

# IBM U2 and Service Oriented Architecture (SOA)

## An overview

Skill Level: Intermediate

[Michael Rajkowski \(mrajkow@us.ibm.com\)](mailto:mrajkow@us.ibm.com)

Software Engineer

IBM - U2 Client Support

21 May 2009

Service oriented architecture is an application framework that takes everyday business applications and breaks them down into individual business functions and processes, called services. SOA lets you build, deploy, and integrate these services independent of applications and the computing platforms on which they run, making business processes more flexible. In this article, learn how IBM® UniData® and UniVerse® (U2) technology relates to SOA.

## Introduction to U2 and SOA

A service oriented architecture (SOA) is not a product, but rather a business framework that allows for a common business blueprint. An SOA is meant to be architected in a way that enables heterogeneous business processes and disparate data sources to interact harmoniously.

The question becomes, "Can IBM U2 technology play in the SOA game?" The answer is, "Absolutely, positively, yes!"

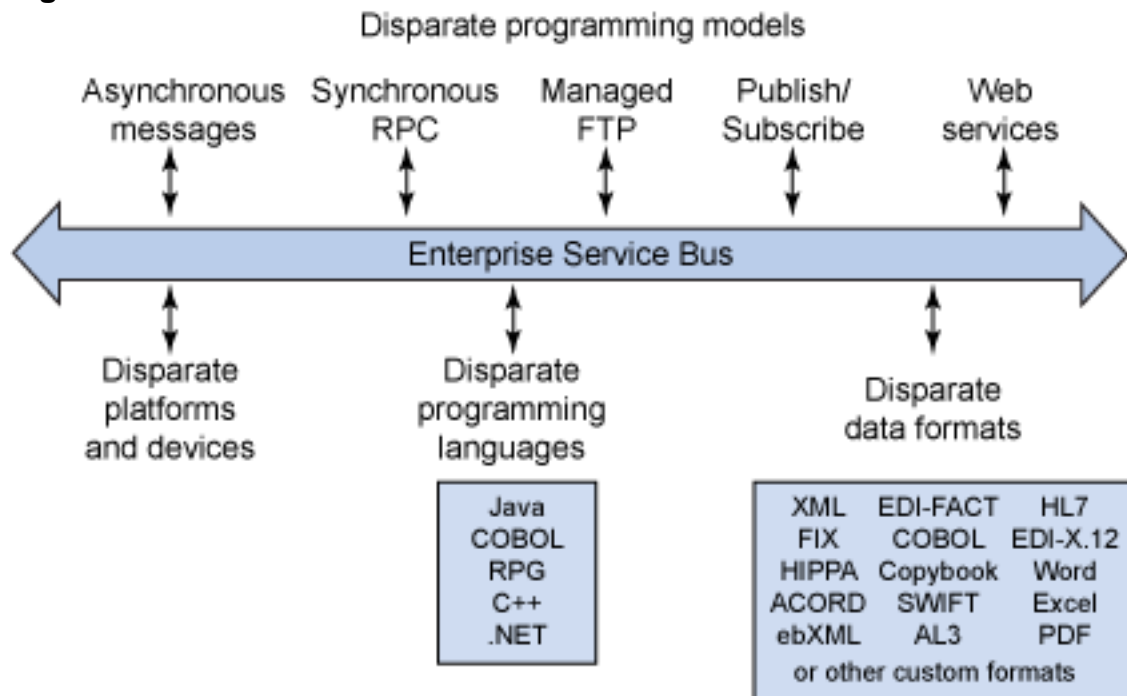
However, before getting into the details of IBM U2 technology as it relates to SOA, it may be helpful for you to understand more about the SOA framework story. IBM suggests that in order for business processes, or business services, to effectively communicate with one another, a common messaging bus is needed in the SOA framework. IBM calls this common messaging bus the Enterprise Service Bus (ESB).

## The Enterprise Service Bus (ESB)

The ESB can be thought of as the common communications medium for applications, much like telephone systems are the common communications medium for people. The ESB enables software applications running on different platforms, written in different programming languages, and using different programming models, to communicate with each other with little or no disruption to the applications themselves. Applications communicate with the ESB in ways that make the most sense for each individual application. Messages can get on the bus speaking the language of one application and get off the bus speaking the language of another.

Figure 1 helps explain the ESB framework.

**Figure 1. ESB as the common communications medium**



The ESB is not a single product, but rather a set of products that work together to provide a framework for SOA-based applications to be deployed. This set of products is determined by the deployment, business, and security needs of the applications. The beauty of this framework is that it enables you to add components and functionality on demand to meet business requirements.

## U2 and SOA

Now that you have a better understanding of SOA and the ESB, let's return to U2

and SOA. First we'll discuss U2 and SOA and then address the issue of deployment of the U2 service to the ESB. Prior to looking at the U2 BASIC routines, it is important to understand the concept of why there is a need to develop a U2 service.

