LET'S NOT REINVENT THE WHEEL

By Brenda Perras

Actuarial modelers need to engage and leverage IT expertise in the design, maintenance, and control of actuarial models.

With the movement to principle-based reserve (PBR) models in the United States, there is increased emphasis on actuarial model risk and model governance. Not surprisingly, model risk and model governance are fast becoming hot topics in the actuarial industry as companies search for solutions to increase transparency and manage model risk for some of their key decision-making tools. Models are experiencing increased scrutiny by regulators, auditors and management.

Actuarial model systems now rival or surpass the largest pieces of commercial software in terms of complexity: vast, heterogeneous data sets are manipulated by multiple pieces of software code, both custom and off-the-shelf, and perform highly specialized calculations. Unfortunately, actuarial system design and development methodologies have not evolved at the same pace as the models themselves. According to the 2012 SOA Actuarial Modeling Controls survey, life insurance firms do not rate themselves highly when self-reporting on most aspects of their model governance frameworks.

While there has been an increase in calls to adopt similar best practices to those already widely employed throughout the information technology (IT) field, only a few industry leaders are actually putting these practices in place. The simple answer for the rest of the industry: Rather than reinventing the wheel, reach out to the IT discipline to learn approaches developed over the past half century, and apply them in developing a rigorous systems development framework for the construction and maintenance of actuarial models.

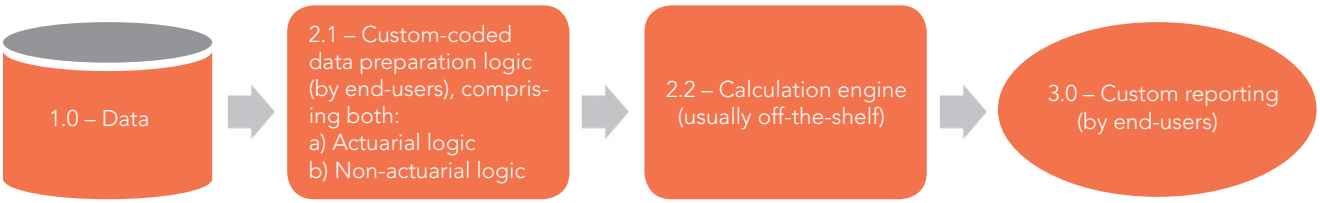
Actuarial Models Have Become IT Systems

Fifteen years ago, available computing power typically only allowed annual projections of quinquennial ages on large blocks of business, and only a few scenarios. Today, computing power is such that we can run thousands of monthly stochastic scenarios, rarely needing to use age groupings on

seriatim data for principle-based valuations. The result is much more complex actuarial models that have become IT systems unto themselves.

All but the simplest actuarial models can be classified as heterogeneous systems containing:

1. A **data component** (the input to the model);
2. Code to perform calculations:
   2.1. A **custom-coded component** developed by the modelers (end-users) to prepare the data for the calculation engine;
   2.2. The **calculation engine** itself. This is either an off-the-shelf component provided with the modeling software package, or end-user code that produces the model output;
3. **Reports** produced by custom code or custom external spreadsheets based on the model output.



While it is ideal to minimize end-user coding by actuarial modelers, in practice it is never 100 percent avoidable. The end-user component (2.1 above) is itself often complex in design and large in terms of code quantity, including things such as:

- Data manipulation and translation, e.g., consolidation, sampling, approximation and enrichment techniques;
- Product features and methodology coding;
- Assumptions setting.

The efforts to build and integrate an end-user data preparation piece (2.1), calculation engine (2.2), and custom reporting (3) have reached the scale of software development projects in their own right. Companies can therefore benefit from applying IT best practices to actuarial modeling (2.1a and 3) and shifting end-user coding of non-actuarial logic (2.1b) presently performed by actuarial professionals to software development professionals.

IT Best Practices Should Be Used Throughout the Actuarial Model Development Life Cycle

The past few years have seen extensive literature highlighting key issues with the current actuarial modeling governance landscape. For instance, over half of life insurance respondents to the SOA Actuarial Modeling Controls survey did not have a formal process to implement code changes, a way to detect unintentional model changes, or a formal code integration process.[2]

In addition, most life insurance companies rate their own model governance and change control practices poorly according to the SOA survey.[1]

CONTINUED ON **PAGE 18**

The opportunities for applying IT approaches to model development include minimizing effort of actuarial staff in areas where non-actuarial staff would be better employed. They also include ensuring actuarial model developers follow a prescribed software development life cycle with IT guidance—just as IT practitioners benefit from well-established best practices under the "systems development life cycle" (SDLC).[3]

A COMMON REFRAIN: "BUT THE IT DEPT. IS TOO SLOW"

A common argument made by actuaries against getting greater IT involvement is "but whenever we request something from IT, it takes months. Their principles and governance have made them too slow." The perception is that IT takes significantly longer to get things done than actuarial modelers would, and that in many situations, business decisions need to be rapidly analyzed and taken. The actuarial modeling industry's shoot-from-the-hip approach is often justified by the stance that "our work is based on judgment and approximations" or the conviction that adding process will decrease speed resulting from unfamiliarity with the benefits of a controlled systems framework.

