

K-SITE RULES:

Integrating Business Rules in the mainstream software engineering practice

José L. Martínez-Fernández⁽²⁾, José C. González^(1,2), Pablo Suárez⁽²⁾
(1) DAEDALUS – Data, Decisions and Language, S.A., Avda. Albufera 321, Madrid, Spain
jmartinez@daedalus.es, jgonzalez@daedalus.es, psuarez@daedalus.es
(2) ETSI Telecomunicación, Universidad Politécnica de Madrid

Keywords: Business rules, rule engine, rule language, software development lifecycle models.

Abstract: The technology for business rule based systems faces two important challenges: standardization and integration within conventional software development lifecycle models and tools. Despite the standardization effort carried out by international organizations, commercial tools incorporate their own flavours in rule languages, making difficult the migration among tools. On the other hand, although some business rules systems vendors incorporate interfaces to encapsulate decision models as web services, it is still difficult integrating business rules in traditional object-oriented analysis and design methodologies. This is the rationale behind the development of K-Site Rules, a tool that facilitates the cooperation of business people and software designers in business applications.

1 INTRODUCTION

A Business Rule (BR) can be seen as an expression used to make explicit the knowledge about the business present in an organization. According to the Business Rules Group¹ two definitions can be given for the term business rule, depending on the point of view adopted: from the business point of view, where a business rule is “*a directive, intended to influence or guide business behaviour, in support of business policy that has been formulated in response to an opportunity, threat, strength, or weakness*”, from the Information Technology (IT) point of view a BR can be defined as “*a statement that defines or constrains an aspect of the business. It is intended to assert business structure, or to control or influence the behaviour of the business*”.

A Business Rules Engine (BRE) or Business Rules System (BRS) is a software system in charge of executing IT business rules. These systems usually include tools to support the definition, verification, validation and management of business rules.

In the last few years, the use of business rules systems is being widely adopted by companies to encode business knowledge. But, why? What are the benefits of using business rules? There are three main reasons: First, the knowledge about the business is stored in a somehow centralized, easily

accessible storage, and this knowledge can be modified, updated and maintained in a quick and simple way. Second, companies need ways to foster their reaction to new business opportunities, and the ability to adapt their IT systems to take benefit of these opportunities becomes a definitive advantage. Third, from the IT perspective, the use of business rules allows a reduction of the effort needed to adapt software systems to new requirements. The ideal situation is one in which no code modifications are needed to satisfy these new restrictions. Let's try to explain this fact with an example: suppose you are managing a web site selling electronic devices and you want to make a 10% special discount to customers coming for their first time to the web site. You can implement this restriction using an if-then sentence in your code. Now, suppose you want to extend the special discount to every customer for a period of time. Then you should change your code and recompile it to include this new restriction. On the other hand, if you express the restriction as a business rule, you only have to change the business rule definition without changing the code of the application, and with no need to recompile your program.

Nowadays, there are a lot of commercial products implementing business rules engines. Blaze Advisor (Fair Isaac, 2007), ILOG JRules (ILOG JRules, 2006) and JBoss Rules (Red Hat, 2007) are

¹ <http://www.businessrulesgroup.org>

some of the most important ones. Of course, you have to make a decision about which product you are going to include in your IT architecture. Each product uses its own language to define business rules, so a standardization problem arises. Although there are several efforts to define a standard rules language such as RuleML, SWRL (Semantic Web Rule Language) or RIF (Rules Interchange Format), these languages are not implemented by commercial products.

On the other hand, if business rules are going to be used to implement software components in companies, there should be a way to include them in the software development lifecycle. For this purpose, tools supporting the harvesting, definition and implementation of business rules should be provided, as well as their utilization in the corresponding stages of the development lifecycle.

K-Site Rules is a tool that covers these two main problems. The first one, by helping in the standardization of business rules representations by providing automatic translations among SWRL and the languages interpreted by commercial products. The second one, providing a simple way to assure configuration management in the business rules development process and development tools, which can be easily integrated within standard software development lifecycle models.

The remainder of this paper is structured as follows: Section 2 describes key issues relating the integration of BRSs in the development cycle. Section 3 describes the architecture of the K-Site Rules solution, focused in assuring the independence of the commercial BRS selected to implement business rules. Section 4 depicts a use case for K-Site Rules and Section 5 includes some conclusions extracted during the design and development of K-Site Rules.

