

# RPG and the AJAX & SOA Technology Trends, Part 1

By Ted Trichew

Software Architect with Databorough

Ted has over 20 years of software development experience, including 16 years at IBM Canada Laboratory in Toronto. At IBM Ted worked on mid-range software, including architecture and development of original releases of C/400 and ILE/RPG. Ted also has 5 years experience as product manager of the Ironside Business-to-business solution suite which provided one of the first rich Internet user interfaces available on the market, and also provided integration in iSeries ERP systems including JDE World, BPCS, PRISM, MAPICS and JBA System 21. His most recent experience was at SSA Global as an architect working on the SSA Technology Architecture platform which provides a modern rich user interface, as well as service oriented architecture for SSA's solutions. Ted has a thorough understanding of the issues facing businesses that want to modernize their legacy applications.

## Abstract

Two technology trends that have caught the attention of enterprises, development tool vendors, and industry analysts because of their potential business value are AJAX and SOA (Service Oriented Architecture). AJAX brings the promise of client/server type rich user interfaces to the web. Service Oriented Architecture brings the promise of creation of shared re-useable services that can be used to implement independent business services which in turn, increase the flexibility and agility of a company. In many ways these two trends go hand in hand in the creation of a "modern" business application. But with all the hype, there is the reality that many enterprises run their business using "legacy" RPG applications. So the question is how can a company realize the promise of a rich AJAX UI and re-usable business services in their legacy environment?

## The Technology Trends

Every year, industry analysts such as *Gartner Group* and *Forester Research* publish their technology trend forecasts that describe technologies that are expected to have a big impact on enterprise developers over the next decade. For 2006, a number of technologies have been identified and organizations should start exploring how to leverage these technologies to solve compelling business problems that they may have. The technologies identified that potentially could have a "high impact", and could pay the biggest dividends for businesses over the next decade are AJAX and SOA.

When looking at technology trends you have to be careful to take into account where the technology is in its "technology life cycle".

This life cycle has four stages:

**Invention:** Technology is invented to solve a particular problem or class of problems, and it usually does very well at solving the problem, or class of problems.

**Adoption:** Technology is adopted; people jump on the bandwagon and try to solve ALL problems using it, the “Swiss army knife” approach. Even though a Swiss army knife has a knife attachment, it doesn’t mean that it can effectively be used to replace a carving knife.

**Disillusionment:** People get disillusioned because the technology does not solve ALL the problems and jump off the bandwagon. They discovered that the Swiss army knife is OK for occasional use, but they are better off using the right knife for the right job.

**Maturity:** The technology is still around, and it still solves the problems it was originally intended to solve, and minor enhancements are made to it to keep it up-to-date.

We are currently seeing the maturity in a number of technologies such as object-oriented programming, Java, XML, and HTML but also are seeing a new wave of technologies being developed such as AJAX and SOA that are in the invention and adoption phase. At this stage in the technology lifecycle there is a lot of hype so we need to explore what business problems, or class of business problems these technologies were meant to solve, and how well these fit with the business problems you are trying to solve in your company.

## **The Business Problems**

Over the past few years many organizations have tried to provide a more compelling user interface to their existing 5250 based RPG applications. One of the main business drivers to modernize the user interface is that modern graphical user interfaces are thought to increase user productivity. Another reason is that packaged software vendors found it difficult to compete in a market where many vendors provided visually compelling graphical user interfaces. Which software package would you buy if you were a CIO, a green screen application, or a web-based application?

There are many ways to modernize a user interface, but by far the fastest path is to use “screen scraping” technology such as *IBM’s WebFacing Tool*. Although this should be considered a more tactical UI modernization solution, since the resulting interface uses the original application flow, and in general will not increase the user’s productivity.

## **The need for Rich User Interface**

In general thin-client UI technology does provide a more graphical user interface and does have the advantage of a zero client install so applications can be deployed to easily “reach” many users. However, thin-client applications do not provide the rich graphical user interface, responsive performance and highly interactive functionality found in client/server applications. The tradeoff is that client/server applications have to be deployed on every client desktop and this has been very costly for IT organizations to

support. To help alleviate some of these client/server deployment issues, Microsoft has recently introduced their “ClickOnce” deployment technology.

