Why Business Rules?

A Case For Business Consumers of Information Technology
Why Business Rules?

A Case For Business Consumers of Information Technology

White Paper

© ILOG, March 2006 - Do not duplicate without permission.
ILOG, CPLEX and their respective logotypes are registered trademarks.
All other company and product names are trademarks or registered trademarks of their respective holders.
The material presented in this document is summary in nature, subject to change, not contractual and intended for general information only and does not constitute a representation.
Table of Contents

1. Introduction........................................................................................................................................... 2

2. Business Policies and Business Rules................................................................................................. 3

3. Benefits of BRMS to Policy Managers ............................................................................................... 3

   Implementing rule changes according to the business rule life cycle .................................................. 3
   Author Rules in Business Language ...................................................................................................... 5
   Business Action Languages.................................................................................................................. 5
   Decision Tables and Decision Trees ..................................................................................................... 6

   Manage rules throughout their life cycle, from creation through testing, deployment and retirement .......................................................................................................................... 7
   Rule Organization ............................................................................................................................... 7
   Search 7
   Versioning and Auditing ....................................................................................................................... 8
   Analysis of Rules ................................................................................................................................. 8
   Testing 8
   Simulation ........................................................................................................................................... 9

4. What to Look For in a BRMS .............................................................................................................. 9

5. Conclusion .......................................................................................................................................... 9
1. Introduction

To the business consumer of information systems, the relationship with the IT personnel who build and maintain those systems can be a paradoxical experience: on the one hand, a partnership with IT delivers significant productivity benefits by automating routine tasks and improving access to information. On the other, the process of building automated systems often requires freezing business policies into software systems, limiting the business sponsor’s flexibility to adapt their operations to dynamic market conditions, individual customer demands, or changing regulatory environments.

The relationship between business and IT has often been fraught with misunderstanding, frustration and even hostility. Part of this disconnect between the IT organization and the business groups is a natural disparity between the work cycles of the two groups.

To the business, the IT system development cycle appears to be a long, drawn-out process. Further, once the policy managers approve the requirements it is often a leap of faith that the implementation is faithful to those requirements. It is not easy for policy managers to follow their requirements through the implementation process once they are translated into technical forms. In the implementers’ defense, what they receive as requirements is often not sufficiently detailed and precise; many requirements are “refined” in the development process, at best with the concurrence of the business people; at worst, at the discretion of individual programmers.

More importantly, though, once the translation is accomplished, the process for changing a business policy requires going through the same complex consultation and translation process for the changes. Thus, the policy changes become entangled with a software development cycle that is driven by a host of technology and resource issues, and is thus insufficiently responsive to the business sponsors’ needs. In effect, the owners of the business policy lose control over its expression and evolution when it is handed off to be embedded in a software solution, and never regain control.

However, business rule management systems (BRMSs) enable a giant leap forward in bridging the gap between business people and IT. Using BRMSs, the business people define their business policies and business rules and provide the clear communication between the policy managers defining the requirements and the developers implementing the application system solution.

Business people have control on exactly how their business rules are being executed, and, perhaps more importantly, the system is designed to facilitate change when business rules change. In traditional information systems, the business policies get hard-coded into the application. When changes need to be made, the entire software development life cycle must restart – understand the requirement, design it into the system, make sure it doesn’t adversely impact anything else, implement the change, test it and deploy it. IT rarely wants to go through this process for a single change, so requirements must be bundled into reasonable work efforts, further slowing down the process.

With a BRMS, the business logic (again, the part most likely to require change) is separated out and can be changed without impacting the remainder of the application. An assessment of the impact of the change on other business rules must still be done, but this is much simpler than a full system impact assessment. A single change can be assessed, implemented and tested in a very short timeframe (often hours). You do not have to wait until you have additional changes to make the effort worthwhile.

Once the problem is stated in this way, the solution becomes clear: the expression of business policies and the ability to change them need to be rigorously decoupled from the technology that implements them, allowing:

- Business policy experts to manage and evolve business policies using the methods and vocabulary with which they are most familiar
- Technology experts to manage and evolve technology using the methods and vocabulary most suitable to their tasks.

