dglaubman@acm.org David Glaubman www.linkedin.com/in/dglaubman

Innovation and Invention at Risk Management Solutions

A call to arms

Hemant Shah, founder and CEO of RMS, addressed the company's software groups as we sat crowded around the cafeteria. 2006 would be another good year, our catastrophe models lead the industry, and most importantly, we are trusted by our customers. However, he said, "*we are unable to metabolize our clients' appetite*" for faster turnaround, more relevant analytics and new lines of business. I have never forgotten those words, and they have inspired the trail of innovation I sketch below.

A new model for exposure data

Our EDM (Exposure Data Model) was the insurance industry standard for describing locations subject to damage by catastrophes such as earthquakes or hurricane, but it was badly in need of an overhaul. Data modeling decisions that made sense in the early '90s, when RMS modeled 2 perils and simple insurance terms, did not fit the complex coverages and multiple perils our customers underwrote. Schema changes were brittle, expensive and error-prone and our products were tightly coupled to the physical schema.

Informed by tenets of domain driven design, I circulated a four page pamphlet setting out the issues I saw and describing a frame for addressing them. I convinced Jeff Kilbreth, the new VP of product management, to sponsor me in developing a new data model. I spent the next 6 weeks developing a conceptual data model in which entities corresponded to domains (structural characteristics, geography, insurance and the like) instead of jumbled together because software once acted on them at the same time. We hired Sudha Raghavan as a database programmer to help build out the logical data model, and after many review sessions we obtained sign off from internal and external stakeholders.

Implementation was slow and incomplete - we now had the ability to model real world complexity in high fidelity, but our aging software products did not know how to make use of this new power and flexibility. Full implementation of the new EDM would have to await our next generation of models, which would support ground up simulation on an entirely new codebase.

Insurance as code, not data

When Philippe Stephan became our CTO in 2009, I sent him a document proposing "a strengthening and simplification of the insurance logical model" with numerous worked examples that our customers could not represent in our current product. He called me a couple of days later to say that he found my document very interesting, but felt it described "code, not data." I said that I could do that, and found myself reporting to the CTO and trying to figure out how to represent contracts and compute their losses in a simulation, at scale.

The challenge I faced in building a Contract Definition Language (CDL) was three-fold. First, I needed to broaden my understanding of the variety and subtleties of insurance terms and conditions throughout the world. This was particularly difficult since both RMSers and our customers tended to think in terms of the existing system and how to work around it, rather than the actual wordings crafted by lawyers, analyzed by actuaries and evaluated by adjustors. Second, I needed to express these terms and conditions in a form understandable by both humans and machines. Third, I needed to transform this representation into compact, efficient code.

The first challenge is on-going a decade later, but product manager Cody Stumpo and others quickly helped me achieve the necessary minimum of domain understanding. As for the grammar: the primary design tenet was that it must be simple to express simple contracts, and possible to express arbitrarily complicated ones. The language is designed to minimize cognitive load: a 'deductible' in CDL looks and behaves the way an insurance professional expects. The most complicated aspect of a contract is the interaction of its clauses: rather than task the user with specifying this, the compiler figures out the order in which interacting terms must be evaluated.

Within a few months, I completed a working proof of concept in F#, using its integrated parsing tools. The generated code was not very efficient, but the success of the PoC justified hiring a compiler engineer, Raj Bains. In short order, the two of us built the next version, which convincingly beat a more conventional implementation with respect to speed, size and setup time. The first production version for use in the simulation engine was built by a small team led by Raj in 2012, and its descendants are used in RMS products today.

Advantages of CDL include its expressive power, transparency and ease of use. Compared to a monolithic financial engine, it is future proof in two senses: users can craft new types of insurance (just as programmers can write almost anything in C), and when new capabilities are needed, the language can be extended in a backwards compatible manner. In contrast, supporting new insurance features in the legacy system required changes to a large, opaque financial engine, schema changes and, often, database migrations, all of which led to delay and frustration for our customers.

Reinsurance and beyond

Reinsurance is insurance on insurance. The difficulty in accurately modeling reinsurance is expressing and calculating the precise subject of the reinsurance contract, since it might cover only certain types of loss to some collection of primary contracts. Furthermore, it might be written to respond to whatever loss is remaining after other reinsurance contracts pay.

After several unsatisfying attempts to express these relationships in CDL itself, I hit on the idea of introducing a new entity, *position*, which could serve as the source of claims on a contract, and the sink of a contract's payout. That is, if the contract was a function *f, f(claims) = payout*. A position has a name, and is composed of operations on other positions, forming a directed acyclic graph. The leaves of the graph can be initialized with, for example, the output of a catastrophe model, and any position can serve as the subject of a CDL contract. Thus a variety of objects - insurance policies, reinsurance treaties, portfolios, inuring relationships, filters and groups - reduced to two: contracts and positions.

The *aha* moment came a few weeks later - consider a contract as the position whose value is given by the payout function of the contract evaluated against its subject position. Now there was just one entity, *position*, which is closed under the operations *group, invert, scale, filter* and *payout(contract).* Over the next couple of days, the Structure Definition Language was born, enabling customers to accurately model and analyze their entire book of business, gross and net of any/all reinsurance. (A good thing).

Applications

The Risk Data Open Standard (RDOS) was released to the public in January 2020. The RDOS Steering Committee includes some of the earth's largest insurers and reinsurers.

The RDOS Exposure schema is the direct descendant of the new EDM logical data model.

RDOS includes the Contract Definition Language (CDL) and Structure Definition Language (SDL).

CDL/SDL are key enabling technologies for ExposureIQ (2019) and TreatyIQ (2020).