes. That's what gets the business excited as well.

Poulin: The functions that you are able to pull out of the repository and integrate with your system directly translate into schedule and quality improvements.

Q: Let's drill down a little more on ROI. What are the elements of an ROI for a metadata management catalog?

A: It comes down to cost savings in development and maintenance minus the expense of implementing and maintaining the reuse environment.

Poulin: The ROI is the sum of all your benefits minus the cost that it takes you to support the program. The benefits basically result from the amount of effort that you saved or costs you avoid spending during development. The rule of thumb is that about 80 percent of your development cost would be avoided by reusing something that already exists.

The second major element of the benefit is your maintenance savings. Remember, software that only takes a couple of months to develop is going to be around for years or decades. The maintenance over that time is a significant lifecycle cost of software, so it needs to play into your ROI model.

And then, finally, you have the cost to implement and support the reuse infrastructure, which includes tools and procedures. If the benefits balance out those costs, then you will have a return on investment model that has documented very lucrative results for all of the organizations I've worked with that practice reuse.

Marks: I would also build on top of that the appropriate process metrics to go with it. All of those ROI numbers support business processes that you can get a metric around, like how fast an IT initiative is launched when reuse is in place or how quickly you can change directions.

Poulin: To support that point, reusable components are historically proven to have about ten times better quality, as measured by defects in lines of code, than non-reusable assets. This is, in part, because you spend more time developing them and touching them. You also work out the bugs over time by virtue of using the components of over and over again.

Marks: Think about how that quality translates into time to market. You don't have to sit and rework, so you can put things into production quicker. Reuse has got a very high IT benefit and a big business impact.

Q: What's the bottom line on implementing reuse?

A: For a successful reuse initiative, target a specific business result, partner with a reuse technology vendor and measure the value gained from the initiative.

Marks: Make sure there's a clear business outcome that you're looking for and make all the decisions appropriate to supporting that business outcome.

Poulin: Establish your metrics so you can assess the value of the project and the tool.

Carlson: Pick a tools vendor who will be a true partner and work side-by-side with you to get going.