This separation of concerns is what BRMSs do. A BRMS is a suite of tools that policy managers and software engineers use to build systems in which business policies are abstracted out of software code, where they can be directly authored, modified and managed independently of the underlying software system.
2. Business Policies and Business Rules

In order to explain how a BRMS accomplishes this feat, we need to agree on some nomenclature.

Every organization has people responsible for setting the policies by which they do business. In this context, a policy is a statement of guidelines governing business decisions. An insurer may have an underwriting policy, for example, that says “an ‘underage applicant’ for insurance on high-powered sports cars is not eligible for coverage.”

A policy statement like this is not sufficient to form the basis of an automated decision. Someone has to translate the policy into more specific statements that specify the details of how the policy is enforced. We call that a person a policy manager. In the insurance example, the policy manager is an underwriter.

The specific statements that enforce the policy are business rules. Business rules are translations of the policies into detailed conditions and actions that unambiguously enforce the policy. For this to be possible, we start with an understanding the business process, business decisions and information included in the specific domain of the business policy. The understanding of the information will lead to the writing of a vocabulary that will be used in writing of the rules. An object model will be developed by software developers and mapped to specific data sources within the software infrastructure, but to the policy manager, the model remains an abstraction that describes their decision-making domain.

The business rules expand upon the policy, by stating the detailed circumstances under which it is applicable and the actions that enforce it. One policy can translate into many business rules. In the insurance underwriting policy described above, for example, the rules need to define the terms of the policy (what is an “underage applicant” and what is a “high-powered sports car”) and probably also need to specify variants of the policy enforcement. State regulations may require that “underage” be defined differently from state to state, and the notion of “high-powered sports car” may change over time, as underwriting experience shows specific models to incur more or less underwriting risk.

Even one policy domain, such as personal auto insurance underwriting in our example, can require hundreds or thousands of rules, constantly changing over time, and differing across jurisdictions (or, for other domains, customers, products, regions or any other partition of a policy domain). We call the process by which a company manages changes to policies and their enforcement, the business rule life cycle.

A BRMS provides policy managers with tools to efficiently define and manage business rules through the business rule life cycle. Specific software systems, called business rule applications, are built around the BRMS to invoke the rules and carry out the intended business policy. For insurance underwriting, for example, the underwriting application might invoke business rules to make a decision between rejecting an application, accepting it automatically, or referring it to a human underwriter for special consideration.

3. Benefits of BRMS to Policy Managers

From a policy manager’s perspective, the benefits of a BRMS stem directly from the ability of software developed using a BRMS to accomplish three separate things:

1. Implement rule changes according to the business rule life cycle
2. Author rules in business language
3. Manage rules throughout their life cycle, from creation through testing, deployment and retirement

Each of these is explained in detail in the following sections.

Implementing rule changes according to the business rule life cycle

Software is developed and deployed through a cycle of activities that is driven partly by business requirements, but also by technical and engineering demands (such as product upgrades or changes in other software systems...
with which it must integrate) not directly related to those requirements. The business rule life cycle, on the other hand, can be driven by a variety of non-technology demands. For example:

- **Market Dynamics**: Successful businesses need to be able to change their policies in response to market changes.

- **Regulatory Change**: In financial services, insurance, and other highly regulated industries, business rule changes can be demanded by the review cycle of regulatory agencies, or by randomly timed regulatory changes from regulators and courts. Government agencies administering benefit programs or complex laws must frequently deal with rapidly breaking policy changes dictated by executive fiat or legislative acts.

- **Customization and Personalization**: More and more businesses are enabling their applications to behave differently for different classes of customers or for individual customers. This may involve the same basic application that applies different rules based upon each customer’s contract terms, or it may involve much deeper customization of entire functions for each individual customer. Third-party logistics providers, for example, are masters of the art of moving material around the country and world. They provide these services to other businesses – manufacturers with geographically distributed plants, or large retail chains, for example – who need logistics services, but whose area of expertise is quite different. But the provision of these services will be “tuned” differently for each customer: an automobile manufacturer with a just-in-time manufacturing philosophy will want its services skewed toward absolute predictability and reliability, over and above cost, whereas a discount merchandiser may wish to accept to delay, in favor of cost savings. The third-party logistics provider customizes its basic competence in logistics for each of these customers quite differently.

