Business rules engine

A **business rules engine** is a software system that executes one or more business rules in a runtime production environment. The rules might come from legal regulation ("An employee can be fired for any reason or no reason but not for an illegal reason"), company policy ("All customers that spend more than $100 at one time will receive a 10% discount"), or other sources. A business rule system enables these company policies and other operational decisions to be defined, tested, executed and maintained separately from application code.

Rule engines typically support rules, facts, priority (score), mutual exclusion, preconditions, and other functions. Rule engine software is commonly provided as a component of a business rule management system which, among other functions, provides the ability to: register, define, classify, and manage all the rules, verify consistency of rules definitions ("Gold-level customers are eligible for free shipping when order quantity > 10" and "maximum order quantity for Silver-level customers = 15" ), define the relationships between different rules, and relate some of these rules to IT applications that are affected or need to enforce one or more of the rules.

**IT use**

In any IT application, business rules change more frequently than the rest of the application code. Rules engines or inference engines are the pluggable software components that execute business rules that have been externalized from application code as part of a business rules approach. This externalization of business rules allows the business users to modify the rules frequently without the need of IT intervention. The system as a whole becomes more adaptable with business rules that can be changed dynamically, although QA and other testing would still be required.

**Design strategies**

Many organizations' rules efforts combine aspects of what is generally considered work-flow design with traditional rule design. This failure to separate the two approaches can lead to problems with the ability to re-use and control both business rules and workflows. Design approaches that avoid this quandary separate the role of business rules and work flows as follows:[1]

Business rules produce knowledge; work flows perform business work. Concretely, that means that a business rule may do things like detect that a business situation has occurred and raise a business event (typically carried via a messaging infrastructure) or create higher level business knowledge (e.g., evaluating the series of organizational, product, and regulatory-based rules concerning whether or not a loan meets underwriting criteria). On the other hand, a work flow would respond to an event that indicated something such as the overloading of a routing point by initiating a series of activities.

This separation is important because the same business judgment (mortgage meets underwriting criteria) or business event (router is overloaded) can be reacted to by many different work flows. Embedding the work done in response to rule-driven knowledge creation into the rule itself greatly reduces the ability of business rules to be reused across an organization because it makes them work-flow specific.

To create an architecture that employs a business rules engine it is essential to establish the integration between a BPM (Business Process Management) and a BRM (Business Rules Management) platform that is based upon processes responding to events or examining business judgments that are defined by business rules. There are some products in the marketplace that provide this integration natively. In other situations this type of abstraction and integration will have to be developed within a particular project or organization.

Most Java-based rules engines provide a technical call-level interface, based on the JSR-94 application programming interface (API) standard, in order to allow for integration with different applications, and many rule engines allow
for service-oriented integrations through Web-based standards such as WSDL and SOAP.

Most rule engines supply the ability to develop a data abstraction that represents the business entities and relationships that rules should be written against. This **business entity model** can typically be populated from a variety of sources including XML, POJOs, flat files, etc. There is no standard language for writing the rules themselves. Many engines use a Java-like syntax, while some allow the definition of custom business friendly languages.

Most rules engines function as a callable library. However, it is becoming more popular for them to run as a generic process akin to the way that RDBMSs behave. Most engines treat rules as a configuration to be loaded into their process instance, although some are actually code generators for the whole rule execution instance and others allow the user to choose.

**Types of rule engines**

There are a number of different types of rule engines. These types (generally) differ in how Rules are scheduled for execution.

Most rules engines used by businesses are forward chaining, which can be further divided into two classes:

- The first class processes so-called production/inference rules. These types of rules are used to represent behaviors of the type **IF** condition **THEN** action. For example, such a rule could answer the question: "Should this customer be allowed a mortgage?" by executing rules of the form "IF some-condition **THEN** allow-customer-a-mortgage".
- The other type of rule engine processes so-called reaction/Event Condition Action rules. The reactive rule engines detect and react to incoming events and process event patterns. For example, a reactive rule engine could be used to alert a manager when certain items are out of stock.

The biggest difference between these types is that production rule engines execute when a user or application invokes them, usually in a stateless manner. A reactive rule engine reacts automatically when events occur, usually in a stateful manner. Many (and indeed most) popular commercial rule engines have both production and reaction rule capabilities, although they might emphasize one class over another. For example, most business rules engines are primarily production rules engines, whereas complex event processing rules engines emphasize reaction rules.

In addition, some rules engines support backward chaining. In this case a rules engine seeks to resolve the facts to fit a particular goal. It is often referred to as being **goal driven** because it tries to determine if something exists based on existing information.

A third class of rules engine might be called a deterministic engine. These rules engines may forgo both forward chaining and backward chaining, and instead utilize domain-specific language approaches to better describe policy. This approach is often easier to implement and maintain, and provides performance advantages over forward or backward chaining systems.
Resources


References


External links

- Guidelines for deciding whether to use a rules engine? (http://herzberg.ca.sandia.gov/guidelines.shtml)
Article Sources and Contributors


**License**

Creative Commons Attribution-Share Alike 3.0 Unported
creativecommons.org/licenses/by-sa/3.0/