### **The need for Better Application Feel**

In order to achieve the business benefits of compelling modern user interface you not only need to improve the “look” of the application, but you also need to improve the “feel” of the application. Improving the feel of the application involves changing the original application screen content and screen flows to be more stream-lined. So rather than having 20 screens for an order entry process the user interface may be consolidated into 3 screens. For example, one of the most requested features for iSeries ERP Order Entry applications is to provide “Fast Order Entry” capabilities that will improve the productivity of the order entry clerks. Order entry clerks do not want to go through all the mouse clicks or key strokes necessary to go through all the screens required to create an order when in the majority of cases only a subset of information is needed. Creating a more modern user interface on top of existing 5250 screen flows will not solve this problem; you actually need to change the screen flow and UI logic of the application. This is difficult to do with screen scraping technology since the screen flow and UI logic are determined by the application logic which has not been affected by the screen scraping program. What is required in this case is a more modern event driven application architecture where the UI actually drives the order entry business logic as the user interacts with it. Unfortunately most iSeries RPG applications are not architected in such a way that you can easily change, or even access the business logic. This means that in order to truly modernize the application user interface and change the flow of the user experience, you also have to modernize the overall application structure to make it easier to access the business logic.

### **The need to integrate with existing Business Logic**

As organizations deploy new applications such as Customer Relationship Management (CRM), and self-service eCommerce they have found the need to integrate to the existing business logic locked in their RPG programs. The main reason for this requirement is that companies do not want to duplicate their business logic in multiple places for obvious maintenance reasons, but still want to provide a seamless interface between these new packages and their existing packages such as ERP systems. When a customer service representative is using a CRM package they would like to be able to view the customers order information as well as be able to create a quote without leaving the CRM interface to go to another application. Since this information is handled through the order management module of the ERP system, there must be a seamless integration between the CRM system and the ERP system.

An entire industry has spawned to solve this integration requirement where integration companies have introduced “adapters” to packaged iSeries software applications such as ERP systems. In most cases this is because ERP vendors have failed to provide the necessary interfaces to integrate with their packages. However, these integration adapters are limited for the main part to only two or three mainstream ERP applications such as SAP and JDE.

## Rich Applications & AJAX

A rich application is one that provides a user experience that is highly interactive with a rich graphical user interface and very responsive performance. Essentially client/server desktop applications are classic examples of rich applications. The main trade-off between thin-client applications and client/server applications has been the issue of “richness” vs “reach”. Client/server applications are rich, but cannot reach as many users because they need to be deployed locally on every client desktop. On the other hand, thin-client applications have a ubiquitous reach since they can be centrally deployed on a web server, but they are not as rich or highly interactive because they use page based HTML pages which require refreshing on every client interaction.

Rich Internet Applications (RIA) is a new class of application that brings web applications closer to the client/server user experience. AJAX is a class of RIA based on existing web technology such as JavaScript. There are also Flash plug-in based RIA solutions such as Adobe’s *Flex* and Laszlo Systems *OpenLaszlo*. Not to be left out, IBM and Microsoft both have RIA solutions. IBM has *Workplace Client*, which is based on the Eclipse SWT UI framework. Microsoft has Atlas which can be considered as an AJAX technology, but is also trying to bridge the gap between client/server and web applications with the introduction of *Windows Presentation Foundation (WPF)* as a new way to build client/server applications that can be deployed on a web server. Expect WPF to be available by the end of 2006 or early 2007 as part of the *Microsoft .Net Framework 3.0*.

AJAX is the RIA technology that analysts agree enterprises should put an effort into investigating. A majority of enterprise applications are based on querying, modifying, updating and deleting data. To build these types of applications a rich UI technology such as AJAX needs to have highly interactive grids/tables and forms. Let’s look at a couple of enterprise application UI patterns in more detail:

### ***Filter/Summary/Detail***

Many enterprise applications are based on a common application UI pattern of “Filter” (or Search), “Summary”, and “Detail”. That is, you provide a filtering mechanism that allows a user to specify what they are looking for from a business perspective. For order management, this may be a filter for querying order status, which may include a choice of “open orders”, “shipped orders”, “orders by PO#”, etc. Once the filter is specified, a summary of results is shown, typically in a grid. A user can view the details for an item in the results set by selecting the item in the grid.

In traditional thin-client (Servlet or CGI generated HTML) implementation, this would be handled through a number of screens; each would be refreshed as the user interacts with the screen. In an AJAX or client/server application, this can all be handled on one page since AJAX applications allow partial refresh of page results. This means that as the user selects an order that they want to see more details on, the resulting order detail can instantly be shown on the same screen without a painful page refresh. In fact some AJAX frameworks allow you to show the results inline in the grid.

