SOCIETY OF ACTUARIES
MODELING SECTION

The Modeling Platform

ISSUE 8 • NOVEMBER 2018

The Actuarial Paradigm

By Bob Crompton

Page 6

The Modeling Platform

Issue 8 • November 2018

Published biannually by the Modeling Section of the Society of Actuaries.

475 N. Martingale Road, Suite 600
Schaumburg, Ill 60173-2226
Phone: 847.706.3500 Fax: 847.706.3599
www.soa.org

This newsletter is free to section members. Current issues are available on the SOA website (www.soa.org).

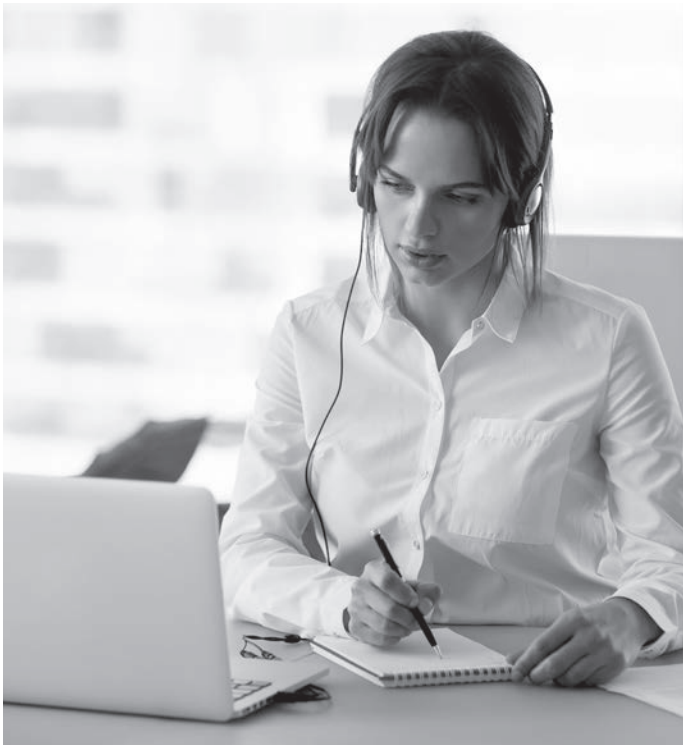
To join the section, SOA members and non-members can locate a membership form on the Modeling Section webpage at https://www.soa.org/sections/modeling/modeling-landing/.

Publication Schedule
Publication Month: April 2019
Articles Due: Jan. 24, 2019

2018 SECTION LEADERSHIP

Chairperson's Corner

By Scott Houghton



Well, the past year went by very quickly! My term as Modeling Section chair ends on Oct. 17; I'll be leaving the section leadership in the capable hands of Brenna Gardino for the next year, who will assume the role of chairperson. Daphne Kwan will succeed Brenna as vice chairperson.

On a lighter note, I've always been a bit envious of the Financial Reporting (FR) Section as they have the green jacket, a tangible item the chairperson can pass to their successor. I've been looking (but am quickly running out of time) to find something catchy for our section. I've rejected a claret jug sculpture made of recycled shredded U.S. currency (a "financial model") as both unoriginal (has golf theme similar to FR section) and not quite catchy enough.

As I look at my term in reverse, I hope that Modeling Section members agree we've brought value to your section membership. I'd like to thank our volunteers and Society of Actuaries (SOA) support staff, who make this newsletter and our webcasts, SOA meeting sessions, research, webpages and other content possible.

I hope you had a chance to catch our SOA Annual Meeting & Exhibit sessions Oct. 14–17 in Nashville, Tennessee. The section sponsored a hot breakfast and sessions on assumption setting, model sustainability and model management. If you were not able to attend, and perhaps are reading this and haven't yet completed your 2018 continuing education requirements, section members may listen to and view recordings of past webcasts that are more than 1 year old for free as a section member benefit. The free Modeling Section webcast recordings can be accessed by logging in using your SOA username and password at *engage.soa.org*. Look for the Modeling Section Community.

Looking forward, our models are changing with principle-based reserving (PBR), changes to U.S. GAAP, and International Financial Reporting Standard (IFRS) 17. We are starting the process of making big plans for 2019 section activities. If you have thoughts on meeting sessions, articles, webcast topics or research topics, please reach out to me, Brenna, or the other volunteers listed in this issue. ■

Scott Houghton, FSA, MAAA, is an actuary in Simsbury, Connecticut. He can be reached at *Scott.David.Houghton@gmail.com*.

Letter From the Editors

By Mary Pat Campbell, Phillip Schechter and Jennifer Wang

We welcome aboard our newest co-editor, Jennifer Wang. Thanks to Jennifer, we have started a new article series: Modeler Q&A. These pieces are intended to focus on technical modeling problems and pet peeves. We would like to feature real-life modeling situations that our readers encounter, and how they approach their everyday modeling work.

We invite our readers to answer the questions yourselves, or to interview a modeler you know with your own burning questions, and submit your responses to be featured in a future issue.

Here are 10 questions to get you started—feel free to add your own or to skip ones that are not pertinent to your or your interviewee's work.

1. What kind of modeling work do you do, and what software platform(s) do you use?
2. How do you plan or prepare for a modeling change?
3. When you find a bug that has an immaterial impact on results, can you let it go or not?
4. Documentation—as you go along or after you're done?
5. Do you have a modeling pet peeve?
6. What's the most frequent piece of code that you can never remember the syntax for?
7. What was the last problem you encountered that had an easier-than-expected solution?
8. What's something new you picked up recently that you'd like to share?
9. For consultants: What do you wish your clients understood about your modeling work?
10. For clients: What do you wish your consultant understood about your models?

We have a couple of initial examples in this issue: Daphe Kwan interviews David Yu of Prudential, and Uri Sobel interviews Ryan Krisac of Penn Mutual Life. Check them out! We hope to hear from you for upcoming issues. You can email any or all of the editors with your Q&A submission.

Other articles included in this issue are:

- Bob Crompton points out the actuarial paradigm of descriptive models in "The Actuarial Paradigm"—might that paradigm be changing to more causal models?
- Crew Sullivan and Igor Nikitin look at practical model-building issues in "Applying FinTech and IT Practices to Building Actuarial Models," using processes developed in other fields to inform actuarial modeling processes.
- Dodzi Attimu develops a formal framework in "What's a Model? A Framework for Describing and Managing Models," which may provide clarity on issues such as whether assumptions are model inputs or parts of a model itself.
- Linda Chow, Jeremy Levitt, Laura Donnelly Knab and Yuan Yuan continue their series on long-term care modeling, looking at model validation and other model governance processes.
- Jennifer Wang details the model-related sessions from the 2017 SOA Annual & Exhibit.

We are always looking for new authors. The submission deadline for our spring 2019 issue is January, 24. 2019. Even if you can't meet our next deadline, we'd be happy to talk with you about article ideas. Contact any or all of the co-editors if you're interested.

