Web Services Journal
www.sys-con.com/webservices

Business Rules

Building Rules into Web Services Applications

Use past successes – and failures – to find the best path to your project's needs

by Henry Bowers

The emergence of Web services is forcing sites to substantially rethink how existing applications can and should work together.

Previously, considerations about where certain functions should execute focused principally on the tier. Should the client, the application server, or the database management system perform this processing? Web services, however, force the question in a horizontal direction: Which server on this tier is best suited to handle this processing? And, even more frequently, how can this application be partitioned intelligently so the right pieces are on the correct servers? Design tools are responding to these changes with model-driven architecture and services-oriented application development, two approaches that capture and represent the salient factors in the design of loosely coupled, distributed applications.

However, most sites that have committed to pilots of Web services cannot wait for best practices to mature past these nascent technologies. These sites need to partition applications now. Thus, they are highly dependent on learning from the successes and failures of other sites working on similar challenges. From these experiences, they can begin to divine the correct path for their own projects.

This approach is hampered by the limited amount of information available about Web services projects. The lack of information is particularly acute as regards the partitioning into Web services of projects that handle business rules. This article discusses several ways in which applications can be intelligently partitioned to make the best use of business rules. The models are drawn from our client consulting experiences with this particular problem.

Using Business Rules

Ask most IT professionals what they place on their application servers, and they'll respond "business logic." If you ask for a more specific definition of business logic, they will usually reply with application details from which flow explanations of implementation particulars. Because business logic is perceived as unique at every site, little effort is made to view its elements as anything but a series of unique detail-level components. This view, however, does not generally reflect the reality. Companies in like industries generally have similar processing to perform (over and above standard accounting applications). For example, insurers all have claims and underwriting systems; they all must perform actuarial analysis of pricing, process claims more or less the same way, and issue policies using similar constraints. In addition, their regulatory reporting requirements look a lot alike.

What distinguishes one insurer from another is the criteria used for accepting an applicant, establishing the kinds of insurance the candidate qualifies for, pricing the policy, and applying policy-specific features and benefits. These distinguishing characteristics are implemented through business rules. Rules determine who qualifies for what, at what cost, and with what options. Because of this, insurance companies, mortgage lenders, financial services companies, government agencies, and hundreds of other businesses rely on rules as a central part of their business logic. In fact, it is fair to say that business rules capture the unique competitive proposition of a business. Because of this central role, the specification, enforcement, and management of rules should be a separate and specialized portion of business logic.

Transitioning Rules to Web Services

Many rules implementations today do not distinguish rules from straight data processing very well. In fact, rules are most often implemented in pages of if/else statements buried in larger C, COBOL, and Java applications. One client had more than 400 pages of if/else statements; as a result, maintaining the business rules or adapting them to a new business product was very difficult, if not impossible.

Companies that haven't carefully segregated rules from other business logic are confronted with two choices: divide the application as best they can (and one would hope with segregation of rules a key objective) or simply slap a Web services front end on the monolithic application. Given the tight budgetary constraints at most IT sites today, the latter approach is the most common. It doesn't require a lot of effort at a programming level. Such implementations generally place software such as a principal mortgage application module on an intranet as a Web service. Other applications send the module large SOAP-wrapped XML records containing all relevant applicant data for approval. When the software approves the application, it returns to the client the necessary text to drop into loan document templates.

One of our customers is using a variant of this approach to enable its agents to sell policies in the field. Previously, agents would fax the data to an underwriting department and wait for an answer. Shortly, they will use wireless access to virtual private networks (VPNs) and Web services to get approval and the necessary documentation back on their machines with only a slight delay. This allows the agents to close a policy sale on the spot. The advantage of this approach is its simplicity. Existing software that a site knows and trusts is unchanged but for the front-end means of access. This solution is quick, inexpensive, and effective. However, it does not make optimal use of business-logic components.

Better Granularity

The second approach is to separate business rules from the business logic. Depending on the construction of the applications, this can be fairly easy. For example, the 400 pages of if/else statements mentioned earlier are located in a handful of functions. These routines can easily be wrapped up as a Web service. This service is sent the same parameters with which the original functions were called and returns the same values after executing the rules. This straightforward division is somewhat more complex if global variables in the application are modified, but even this effort is worthwhile given the advantages of this approach.