In fact, IT principles and governance do not result in lengthy turnaround times;[4] rather, it is typically cases where accepted IT best practices have been deviated from—or a legacy of past implementations that deviated, cutting corners in the name of "saving time"—that result in slow delivery. Further, the fact that model approximations are used does not address the risk of material errors going undetected as a result of undocumented and untracked changes made to an opaque system.

IT IS NOT ENOUGH TO SELECT AN ACTUARIAL STAFF MEMBER WHO HAS YEARS OF CODING EXPERIENCE. DO THEY HAVE APPROPRIATE TRAINING AND EXPERIENCE IN FOLLOWING THE DESIRED APPROACH FOR YOUR COMPANY'S MODEL DEVELOPMENT FRAMEWORK?

One material error may simply have been offset by another in the scenario output, or not have been triggered in the particular scenario tested.

A properly designed, well-understood, well-documented, thoroughly tested, and properly maintained system will allow for rapid analysis in the form of quick configuration or input changes—at the same time mitigating the risks presently posed by models that lack transparency in their original design, testing practices and maintenance.

NO ONE-SIZE-FITS-ALL SOLUTIONS

Organizations must balance their appetite for risk and the cost of additional controls and training when selecting a modeling framework most suitable to their business needs. They should create a framework of principles and adapt them to the models and situations at hand based on the regulatory environment and company leadership's objectives. Companies have different categories of models that require varying degrees of adherence to the principles.

Ensure Staff Have Appropriate Training and Experience

This is the low-hanging fruit: The staff coding end-user components should have sufficient systems design and programming training and skills, and likewise for staff documenting the system; presently this is often not the case. The costs associated with having these tasks completed by someone without the right skill set include:

- Increased maintenance costs;
- Increased change control risk;
- Increased model risk.

Work with IT to Identify Best Practices Most Suited to the Organization and Its Models

There is no need to develop the elements of a governance policy from scratch—model stewards can work with IT subject matter experts to get a full understanding of each, and the benefits and applicability of each to the organization's situation. The key to efficiency with framework elements such as these is having adequately trained staff and management that support them. Building a model with

thorough documentation at the outset will make all future maintenance and changes to the model much easier and reduce cost in the medium and long terms.

The following is a non-exhaustive list of best practices that should be considered in the actuarial model development framework:

Design:

- Completing high-level and detailed designs before "jumping in and coding";
- Ensuring algorithmic and computational efficiency;
- Designing for modularity and "white box" transparency, avoiding "spaghetti code" and hard-coded "magic numbers." Today's models often have data and business assumptions buried in code without documentation, resulting in a "black box" that even insiders only vaguely understand.

Coding:

- Ensuring proper parametrization—allowing rapid testing of different inputs without needing to change code in multiple places;
- Following standard coding practices—function, object and variable naming conventions; proper levels of commenting;
- Holding code reviews—while the focus often tends to be on results during model reviews, having multiple sets of eyes review code can quickly identify errors in logic, design, etc., and is a critical protocol with code changes.

Testing:

Proper testing is a critical piece of a system development framework. It is presently common practice to rely on a review of the results under a single scenario to identify actuarial model errors, and this is often focused on the change from one period to the next. This is insufficient for a number of reasons. There is much to learn from the IT field's evolution over the past several decades here, including:

- Using a dedicated testing team—it is notoriously difficult for coders to uncover defects in their own code;
- Creating and following a test strategy, with a test plan and test cases;
  - Testing too narrowly tends to miss side effects, and ad hoc testing tends to be not easily repeatable or verifiable;
- Creating and using a "test suite"—a collection of test cases that can be automatically run and re-run, with an automatic process built in to review results for correctness;
- Using regression testing: ensuring changes did not have unintended side effects, "breaking" something else in the model;
  - Testing focused only on the expected changes will often miss side-effect defects introduced—for instance, an error in a valuation model on a small but growing block—this may not be caught if the focus is on reviewing results for period-to-period changes;
- Maintaining distinct test environments separate from development and production environments to ensure the right version of code is tested and promoted when appropriate.