Thanks! ■

Mary Pat Campbell, FSA, MAAA, PRM, is vice president, insurance research, at Conning in Hartford, Connecticut. She can be reached at *marypat.campbell@gmail.com*.

Phillip Schechter, FSA, MAAA, is a vice president at Global Atlantic Financial Group. He can be reached at *phillip.schechter@gafg.com*.

Jennifer Wang, FSA, CERA, MAAA, is an actuary at Milliman. She can be reached at *jennifer.wang@milliman.com*.

SOCIETY OF ACTUARIES®

SOA Job Center

The Premier Source For Your Actuarial Career

Thinking about taking the next step in your actuarial career? Check out the SOA Job Center and get access to benefits such as a free resume review, career resources and more.

Learn more at ***Jobs.SOA.org***

The Actuarial Paradigm

By Bob Crompton

"And new Philosophy calls all in doubt, the element of fire is quite put out; the Sun is lost, and the earth, and no man's wit can well direct him where to look for it."

—John Donne

We live in an age of hype and melodramatic overstatement. Every problem is A CRISIS! Every unfortunate event is A TRAGEDY! Every new development is A PARADIGM SHIFT! And so we trivialize the world.

In this article I skip the melodrama as I discuss the actuarial paradigm and how paradigm shifts (in the nontrivial sense in contradistinction to PARADIGM SHIFTS!) might occur.

A FEW REFERENCES

The first reference is Thomas Kuhn's seminal essay on paradigms—*The Structure of Scientific Revolutions*, originally published in 1962.[1] This is the essay that originated our use of "paradigm" to mean our worldview of any particular area of study.

When reading Kuhn's work, realize that not every philosopher of science agrees with his epistemology. Nevertheless, Kuhn has some important and influential things to say about how we perceive the world.

The next reference is the article "Generalized Models of the Insurance Business (Life and/or Non-Life Insurance)" by Dr. W.S. Jewell, published in 1980.[2] The late Jewell was an internationally recognized expert in risk analysis and professor of operations research at Berkeley University. He was a recipient of the Halmstad Memorial Prize, as well as active in the International Actuarial Association.

This paper applied Kuhn's concept of paradigms to actuarial science, although his focus and intent were somewhat different from mine. Professor Jewell wanted to highlight the scope of actuarial work in various life and nonlife insurance settings, as well as how academic research was affecting practice in each of these areas.

In one way, however, Jewell was a victim of his time. In his paper, he recommends the use of APL by actuaries. The symbolic logic

nature of APL has seduced many actuaries into the same recommendation, but clearly these actuaries were never responsible for code maintenance—or even thought about code maintenance.

Finally, there is the article, "Current Actuarial Modeling Practice and Related Issues and Questions" by Dr. Angus Macdonald, published in 1997.[3] In this article, Macdonald points out that in actuarial science, our paradigm is reflected in our models. In addition, he provides his views on the hierarchy of models and how that points to the need for software that matches our intentions.

THE ACTUARIAL PARADIGM

The actuarial paradigm consists of applying probabilistic discounting of cash flows based on assumptions, often of a long-term nature, developed from observational techniques, only without recourse to a causal model in order to determine the value of future contingent benefits.

In practice, the probabilistic discounting is typically rudimentary and is sometimes dispensed with for simplicity when inclusion has an immaterial effect.

The lack of a causal model is something I seldom see mentioned in actuarial literature, perhaps because it seems so usual to us. But to those in other disciplines, this is one of the defining features of actuarial science and is often considered to be a highly flexible and desirable feature.

In "Clinical Versus Actuarial Judgment," Robyn Dawes, David Faust and Paul Meehl study the application of the actuarial paradigm in psychology. In this study, the actuarial approach to diagnosis and prediction of behavior was found to be superior to clinical judgment.[4]

In an essay from *The University of Chicago Law Review* titled "The Shaping of Chance: Actuarial Models and Criminal Profiling at the Turn of the Twenty-First Century," Bernard Harcourt makes this statement about the actuarial approach in criminal law:

> One intriguing and recurring hypothesis is that the late twentieth century ushered in a new probabilistic or actuarial paradigm. The idea is that there was a shift toward a new mode of bureaucratic management of crime involving a style of thought that emphasizes aggregation, probabilities, and risk calculation instead of individualized determination—a new probabilistic episteme modeled on an actuarial or risk analysis approach to crime management. Although this thesis captures an important aspect of the way we think about criminal law at the beginning of the twenty-first century, it is crucial to emphasize that the turn to probabilistic thinking pre-dates the twentieth century and in fact helped bring about the era of individualization that marked the early twentieth century.[5]

It should be noted that Harcourt believes the actuarial method in criminal law has been an unfavorable development.

Author David Wick recounts how physicist Imre Fényes regrets the insouciance with which actuaries simply take a statistical view of the world without bothering about the "why."[6]

SEEING OUTSIDE THE CONFINES OF THE PARADIGM IS HARD WORK

Because the paradigm guides us in how we understand the world as well as what is considered of importance and worthy of study, it is extremely difficult to comprehend the boundaries of our paradigm. No matter how many times we are admonished to think outside the box, it's just not so simple to do so.

Joseph Priestley, the discoverer of oxygen, believed it to be "dephlogisticated air." To us, this phrase is meaningless because our paradigm does not include any reference to phlogiston. Priestley, on the other hand, was never able to conceive of oxygen as being a gas in its own right because his paradigm did not include this concept.

Kuhn tells of an experiment in psychology[7] in which the subjects were shown short, controlled exposures to playing cards, then asked to identify them. Some of the cards were the wrong color—for instance, the ace of spades was red, and the four of diamonds was black.

For the unorthodox cards, recognition took considerably longer than for the orthodox cards. Most of the subjects were able to identify the unorthodox cards after extended exposure to such cards, but a few subjects could never make the mental adjustment and experienced severe distress from viewing the unorthodox cards. One subject is recorded as saying, "I can't make the suit out, whatever it is. It didn't even look like a card that time. I don't know what color it is now or whether it's a spade or a heart. I'm not even sure now what a spade looks like. My God!"

The effect of the paradigm is so pervasive that Kuhn makes the following observation:

> . . . something like a paradigm is prerequisite to perception itself. What a man sees depends both upon what he looks at and also upon what his previous visual-conceptual experience has taught him to see.

To a large extent, we see only what we expect to see. We see what our worldview (paradigm) has taught us to look for.

Kuhn believes that there is an objective reality, but that our ability to perceive it is limited by our mental constructs. Our epistemology is always smaller than reality. Furthermore, because reality is too large for random investigations, our paradigms guide our investigations only to those areas of reality mapped by our paradigm.