---

**Figure 1: The business rule and software development cycles.**
All these drivers have one thing in common: they require a much higher degree of flexibility and responsiveness in translating policy into actionable rules. The result is that it can be very difficult to “marry” the resulting business rule life cycles to a traditional software release schedule. A business rule application mitigates this problem, by allowing separate cycles for software development and rule management, each with its own drivers and timing. This is shown schematically in Figure 1.

Note that the software development cycle, represented at the top of the diagram, is driven over fairly long timeframes, by major functional enhancements or platform upgrades. New software releases are major engineering efforts, characterized by traditional design, construction and testing methodologies, and are relatively infrequent. Policy changes on the other hand, represented in the bottom half of the diagram, are driven by business events which result in rule changes. The business events may be very frequent (as in selection of cross-sell or promotion opportunities in a retail online store, for example), or quite infrequent (as in the case of negotiated contract terms for a specific customer), but are in any case not on the same schedule as the software changes. That is, so long as the rule changes represent variations of decision making based on data already managed by the application, they can proceed according the business demands of the policy maker, not the technology demands of the application development process.

**Author Rules in Business Language**

Separating the business rule life cycle from the application development cycle enables more timely response to business change. However, to also empower the policy manager to enact these changes directly, a BRMS has to provide another important service: a means of expressing the business rules in a format and language that is familiar to the policy managers, and easily manipulated by them.

There are really two elements to meeting this requirement:

- Specification of a business language that can unambiguously express the policy manager’s intent, yet remain familiar and easy to manipulate.
- Translation of this language into something “executable” by the software application.

**Business Action Languages**

In many cases, these two elements can be met by means of a straightforward mapping between business vocabulary and technical software artifacts.

Software developers typically express their understanding of the information required to run a business in terms of an object model. An object model is a formal means for expressing concepts (like Customer, Order, and Item, for example), their relationships (a Customer can have multiple Orders, but an Order belongs to exactly one Customer), their content (a Customer is described by a Name, Street Address, City, State and a Customer Class - either Platinum, Gold, or Silver), and operations that can be performed on them (adding a message to an Order). The software developers will implement the object model to allow programs to manipulate the data. All the terms in the object model can be translated into a vocabulary that the policy managers can use when writing business rules. Combined with a simple if-then syntax (IF some set of conditions are true THEN a certain action should be taken), the policy manager can express business rules in statements that are still in business vocabulary but are precise and automatable. We call this combination of the business vocabulary mapped to the object model and the syntax a business action language. The small object model discussed above might be verbalized by a vocabulary as shown in the following table.
<table>
<thead>
<tr>
<th>Concept, Data or Operation</th>
<th>Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>the customer</td>
</tr>
<tr>
<td>Name</td>
<td>the customer’s name</td>
</tr>
<tr>
<td>State</td>
<td>the customer’s state</td>
</tr>
<tr>
<td>Customer Class</td>
<td>the customer’s value class</td>
</tr>
<tr>
<td>OrderTotal</td>
<td>the customer’s historical total order value</td>
</tr>
<tr>
<td>Order</td>
<td>the order</td>
</tr>
<tr>
<td>Discount</td>
<td>the order discount</td>
</tr>
<tr>
<td>Amount</td>
<td>the order amount</td>
</tr>
<tr>
<td>Date</td>
<td>the order date</td>
</tr>
<tr>
<td>Add Message</td>
<td>add the message &lt;message&gt; to the order</td>
</tr>
<tr>
<td>Item</td>
<td>the item</td>
</tr>
<tr>
<td>SKU</td>
<td>the catalog number</td>
</tr>
<tr>
<td>Quantity</td>
<td>the quantity</td>
</tr>
<tr>
<td>Price</td>
<td>the item price</td>
</tr>
</tbody>
</table>

Table 1: Mapping business object model to vocabulary

Now suppose that a policy manager needs to enforce a rule that says that all “gold” customers in Minnesota who place an order in January receive a 10% discount. The policy manager would write a rule like that in Figure 2.