2 ROLES IN BUSINESS RULES SYSTEMS DEVELOPMENT AND INTEGRATION

As mentioned in the introduction, the development of decision support systems based on business rules must face two main problems: there is a great variety of different rule engines supporting different rule languages and it is difficult to foresee the point in the development cycle where rules construction should be carried out. In order to make business rules application development independent of rule

engine and rule language selected for implementation, three basic issues must be analyzed:

- The way in which applications can interact with rule engines regardless of the rule engine product selected
- How applications are going to transmit rule definitions to rule engines regardless of the product selected to implement business rules
- How the development of business rules is going to be integrated in the development lifecycle model for the rest of the application being constructed.

The following three subsections are devoted to each one of these main issues.

2.1 Rule Language Standardization efforts

A lot of effort is being devoted to the definition of standard rules languages to assure that a rule can be defined once and implemented in different rule engines. Among these efforts, it is worth highlighting the following ones:

2.1.1 Simple Rule Markup Language (SRML) and Business Rule Markup Language (BRML)

SRML language (Thorpe&Ke, 2001) was an initiative started by ILOG, a company specialized in BRSs development. The main goal was to define a generic XML based language with constructors from several proprietary languages. In this way several BRSs could share the same set of business rules. This initiative has been substituted by the W3C Rules Interchange Format (RIF) proposal.

On the other hand, BRML was a language defined by IBM under the framework of an eCommerce project. The idea was also to define a language to interchange rules definitions and now it has also been substituted by the RIF initiative (described in 2.1.3).

2.1.2 RuleML

RuleML stands for Rule Markup Language (Boley et al., 2005) to publish and share access to rule databases across the web. It is inspired by SQL (Structured Query Language) and Prolog and uses XML to define the structure of the rule. This structure is based in three main elements: a head for the rule (the rule antecedent), a body (the consequent) which are formed by *atoms*. An atom is

a relation among constant values. Rules engines exist specifically built to interpret RuleML language, such as JDrew and Mandarax.

2.1.3 Rule Interchange Format (RIF)

The W3C (World Wide Web Consortium) has created a workgroup to develop a format to simplify rules interchange. This format is called Rule Interchange Format (RIF). The main goal of this workgroup is to define a rule language and necessary extensions to allow rules translation among different BRSs. In this way, rules definitions could be shared and reused by different engines. A draft for a Core Condition Language (Boley&Kifer, 2007) has been the first step taken by this group along with a set of use cases where the standard could be applied.

2.1.4 Semantic Web Rule Language (SWRL)

During last years a great effort has been devoted to the development of the so called Semantic Web, an attempt to put meaning in the web in a way that machines and applications are able to interpret. Of course, the business rules world is not aside these efforts and an expression of business rules using Semantic Web approaches has already been built. This effort is called Semantic Web Rule Language (SWRL) (Horrocks et al., 2003) a rules language combining OWL (Web Ontology Language) with RuleML. For this purpose, SWRL includes an abstract syntax for Horn clauses, with a knowledge base expressed in OWL. The proposed rules are defined as an implication between an antecedent (the head of the rule) and a consequent (the body of the rule) where both elements can be constituted by atoms. Among other things, an atom can be some condition over an OWL class or instance. In this way, the elements considered to build the rule must be found in the domain defined by an ontology.

2.2 JSR-94, a standard programming interface for rule engines

JSR-94 (Toussaint, 2003) is the name given to the standard JAVA application interface programming (API) defined to make applications independent of the rule engine product used to implement business rules.

This specification states a set of basic operations that every rule engine should provide. This set of operations is selected based on the assumption that all users of a rule engine needs an

execution of a cycle including the following steps: rule analysis, inclusion of new objects to the engine, firing rules and obtaining result objects from the engine. This interface does not state the language in which rules must be supplied, and neither gives semantic to the rule execution cycle.

All rule engine products with relevance in the market implement this standard, including: Blaze Advisor, ILOG JRules, JBoss Rules and Jess.

2.3 Stages in the business rules development process

There exists a methodology to harvest and develop rules languages called PROTEUS (Ross, 2006). K-Site Rules proposes a simplification of this methodology and provides development tools to support the stages in this reduced methodology.