However, both Jewell and Macdonald make the point that regular contact and discussions with those from other fields, who have different ways of thinking about problems and issues, help us to understand where the boundaries of our own paradigm are, and what shape our paradigm might take if we have a true paradigm shift.

This sort of intellectual cross-fertilization is difficult for working-stiff actuaries. For now, such activity appears to be mainly confined to academic actuaries.

WHAT CAUSES NEW PARADIGMS TO EMERGE?

Kuhn makes it clear that new paradigms do not emerge from old paradigms, nor are they in any way a reinterpretation of data from the existing paradigms.

A new paradigm emerges when there are systematic anomalies between reality and the expectations created by the existing paradigm. Furthermore, these anomalies must be of such a degree that there is a failure of extended attempts to adjust or refine the existing paradigm to address the anomalies.

Such failures result in attempts at alternate explanations of the anomalies. Whenever a successful explanation occurs, and when enough people come to see reality in a new way, then we have a paradigm shift.

It should be clear that Kuhn sees paradigm shifts as being in some ways analogous to the operation of Hegel's historical dialectic. As such, paradigm shifts will only occur at times of intense intellectual turmoil.

HISTORICAL EXAMPLE OF A PARADIGM SHIFT

One of the early modern paradigms of chemistry was the phlogiston paradigm. In this view, phlogiston was one of the basic types of matter and was involved in burning. Burning substances released phlogiston, which was absorbed by the air. Wood, for example, was viewed as a combination of ash and phlogiston. When burned, all that was left was the ash, with phlogiston released into the air.

This explained why many materials lost weight when burned—the phlogiston component of the material had been released. However, when it became clear that some substances gained weight with burning, chemists attempted to adjust the phlogiston theory by concluding that phlogiston was lighter than air, or that it had negative weight.

> To a large extent, we see only what we expect to see. We see what our worldview (paradigm) has taught us to look for.

It wasn't until Antoine-Laurent de Lavoisier discovered that combustion requires oxygen and that oxygen has mass, that the phlogiston theory began to be replaced by the oxygen theory of burning.

We mustn't conclude that the proponents of the phlogiston theory were simpletons. Many of them were brilliant men. However, they were trapped in their paradigm, and it wasn't until anomalies in the phlogiston theory appeared that they were forced to think outside of their paradigm.

WHAT ARE NEW PARADIGMS LIKE?

A new paradigm is not cumulative—that is, it is not a new development based on what has come before. It is something entirely new such that we cannot even make analogies between the two worldviews.

Much that was important in the old paradigm is either trivial or of no interest in the new paradigm. For example, if we develop a new accounting framework, it may cause much wailing and gnashing of teeth, and it may cause significantly different financial results. But it is not a paradigm shift because we recognize it as still based on double-entry bookkeeping with debits on the left and credits on the right. We are still concerned about assets, liabilities and profits.

A paradigm shift occurs when we develop an entirely different way to think about commercial activity. Rather than a new and better accounting framework, we will have some novel way to measure and allocate changes in material welfare. We should expect such terms as "profits" and "assets" to become meaningless when such a paradigm shift occurs.

Whatever the new paradigm consists of, we can be sure that there is nothing in the existing paradigm that points toward the new paradigm. Perhaps the best we can do is to ponder what is missing from the current accounting paradigm, much like Sherlock Holmes noticing that the dog did NOT bark.

This is how all paradigm shifts work—new developments occur as improvements of what has come before. Then the ground shifts under our feet without warning, and we wake up in an entirely new world. The times of such changes are often stressful and angst-ridden since significant psychic energy is required to accustom one's self to the new world.

HOW MIGHT NEW ACTUARIAL PARADIGMS EMERGE?

Attempting to prognosticate about new paradigms is almost certainly guaranteed to be wrong. However, there is something oddly satisfying in making an attempt to see the future.

Jewell, in his paper, states that his judgment is that progress in actuarial science will be evolutionary rather than revolutionary—meaning that he expected no paradigm shifts in actuarial science in the foreseeable future.

I believe this judgment is correct as long as we consider actuarial science to be hermetic and unaffected by emerging technology or by other professional and academic disciplines. Once we admit the effects of emerging technology or developments in financial economics, then the possibility of a paradigm shift in actuarial science appears much more possible.

The possibilities that I consider are based on potential changes in technology as well as possible developments in theoretical constructs that will affect insurance and risk transfer in general.

The first possibility is that our paradigm will change because some replacement for money is developed. Perhaps biometric tracking will improve to the point where a person's cumulative productive activity net of economically dissipative activity can

be used to determine his share of available material welfare. In such a case, the term "cash flow" will be meaningless. It is difficult to visualize what form of benefits will be provided by insurance in such a case, or even what form insurance will take in this scenario.

Just as it was discovered that money had more functions than merely a medium of exchange, when we reach an abstraction of money, it seems likely that we will discover that this abstraction has more functions than money has. Such an abstraction is likely to open up large new fields of economic endeavor. It is clear that if insurance remains a viable and needed activity, the actuarial paradigm will undergo drastic changes to meet this eventuality.

A second possibility is that our understanding and conception of mortality, morbidity, accident, disaster and other insured events reach a point where we can incorporate causal models into our paradigm. The effects of such a change would reduce the statistical elements of actuarial science, but whether the reduction is a little or a lot will depend on the nature of the causal models. Perhaps these models will be generated by some form of artificial intelligence, in which case perhaps the statistical elements will also be generated by artificial intelligence, leaving actuaries with only a monitoring and oversight role.

A third possibility is that biometrics and biometric analysis advance to the point where the cost of risk can be assessed atomistically—that is, actuarial values are determined for each basic unit of risk based on such unit's individual characteristics rather than based on statistics developed from group averages.

This possibility seems to me to be the most likely since much of what we today call artificial intelligence can be termed correlation engines. These programs are typically weak at imputing cause and effect, but work well for finding hidden or unnoticed correlations.

CONCLUSION

Paradigm shifts do not occur often, but when they occur, they are accompanied by intellectual turmoil and result in a change in our worldview. When such a shift occurs, much of what we were concerned about in the old paradigm will become unimportant or meaningless. Likewise, the things that are important in the new paradigm are things that were either unimportant or were not noticed under the old paradigm.

Because of advances in technology and changes in areas such as financial economics, actuarial science might experience a paradigm shift in the foreseeable future. ■

Bob Crompton, FSA, MAAA, is a vice president of Actuarial Resources Corporation of Georgia, located in Alpharetta, Georgia. He can be reached at *bob.crompton@arcga.com*.

ENDNOTES

1 Kuhn, Thomas S. 1996. *The Structure of Scientific Revolutions,* 3rd ed. Chicago: University of Chicago Press.

2 Jewell, W.S. 1980. Generalized Models of the Insurance Business (Life and/or Non-Life Insurance). *Transactions of the 21st International Congress of Actuaries, Zurich and Lausanne* S, 87–141.

