The Logic of Insurance: an Ontology-Centric Pricing Application

Ludovic Langevine[1] and Paul Bone[2]

[1] Mission Critical IT, Brussels, Belgium
llg@missioncriticalit.com
[2] Mission Critical Australia, Melbourne, Australia
pbo@missioncriticalit.com

Abstract. We have developed a new pricing and scoring application for a large insurance company's car insurance products. The business logic of this application is entirely expressed in an OWL ontology and SWRL rules. The ontology and rules are not merely documentation or a specification of the application, they form the business logic of the application. In other words the application is *ontology-centric.* The application efficiently generates quotes, calculates the pricing of contracts and determines the applicability of a product based on an acceptance policy.
We report on our experience developing and deploying this system, as well as integrating it with legacy systems and new applications using mainstream technologies. We will show the significant quality and agility benefits of an ontology-centric approach that also delivers the expected performances on typical enterprise hardware.

Scope and context of the project

In an insurance company, actuaries define how products can be configured, how they are priced and at which market segments they are aimed. They document these details as reports and spreadsheets. IT people use these as the specifications of the systems to be built, creating multiple scalable systems, such as: product configurators, pricing services, scoring systems and acceptance checkers. These systems can be used through multiple channels, e.g. a call center, a Web site, insurance brokers, or partner car retailers. In this scenario, the introduction of a new insurance product or changes to existing acceptance rules or other aspects of existing products can take months and is subject to interpretation errors.

The development of Web insurance comparators and faster customer turnover represent challenges that insurance companies have to address in order to compete effectively in the market. The time-to-market for new or altered insurance projects must be as small as possible, this means the delay between the actuaries' reports and actual changes to IT applications must be reduced.

To meet this challenge, a French subsidiary of the UK-based insurer Aviva decided to seek help from an external expert consultancy - LiteHouse Advisory - who proposed them an innovative solution and architecture which allowed them to externalize the business logic of its car insurance products into an ontology. The ontology drives the behavior of an application which implements the products' definitions. This application is then used by other systems such as the customer-facing Web site, and the sales-support application used by the call center. Changes to insurance products are made by changing the ontology and rules which, after testing, are used directly by the production application. This reduces the time-to-market dramatically.

The external consultant led the selection process to implement this solution, and Mission Critical IT was chosen to support the development of the ontology, to provide the needed technical components for IT to leverage it and to assist the IT department in the integration with other systems. A key component of the application architecture is Mission Critical IT's ODASE[3] platform for ontology-centric development. The application went live on the 10th June 2014.

Functionality

The application exposes three services:

– Pricing: execute the product policy defined by the actuaries to produce quotes (determine the eligible products and their prices, apply the default selection based on questions answered by the customer) or compute the yearly premiums of existing contracts on renewal. This pricing service includes a configuration service, as it returns the set of optional insurance features a customer may chose and the default choices.
– Acceptance: depending on the available data about a customer (including incomplete data when some of the questions have not been answered yet) the application executes the accepting policy to determine whether an insurance offer can be made to the client or whether their contract can be renewed.
– Scoring: depending on the data about a new offer or a change of an existing contract, this service computes a statistical score which is used by the backend to decide whether the customer should be asked for additional documentary evidence (e.g. a statement from the previous insurer in the case of a new customer).

Ontology-centric development and integration

The concepts and properties of the domain are expressed in OWL. Examples of concepts are *Vehicle*, *Person*, *Guarantee* and *Premium*. The ontology contains 208 concepts and 598 properties. SWRL rules are used with the axioms to provide reasoning. Two examples of rules are: a single rule defines the premium including taxes as the sum of the premium without taxes and the applicable taxes; a set of rules define which optional guarantees can be offered to the customer for a given quote and which of these should be selected by default.

[3] `http://www.missioncriticalit.com/odase.html`

Three groups contributed to the project: *actuaries*, acting as subject matter experts (SMEs), two *Cobol developers*[4] and *ontologists* from Mission Critical IT. It is worth noting that the ontological approach was very well received by the developers as they wanted to upgrade their skills and leverage their deep knowledge of the business. The developers created the ontology with coaching from the ontologists. The ontology was created based on interviews with the SMEs and then refined using an iterative Scrum agile methodology with three-week sprints.