2.3.1 PROTEUS Methodology

PROTEUS is a methodology providing a set of steps and techniques defined to allow an easy capture, expression and organization of business rules.

Among its main characteristics is worth mentioning: the ability to express, organize and exhaustively capture business rules; its business orientation; the inclusion of a guide to facilitate the requirement analysis; the possibility to build a business model with the involvement of the user; the harvesting of business rules from the products delivered with the business model and the development of a document where software developers can get answers for questions about the business. Although this methodology is business oriented, an exhaustive list of requirements must be obtained in order to allow software architects the design and development of the rule based system. This methodology defines a business rule as a sentence (directive, business expression or formal expression) about a specific theme. Every rule can be classified, taking into account different aspects, into four main classes: business category, referring the basic function of the rule in the business execution; functional category, according to the main operation or effect (computation, exclusion, projection) of the rule; abstraction level category, taking into account how strict must the application of the rule be and; finally, the system category, which takes into account the objective of the operations and actions related to the rule.

This exhaustive methodology has been taken as the baseline for the development cycle of K-Site Rules, described in the next section.

2.3.2 K-Site Rules development cycle

The development steps defined in K-Site Rules are a simplification of the PROTEUS approach. Before describing the methodology, some definitions must be given according to the point of view considered in K-Site Rules: a decision service is a set of atomic business rules and actions (i.e.: operations included in business objects); an atomic business rules is a rule that can be expressed in one sentence.

Four basic steps are considered:

- *Workflow definition*

A flow must be defined among sets of atomic rules, making easier to understand the operations performed by the rule and allowing the reuse of already defined atomic rules and rulesets.

- *Atomic rule definition*

Definitions for atomic rules are supported in different formats. In K-Site Rules an atomic rule can be defined using natural language, decision tables and decision trees. Concepts and facts available to define rules are taken from JAVA object models. Atomic business rules are expressed in the standard SWRL language.

- *Decision service validation*

Once atomic rules have been defined and included in the decision service workflow and this decision service is complete, validation tasks must be done. The main goal is to test if the functional behaviour of the decision service is correct or not. In order to perform this validation, a target rule engine must be selected and K-Site Rules will automatically translate the decision service to the selected engine, performing predefined validations. Mechanisms to provide data for these validations are, of course, supported by the tool.

- *Decision service publication*

A version control system is also integrated in K-Site Rules, to manage different versions of business rules and also to publish or deploying rule definitions.

These developments steps are exemplified in a use case in section 4.

The PROTEUS methodology considers an initial phase to build the business model for the organization. In a first version of K-Site Rules, this business model is inherited from the organization, so valid concepts and facts are going to be those

already defined in the target organization and included in some kind of JAVA business objects repository.

3 A COMPREHENSIVE ARCHITECTURE FOR BR SYSTEMS DEVELOPMENT AND INTEGRATION

Most of Rule-Based Systems offer an architecture similar to the one showed in Figure 1.

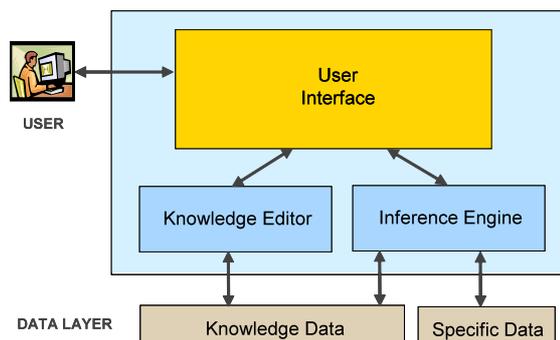


Figure 1: Rule-Based System general architecture

The user interacts with the Rule-Based system through a User Interface. It allows access to a Knowledge Editor that supports the creation or modification of new rules, possibly using natural language. These rules are stored in a repository which constitutes the Knowledge Data element in Figure 1. Along this process, the Knowledge Editor could make use of previous data stored in this Knowledge Data repository. Rules stored in this repository are typically structured in form of if-then-else logical propositions. On the other hand, the Inference Engine or Rules Engine obtains inferences starting from the Knowledge Data and the Specific Data. Obviously, the Inference Engine can use intermediate data generated in successive steps along the inference process.