3 Macdonald, Angus S. 1997. Current Actuarial Modeling Practice and Related Issues and Questions. *North American Actuarial Journal* 1, no. 3:24–37.

4 Dawes, Robyn, David Faust, and Paul Meehl. 1989. Clinical Versus Actuarial Judgment. *Science* 243:1668–1674.

5 Bernard E. Harcourt. 2003. The Shaping of Chance: Actuarial Models and Criminal Profiling at the Turn of the Twenty-First Century. *The University of Chicago Law Review* 70, no. 1:105–128.

6 Wick, David. 1995. *The Infamous Boundary—Seven Decades of Controversy in Quantum Physics*, 78. Boston: Birkhäuser.

7 *Supra*, note 1, pp. 62–64.

Applying FinTech and IT Practices to Building Actuarial Models

By Igor Nikitin and M. Crew Sullivan

My name is Igor, and I have held a variety of business and technical roles as an actuary. An acquaintance approached me with an idea of starting a FinTech company. Impressed with his enthusiasm and knowledge, coupled with my own curiosity and desire to obtain experience in lauching a startup, I decided to join the company as a part-time technology co-founder. At the time, I felt confident in my programming knowledge.

The most shocking thing that I quickly learned in a startup is best described by a quote from the Game of Thrones*: "You know nothing, Jon Snow."*

- The culture was very different, until I realized that it must be.
- The cost constraints looked impossible, until I learned that they were not.
- The problems I had to solve were all new and uncomfortable, until I got used to them.
- And my awesome programming . . . Well, I fired myself from technology lead for . . . knowing nothing.

At the same time, I started to learn furiously the knowledge I discovered I lacked, which seemed to be literally everything at first! I also quickly noticed a great amount of synergy between my work in a startup and my work at both jobs. Here is my story of applying knowledge from a FinTech startup to modeling in a big insurance company.

IN-HOUSE MODEL OR VENDOR-BUILT?

Actuaries need models to do their work. Models are just software, which can be bought from a vendor or built in-house. For most companies, vendor systems offer a positive user experience at a reasonable price; but for some companies, in-house systems are the only way to satisfy business needs.

Common reasons for requiring an in-house model are innovation and speed of modification.

- No vendor would develop software for products that don't exist in the marketplace. Only the company innovator knows what these products are and can develop its platform accordingly.
- Innovation in the institutional space is often deal-driven, and includes the risk of being unable to transact due to waiting for vendor modifications of the modeling platform. In-house software is faster to change.

On the other hand, an in-house platform can have its own drawbacks and dangers, such as high cost of original build, scalability, transparency, build quality and maintainability. My perspective is that of a modeling actuary in a highly innovative company that opted for an in-house platform. In this article, my colleague Crew Sullivan and I will detail how to build an in-house modeling platform using a mix of technologies and processes to overcome typical drawbacks of an in-house system.

PROCESS STRUCTURE

Here are the steps we followed to build a successful in-house modeling platform:

1. Study the past. Why do we do modeling platform conversions?
2. Do we have necessary resources?
   a. Right skill sets
   b. Time
3. Do we have management buy-in and commitment?
4. Design phase (aka trade-offs, trade-offs, trade-offs)
   a. Design goals
   b. Object-oriented vs. procedural programming
   c. Software design principles
   d. Design patterns
   e. System blueprint (UML)
   f. Language choice
5. Execution phase
   a. Style guide
   b. XML doc
   c. Version control system (Git)
   d. Input structure
   e. Error handling
   f. Build order
   g. Unit testing framework
   h. Optimization
6. Testing phase
   a. Automate your regression test process
   b. Build a quality test bank
7. Maintenance phase
   a. Maintain UML
   b. Don't make a mummy! If you need surgery, don't use a bandage.
   c. Maintain documentation

STUDY THE PAST: WHY DO WE DO MODELING PLATFORM CONVERSIONS?

Companies switch modeling platforms for a number of reasons, which may include:

- **Scalability.** Excel models tend to run into this limitation, and the story typically goes like this: "My original platform priced my first contract in a day of runtime, and it was awesome. But now I have dozens of contracts in our pricing pipeline and a growing valuation block. My existing runtime is unacceptable, and I have no way of scaling it."

- **Flexibility.** "I came up with an awesome new product feature that a client wants, but there is no way to model this in my modeling platform. Modification will take a long time and/or will be very expensive to develop." Closed vendor systems tend to suffer from this the most.

- **Transparency.** There are two equally important flavors of this: user transparency and developer transparency. If users can't see and easily control calculations in an innovative business (think research and development), they will push for platform change. If developers don't know how to modify the platform, or you have a single person who knows it, you have an unacceptable operational or key person risk.

DO WE HAVE NECESSARY RESOURCES?

There are two critical resources that are necessary to build a quality in-house modeling platform: people with the right skill sets, and time. Let's examine both.

People With the Right Skill Sets

Would you go to a brain surgeon to fix a toothache? Then maybe you should think twice before going to an actuary to design and build software for you. We are great at insurance and many other things, but we generally know very little about programming and nothing about software engineering. To build a modeling platform, you need a dedicated team possessing the collective knowledge of actuaries, software engineers and programmers.

- **Software engineers** know how to build software and understand all the considerations that go along with it. They can take your business goals and design a system tailored to meet them. Software engineers will need help with business knowledge, but they can educate you on what is possible, the different techniques of achieving your goals, and the trade-offs involved.

- **Actuaries** possess the necessary business knowledge, but need help with software design, programming and software shop operations.

- **Programmers** are needed to do the actual work of writing the code. They need the help of both actuaries and software engineers to write code in an optimal way.

A stable team that cross-trains on actuarial, software engineering and programming topics can become a development powerhouse, requiring minimal business explanations and displaying impressive efficiency.

Time

Following a proper software development process requires extra time up front but pays off in faster speed of change and less maintenance over time. It is important to explain this to stakeholders and ensure they are onboard with giving you necessary time. Failing to do so results in one of two things:

- Burning out the development team with an unrealistic delivery schedule, which wastes a lot of time on hiring and training replacements.

- Cutting corners and failing to deliver the advertised quality, which can manifest as maintainability, flexibility, clarity and/or runtime issues.

MANAGEMENT BUY-IN AND COMMITMENT

Management buy-in is critical and can be a challenge to obtain. Building a maintainable model platform requires up-front investment in design, training and testing capabilities. The trade-off is that building quality software is an investment that pays off in lower-cost maintenance over time. When considering resource levels over time, usage of the model platform should be considered. The flexibility of the design may mean different functional areas could leverage the platform (as it did with us). It should also be identificed whether the modeling

team can act as a pooled resource across user groups in an efficient and cost-effective way. The challenge here is to make believers of the long-term value that the home-grown platform will provide.

Beyond the investment to build the system, management may also be concerned with having the resources with the right skills to support the model platform over time. After all, what good is a fast, flexible system if no one can interpret the design and make changes. Management support of a dedicated staff and robust model governance practices is important to the long-term success of this approach.