### ***Interactive Update***

A more complicated application enterprise UI pattern is an “Interactive Update”. An example of this would be a line-at-a-time order entry application where you start with a screen that contains an editable grid. The user enters an item number in the “item #” column, and the UI control could asynchronously populate results in the drop down based on the partial item number the user is typing. Once the users enters the item number and presses the tab or enter key, fields on the line such as “description”, “price”, “UOM” are filled in and the cursor is moved to the next editable field which is “quantity”. The user enters quantity and presses the tab or enter key. If everything is successful, the cursor is moved to the next line and the user can enter another item. If there is a business event such “insufficient inventory”, a visual indication is given to the user in the form of a pop-up message which allows the user to take action in terms of accepting backorder, partially filling the backorder, or cancelling the item. Once the user takes action, the cursor is move to the next line, and the user continues by entering another item number.

This scenario is highly interactive, and the user interface and business logic is driven by UI events such as alpha-numeric key pressed, tab key pressed, enter key pressed, back order button pressed. This is not an application that can be created using thin-client technology; it requires an RIA technology such as AJAX.

### **So what is AJAX?**

AJAX, which stands for *Asynchronous JavaScript with XML* is not really new technology in that it uses existing web technologies. AJAX is a collection of existing technologies used to build dynamic web pages on the client side. Data is read from the server or sent to the server by JavaScript requests. However, some processing at the server side is required to handle requests, i.e., finding and storing the data. This is accomplished more easily with the use of a framework dedicated to process AJAX requests. There are a number of open source and commercial frameworks available that handle AJAX, some of these frameworks such as *Prototype* and *Dojo* are based on the use of JavaScript libraries to handle the rich UI controls, while others such as *ZK* and *Backbase* are based on dhtml, or a XML format definition languages to handle the rich UI components. Google which is one of the pioneers of AJAX technology uses a different approach. They provide an AJAX framework called the *Google Web Toolkit* (GWT) that allows application developers to code the user interface using Java, and then using Google tools to generate the client side JavaScript and HTML for the AJAX UI.

### ***Standards what standards?***

As you can tell from the discussion above, there are no standards for AJAX frameworks. In fact, it seems that there is a new AJAX framework being introduced every month, so you have to be careful in deciding which framework you are going to use. Many of the problems with lack of standardization are related to interoperability of different AJAX frameworks. The OpenAJAX Alliance which boasts more than 50 members is one of the organizations that is trying to tackle standards around the issues of interoperability between AJAX frameworks. The W3C is also trying to add some clarity through its *W3C Web Formats Working Group* which is working on development of a language for defining client side web applications. This will be based on existing UI web format

definition languages, such as *Mozilla's XUL*, *Microsoft's XAML*, *Adobe/Macromedia's MXML* or *Laszlo Systems' LZX*.

### ***Development Tools***

One of the challenges with AJAX has been the lack of development tools to make AJAX programming easier. Debugging JavaScript can be quite challenging, and many of the AJAX frameworks are based on a lot of fairly sophisticated JavaScript code. So when choosing an AJAX framework, make sure that development tools is a high priority evaluation criteria.

### ***Integration with Legacy Applications***

In order to provide a truly rich interactive experience, you will need to integrate your rich UI application with your legacy applications in a fairly “chatty” or transactional manner. Traditional legacy integrations have not needed to be fine grained, due to the page based nature of thin client web interfaces. However, all this changes with AJAX, as we are providing a highly interactive user experience that will require a highly interactive “real-time” integration mechanism.

## **Conclusions**

When you are assessing any new technology you should always assess them based on the business problems your organization is trying to solve, and where they are in their technology life cycle. AJAX is an exciting new technology trends in the early stages of its technology life cycle that industry analysts like *Gartner Group* and *Forester Research* have forecasted to potentially have a “high impact” on enterprises. AJAX brings the promise of client/server type rich user interfaces to the web, and can be used to create highly interactive, event driven and compelling web-based user interfaces. However, the key to unlocking the promise of AJAX lies in leveraging of legacy business logic and data. In Part 2 of this article we will examine the Service Oriented Architecture (SOA) technology trend, and how it can go hand-in-hand in creation of a modern application architecture that provides common access to your legacy application business logic. We will also examine a tools-based approach to help with this modernization effort.

For more information on the tools and methods described in this paper, please contact Stuart Milligan at:

[stuartm@databorough.com](mailto:stuartm@databorough.com)

705 719 7952