If

the customer’s state is “MN” and
the customer’s value class is “Gold” and
the order date is between “January 1, 2005” and “January 31, 2005”

Then

set the order discount to 10% and
add the message “As a gold customer you receive a 10% discount on today’s order.” to the order

Figure 2: A rule in a business action language for computing a discount

Note the generic if-then structure of the rule, the use of the established vocabulary (underlined and mapped behind the scenes to the object model) to define the rule’s conditions and actions, and the readable, though precise, language of the rule.

Decision Tables and Decision Trees

Sometimes it makes sense to express a rule as an if-then statement, but often it is more useful to express a set of rules that involve the same conditional terms in either a table or tree form. A decision table is defined by the set of conditional terms as column headers with each row representing a single rule and the result of each rule in the final column.

In the order example above, a decision table like that in Table 2 might be used to compute the customer’s class based on their total order history and state.
<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Historical Order Volume</td>
<td>State</td>
</tr>
<tr>
<td>1 between $1000 and $2000</td>
<td>NY, CA</td>
</tr>
<tr>
<td>2 between $2000 and $2500</td>
<td>NY, CA</td>
</tr>
<tr>
<td>3 between $500 and $1000</td>
<td>IA, MO, IL</td>
</tr>
<tr>
<td>4 between $1000 and $2500</td>
<td>IA, MO, IL</td>
</tr>
<tr>
<td>5 greater than $2500</td>
<td>Any</td>
</tr>
</tbody>
</table>

Table 2: Decision table for customer class

A decision tree also combines a set of business rules into one graphical display, but rather than each rule being a row, each rule is a path through the tree, branching at each conditional term. The density of the tree depends on how many possible values there are for each condition – with a few (e.g. true/false), a tree is a good display choice; with many (e.g. each of 50 US States), a table is often more readable. Both decision tables and decision trees are extremely useful for viewing multiple related rules at once, to ensure that you have addressed all scenarios and that the rules makes business sense as a group.

**Manage rules throughout their life cycle, from creation through testing, deployment and retirement**

The advantages gained by with the ability to quickly identify which rules need to change due to a change in business policy, simulate the proposed change prior to going into production and then implement all the rule changes in the timeframe the business needs is a competitive advantage to organizations. The business rules by which an organization implements its business policies define how the business operates and are valuable assets thus should be managed accordingly. This means:

- Organizing rules in a way that is logical to the policy manager
- Making rules accessible to the policy manager and searchable by relevant criteria
- Versioning rules and maintaining an audit trail
- Analyzing rules for consistency, completeness and business efficacy
- Testing the implementation of rules to ensure it is faithful to the business intent
- Simulating rule changes to understand the impact before implementation

**Rule Organization**

A typical enterprise has many policy managers who are responsible for thousands of rules. To make this efficient, a BRMS needs to provide simple, transparent capabilities for organizing rules. Rule organization takes place in a repository that manages all rules for one or more projects. Typically, business rules are grouped by high-level content areas which are then broken down further by tasks or decisions. For example, a property and casualty insurance company might start with lines of business (Fire, Auto, Property, etc.) and then have Rating Rules, Pricing Rules and Claim Processing Rules under each.

**Search**

To effectively manage policies implemented as large numbers of rules, a BRMS provides the ability to search the rule repository for specific rules, based upon criteria other than organizational structure of the repository. A BRMS enables the assignment of additional metadata properties (built-in or customized to your own needs) such as who authored the rule, when the rule was last modified or the effective date range for the implementation of the rule. It may also be necessary to search by more application-specific criteria such as the states (jurisdictions) to which
the rule applies. These properties are in addition to the ability to search based on the vocabulary terms used in rules and the actions taken as results. The more accessible the information, the better business people across the enterprise can use and build upon the rules already written.

For example, a regulatory compliance officer for the property and casualty company using the physical organization and metadata mentioned above, could collect and inspect all the rules for California in effect for a given date, across all lines of business and coverage types, by issuing a query for those specific rules.

Versioning and Auditing

One key requirement of policy management is the ability to track individual rules and sets of rules as they change over time, and to verify the state of a rule as of any given transaction. A BRMS facilitates this type of audit, by keeping each version of a rule in the rule repository, and tracking when each version was deployed and active. The versioning system may also track the identity of each individual who changed a rule, as well as require annotation of the changes. In this way, the BRMS creates an automated audit trail for policy changes in the organization.