K-Site Rules implements a layer allowing the integration of Rule-Based Systems, and the architecture of the tool was thought in accordance with the following objectives:

- Make rules development independent from rules engines, allowing the integration with several different engines (view Figure 2).

- Develop and maintain rules in a simple and intuitive way.
- Coordinate all the processes involved in rules definition for making easier the reuse of the rules.
- Control access to rules according to the user profile.

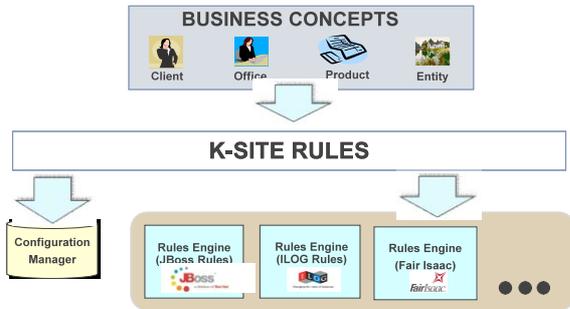


Figure 2: A global vision of K-Site Rules Integration

With the aim of reaching all these objectives, the basic architecture of K-Site Rules is depicted in Figure 3.

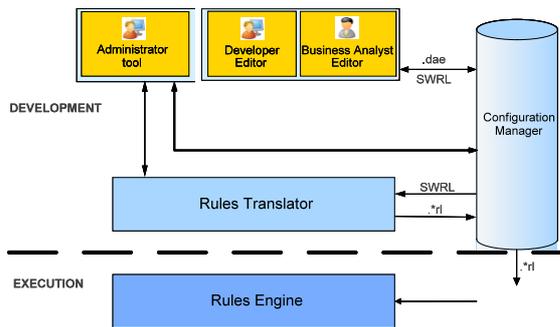


Figure 3: K-Site Rules Architecture

The most important elements of this architecture are the next ones:

3.1 Rule Editors

K-Site Rules offers an editor for each different role. In particular, we include in our architecture a Business Analyst Editor and a Developer Editor. The Developer Editor was developed as a plug-in in an eclipse-based IDE. This plug-in allows the integration of the editor in a modelling tool for the definition of business rules. The Business Analyst Editor was developed as a friendly web tool allowing access to the system through a standard browser. It can also be integrated with the corporate

access control system. Both editors allow the user to create atomic if-then-else rules in three different ways: through guided natural language, decision tables or decision trees. The editors generate Knowledge Data in an intermediate standard language known as SWRL. Some data (.dae) must also be stored in the Knowledge Data repository in order to reproduce graphic representation of rules and rule flows. The editors are also integrated with the Configuration Manager (Figure 3).

3.2 Rules translator

Independence among the rules development and the rules engines in K-Site Rules makes necessary the incorporation of a translator in the architecture. It performs the translation of the intermediate SWRL files to the specific Rule Engine files (Figure 4). These languages are those provided by rule engines like ILOG JRules (IRL) or JBoss Rules (DRL). The interpreter is also integrated with the Configuration Manager (Figure 3).

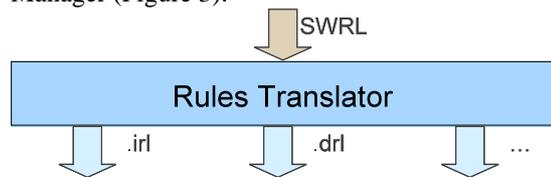


Figure 4: Translation process

3.3 Configuration Manager

The Configuration Manager facilitates the reuse of Business Rules, validates the deployment of Business Rules, audits changes in the Rules and controls and maintains historic data about Business Rules versions. In addition to these features, the Configuration Manager communicates all involved user roles (business analyst, developer and administrator) in a transparent way, so that the changes carried out by a role will be visible for the other. It controls the access permissions to rules for each user too, and makes it possible the integration with the corporate configuration manager. K-Site Rules provides support for different configuration servers, like Subversion (Collins-Sussman et al., 2007) or Clear Case (IBM, 2007).

3.4 Administrator tool

In addition to these elements, K-Site Rules incorporates an Administrator Web Tool that

permits access, through a standard browser, to configuration and control parameters of all components constituting K-Site Rules. It allows the definition and configuration of all necessary repositories in the Configuration Manager System. Moreover, it allows the management of users with access to K-Site Rules, including access and deployment permissions of Business Rules.