The first advantage is that other applications can now make use of the same rules. Suppose, for example, that a mortgage insurance company wants to establish what proportion of its loan portfolio is at risk. It might obtain updated information on its customers and run the data through the qualification engine to see who would be rejected or assigned a lower credit score. To do this analysis, the company needs to access the business rules without dealing with the other processing of a loan application. It is not interested in receiving text streams for loan documents; it wants only the results generated by pushing data through the qualification rules.

Web services provide a unique opportunity to implement this kind of reusability by carving out sets of rules into callable services. In addition, they encourage the conversion of many standalone data formats into XML, which has the benefit of simplifying overall application integration (at the cost of performance, which in high-volume settings is far from negligible).

Sites that choose this path are often surprised at how quickly benefits become apparent. The separation of rules processing as Web services can be implemented progressively. Complete conversions of applications are unnecessary. Also, once rules are segregated as Web services, new opportunities for their use appear quickly. For example, business analysts can much more easily perform what-if scenarios by rolling hypothetical data through the rules.

A common implementation detail of this approach is the migration of rules from a code-based implementation to a business rule engine (BRE), an integral component of a business rule management system. A BRE applies business rules to application data, generally in a highly optimized fashion. However, its real importance is that it enables firms to specify rules in business terms rather than in code. This step permits sites to move rules formulation and testing outside of IT and into the hands of business analysts. These analysts now can formulate rules using their own vernacular, test the results, and go live when they're ready. Meanwhile, programmers are freed from the conversion of business policies into code and so can focus on other portions of the application.

By using Web services and a BRE this scenario becomes possible. The rest of the application then interacts with the rule sets, which are now flexible and highly amenable to use in one-off situations. Because of this benefit, BREs are increasingly viewed as a central best practice for the implementation of policy-intensive applications.

Toward Business-Rule Vending

As companies gain expertise with Web services and BREs, they find that deploying and reusing volatile business rules presents new management challenges. With business rules that change frequently, the task of deploying rules varies as the changes occur. Ensuring that all affected applications have been updated becomes increasingly tedious and error-prone. Companies with this predicament may avail themselves of an additional implementation model that simplifies and automates distribution of updated rules. This implementation model introduces the concept of business-rule vending.

The rule-vending service is implemented as a Web service with a BRE and an interface to a business-rules repository (see Figure 1). This vending service receives requests from other Web services and does two things with these requests: it determines what business rules in the repository are appropriate to apply, and it returns those rules in the requested format. Determining which business rules to apply and return consists of matching the context and the origin of the request with the sets of rules available in the repository. The matching logic is implemented using rules as well (these matching rules are stored and maintained with the other rules - in the repository). The rule-vending service has the ability to return an XML-based file containing the applicable business rules, a link to a Web service that applies a specific set of business rules, or the results produced by executing the business rules on the spot.

This vending service simplifies distribution of rules by providing a central (and potentially a single) location for business rule location and distribution. It also enables sites to implement even more sophisticated arrangements such as publish-and-subscribe models for business rule distribution.

The use of BREs in conjunction with an atomic implementation of rules enables sites to have extremely effective and efficient rule processing via Web services.

Author Bio

Henry Bowers has spent more than 15 years in the high-tech sector, building and managing software products for both private industry and government. He has more than seven years' experience working with rules-based systems and business rules in general. Henry is currently a product manager for business rules at ILOG (www.ilog.com). For more than 10 years, ILOG's innovative enterprise-class software components and services have helped companies maximize their business agility and improve operating efficiency.

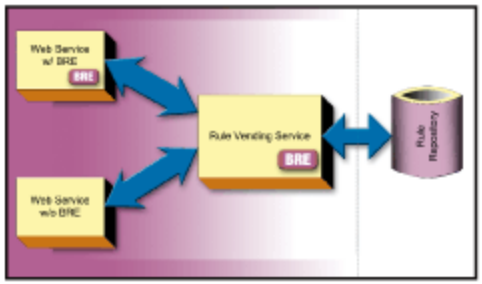

FIGURE 1 **Rule Vending: Applications call a Web service to process rules. This service calls another Web service to provide it with the freshest rules required by the application.**

Figure 1

Reader Feedback

Business Rules Engines

Posted by Peter Idoine on Feb 10 @ 05:38 PM

Hi, some excellent observations Henry. I have been working on a number of projects that have been able to abstract the business rules from the application and given the ability to the business (not a developer or BA) to create and change business rules outside of the application code. Its worked well - your insurance example is a good one - dear to my heart having delivered a number of solutions for different insurance companies. Have also used the approach for Cost disbursement rules, logistics rules, rating rules. We used www.idiomsoftware.com in all our projects.

Rgds Peter

(read & respond...)
