Article from

The Modeling Platform

November 2019

Issue 10

Article from

The Modeling Platform

November 2019

Issue 10

The Importance of Centralization of Actuarial Modeling Functions, Part 1: Focus on Modularization and Reuse

By Bryon Robidoux

This article is a response to the April 2019 issue of *The Modeling Platform* that contained discussions on the pros and cons of centralizing and decentralization the modeling departments.[1] This article will look at these activities from a software engineering perspective. It will give clear insights as to why large corporations, such as Prudential and New York Life, would find it critical to centralize their modeling departments and explain how to get the most out of centralization.

Why would Prudential and New York Life want to spend their time, money and effort to centralize their modeling departments? As an insurance organization grows, especially to a very large size, it becomes more likely that different products will be splintered into silos to better focus on the product lines. This makes sense from a product management standpoint, but it starts to create problems with modeling.

To see why, let's create a fictitious insurance corporation called ZZZ. When ZZZ opened its doors 10 years ago, it offered fixed indexed annuity (FIA), variable annuity (VA) and universal life (UL) products and put them in independent branches of the organization. Today, they decide to offer indexed universal life (IUL) and variable universal life (VUL) products and have the UL team model them. The options and option budget calculations are going to be similar between the FIA and UL plans. Similarly, the VUL is going to offer the same set of mutual funds, especially volatility-controlled funds, as the VA line so their account-value behavior is similar. What options are available to ZZZ as they try to model their new product offerings?

There are three ways to move forward in this situation:

1. Start from scratch
2. Copy existing code
3. Build reusable libraries that can be shared

OPTION 1: STARTING FROM SCRATCH

Starting from scratch is hardly ever a viable option, even if there is a strong desire to do so. This can be a very expensive and risky endeavor because, no matter the good intentions of starting with a clean slate, there is a good chance that reality will set in and the new model will start having all the same blemishes as all the other models in the organization.

In behavioral economics, it has been shown time and time again that there is a bias to underestimate the time and effort needed to accomplish a project even with experienced professionals. [2]

OPTION 2: COPYING EXISTING CODE

Copying actually has two paths it can follow. The team can either copy and periodically synchronize with the original, or they can copy and modify by throwing out calculations or adding new calculations.

Synchronizing is very challenging and no small task due to continual coordination and reconciliation. Copying and then modifying is the most likely option due to stakeholders wanting to work independently to manage their own priorities. Actuaries are motivated to copy because it appears to be the cheapest and easiest to implement. Copying and decentralization are really one and the same activity.

Copying appears to be cheap because of the divide-and-conquer fallacy. This is the idea that a model can be copied to better divide and conquer the workload so that deliverables can be parallelized, finished independently and faster. Copying actually increases the workload because each copy takes on a life of its own. The models will have to be developed, tested, run, audited, controlled and managed separately. The model will start diverging due to inconsistencies, at worst, that should not exist or, at best, are annoying. All the differences manifest themselves with various models giving different results for what is supposed to be similar or identical behavior.

With the insurance companies offering ever more complex products and dealing with ever more complex regulation, senior managers need to eliminate as much noise as possible. They should not have to be thinking which model produced the results and trying to mentally juggle the differences.

Even if the calculations remain identical, the more the original and the copied model diverge, the more infrastructure and adaptions are required to release the model into production. If the only requirement for a model to go to production is that it spits

out accurate numbers, then the infrastructure needed to feed the model will be overlooked. This will result in manual, tedious and error-prone processes. The worst thing that can happen is an actuary creates one, two or 10 spreadsheets to bridge the gaps between model inputs or results.

Further, the design of the model is directly related to the service level that can be provided to stakeholders. The stakeholders will want to do what-if and other analysis outside of the normal production runs. This additional analysis will likely take forever, requiring an army of people, or the results will be unreliable if the manual adaptation gets out of hand.

Decentralization: An Expensive Choice

When starting from scratch or copying, there are multiple groups effectively maintaining the same functionality and solving similar problems. In the end, what had been done to reduce timelines has just increased work, headcount or both. Once a model goes to production, the process dictates the structure of the company and not the other way around.

With copying, I often relate the project to a tractor pull. It starts fast, but eventually the sled weight will bury the tractor in the mud. The fast start will lead to short-term decisions that don't scale. As the project grows, it will start being very complex and error prone. Eventually, new feature delivery will grind to a halt and stakeholders will find new sources for their results. Copying is therefore a short-term gain for a very expensive long-term loss, which makes the pros of decentralization a mirage. Just like in finance, there is no such thing as a free lunch. The piper wants his payment and his wealth derives from immediate gratification from making copies!

To achieve a smaller, better, faster and cheaper modeling operation, centralization is the correct move. To really get the most out of centralization, the goal must be to focus on modularization and data and logic reuse by using software engineering practices.

Applying Software Engineering Concepts

In actuarial modeling departments, there seems to be a mental separation between software engineering and model development. In reality, there is no difference. To demonstrate the point, here is the mapping of roles from the modeling department to the software development department:

- Actuarial modeler ⟶ developer
- Model steward ⟶ application technology lead

- Model governance ⟶ development operations
- Stakeholder sign-off ⟶ user acceptance testing
- Setting up runs and their switches ⟶ configuration management

The list is by no means exhaustive. The pain we feel in the actuarial modeling department is directly related to the difference we believe there is between the activities. The software engineering profession already has a lot of our modeling challenges solved.

While possibly provocative, I think the modeling department should report to the chief technology officer rather than the chief actuary. The modeling department is more an extension of the modeling platform than the insurance organization itself. Actuarial modelers are just customizing the platform so the business can run the calculations it needs.

To really get the most out of centralization, the goal must be to focus on modularization and data and logic reuse by using software engineering practices.