The ODASE platform includes a tool that generates a Java API from an ontology. It creates a Java class for each OWL concept and Java getters and setters for each OWL property. OWL subclass axioms are translated into up- and down-casting methods. This ontology API interacts with the RDF stores and the OWL/SWRL reasoner which enforces the business logic expressed in the ontology at runtime. This API provides a type-safe and familiar interface to the business logic for Java developers.

The application is pure Java and runs as a plain servlet in a Tomcat container. The ontology files (OWL and RDF) and the ontology API are packaged within the WAR artifact with the runtime libraries of the ODASE platform. Demo applications were also developed using the API, these were used by the actuaries to test and refine the ontology during its development.

The actuaries often work with statistical and spreadsheet software. Specific import tools have been developed to extract data from Microsoft Excel spreadsheet and translate them into RDF instances. More than 5,000 instances have been generated this way to provide the ontology with all the required parameters.

The servlet listens to requests with two protocols: HTTP REST requests for modern client applications (the company Web site and external Web-based insurance comparators) and MQ Series requests from applications running on the legacy AS/400 system (backoffice and sales-support application for the call center). The amount of hand-written code specific to these services is very small: less than 2,000 lines of Java code.

The REST requests use RDF/XML formatted data, which is already in the business model defined by the ontology. However the MQ Series requests use Cobol data structures in the model of the legacy applications. A semantic integration with the AS/400 legacy system has been developed through the generation of intermediation ontologies from the description of these Cobol data structures.

Performances

The use of runtime reasoners and interpreted rules for such complex computations and inference tasks was perceived as a major risk for this project. In our opinion, this skepticism is mostly due to the lack of visibility of the Semantic Web technologies. To mitigate this risk, performance tests were conducted

[4] The Cobol developers were selected for their abstract reasoning and strong understanding of the business.

throughout the project, under the supervision of the client's IT department. Finally two load-balanced Intel Xeon systems proved enough to cope with the load during the busiest business hours.

ODASE parallelization mechanisms have been used to leverage the multiple cores of the servers. The query-based nature of its reasoners allows good response-times: the most complex request is answered in less than 400 ms during peak times. Quotes can be generated in 250 ms. The application serves about 30,000 requests per day.

Benefits

Business — IT alignment Traditionally such a business has, at least, two separate views of the business model; one maintained by the business and one (or more) maintained for the sake of IT applications. This requires extra effort and problems will occur when the models are out-of-sync. Ontology-centric development allows a single model to be shared by business and IT.

Agility IT and business people now share the ontology and understand the same concepts. The delivery of a new pricing or a change to the acceptance rules can be achieved in a few hours. The delivery time of new products has also been reduced. Changes in the ontology are reflected in the generated Java API when necessary and IT can modify applications as necessary, which is much less effort than implementing new products directly in Java. With ontology-centric development changes like these have a dramatically reduced time-to-market. As a matter of fact in the month since deployment, two change requests have already been successfully applied.

Quality In the three months following deployment approximately 200 defects have been identified in the whole project. Only two of these were in the ontology (approx 1%). The majority of defects were in the AS/400 screens and integration layers, followed by the Java Web application. Using ontology-centric development dramatically reduces the risk of bugs occurring in or effecting the business logic, where subtle bugs might be very expensive (for example, by miscalculating the price of a contract). This high level of quality is made possible thanks to the externalization of the business logic in a high-level formal language: the application is built on top of an executable and testable specification, rather than being the result of human interpretation of large and often inconsistent informal specifications. Moreover, because the business logic is implemented declaratively, any given output of the application can be explained in business terms, pinpointing the set of axioms and rules which are responsible for the result using the ODASE platform's declarative debugging tool. The maintenance burden of the ontology was so low that two of the three developers (two from IT and one Mission Critical IT ontologist) were reassigned to other teams.