A great impedance mismatch exists between SBVR / behavioral business rules vs. DMN / decision rules. At the heart of the mismatch is the issue of state-based vs. modeler-invoked evaluation of rules. No current technology addresses this problem – a huge gap (or big opportunity!) in the marketplace. It’s time to move beyond limited-function decisioning systems to comprehensively rule-based business solutions.

The Great Rules Impedance Mismatch

by Ronald G. Ross

Modeler-Invoked Evaluation of Rules

Evaluation of rules under OMG’s Decision Model and Notation (DMN) is modeler-invoked. Evaluation of a set of designed rules occurs where and only where someone (a modeler or coder) indicates that the evaluation should occur, usually at some point in a process model.

Overtly-designated decisions points for evaluation of rules works well for the kind of business decisions where waiting as long as possible to make a decision tends to produce the best business results. Those are indeed important cases. One noteworthy variety is what I call best-fit match problems.¹

Modeler-invoked evaluation does not work well for behavioral business rules (and many definitional rules as well). For these business rules, which are legion, not catching violations as early as possible – preferably as soon as they occur – can cause snowballing errors downstream in the work being conducted and therefore extensive rework.

An additional issue is that modeler-invoked evaluation depends on the ability / wisdom / good intentions of the modeler or coder to decide where (in a business process) to invoke the decision. In the age of AI, that just doesn’t seem very smart. In any case, why should anyone at all be involved in the choice of where to evaluate rules (again, except in those select cases where it can provide better business outcomes)!

State-Based Evaluation of Rules

Visualize a soccer game. Behavioral rules (no tackling from behind, offside, etc.) are basic to the game. Formal soccer games always feature a referee, watching the game close-up but not actually part of the play. When this ‘watcher’ sees a violation, s/he stops the on-going activity, resets the play, and issues a warning or penalty. If you left the monitoring of rules to the players, just imagine the chaos! (Yes, the referee presumably does have some sort of decision process or

mechanism in his/her head to decide when a violation has occurred, but that abstracts the basic issue of the behavioral rule itself being violated several steps too far.)

In state-based evaluation of rules, current states of affairs are captured and monitored in real-time by a ‘watcher’ external to processes. This ‘watcher’ automatically intervenes whenever a violation of a rule occurs. The watcher resets the activity – i.e., the state of affairs is corrected if possible, some sanction(s) are perhaps issued, messages are possibly sent, procedures may be invoked, etc.² By this means errors are corrected automatically and ASAP, so they don’t compound themselves downstream.

No modeler or coder has to indicate when state-based evaluation of rules is to occur – the external watcher is completely automatic. State of affairs reigns supreme. What a boost to productivity! It also makes evaluation of rules very smart. What do I mean by watcher and state of affairs? The watcher is some kind of software platform – possibly a rules engine; in an SBVR-style approach³ the state of affairs would be a factbase.⁴

Another reason state-based evaluation is so important is that the great majority of behavioral business rules can be violated at multiple events. Consider the simplest example you could possibly imagine:

   Behavioral Business Rule: An employee must have a name.

The obvious event where this rule could be violated is upon first learning an employee exists. According to the rule, he/she must have a name from the onset. (You might say some decision is involved at that point, but it’s not really a natural or basic one. It’s certainly not a rich business decision.)

Far less obvious is the fact that if anyone tries to remove or discontinue the name of an existing employee, that too produces a violation of the rule. Catching violations at this second event is just as important as at create-time in maintaining valid state of affairs. What ‘decision’ does this second event correspond to? Certainly not anything remotely obvious. Even if there were one, however, why depend on a modeler or coder to ensure that the ‘decision’ is invoked? That’s definitely not smart.

This example is by no means rare or exceptional. In fact, the large majority of rules from laws, regulations, contracts, agreements, service level commitments, MOUs, business policies, etc., follow the very same pattern. Here are two additional somewhat more involved examples⁵.

   Behavioral Business Rule: A student with a failing grade must not be included in a sports team. What happens when a student already on a sports team (who presumably didn’t

---

URL: http://www.brcommunity.com/a2013/b688.html
³ For explanation and discussion of SBVR, refer to the SBVR Insider section under Standards on www.BRCommunity.com.
⁴ USoft implemented such an approach over two decades ago using SQL databases.
have had any failing grade to begin with), now receives a failing grade in the current semester?

Behavioral Business Rule: A customer must be assigned to an agent if the customer has placed an order. What happens when an agent currently assigned to a customer retires or leaves the company?

I call the events where a behavioral business rule can be violated flash points. Flash points are inevitable when you treat behavioral business rules as a first-class citizen.

Note that the various flash points for any given rule can occur in different process models, procedures, use cases, etc. – or at various points in random (unmodeled) business activity. Who or what is responsible for ensuring consistent results across all the flash points of each rule? Is it really any mystery why business processes often give such inconsistent results on things you’d think are basic? Or that data quality only seems to get worse and worse?

**Conclusion**

In modeler-invoked evaluation of rules the burden of validating states of affairs for behavioral business rules is shouldered by modelers or coders. That’s simply not smart. It’s also a major factor in why business software remains so complex and brittle. The solution requires a paradigm shift – adding state-based evaluation as equal partner in rule platforms. For the record, it’s not a question of one or the other – you need both for truly smart systems.