A sound modeling department needs to be a wide spectrum of technology and actuarial skills all working together. New regulations have sophisticated requirements. Senior managers need to have the flexibility to look at numbers and do whatever analysis is required to make better and faster decisions. This can't be done with cumbersome and clunky models with a million manual processes.

Software engineering is the art of abstracting sets of related concepts so they can be dealt with in a uniform manner. Good software engineering involves the following key aspects:

- One should focus on modularization.[3]
- Each of the units should be tested to ensure they work properly before being merged into the main production branch.
- Units should be simple reusable components.
- Each component is divided into abstractions that semantically map to the problem at hand.
- One should develop to pattern, not to the specific problem.

A project backed with good software engineering practices will start slow like a large rocket and then accelerate once a critical amount of functionality is built. The idea to remember is that nothing is ever new. It is usually just a slight extension on what already exists. With a little patience, focusing on work product reuse will pay large dividends and allow the modeling department to do more with less and easily adapt to new changes.

Modern project management, such as agile development, is based on the axiom that good engineering practices are modularized. This is why there are one- or two-week sprints to build small units, get buy in, make necessary adjustments and move on to the next small units of work. This very iterative process leads to a much better product with faster feature delivery.

OPTION 3: BUILDING REUSABLE LIBRARIES

Now back to the last option for company ZZZ, which is to build reusable libraries for common calculations. This means that the group deemed to be the subject matter expert builds a library and shares it with the rest of the corporation so that everyone benefits from the expertise. Rather than organizing the insurance company in terms of product lines (which can have many redundancies), the company can be organized in terms of common services. This service-oriented corporate structure would strive to only do a task once, have only one source of data and make code easily extensible to avoid redundant logic. For example, there would be a team responsible for providing the option and option budget calculations for both FIA and IUL products. This promotes consistency and greatly speeds up the rate at which enhancements can be added to models.

Code modularity and work-product reuse are not easy to implement. They require coordinating with other groups and living within their response times. There is also an entire discipline of software engineering above and beyond actuarial science to learn. Our modeling platforms try to shield us from software engineering, but this effort is futile. Trying to shield modelers from software engineering is equivalent to playing the whack-a-mole game. By hitting one mole, it will cause another to pop up somewhere else.

If software engineering, modularization and work product reuse are so important, then why are they not common practice in the actuarial modeling department? By the design of our modeling platforms, actuaries are strongly encouraged or mandated to put all their work products directly into the model. Once in the model, it is locked away from other projects that might need that same logic. This is the monolithic-system problem.

The monolithic-system problem forces modelers to copy logic, which causes issues mentioned earlier in this article. It also creates challenges with unit testing and project management because it makes unitizing the work very difficult. (An example of a unit of work might be a formula table in Axis or an extended formula in Prophet.)

In our models, many units are merged together, there are tons of switches that have to be set correctly, and many components have to be put into place in order to get the model to run. These components have to be set up just perfectly to make sure that all the execution paths are exercised during testing. This requires actuaries to put changes into the model and then sort out all the issues. This forces the testing team to run the entire model to find problems. Depending on the size or number of run(s), this could be a multiple day turnaround to analyze changes, determine the sources, explain to the developers any issues, have developers fix and/or dispute the perceived problems, and get the model back for the next round of tests. This leads to project management issues, especially in agile project management. Depending on the complexity, it is difficult to build the units, assemble then together and test within a typical one- or two-week sprint. The sprint can be lengthened, but this cuts down tremendously on the agility of the project.

This is backward to what should be done. With proper unit testing, the actuary would find all the problems *before* the model is fully built. The tests should take seconds and not days. The developers should be able to run independent of the testing team so they can get very fast turnaround. Then all that would be required is a little bit of integration testing to make sure all the units play together nicely with the model. This is much less work than the current practice.

CONCLUSION

The expense of decentralized models stems from three main problems: the perceived difference from actuarial modeling and software engineering, the divide-and-conquer fallacy and the monolithic-system issue. The act of copying the model to try to divide and conquer the workload actually creates more work. This cheap act of copying creates a massively expensive modeling department and does not scale in the long haul. Decentralization and copying are the antithesis of sound modeling and good software engineering practices.

By not accepting that actuarial modeling and software engineering are the same jobs, the actuarial profession is struggling with problems the software engineering profession have already solved. If we harness software developers' expertise and their tools, we can reduce many of the challenges we face.

Centralization of modeling is a good start, but it isn't the end game. To make the centralization really pay off, modeling departments need to go one step further and focus on modularization so that logic and work products can be reused as much as possible. This will speed up development throughput and testing. It will make it easier to audit, document and maintain the model. If done correctly, this will reduce unnecessary head count and make the models smaller, better, faster and cheaper to operate and maintain. ■

Bryon Robidoux, FSA, CERA, is actuary ALM, Reinsurance Group of America. He can be reached at *bryon.robidoux@rgare.com.*

ENDNOTES

1 Kwan, Daphne. 2019. Journey to Centralizing Modeling Function. *The Modeling Platform*, no. 9. *https://sections.soa.org/display_article.php?id=3361692&view=582866*; Kerr, Dean, Josh Chee, and Jay Boychuk. 2019. Centralizing Model Development: Is It Worth It? *The Modeling Platform*, no. 9. *https://sections.soa.org/display_article.php?id=3361693&view=582866.*

2 Kahneman, Daniel. 2011. *Thinking, Fast and Slow*. New York: Farrar, Straus and Giroux.

3 "Broadly speaking, modularity is the degree to which a system's components may be separated and recombined, often with the benefit of flexibility and variety in use." *Wikipedia*, s.v. "Modularity," accessed Aug. 29, 2019, *https://en.wikipedia.org/wiki/Modularity#cite_note-MWModular-1.*