Finally, the dashed line in Figure 3 represents the separation between the elements involved in the development or execution procedures respectively.

The main features of K-Site Rules are summarized in this couple of definition equations:

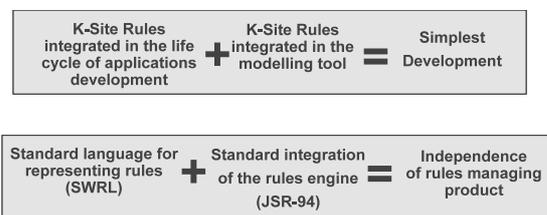


Figure 5: K-Site Rules definition equations

4 USE CASE

We will pay now attention to the process followed by a developer, a Business Analyst or an Administrator to make use of K-Site Rules.

4.1 Developer use case

For the developer, K-Site Rules will be offered, as said before, as a plug-in integrated in a modelling tool. The developer will start from a UML model that represents all the business objects included into the Business Model, its attributes and methods, and all relations among them. Related to this model, there will be a kind of elements stereotyped as “Decision Service”, with only an associated method. The right button click of mouse over one element of this type will launch the K-Site Rules plug-in. Login and password will be required and then, associated rules information will be downloaded from the tool’s repository. If there is some rule information associated to the corresponding Decision Service (Business Rule), all this information will be downloaded. Then, a tab control will be shown in the plug-in view, with a flow diagram that represents the Decision Service.

This flow diagram represents the Business Rule or Decision Service itself. It is represented as a conventional rule flow, with a start point, a sequence of rule sets and flow packages connected through arrows and an end point. Clicking over a flow package, we can see a new flow diagram with the same features mentioned. The flow screen will offer options related to the business rule publication, i.e., options as commit, update, and so. Besides, options related to business rules validation, that allow a business rules proof by means of tables that permit the introduction of the needed information, will be shown. All these options are related to the complete Business Rule or Decision Service. From this screen we can see the properties associated to the Decision Service too.

When we double click over a rule set element in the flow diagram, the plug-in will open the atomic rules editor. This editor will show all the atomic rules (including decision tables and decision trees) associated to the selected rule set. From this Rules Editor we can create, modify or delete atomic rules associated to this rule set by means of three different tools: a natural language editor, a decision table and a decision tree. All of them allow access to the properties associated to each atomic rule. In addition, all of them allow for version control options referred to the atomic rule (commit, update, etc.) The natural language editor permits to guide the rule creation process in business language. In the decision table, the atomic rules are introduced through a table form. Finally, in the decision tree a tree form of data introduction is enabled. All these atomic rules are automatically translated to the SWRL standard language as was said before.

Finally, the plug-in offers a panel that enables the translation of standard SWRL files associated to the atomic rules to the specific inference engine files (view Figure 5). It represents the implementation of the Translator of our architecture (view Figure 3 and Figure 4). This panel allows the developer to select a rule set or a set of them and a concrete destination Rule Engine, and translate all SWRL of the associated atomic rules to the destination Rule Engine language. The Translation panel also allows navigation through the original SWRL files and the translated ones as a comparison utility. Other options offered by the Translation panel are: 1) the possibility to export the generated files to an adequate directory when the tool detects the presence of projects characteristic of a certain Rule Engine (present through other plug-ins offered by

third parties in the Modelling Tool); 2) access to the rules debugging tools of the destination Rule Engine.

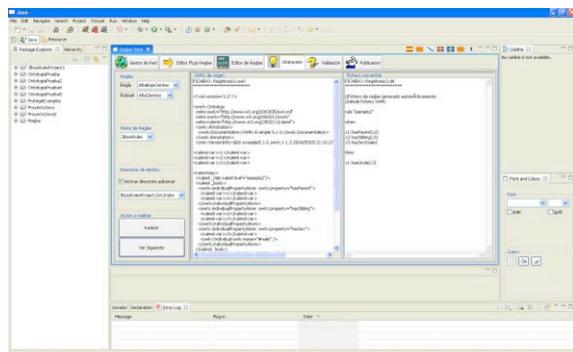


Figure 6: Translator screen

4.2 Business Analyst use case

The Business Analyst tool was thought as a web application that can be connected through a common standard browser. This tool is devoted to allow business analysts, usually not having technological knowledge, the definition and validation of business rules without the intervention of developers. To access this tool, a user and password is required. Then, a rules screen is offered to the user. In the left part of this screen, the user can select the Decision Service (Business Rule) required through a combo box. In the centre and right part of the screen, a table with all versions of the selected Decision Service and its components is offered. When the business analyst selects a version and click on the Edit button, the web tool opens a flow diagram screen. This screen is similar to that explained in the developer tool, with identical options and identical access to the Rules Editor screen. The Rules Editor is similar too, and, as in the other case, it allows the creation of rules through natural business language, decision tables and decision trees. For more details, the reader should see the developer use case.