HAVING MULTIPLE SETS OF EYES REVIEW CODE CAN QUICKLY IDENTIFY ERRORS IN LOGIC, DESIGN, ETC., AND IS A CRITICAL PROTOCOL WITH CODE CHANGES.

Change control and version control:

The need for change and version control depends on factors such as the number of modifiers or users of the model. A model created by a single developer may not need such formal controls if the developer is disciplined enough to track changes, archive each version of the model, and store the present "production" version in a specific location.

However, as soon as multiple developers are involved in a model's creation or maintenance, and always for business-critical models, change and version controls should be employed.

CONTINUED ON **PAGE 20**

"REGRESSION AND STRESS TESTING SEPARATE THE PROS FROM THE AMATEURS."

—BOB LEWIS

- Change control: ensuring modifications to the production model are strictly controlled, and are only put in production after having been deployed and validated in testing environments;
- Version control: ensuring every change is tracked and reversible—including what was changed, who made the change, and when—with the ability to revert back to prior versions of the model in case of defects being uncovered after testing.

Documentation:
Documentation extends beyond simply commenting code. Much as a consumer or industrial product comes with instructions and specifications, professionally developed models require multiple layers of documentation. If documentation is skipped or cursory, the delivery of the initial model may be "faster" but the time taken to maintain it in the future increases significantly. For example, years down the road a new team may be called on to convert a valuation model. If documentation is insufficient, the new team must first spend time figuring out "how does it work today?" (akin to archaeology), before even considering how it can be converted tomorrow. The same is true when knowledge walks out the door due to staff turnover or rotation.

- High-level documentation: "box" level description and diagrams of the model's key components and data flows;
- Process documentation: how to run the model and how to change the model;
- Detailed description of all approximations used in the model; for instance, how product features and regulatory requirements are accounted for;
- Input/output specifications: detailed descriptions of how each input data field is transformed by the end-user code to produce each output field, and the significance of each output field.

Conclusion

It is widely accepted that actuarial modeling development and governance need improvement to catch up to advances in modeling complexity and today's regulatory environment. Rather than try to concoct best practices in isolation, the most efficient approach is to leverage expertise from the IT discipline's long head start, ensure staff doing actuarial coding have appropriate levels of training and a framework to follow, and shift development of non-actuarial model portions to IT.

Tip:

These articles have additional tips and strategies that can help you get started:

1) http://www.actuaries.org.uk/research-and-resources/documents/report-actuarial-processes-and-controls-best-practice-working-party
2) http://www.louisepryor.com/wp-content/uploads/2011/09/managing.pdf

ENDNOTES

1 Kaufman, Sara V., Jeffrey R. Lortie and Jason A. Morton. 2012. Actuarial Modeling Controls: A Survey of Actuarial Modeling Controls in the Context of a Model Based Valuation Framework. December. Society of Actuaries. Retrieved March 16, 2015. *https://www.soa.org/Research/Research-Projects/Life-Insurance/Actuarial-Modeling-Control.aspx*

2 Kaufman, Sara V., and Jeffrey R. Lortie. "Session 177 Panel Discussion, Actuarial Modeling Controls for a Model-Based Valuation Framework." April 2013. Society of Actuaries. Retrieved March 16, 2015. *https://www.soa.org/Research/Research-Projects/Life-Insurance/Actuarial-Modeling-Control.aspx*

3 "Systems Development Life Cycle." Wikipedia. Wikimedia Foundation, Inc. Retrieved March 16, 2015. *http://en.wikipedia.org/wiki/Systems_development_life_cycle*

4 Selig, Gad J. 2008. Implementing IT Governance: A Practical Guide to World Class IT Management Using Current & Emerging Best Practices. GPS Group, Inc. Retrieved March 16, 2015. *http://www.atilim.edu.tr/~mrehan/ISE511-Text.pdf*

5 Lewis, Bob. 2013. 10 Old-School Principles That Still Rule. Oct. 17. InfoWorld, Inc. Retrieved March 23, 2015. *http://www.infoworld.com/article/2606417/it-management/123467-10-old-school-IT-principles-that-still-apply-today.html#slide5*

Brenda Perras is a senior manager in the Life Actuarial Practice at PricewaterhouseCoopers. She has over 15 years of actuarial modeling experience and extensive experience in model governance and controls. Brenda can be reached at *brenda.k.perras@ca.pwc.com*.