

# All You Need to Know About – Service Oriented Architecture (SOA) and System i



## What does looksoftware mean by SOA exactly?

SOA is a set of architectural and design principles that are probably new to most iSeries people. SOA considers an enterprise as consisting of many processes and services. SOA comes with the expected set of acronyms and jargon, such as "loose coupling" and "granularity" and is typically implemented with Web Services. SOA promises to deliver increased business agility by enabling IT professionals to easily reuse and integrate applications in a standardized way. By reusing chunks of application functionality across platforms, we can begin to refine, streamline, and perhaps, even create new business processes - on the fly.

A simple way of understanding Web Services (and avoiding technical discussion of WSDL, SOAP, and UDDI) is to think of them as a standard way of calling a program that may be located on another platform. By using tools that create Web Service wrappers (and generate the SOAP and WSDL etc.) for your existing applications, cross-platform integration can be achieved much more painlessly than in the past.

Our new product, '**so**architect', provides iSeries users with a practical path to Web Services and SOA because it generates Web Services from existing applications. This means iSeries users are able to reuse their existing, monolithic, applications and make them much more flexible by providing integration based on Web Services. The **looksoftware** modernization suite provides access to the presentation, application and data layers of iSeries applications to create reusable adapters that may expose chunks of application functionality – see the example below titled '*Exposing an iSeries Application Function as a Web Service*'.

## What do we mean by the 'Dynamic Environment'? Are we referring to real-time transactions, or is it more than this?

For ten years Gartner has been talking about the real-time enterprise, supported more recently by IBM's 'On Demand' and HP's adaptive enterprise messages. The idea being that as business agility improves, so our systems and processes can adapt to change more quickly.

**looksoftware's** 'Dynamic Environment' is designed to accommodate change in real-time, so your applications can continue to be enhanced without repeated conversion, collection or re-compilation. Objects are created dynamically, based on the execution state of your applications and your custom rules. This 'just-in-time' process allows you to modernize existing applications and compose new applications much more quickly than previous generation batch-based solutions. This results in increased business agility and significantly improved ability to respond to the inevitable new events that seem to arise in business all the time!



## **Composite Applications - how do they fit with the idea of SOA?**

Composite Applications are about creating new systems by reusing old ones in clever ways. Reuse is the common theme of Composite Applications and SOA. If we are able to access our existing applications as useful chunks of function, termed 'services', then we have a means of accessing even monolithic applications in a more modular way. That makes the delivery of Composite Applications feasible – because now you can assemble/access the services, perhaps add some business rules and deliver the 'new' solution.

## **What's the profile of the typical client wanting SOA at the moment?**

Any organization that has a significant investment in existing applications should be interested in service-enabling their applications and, of course, there needs to be a clear business driver. More often than not, these applications have evolved over years of refinement and provide robust support for the organization's business processes – in many cases reusing these applications makes the most business sense.

Another relevant sector is the ISV community. These are software companies that have produced vertical applications who can't afford the costs, risk and timeframe required to develop them from scratch again. By offering their customers modern, flexible, web standards based integration, these ISVs will be well positioned to retain and grow their customer base.

## **Does SOA and your 'soarchitect' offer any special advantage to iSeries users in particular?**

**soarchitect** is designed specifically for iSeries customers that need easy-to-learn tools that support their current RPG environments. Typically, they don't want to pull apart their systems or redevelop them. **soarchitect** supports the most popular iSeries technologies like 5250, RPG, Cobol and DB2.

## **Isn't SOA just a new term for existing ways of gluing systems together? If not, what is new or different about the idea of SOA?**

The key benefit of SOA is *simplifying* application integration. At **looksoftware**, we know that a key differentiator of our products is their ability to make it easier to 'glue' application components together. SOA's goal though is to encourage applications to be structured into service based components that can be easily and usefully reused – as described in the Jacob's example in the [soarchitect flyer](#) – where the same function or service is used in multiple channels as a component in a number of different solutions.

## **soarchitect creates web services from existing applications. What sorts of applications are best suited to this model?**

Applications that are suited to service enablement include custom RPG, COBOL, 4GL or Case generated applications using for example, Synon or LANSAP and Asset. Also, ERP applications like BPCS, System/21, PRMS etc, are very suitable, particularly if they have been customized.



**soarchitect's** adapter support allows programmatic access to any iSeries application at any of the three layers, so it is feasible to provide service access even to large monolithic applications. There will be occasions however, when an application may not be able to be effectively service-enabled because the existing application structure (even with intelligent adapter access) cannot easily provide access at the required granularity. In cases like this, it may be necessary to invasively restructure or perhaps even replace the underlying application.

### **Extending an iSeries application to consume a Web Service**

Let's take a look at a practical example of extending an RPG 5250 iSeries application to consume a Web Service. The customer maintenance function includes the customer's address details. Addresses are often very important data!