DESIGN PHASE (AKA TRADE-OFFS, TRADE-OFFS, TRADE-OFFS)

When creating a large or sophisticated piece of software, it is crucial to spend time to think through the software design before any code is written. Most actuaries have experience with writing relatively small pieces of code (100 to 2,000 lines) while being the sole developer. In contrast, software companies develop much larger systems with dozens of developers working on various parts of the code simultaneously. On this scale, multiple issues arise that most actuaries never experienced.

Design Goals

In-house systems can and should be designed to meet specific business goals. For example, business can desire the fastest possible model runtime (systems that need to do real-time market analysis), fastest possible development time (prototypes for new products/markets), clarity of the code (mature systems that will be maintained for a long time), flexibility of the code (pricing systems, systems that support multiple products, systems that change often), or some other priority. These goals often contradict each other, but a software engineer can make trade-offs to tailor the system to meet business goals. The first step in designing software is to pick your main design goal. This provides guidance for the software engineer regarding the qualities of the platform that should be maximized.

We selected flexibility and clarity as primary design goals for our actuarial pricing modeling platform.

Object-Oriented vs. Procedural Programming

Code organization can be broadly described as procedural or object-oriented. Procedural code is typically used for applications requiring extreme performance, such as sophisticated real-time market data analysis, or small applications with only a few hundred lines of code. Object-oriented code sacrifices some speed for clarity and maintainability. Generally, the object-oriented approach should be favored since it is easier to maintain. Design of large object-oriented systems requires an experienced software engineer who understands your specific business application.

We selected object-oriented design for our pricing platform since it provides superior flexibility and clarity over procedural design. Notice how our choices are driven by our design goals.

Software Design Principles

Software design principles are a set of the most general and highest-level aspirations for software. No system could or should strictly follow these. These principles are useful to keep in mind when making design decisions. They describe what makes a design good or bad, and help in understanding the tradeoffs being made. A good introductory discussion is available at *https://wiki.base22.com/display/btg/Core+Software+Design+Principles*. Here are the principles:

- Separate code that varies from code that stays the same.
- Program to an interface, not an implementation.
- Favor composition over inheritance.
- Strive for loose coupling.
- Classes should be open for extension, but closed for modification.
- Depend upon abstractions, not concretes.
- Principle of Least Knowledge (interact only with your immediate friends).
- The Hollywood Principle (don't call us, we'll call you).
- A class should have only one reason to change.
- Design to avoid rigidity, fragility and immobility.
    - It is hard to change because every change affects too many other parts of the system (rigidity).
    - When you make a change, unexpected parts of the system break (fragility).
    - It is hard to reuse in another application because it cannot be disentangled from the current application (immobility).

Applying these principles takes some practice; hence the experience of the software engineer matters in applying these correctly.

Design Patterns

A design pattern is a general repeatable solution to a commonly occurring problem in software design. You can think of design patterns as proverbial wheels that you can use to build a vehicle, without having to invent them. Design patterns introduce technical ways of achieving the aspirations laid out in software design principles. An outstanding introduction to design patterns is *Head First Design Patterns* by Eric Freeman and Elisabeth Robson. Just like with design principles, experience is required to apply design patterns correctly.

For our pricing platform, we heavily used strategy and factory patterns. We also used a modified command pattern to achieve our flexibility goal of supporting multiple products in a single platform.

System Blueprint (UML)

Unified Modeling Language (UML) is commonly used by software engineers to design software. It is a standardized diagram convention that lets you describe structures of software, database, use cases and so on. UML class diagram describes the objects in the system, the responsibility of each object, and the data flow between objects. For a sophisticated system, such as a complex pricing platform, a detailed UML class diagram must be completed before any code writing takes place. Completion here also means review with business partners to make sure the platform can support changes in the foreseeable future. The main reasons for completing UML before writing code are:

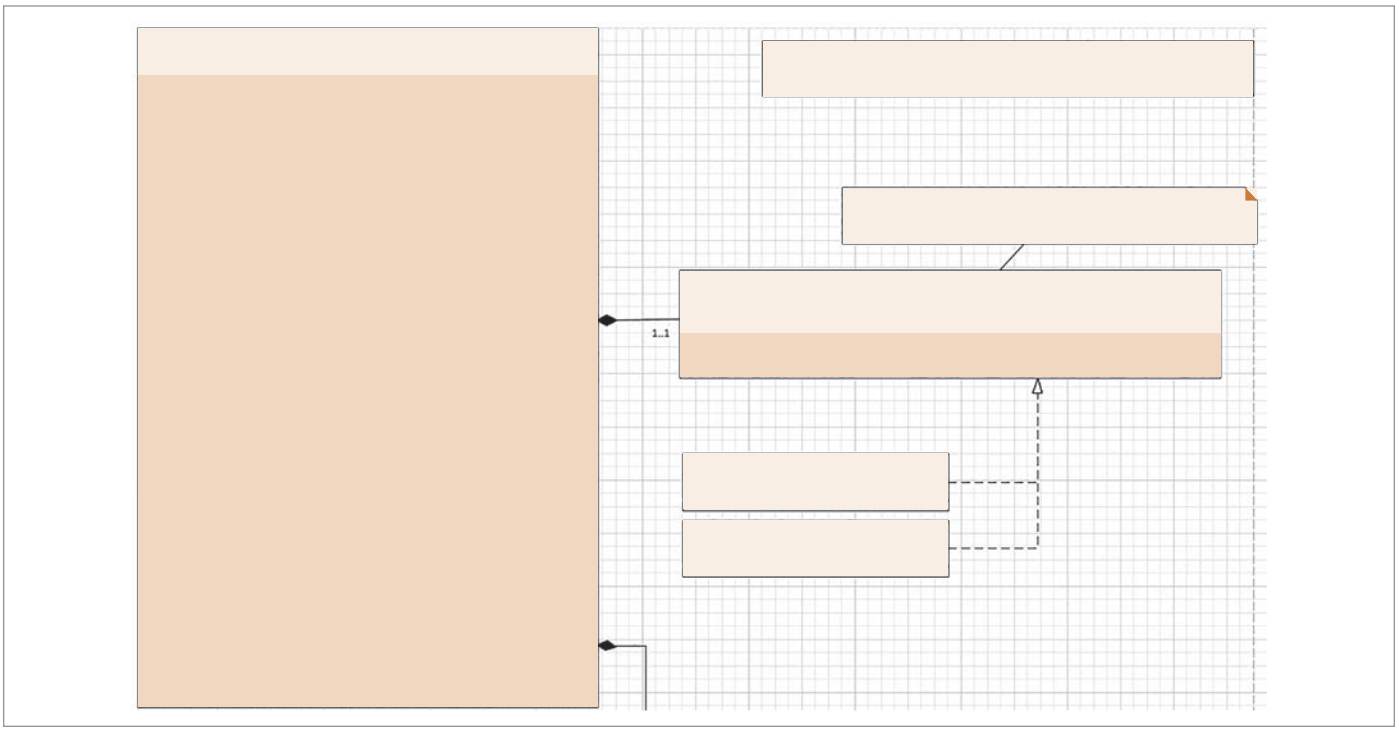
- Identify the responsibility of each object and how each will communicate. This may not seem important for a small system with a handful of objects, but it is critical to document this for a large system with hundreds of objects and multiple developers.