Analysis of Rules

Business policies can be quite complex, and the formalization of the expression of the rules that implement a policy enables more systematic analysis of dependencies, consistency and faithfulness to the business intent. Having rules in a formal syntax, the BRMS can highlight where the result of an action in one rule is used in the condition part of another rule – since the second is dependent on the first, you know changing the first will impact the second. If you have long dependency chains, the ability to highlight the dependencies further down the chain is crucial to understanding the total impact of changes. A BRMS can also identify inconsistencies and redundancies in your rules:

- Whether the rule is self-inconsistent (because it specifies conditions that cannot be true, for example).
- Whether rule conflicts with another rule.
- Whether the rule is redundant (because it is equivalent to another rule, for example).
- Whether a set of rules has gaps in the scenarios addressed

Finally, having rules organized and formally expressed enables the policy manager to review sets of rules to ensure they really do result in the appropriate set of actions. A BRMS cannot make this determination for the policy manager, but it can lay out the information in a manageable way.

Testing

Policy managers authoring rules in a BRMS need to have tools that permit them to verify that the rules they write actually work as required. In its simplest form, that calls for a means to answer the question "if I run my rule on these specific values, will it fire and produce the expected results?". This might be called "unit testing" for rules.

It is also necessary to test rules in aggregation, to determine if, collectively, they implement fully and correctly the intended policy. This requires the ability to specify a complete rule set, and a complete set of input data, and again verify that the results are as expected. This can be called "functional test" of rules.

Finally, because even with constantly evolving policies, businesses require stability in most of their functions, there needs to be a mechanism for verifying that a proposed new rule set satisfies standing constraints on its functionality. This may require running many complete input data scenarios against a candidate set of rules, and evaluating the aggregate results. This is "regression testing" of rules.

All three testing types are supported by an enterprise BRMS.
Simulation

One utility of a BRMS is that, because policies are given a precise operational form, as rules, a proposed set of policies (set of rules) can be evaluated rigorously to determine if it yields the desired results. One version of that is the regression testing of rules mentioned above, but it is also possible to use a BRMS to do more prospective verification of rules. Suppose, for example, an insurer’s underwriting goal is to accept 80% of tendered risks, without exceeding a specified aggregate exposure index. A BRMS “simulation” capability would facilitate this as follows: underwriting managers would craft detailed policies with the intent of meeting this goal, and translate these into a “candidate” rule set. This candidate rule set would then be run against all application received in, say, the last month, with the results of each evaluation being recorded. Finally, key performance indicators, including the percentage of yield, and the aggregate exposure index would be computed on the results, permitting the underwriting manager to discern whether the particular policies written into the rules would have met the overall business goal.

4. What to Look For in a BRMS

From the perspective an enterprise policy manager, evaluating a BRMS can be a daunting task: a BRMS delivers its benefits by enhancing the function of enterprise applications, not by replacing them wholesale. As a result, a BRMS is by its very nature shared between policy managers and software developers, and both perspectives need to be accommodated when evaluating a BRMS. There are, nevertheless, some important features a business user should look for in a BRMS:

- **Address the needs of business and IT**: A BRMS should support tools designed specifically for policy managers that isolate them from the concerns of application developers, and have at the same time, tools that support and simplify the developers’ task of integrating the BRMS into software applications.

- **Provide domain-appropriate rule artifacts**: As discussed above, there are many ways to express a business rule. A good BRMS should support those that are most natural for a given business, or provide a means to extend and alter its default rule languages to accommodate the business’ specific needs.

- **Address the need to manage the entire business rule life cycle**: Typically, a successful business rule application will eventually encompass hundreds to perhaps tens of thousands of rules, which will undergo constant modification and extension as policies change. To be fully effective, a BRMS must support the policy manager at every step in the business rule life cycle, in order to make that large collection of rules intelligible, verifiable, and manageable.

5. Conclusion

Managers responsible for policy in decision-rich enterprises have benefited enormously from information technology. But as the pace of change has increased, and the reach of automation has been extended, traditional software applications have proven unable to deliver the flexibility and agility these managers require. Applications constructed with BRMSs, on the other hand, can deliver that flexibility and agility, and enhance the entire policy management infrastructure of the enterprise at the same time.