For example, if you're in the logistics business, delivery failures impact productivity and therefore, profitability. If addresses could be validated as they enter the system against a postal database, then most delivery failures could be eliminated, improving customer service levels – and profits! Web Services provide a relatively simple mechanism of integrating the logistics application with the external Web Service module that checks the supplied address against the postal authority's address database. We'll assume the Web Service expects to receive an address as input and returns a valid/not valid flag. The simple, non-invasive way to extend the existing application is to do the following:

1. Intercept the address as it is entered on the customer address details screen.
2. Call (consume) the Web Service and pass the address entered to return a valid/not valid flag.
3. If not valid, display the error dialog and the steps required to correct the problem. If valid, allow the unchanged host program to continue updating the database with verified address information.

The result is that only correct addresses reach your DB2 database, reducing the cost of delivery failures and increasing profits.

### **Exposing an iSeries Application Function as a Web Service using looksoftware's soarchitect**

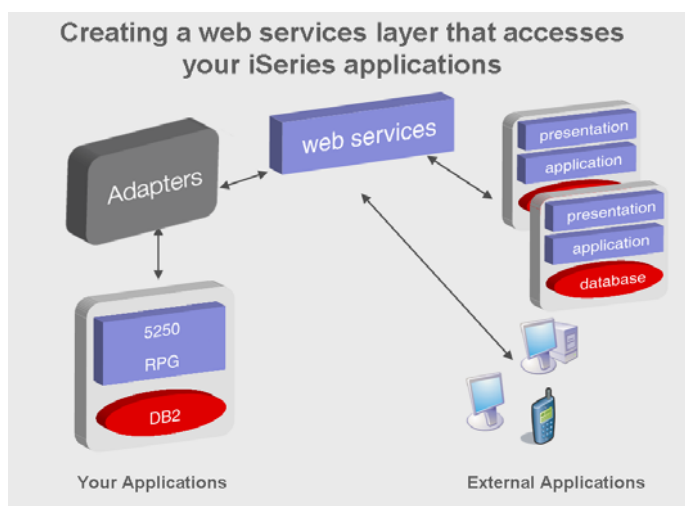
In this example, the business need is for an external business partner, for example a sister company, to have access to your iSeries customer credit check function. The business partner doesn't have - and doesn't need - access to the iSeries application which supports your business processes. They need access to a function or service provided by your application, named customer credit check. The business partner has requested a Web Service interface that accepts a customer number and returns a credit status flag and the value of all orders received this month. The 5250 customer inquiry program can provide this information easily and quickly when used by an experienced operator. The customer details screen displays the credit status flag, and the order inquiry screen calculates the correct value for this month's orders when the operator enters the start date of the current month into the appropriate order search field.



So, for the Web Service to retrieve the required information, it must programmatically access the customer inquiry program and mimic the operator's actions. The simplest, non-invasive means of creating a Web Service interface to the existing application is to create a reusable module that receives the customer number as input and programmatically accesses the iSeries application to perform the following:

1. Start a 5250 session and navigate to the customer inquiry menu option.
2. Populate the customer number into the customer number entry field and retrieve the credit status.
3. Navigate to the order inquiry screen and populate the start of current month date into the appropriate order search field to retrieve the calculated value of orders this month.
4. Save the retrieved credit status and value of orders received this month.

Then, publish the reusable module as a Web Service, and provide the Web Service details to your business partner.



Of course there are other ways to achieve the same end. Perhaps a new program written in RPG or Java could retrieve the requested information; however, that would probably require duplicating some business logic, whereas SOA is about reusing, not duplicating! Alternatively, the existing customer inquiry program(s) could be invasively restructured to separate the presentation logic from the business logic, and the resulting business logic layer could be accessed directly.

For the example above, creation of the Web Service interface is obviously important but technically not at all complex. More important is the definition of the reusable module that programmatically accesses the existing application to expose the required business functionality. Sometimes referred to as "adapters", these intelligent access routines can access any layer of the application -presentation, business logic and database. And since most traditional iSeries applications have their 5250 presentation code tightly bound to the business logic, often the easiest way to access and reuse the logic is via the robust 5250 presentation layer. The non-invasive approach is much more preferable because the underlying application code does not require to be changed, which means service enablement is less risky and much faster.



**Tips to help you move toward Web Services and SOA:**

1. Review your applications and rate them according to strategic value, volatility, health, and the need to integrate them with other applications. Review the list and determine which applications could best benefit business agility if they were service-enabled.
2. Consider service-enablement where it makes sense. There needs to be a clear business benefit, as in the examples above.
3. Determining the "right" granularity of the service is often tricky. A strong understanding of your company's business processes is a prerequisite for success.
4. Extending an existing application to consume a Web Service can be a good way of getting started.
5. There are now tools available, designed for the iSeries, that provide a practical path to service-enabling your applications.
6. Adapters can expose functionality that provides other applications with easy access to specific functions, called "services," from your existing unchanged applications.

