**W3C Workshop on Rule Languages for Interoperability**

# Fair Isaac Blaze Advisor Structured Rules Language - a commercial rules representation

**March 2005**

Author: Paul Vincent, Fair Isaac

Contents:

# 1. Introduction & Context

## *1.1 Background*

Fair Isaac markets and sells the Blaze Advisor rule management solution, which is used around the world for:

- Policy-type rules for process automation, running as "rule services"
- External commercial and government customers, as well as Fair Isaac solutions such as originations and fraud detection
- Third party software vendors who need to utilize high performance rules technology without inventing their own.

Fair Isaac traces its roots in rule use (& representation) from its forerunners:

(a) Neuron Data / Blaze Software, which developed one of the first API-enabled rule engines for commercial use in the mid-1980s (Nexpert Object™, Smart Elements™, Expert) before introducing the Java-based Blaze Advisor in 1998.
(b) HNC, which developed custom rule engines for fraud and originations.
(c) Fair Isaac, which developed both custom rule technology for originations and credit scoring, and a commercial decision engine (Blaze Decision System™).

Fair Isaac's current rules strategy is to continuously improve the Blaze Advisor rules representation and services by merging it with the Blaze Decision System functionality, and selectively embed the features of other Fair Isaac rule engines to provide a single, powerful Enterprise Decision Management solution.

This paper aims to give some insight to W3C workshop attendees into the role and features of the rule language, how it relates to rule management and execution, how it is evolving, and how it fits in the standards world.

## *1.2 Rule Usage*

Blaze Advisor can be used to define the following types of rule services:

- stateless rule services, typically used in conventional transaction processing, for example using the JSR-94 rule service invocation standard
- stateful rule services, typically used in event monitoring type applications
- form validation.

A number of rule execution technologies are available to the technical architect, so that, for example, stateless rule services can be deployed as a Java, J2EE, .NET or COBOL rule service, whereas form validation is typically a W3C XFORM using a mixture of XPATH and web services.

## *1.3 Deployed Rules*

Blaze Advisor users deploy typically 100s to 100,000s of rules for any particular system. It is likely that 1M+ rules have been deployed in commercial use since Blaze Advisor was launched.

# 2 Structured Rules Language

## 2.1 Goals

Blaze Advisor uses a Structured Rules Language (SRL) to provide a formal, near-natural language rules language that is readable by business people, yet powerful enough to implement not only standard business policies, practices and procedures, but also AI-type rule-programming techniques when combined with Rete-based deployments.

## 2.2 Main Features

SRL is a language with an associated XML representation. SRL can be used to define the following types of entities:

- production rules in rulesets (which may be flagged to operate procedurally or use a Rete-based rules engine at run-time)
- "patterns" that define constrained collections against available data (such as a specific class)
- event –condition-action rules, acting in the context of a rule service or locally in a ruleset
- functions (ie procedural algorithms) (although sometimes these will be mapped from an external implementation or library)
- classes, objects, enumerations and variables (although usually these are mapped from external implementations of business objects etc).

Blaze Advisor also supports representational structures such as

- templates for rule entities, allowing design patterns to be re-used for any SRL structure
- "instances" of such templates
- metaphors for other decisioning components that are implemented as specialized SRL templates (decision tables, trees, score models).

The main area of interest is the flexibility it provides to rule developers: clearly this is closely associated with the functionality of the underlying rules engine(s).

- Rule / business term syntax choices (eg `cat's whiskers; the whiskers of cat; cat.whiskers`)
- Priorities (also effective in ordering rules for sequential / procedural processing)
- Effective dates (time and/or date range for rule validity)
- Pattern-based rule conditions (rule conditions can refer to a previously defined pattern, that can match any quantity of data)
- "Rule inheritance" (rulenames can be used in conditions to re-use the conditions of that rule)
- Set operations (eg `aircraft.passengers.checked_bags.weight.total`, maintained for any change to the collections of `passengers` or `checked_bags`)
- Extended boolean logic: data states include `known, unknown, available, unavailable,` and `null`.
- Declarative (unordered) rule statements (for Rete execution) and associated non-monotonic reasoning and Inferencing.

Rulesets features include:

- Parameterizable and can return a value (or return object).
- Specification of local definitions of data patterns, variables, event rules, etc.
- Execution mode (Rete, sequential interpreted, or sequential compiled).

### 2.3 Rule Management

Although SRL provides a high-level language for rules for business analysts, it is often considered insufficient for the purposes of business-level control by business managers:

- Business users should not be expected to define rules in a formal typed language (like SRL) in a developer's environment
- Business users require a business interface that constrains their actions to permissible values (eg web-based forms with drop-down menus)
- Business users require terminology to be in the language of their business domain
- Business users need to be able to update executing rules without IT intervention

To satisfy these requirements, and avoid undue loads on IT departments, Blaze Advisor provides a Rule Maintenance Application Generator that uses HTML business terms defined in rule templates, and constructs a web-based application called a Rule Maintenance Application (RMA). RMA users can then update rule instances in a repository that can then be mapped to SRL for testing / validation and deployment.

In addition, rule metaphors such as decision trees and tables can be edited within RMAs; rule metadata and repository information can be queried; past versions reviewed, and so forth.

### 2.4 Ruleset Orchestration

Rule definitions in rulesets have proved to be difficult to manage in any large quantity without an orchestration mechanism. Blaze Advisor provides a "ruleflow" to provide a WYSIWYG ruleset / function orchestration mechanism.

### 2.5 SRL extensions and evolution

Fair Isaac's emphasis on "decision lifecycle" means that Blaze Advisor is evolving with extensions to invoke / include predictive analytics in rule services. Extensions at the engine level will continue to be defined to accommodate new rule engine types and rule representations.

### 2.6 Standards

Fair Isaac is fully supportive of standard in the business rules domain, as the lack of standards is seen as a barrier for wider adoption of business rule technologies. Blaze Advisor supports or uses technical standards such as XML, HTML, XFORMS, CORBA, ANT, JSR-94 etc, and domain standards such as ACORD. Fair Isaac is also part of the OMG Production Rule Representation team and is also a member of BPMI.

# 3 Final Remarks

Common rule interchange for rule engines remains an interesting proposition. With RuleML being developed to directly represent the rules executed in rule engines, commercial demand for standardized rule representations will likely increase. However, we should note that rule representation is also directly related to data representation, that separation of business rules from data is required to support commercial applications, and that business rules as defined by business people are not necessarily equivalent to executable rules.