JT on EDM

James Taylor on Enterprise Decision Management

16th April 2009

First Look - IDIOM Decision Manager

Posted by James Taylor
Categories: Business Rules, Decision Management, Product News

I got an update from the folks at IDIOM recently. The founders say they got started with data modeling in the early 80s and realized this could not deliver model-driven development because the whole process thing did not work. By the 90s they had found an approach that worked as a model-driven approach but the JVM/Internet/XML storm knocked a lot of vendor-specific model-driven approaches out. Idiom grew from this experience - model-driven for sure but without the need to bet the bank on a vendor-specific approach. Idiom has a focus on decision automation with a strong data element. They got a couple of early insurance customers including New Zealand's largest insurer (they are NZ based) and Allianz in Australia, and have been growing ever since. Their focus in the last few years has been on OEM relationships - software companies who are building decision-centric applications. This is a somewhat slow but very viable approach. Next up is going to be some focus around delivering model generated source code through SaaS

They find, as I do, that doing the decisioning first can bound and constrain (in a good way) the rest of the application. The feedback from their implementers and partners is that you can and should do the decisioning definition first both because it drives the rest of the application and because it lasts longer than the rest of the system.

IDIOM's product supports both Java and .Net and is a really pure Decision Service implementation - XML structures are passed to the decision server which executes decision models and returns the updated XML. A classic stateless decision service where the knowledge embedded in the decision service is your core IP. Their implementation is always stateless and can be wrapped so it can be accessed using SOA, queue services, as a DLL, from workflow engines, enterprise service buses etc. IDIOM generates and delivers to the user source code (Java, C# and compact framework) so users can also include the compiled code in the address space of the calling application if they need to.

While some decision services execute only against the data passed in, the IDIOM decision service can call out to grab additional data when a decision requires it. This was added recently - IDIOM has always had a concept of Tables which are internal and could be loaded at run time and now there is a connector for these Tables so they can be linked to a live database. They try to stay very agnostic with respect to linking the decision service to the execution environment - a philosophical approach I strongly support - but it is pretty limiting for decision services if they

can’t go retrieve additional data. Even though this represents a coupling to the execution environment they reluctantly added this.

The product stores all its definitions in a repository, typically one for an enterprise or Line of Business. One of their largest rule bases is NTI - a niche insurer in Australia who took decisioning very seriously right at the start. NTI has a team drawn from business groups that own the rules and that worked in parallel with a technical development team on a 100 person year project. This repository manages some thousands of decisions across the entire business and generates ~1.8millions lines of Java code as 50 distinct decision models.

An executable or a Decision Model is typically something like the entire underwriting decision - do I have enough data, is it valid, interim calculations, do I accept the risk, at what price and with what terms and conditions, and what needs to happen next. The decision models process schemas as fact models. The tool renders the schema (w3c) as a tree and cuts it down to just the stuff that makes sense for building decisions. Generally the schema contains both the supplied data and the outbound decision results etc. Delegation of control to the decision designer is managed by making only some of the elements of the schema available for manipulation by the Decision Service (these are the values that can be set by the decisions), and the schemas also get annotated with the decisions that can then be defined. In IDIOM’s view the schema is a description of the problem domain at rest while the decision model is a description of how it changes between the resting states.

Decision Model - contains atomic decisions and collections of decisions and is represented as a tree. Can break down a decision into elements - defining how the decision that is required breaks down into its component decision elements. Decisions are linked to the element of the schema that will be updated by the decision. This decomposition defines the execution sequence - there is no support for inferencing. “Below” the decision model there are formulae. Each formula is a sophisticated expression and a very nice expression builder is provided. Hundreds of validation tests are applied at deployment time as the code is generated.

The product generates source code for deployment (so there is no licensed server as such) and supports live re-deployment and updates as a good decisioning platform should. IDIOM regards testing as critically important and makes it part of the analysis/design process - a good designer tests their own decision design. Iterative, step by step testing in the builder is supported, as well as regression testing both in the builder, and through testing harnesses that wrap the run-time. The tool supports logging of the runtime execution of decisions (creating before and after images of the schema documents for instance) so users can manage regression testing and testing for expected values on future deployments.

The tool supports versioning of all elements with effective dating at the decision, formula, and table data levels. It also has a nice PDF generator that documents all or part of the repository in a structured ‘logical english’ format suitable for a business audience.. A future version will allow users to plug-in external function libraries for use in expressions/decision models.

IDIOM also have a forms builder that allows decision models to manipulate the UI, underlying behavior and the forms themselves. Forms are built by mapping the schema to a session schema

and then to a form that uses a CSS to control look and feel. For each field or form section the user can execute a set of UI rules against the session schema - the same approach as executing rules against the main schema but with access to specific meta data elements related to the UI. Both the underlying business and the session rules are run when you link an element to decision models. This approach allows users to build ‘reflexive questioning’ for real time, inline response. The user can use the decision models to define messages, change the layout, add and remove fields etc.

IDIOM is not a “classic” rules management system and has some unique features. At its heart it allows you to build Decision Services, however, and that’s what matters.