One of the most important business issues that IT managers face today is integration. Integration can include anything from applications, application data, email, and instant messaging to hardware. As a result of mergers, acquisitions, etc., IT management is faced with a long list of seemingly daunting business concerns. These concerns may include:

- How do I eliminate multiple customer data files?
- What is the best way to integrate heterogeneous applications?
- How can I quickly respond to new functionality offered by my competitors?
- What am I going to do about the growing customer service issues that arise when multiple applications are involved?
- How can I quickly and effectively increase revenue and grow the business?

What is the solution to these questions? SOA of course! Some of the business benefits of SOA include:

- Connects disparate systems
- Leverages legacy applications
- Utilizes intellectual property investments
- Extends the application environment
- Allows data to become information
- Simplifies management of IT complexity
- Improves speed of IT implementations

SOA can help your IT environment quickly evolve into a superior solution that meets your business needs.

Consider the value of leveraging existing applications for SOA. Often the question is asked, "Why reuse existing assets in a *new* service-oriented architecture? Surely it's easier to start with a clean slate?"

The answer is no, it is not easier to start with a clean slate. It can be faster to expose parts of your application to an SOA than it is to start from scratch, and you benefit from working with an existing solution that has been proven over time.

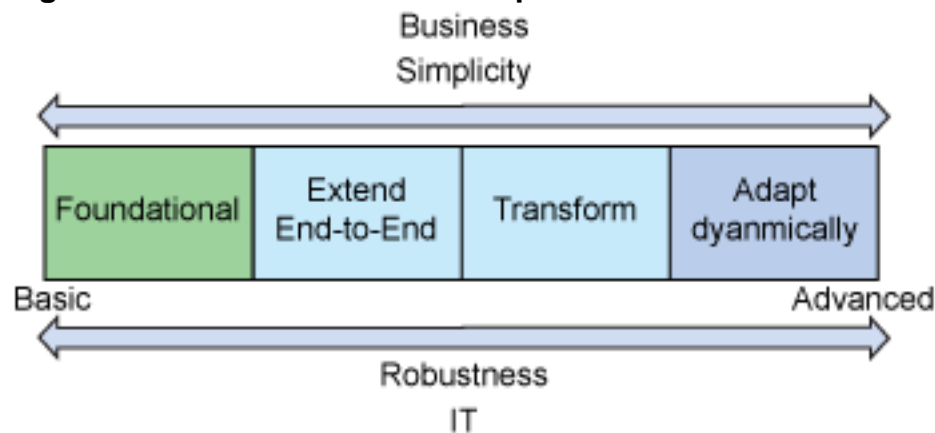
So, how do you approach SOA? The answer is easy. You can learn from the experiences of those around you and take a *smart* approach to SOA. The Smart SOA™ approach is a set of guiding principles developed by IBM. You can use these principles to extend the business value of your deployment.

## Smart SOA

A major tenet of Smart SOA is to recognize that your needs are evolving and that an answer that is good today may not be good tomorrow. A project that was being used ten times per day may someday be used ten thousand times per day. The project that met a hidden, back-office task is now being used as part of a mission-critical, customer-facing business process. Small projects grow up. Sometimes they grow up fast! That is why Smart SOA is about solving basic challenges in a basic way, but doing so in a manner that can grow as your needs evolve. Smart SOA allows you to meet the requirements of tomorrow's advanced deployments without undoing the foundational projects that you use today.

Figure 2 shows the levels of Smart SOA implementation from basic to advanced.

**Figure 2. Levels of Smart SOA implementation**



To keep it simple, we will focus on the foundational stage of adoption. At this stage, the key questions to ask yourself are:

- What aspects of my existing U2 applications can I expose as services?
- How can I maximize the value of my existing investment in an SOA environment?

To answer these questions, you must understand your business needs. For example, if IT wants to create a new business application via a workflow/patterns approach, they must answer the following questions:

- What specific areas of the application are causing the business the most pain?
- What aspects of the application can be defined as a service and reused in multiple workflows?
- What is the business objective for moving to an SOA environment?

Only after these questions are answered should you embark on moving the existing U2 BASIC code into services components.

## Preparing your U2 code for SOA

When creating U2 service components, you must first take inventory of the current U2 BASIC application code. A common issue facing the conversion of legacy U2 applications to SOA components deals with screen I/O's that are embedded in the business logic.

If your U2 code is written in this way, you'll need to separate the screen I/O's from the business logic. Note that this does not mean you have to learn Java or Visual Basic.NET to write a new GUI. A prudent solution would be to keep the screen I/O's and business logic in separate code streams. A major goal of SOA is to have reusable components that other applications can invoke. To realize maximum flexibility in your application, U2 service components should be able to call or invoke any other service components that may or may not be part of the U2 family.

Remember, using business logic components is an important part of having responsive business needs flexibility.