- Multiple developers writing code simultaneously need to rely on communication protocols described in the UML to ensure smooth code integration.

- Conduct role play to identify how various products (including new products) will fit into the platform.This will reveal design flaws that can be addressed much more easily on paper than in the code.

Our completed UML diagram was 70 pages long and could be hung neatly on a large wall (4 meters wide and 2 meters tall). We used Visio to create it, however, you can also use other online tools such as Cacoo. (See Figure 1.) After reviewing it with our business partners, we discovered a major computational inefficiency, which was resolved in three weeks of fixing UML. Fixing

Figure 1
Example of UML Class Diagram



this in the code would have taken far longer and, without this fix, we would have been unable to meet our runtime goal.

Once the UML was finalized we began writing code with a team of five actuaries who had technical guidance from the software engineer. The team worked in parallel using UML as the technical specifications. The model build went very smoothly with no integration issues.

Language Choice

For calculation engines, programming language choice mostly boils down to the following considerations:

1. **Higher-level or lower-level programming language?** Lower-level languages (like C) are faster, but less clear, harder to debug, and require much more programming experience to employ effectively. Manual memory management is a powerful tool, but there are a lot of things that can go wrong, and it requires appropriate training and experience. Higher-level languages (C#, Java, Python) are slower but easier to use, have more "guardrails," and are recommended for less-experienced developers.

2. **Will you need to use specialized libraries that have better support in a certain language?** If you need GPU support, then Python and C++ have some robust free libraries. If you heavily use linear algebra, then MATLAB might make sense. If you don't really need anything special, then pick a free language (MATLAB is not free, for example).

3. **What does my grid support? Can I easily install things on the grid?** If you don't have control over what gets installed on the grid, then it might be necessary to pick a language that caters to the grid. For example, C# would work if your grid already has the .Net framework on it, but you would have a problem if your grid runs Linux and you can't install Mono on it.

For our pricing platform, we picked C#. It is one of the easiest programming languages to learn and use. We did not need any libraries outside of .Net, and our grid already had .Net framework on it.

EXECUTION PHASE

We finally have our detailed UML; we know our design goals; we picked the language to use; and we are now ready to write some code! Here are the ideas and technologies that will greatly speed up the process.

Style Guide

Everyone intends to write "great code," but not everyone interprets that the same due to personal experience and preferences. Great code from a novice programmer usually looks outright awful to an experienced developer. A style guide is a document that lays out the rules of how to write the code. A good style guide includes variable naming conventions, commenting requirements and so on. One of the most popular style guides is Google C++ style guide, which is available at *https://google.github.io/styleguide/cppguide.html*. The benefits of using a style guide include:

- Code written by different developers looks and feels the same. This accelerates the code review process and onboarding of new hires.

- Developers can work with and debug each other's code much easier, since all code looks similar.

- Using a style guide makes code transparent. It reduces the chances of ending up with cryptic code that only the original author understands. Cryptic code is one of the main reasons for model platform changes.

Adherence to a style guide should be continually enforced. It takes time to see the benefits for a programmer who never worked in a team-based development environment.

We used Google's C++ style guide with slight modification for our pricing platform since we used C# and not C++.

XML Doc

XML doc is a technology that allows storage of documentation directly in the code. You can then use programs like Doxygen or write your own interpreter to have HTML documentation generated from tags in your code. XML doc allows for tighter integration of code and documentation. It also allows the use of programmatic hyperlinks, which are easier to maintain compared to hyperlinks in a Word document.

For our pricing platform we wrote our own XML interpreter that generates Microsoft-like HTML documentation.

Version Control System (Git)

Version control software like Git is the most important technology required for development in a team. It tracks versions of the software, allows for very efficient review and merging of the code from multiple developers, and saves enormous amounts of time and effort. Git enables each developer to work in their own branch of code. The project lead can then easily review branches from different developers, and either send them back for additional development or merge them into your most current accepted code. Vincent Driessen wrote a good article on how this should all work (*http://nvie.com/posts/a-successful-git-branching-model/*). You can use online repositories like GitHub and BitBucket, or you can use Git functionality built into Visual Studio and many other development environments.

Input Structure

Most actuaries are familiar with Excel and csv files. However, for large systems these may not be good programmatic inputs; especially Excel, since its data access is very slow. When designing a large system with long-term maintenance in mind, you may want to consider specific file formats like JSON or more robust data storage solutions such as a database. The benefit you are after is generic programmatic data transfer between your user interface and your calculation engine. Adding or expanding a table or adding a new switch should require no coding related to data transfer since all data should be transferred generically. Since data in the JSON file is tagged, your code can handle its transfer from one media to another generically by using metadata. Databases would require some setup of metadata, but a similar approach can be used.

For our pricing platform the Excel user interface creates XML parameter files, which are used by the C# calculation engine. The process is fully automated, so that adding a set of brand-new input tables for a new product requires only defining them in the Excel interface and tagging them with several named ranges. The Excel code and C# code are fully generic for all input tables, and hence require no modification!

Error Handling

You can use exceptions to generate a call trace of the error. This will greatly improve user experience and reduce the time spent on user support. To do this, you simply wrap all your methods in a try catch block generating an error message that contains call description appended with the existing error message from downstream objects. The resulting error message would look something like this.

> Calculate method of Benefit object found negative benefit amount -43. Only positive benefit amounts are allowed for this benefit category.
> Calculate parameters were: benefitName = salariedPlan, category = JointAndSurvivor, amount = -43.
>
> Calculate method of Policy object encountered an exception.
> Policy parameters were: policyNumber = 103945.
>
> Calculate method of Contract object encountered an exception.
> Contract parameters were: contractName = TestCo.

This was one of the favorite features of our model users since it was now very clear why the model didn't run and how to fix it.

Build Order

When building a system from scratch, the most common build order is simply trying to get a runnable skeleton, and then adding actual calculations. Our build order looked like this:

1. **Launcher** is an interface that launches your code locally or on the grid. It verifies that you will not have compatibility issues between your user interface, programming language and grid. Launcher is your simplest runnable model.

2. **Base classes and interfaces to read in inputs** will let you read in your input files, connect to your databases and APIs. This furthers your verification of compatibility and connectivity.

3. **Empty base classes** enable you to have a runnable skeleton of the model.