4.3 Administrator use case

The Administrator tool was thought as a web application that can be accessed through a common standard browser. A user and password is also required, like in the Business Analyst Tool. The Administrator tool offers three main options through a tab panel:

- A screen that allows the administrator to manage the user accounts and groups and the associated permissions.
- A screen allowing the administrator to configure the different repositories required by the tool.
- A console mode screen, allowing the administrator the management of system scripts to simplify administrative tasks such as rule deployment.

5 CONCLUSIONS

This paper presents K-Site Rules, a tool and a methodology for business rules integration and deployment. The main issue addressed by K-Site Rules is the effective cooperation of business people and software engineers in the design of software components, specially in the case of components incorporating declarative business knowledge in the form of business rules. In this sense, the main contribution of this work to the state of the art is the possibility of collaboration between business experts and software developers in a typical Integrated Development Environment (IDE), addressing key issues as configuration management.

K-Site Rules does not incorporate its own inference engine. It relies on commercial or open source software for the debugging and execution of business rules. Instead, it permits the specification of decision processes in the standard SWRL language, translating business rules to the proprietary languages of commercial rule engines.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Center for Industry Technological Development (CDTI, Ministry of Industry, Tourism and Trade), through the project ITECBAN (Architecture for Core Banking Information Systems), INGENIO 2010 Programme. Other partners in ITECBAN are INDRA Sistemas, CajaMadrid, Sun Microsystems and Grid Systems. Special mention to our colleagues at INDRA must be done for their involvement in the specification of K-Site Rules: Fernando Alcántara, Pablo Leal, Juan Carlos Macho and Gonzalo Pando (in alphabetical order).

REFERENCES

- Boley, H., Grosf, B., Tabet, S., 2005. *RuleML Tutorial*, Draft. Available at <http://www.ruleml.org>, last visit: 3/12/2007
- Boley, H., Kifer, M., 2007. *RIF Core Design*. W3C Working Draft 30 March 2007, W3C Consortium. Available at <http://www.w3.org/TR/2007/WD-rif-core-20070330/>, last visit: 3/12/2007
- Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M., 2007. Version Control with Subversion: For Subversion 1.5, Available at <http://svnbook.red-bean.com/nightly/en/svn-book.pdf>, last visit: 3/12/2007
- Fair Isaac, 2007. *Blaze Advisor Business Rules Management System: How it works*. Fair Isaac Corporation
- Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosf, M. Dean, 2003. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, draft 0.5. Available at <http://www.daml.org/2003/11/swrl/>
- IBM, 2007, IBM Rational ClearCase, Available at <ftp://ftp.software.ibm.com/software/rational/web/datasheets/clearcase.pdf>, last visit: 3/12/2007
- ILOG JRules, 2006. Business Rule Management (BRM) with ILOG JRules 6. BRMS without compromise. ILOG Inc.
- Moore, R., Lopes, J., 1999. Paper templates. In *TEMPLATE'06, 1st International Conference on Template Production*. INSTICC Press.
- Red Hat, 2007. JBoss Rules Fact Sheet. Red Hat Inc. Ross G., Ronald, 2006, *Business Rule Concepts. Getting to the point of knowledge*, Business Rules Solutions LLC.
- Smith, J., 1998. *The book*, The publishing company. London, 2nd edition.
- M. Thorpe, C. Ke, 2001. *Simple Rule Markup Language (SRML): A General XML Rule Representation for Forward-chaining Rules*, from <http://xml.coverpages.org/srml.html>
- Toussaint, 2003. *Java Rule Engine API Specification JSR-94*, Draft 1.0, Available at <http://jcp.org/en/jsr/detail?id=94>