Once the U2 BASIC service components have been identified and the screen I/O's removed (if necessary), the next step is to identify other application logic that mimics the new service component. Reducing code redundancy is another positive result of moving to an SOA. Some may consider this an optional step, however, you will see greater business benefits if you take the time to inventory your current BASIC code and document the business functionality it performs.

Finally, passing arguments to and from U2 service components is absolutely acceptable. Think of almost any Web service, or the Web's request and response model. These technologies are all about "give me A, B and C and I will give you back X, Y and Z."

Your application infrastructure is the cornerstone of SOA deployment. How your service components are called, and how you call other service components, is up to you. Fortunately, U2 provides many ways to accomplish this task.

**Service:**

- A reusable business function described by an interface
  - A repeatable business task
  - Not necessarily computerized or automated
  - When computerized, often a Web service, but could be done via messaging like WebSphere MQ, FTP, etc.

**Web service:**

A self-contained, modular application that can be described, published, located, and invoked over a network.

While a service does not necessarily need to be a Web service to be included in an SOA implementation, Web services do provide you with some important benefits:

- Fundamental way to participate in SOA architecture
- Open, standards-based (HTTP, XML, SOAP, WSDL, etc.)
- Self-contained (support of HTTP and XML is sufficient)
- Self-describing (WSDL)
- Web-based (deployable across Web)
- Language independent (U2 SoapSubmitRequest() is an example), platform independent

Once you have created a set of components to call as a service, there are several way to expose them for use as a Web service in U2:

- Deploy to a U2 SOAP Server with the IBM U2 Web Services Developer Toolkit
- Create Web services with IBM U2 Web Development Environment
- Write your own services with one of the U2 Common Middleware Tools
- Deploy the services on the Web server of your choice

Another important item to address is security and authentication. If U2 is the only technology running your business, then you may not need to focus too heavily on security concerns. However, if your business has a heterogeneous database environment or is using the Web as a means of network connectivity and an anonymous entity is trying to invoke your U2 service component, then security and authentication become requirements. U2 now includes a set of the OpenSSL libraries to assist with these tasks.

There are a number of IBM developerWorks articles that can help you with security issues. You will need to know how you are exposing the services before you look into how to make them secure.

U2 now has the ability to expose BASIC routines as well as command line output streams, such as LIST statements, as Web services. This is done with the new U2 Web Services Developer Toolkit (U2 WSD). By utilizing U2 WSD, you have the ability (via a drag and drop wizard interface) to turn the U2 service components into a Web service. U2 WSD also includes a U2 SOAP server to run your Web service and a deployment wizard to speed the process.

Web services provide a great means of effectively having disparate technologies communicate using the same "language." It is in this context that you should use Web services. However, to have one U2 service component communicate to another U2 service component via a Web service can be a benefit, but may be unnecessary.

Another option for deploying U2 service components is through IBM WebSphere MQ (WMQ). WMQ provides guarantees and once-only message delivery. This is accomplished by configuring one or more Queue Managers to receive messages and guarantee that they are delivered to the appropriate target recipients. This is typically performed by applications sending either Send or Receive messages to the queue. IBM U2 has APIs for MQ built in as part of the standard U2 BASIC environment.

## Conclusion

In summary, IBM U2 applications have the ability to effectively engage in an SOA and ESB framework. Best practices have proven that understanding the business requirements first will fast-start a project. In addition, code modifications will probably be required. Screen I/O's will need to be decoupled from the routines used for U2 service components. Security and authentication applications may need to be added as well. Finally, U2 service components need to be exposed as Web services or WebSphere MQ messages. However, the resulting business benefits from deploying a U2 application that enables a true on demand business model will offer a return on investment far beyond the costs associated with the SOA migration effort.

# Resources

## Learn

- [Smart SOA](#): Learn more on how the Smart SOA approach can power smarter business outcomes.
- ["IBM U2: The big picture:"](#) (developerWorks, Aug 2005) Gain a basic understanding of the IBM U2 product line, and gather information about the extended relational data model, architecture, benefits, and associated tools products.
- In the [U2 area on developerWorks](#), get the resources you need to advance your skills in the U2 arena.
- [Manuals](#) for UniData and UniVerse
- Visit the [International UniVerse & UniData User Group](#).

## Get products and technologies

- Download [IBM U2 Personal Edition](#) and get your hands on U2 application development tools and middleware products.
- Download [IBM product evaluation versions](#) or [explore the online trials in the IBM SOA Sandbox](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

## Discuss

- [Participate in the discussion forum for this content](#).
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

## About the author

Michael Rajkowski

Michael Rajkowski is a Software Engineer in the IBM U2 Client Support Group. He has over 20 Years experience with MultiValued databases. Bachelor of Science in Computer Science from NYIT, and a MBA from Dowling College (Degree Concentration: Total Quality Management)