4. **Detailed report** is a dump of all vectors produced by the model. This will come in very handy for developers since all of them will need an easy way to verify their calculations.

5. **Implementations of actual calculations** should be written last. This step can be written in parallel by multiple developers. At this point each developer can run a model, get inputs

and produce a nice report that includes the vectors that he or she is working on. Very efficient!

Unit Testing Framework

Unit testing framework offers an efficient way to run tests for each class. For example, if you have a class responsible for age calculation, you can use unit testing framework to write a test class that will instantiate your age class with a variety of parameters and check the calculation results. This is very useful as a quick regression test that verifies that code modifications did not break something in an existing class. This works very well on classes that sit at the bottom of the call hierarchy. For top-level classes, such as Launcher or Contract, setting up unit tests is too complicated and hence impractical. Top-level classes should be tested using full model runs and analysis of the outputs.

Optimization

Optimization should be done after each functionality goal is achieved. It is faster and easier to optimize the smallest possible amount of code. Some of the more sophisticated integrated development environments (IDEs) have built-in optimization support. For example, Visual Studio has Performance Profiler, which will record time spent on each calculation and show you what took the longest to run. It will also provide counts of each method call. You get the biggest benefit from optimizing objects that get called the most. Here are some ideas of what could cause performance drag:

- **Searching for something more times than you need.** Examples include looking up the same value inside of the loop or on a lower level than you could and getting items one at a time from a vector in a dictionary, when you could get a vector from the dictionary once and then use indexing to get individual items.

- **Inefficient calculation reuse.** For example, if you compute mortality for every benefit on a policy, it might be much more efficient to compute mortality once and reuse it for all benefits. This is a design issue, though, and may or may not be possible to address easily once you have the code written.

We did two optimization rounds for our pricing platform that yielded runtime improvements of 25 times and further four times, for a cumulative 100 times faster runtime. It really pays to spend time on this!

TESTING PHASE

Once the model is complete, it is time to make sure it is production-worthy. Generally, testing can be broken into two parts. Regression testing makes sure a new version of the model didn't break anything. New functionality testing makes sure additions work as expected. There is not much that can be done to improve testing of new functionality, but there is a lot that can be improved for regression testing.

Automate Your Regression Test Process

You should consider automating your testing process since its efficiency or inefficiency will drive the quality of your regression testing. Ideally, you want to be able to run a full regression test of the new model with a click of a button and get a comprehensive report on the test cases that didn't match. You can then compare the test numbers to your test names and see a pattern. ("Aha! All tests containing a particular benefit failed!") Your automation goal should be that the number of tests does not matter and your tools can handle one test case just as easily as 1 million.

We wrote a custom utility that compares model outputs in two folders and produces a report on the largest mismatch in each test case. We then compared this report to our test bank description to see if there was a pattern.

Build a Quality Test Bank

In theory a regression test should cover all possible input combinations, but practically you need to be able to run and analyze it in a reasonable amount of time. The inputs that should not be used should be programmatically blocked with an appropriate error message. The test bank should include tests that verify that the model will not run with prohibited inputs. The task of constructing the test bank can be done in parallel with development of the model. It is a good idea to have a well-organized document describing the tests, since it will greatly speed up analysis of the mismatches.

Our regression test bank contains more than 6 million test cases, since our automation enables us to run and analyze the results in about a day. The test cases cover all combinations of various toggles and switches for all the products that the system supports.

MAINTENANCE PHASE

Once the model is in production, it is important to maintain and preserve its original qualities. Poor maintenance will deteriorate the model and may result in issues that cause the next model conversion. It is important to be diligent on maintenance.

Maintain UML

Larger model changes should be approached similarly to original model design and hence should be first implemented in UML. This will help you think through the various possible ways to implement your changes and identify possible implementation issues. Make sure that the changes you make flow

well with overall model design. UML also helps with training new developers as it provides an uncluttered way of walking someone through the code flow.

Don't Make a Mummy! If You Need Surgery, Don't Use a Bandage

Many changes can be implemented in several different ways in an object-oriented system. Make sure to pick the way that is most sound from the software design principles and design patterns perspective. It may not be the fastest way to implement the change, but it will save you from having to work with a patchwork of various approaches and implementation styles a year into the model's life.

Maintain Documentation

Documentation should be maintained as part of the development process, especially if you use XML doc. It is easy to describe what you did and why you did it in the code as you write it. It is much harder to do later.

CONCLUSION

There is a lot to learn from practices of IT and FinTech industries when it comes to model building. Some tasks are best handled by integrated cross-functional teams. Technology provides a lot of efficiency, but unlocking its potential requires very close cross-functional collaboration. ■

Igor Nikitin, ASA, MAAA, is a director, actuary at Prudential Financial. He can be reached at *igor.nikitin@prudential.com*.

M. Crew Sullivan, FSA, is a VP and actuary at Prudential Financial. He can be reached at *crew.sullivan@prudential.com*.

Modeler Q&A With David Yu

By Daphne Kwan

David Yu, FSA

In this article, we talk to David Yu, FSA, director and actuary, Modeling Center of Excellence, at Prudential Financial. David discusses his experience leading asset and retirement product model development.

Q: What kind of modeling work do you do, and what software platforms do you use?

A: I am responsible for modeling asset-liability projections for financial reporting and internal management purposes. This includes modeling of retirement liabilities, as well as the assets and reinvestments that are shared by multiple lines of business. I develop automated processes to build and run models.

Q: How do you plan or prepare for a modeling change?

A: We follow a well-defined Model Development Life Cycle (MDLC) to prepare for and make model changes. Users prepare business requirements for the model change. We, the modeling team, then review the business requirements and clarify changes with the users. We assess the complexity as well as the model design, and, if needed, discuss this with the platform vendor to better understand the effort required to affect a model change.

Q: What do you do if you find a bug that has an immaterial impact on results?

A: If we identify a minor bug, we then follow a reduced version of MDLC. This requires communication:

- Assess the potential materiality with model users.
- If no immediate changes are needed based on materiality, we will log the bug and fix it before the next model release.
- If immediate changes are needed, we will apply the changes, perform model testing, and release the changes to users.

Q: Do you document as you go along, or after you're done?

A: Our goal is to prepare documentation as we go along. Our main documentation produces technical specs for the models we develop.

Documentation also plays an important role in our internal modeling work. It is a more effective way of communication. We divide duties internally for development, technical review and peer review. We also often need to assign work to different developers, and then integrate the work. All this requires teamwork and effective communication, which is best done by documentation.

Q: What's your modeling pet peeve?

A: Using the annuity module to model retirement products. The annuity module is designed for an individual annuity, and it may not be equipped to model some retirement-specific features. For example, stable value products may have a book value discontinuance feature that provides a contract holder with book value payments in installments, rather than as a lump sum.

Modeling the installment payments can be challenging and we can do it, but it's so much easier if the right model is used from the outset.

