

Externalizing Business Rules from Enterprise Applications: An Experience Report

Isabelle Rouvellou, Lou Degenaro
IBM T.J. Watson Research Center

E-mail: isabelle@watson.ibm.com

Kevin Rasmus
Country Companies Insurance. Bloomington, Illinois

Dave Ehnebuske, Barbara Mc. Kee
IBM Software Solutions. Austin, Texas

Abstract:

Business rules (i.e., implementations of the policies and practices of a business organization) are often embedded within the code of distributed object and component systems. Decoupling and externalizing these rules can provide a number of advantages. This presentation describes our experience building and using a framework that enables enterprises to develop distributed business applications that systematically externalize the time- and situation-variable parts of their business logic as externally applied entities called business rules. Because business rules are external to the applications that depend upon them, the variable business logic contained in them can easily be changed. Because the management of the externalized business rules is done explicitly through a rule management facility, it is easy to understand what rules exist and to locate those that need to be changed. The Accessible Business Rule (ABR) Framework is available as early test function in IBM Component Broker 2.0 and 3.0 (a part of WebSphere Enterprise).

In 1995, Country Companies Insurance began work on a CORBA-based application called LUW (Life Underwriting Workstation). Portions of that project were done in partnership with members of the IBM T. J. Watson Research Laboratory. LUW is an application that supports the decision making process that life underwriters go through in determining whether to accept or reject an insurance policy. An important part of this application is a rules framework that allows business rules to be externalized from code, made explicit for understanding and ownership, and reused for reduced cost of development and consistency of business process. LUW was placed in production in early 1998.

The LUW Rules Framework supports server-side rules written in C++ running on OS/2. The framework supports two types of rules: constraint rules that check to see if a particular constraint or edit has been met, and derivation rules that use an algorithm to arrive at a particular return value. An example of a constraint rule might be "If largest_allowed_vehicle_type = light_truck and gross_vehicle_weight < 2000, then OK". An example of a derivation rule might be "given the policy object, gather data and run a complex algorithm that computes the policy rate". These rules are externalized from the points in code where they are invoked. The places in code where rules are run are called trigger points. LUW supports different types of trigger points. Precondition and postcondition trigger points fire constraint rules respectively at the beginning and the end of a method. An exception is thrown on violation. A commit trigger point fires constraint rules that deal with cross-class edits or constraints on the participants in a unit of work. Precondition, postcondition, and commit trigger points are fully generated by a code generator. Other trigger points, such as derivation trigger points, often require some hand-coding.

In 1997, even before LUW went production, Country Companies began another joint effort with IBM (again with T. J. Watson plus a group from Austin Tx. that had experience in client-side rules) to create the next generation of rules framework; this time called ABR (Accessible Business Rules). ABR was to be a rules framework that would run on multiple platforms (Windows NT, UNIX, and others), support rules written in multiple languages (Java, C++, and others), and

support both client-side and server-side trigger points and rule execution. An administration system was to also be included to assist in the management and deployment of rules. IBM's Component Broker (now a part of WebSphere Enterprise) was chosen as the object transaction monitor and application server to support this. The first release of ABR was shipped as early test function with Component Broker Release 1.3 in July of 1998.

In the course of these two efforts, and the continuing development of ABR, we learned many things. Much of what we learned in LUW became the basic requirements for the ABR effort. We continue to learn as ABR progresses.

One of the early decisions that we had to make was which rules to externalize and which rules to leave embedded in code. This is a cost/benefit question. The benefits vary in their importance depending on business needs. Some benefits considered: reduced development cost, reduced maintenance cost, reuse, explicit documentation of which rules are in use, clear ownership of rules, consistency of rule application, ease of testing, etc. The costs, we found, came down to a single factor; performance. In our first trial, performance was so bad that we would not be able to deploy applications utilizing externalized rules. But through some rework, tuning, and the implementation of caching, performance improved. In the end, balancing performance against benefits, we ended up externalizing about 200 rules.

LUW supported only server-side rules. Server objects are often many levels of abstraction down from what the user knows and understands. We found that rules often need to be in the server for consistency and to make sure they are not circumvented. However, many rules need to be in the client so rule failures can be more clearly linked to objects that a user understands. In ABR, as we moved rules to the client (trigger points placed in the client), we included the flexibility to execute the rule on the client or on the server. This gave us additional control over performance.

A number of other issues were also shown to be important. Rule failures are thrown as exceptions. The content and handling of these exceptions is very important to a system where the user must correct something in order for the rules to fire successfully. Also, manual administration and deployment of rules is difficult and error-prone. An administration system is important in making externalized rules manageable and easy to understand.

Both Country Companies and IBM plan continued use of rules frameworks. Externalized rules, when properly selected and located, can lend considerable flexibility to a system at a reasonable cost. As they are external to the applications that depend upon them, the variable business logic contained in them can easily be changed. Because the management of the externalized business rules is done explicitly through a rule management facility, it is easy to understand what rules exist and to locate those that need to be changed. Analysts, designers, and end users can become comfortable with rules concepts in a short period of time. As performance improves in Java and WebSphere, externalized rules will become even more practical than they are today. It will become difficult to even think of building a system where all of the rules are embedded in code.