Q: What's the most frequent piece of code that you usually forget the syntax for?

A: I would say the coding related to read/write access to external files.

Q: What was the last problem you encountered that had an easier-than-expected solution?

A: The last problem that had an easier-than-expected solution dealt with using externally projected asset (EPA) templates to handle more generic path-dependent external projections. Sometimes we need to incorporate external projections into the model. The projections may vary by scenario paths, and it can be challenging to incorporate them directly into an asset-liability projection. We found a way to funnel the external projections through EPA using out-of-the-box platform features, so that they could be included in the projection without making many changes to the process.

Q: What do you wish consultants understood about your models?

A: I would like consultants to understand our modeling process. It is also helpful to suggest ways to make it more modular. We are often in a situation where we need to provide a process for different businesses. The ability to reuse process components would make the process more effective and easier to maintain. ■

Daphne Kwan, FSA, is a vice president and actuary at Prudential Financial in Modeling Center of Excellence. She can be reached at *daphne.kwan@prudential.com*.

What's a Model? A Framework for Describing and Managing Models

By Dodzi Attimu

The definition of a model is one of the first items addressed in any model risk management program. Thankfully, the Federal Reserve's Office of the Comptroller of Currency's guidance in "SR Letter 11-7" on model risk management[1] provides a good benchmark (if not the standard) regarding what a model is for the financial industry. In this article, we will address some model-related questions that arise in operationalizing a model risk management program. Some of these questions may be philosophical while others are operational. One such philosophical question is whether a model is a *process* or a *unit* that transforms input via computational methods into useful output/estimates. An example of an operational question is whether to classify modeling functionality on a single platform (e.g., Prophet, MoSeS, GGY AXIS, etc.) as a single model or as multiple models.

Regarding the operational situation, a typical scenario is the following: An ALM projection functionality is built on a platform like GGY AXIS for insurance products $Product_1$ and $Product_2$ that generates cash flow (CF) projections that are used for Actuarial Guideline (AG) 43 and C3-Phase 2 analysis and reporting. The question becomes: Does this represent one model or four models (the latter corresponding to a model each for the two products times two business processes) or maybe two or three models?

In this article, we outline a *formal framework*[2] *for describing models* that is inspired by the operational context of governance, management and use of models. This is a coherent and consistent frame of reference to answer relevant questions related to models and their use. The framework also provides a sound and easy mechanism and "language" to articulate and analyze different design approaches for models that may have big impacts on the efficiency of business processes relying on them.

Another helpful feature of this framework is that it leverages the actual operational aspects of the use and maintenance of models. Consequently, we expect the framework and related ideas presented here to be of interest to model developers, model testers and model validators, individuals in model governance or model audit functions, as well as business users of models.

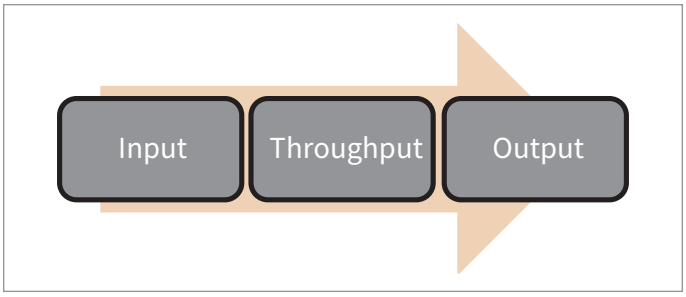
OK, SO HOW DOES ONE DEFINE A MODEL AGAIN?

The Fed guidance in SR Letter 11-7 on model risk management states: "… a *model* refers to a quantitative method, system, or approach that applies statistical, economic, financial, or mathematical theories, techniques, and assumptions to process input data into quantitative estimates."[3]

More pithily, a model is a means to transform input (data, assumptions and other parameters) via a processing component (throughput) into quantitative estimates (output).

Figure 1 provides a depiction of a model showing the constituent parts.

Figure 1
Depiction of a Model



Typically, a model is defined in the appropriate policy (or policies) of a model risk management program and is usually the exact definition in SR Letter 11-7 noted previously or modified based on the operational needs or priorities of the model risk management program. For example, in some programs, any tool that performs any sort of quantitative transformation is classified as a model, whereas other programs add an extra requirement that the transformations involve uncertainty or some element of judgment/assumptions.

In some instances, too, a model is defined as an end-to-end modeling process spanning the sourcing of inputs through extraction, transformation and loading (ETL), model run, and reporting based on the output. We would argue that this latter definition is of a business process that utilizes a model(s) because defining a model as a process provides both philosophical and semantic challenges.[4]

In this formal framework (which is based on the earlier SR Letter 11-7 definition), a model is not a process but an operational unit and is different from business processes that utilize it. Conversely, a business process would be one that may utilize models as tools. Thus, a *model validation* may focus mainly (but not exclusively) on this operational unit, whereas a *business/*

modeling process validation would include validation of all relevant components of the process, including the model(s) used therein.

Formal Structural Definition of a Model

Conceptualizing a model as an operational unit consisting of an input structure, processing component (throughput) and output structure is fairly high level in the sense that we do not go into details about what requirements the processing unit should perform to qualify as a model. That can be determined by a model risk management program as needed. The use of *operational unit* in this framework means that a model could use a collection of software libraries, input file structures and so on. However, if to use the model, the input structure (potentially a collection of structures physically represented by different files) is operated on collaboratively by these components, all these (potentially stand-alone) components are part of a single model.

Defining "Model"

A model is an *operational unit* consisting of an *input structure* and a *throughput* (processing logic/functionality) that acts on the input structure to produce an *output structure*. We will refer to the input structure, throughput and output structure as *operational components* of a model.

Sometimes, confusion arises regarding the use of the term "model" because it can mean a quantitative abstraction of reality (i.e., in the phrase "asset/interest rate/mathematical models") on one hand or an operational unit (what the definition of SR Letter 11-7 addresses) on the other. Consequently, a model (an operational unit) for valuing a portfolio of assets could consist of many different asset/interest rate models (abstractions of the value of these assets/risk factors). Examples include LIBOR Market Model, Black Scholes model, SABR model and so on.

In other words, if a code base (logical specification) encompasses several de facto "mathematical/financial models" but supports an input structure to generate an output structure, we *operationally* have *one model*. Without a framework for defining models from an operational standpoint, there could be (unnecessary) disagreement on what constitutes a single model or multiple models.

Sometimes, confusion arises regarding the use of the term "model" because it can mean a quantitative abstraction of reality on one hand or an operational unit on the other.

In addition, we note that the processing logic of a model consists of all logic that is accessible (reachable) through a unique logical entry point. At a high level, consider this entry point to be synonymous with a "RUN" button or a command that triggers the calculations.[5] The next sections expand on the three operational components of a model.

Input Structure

The first operational component of a model is the input structure. This includes user interface, configuration files and an input data structure that may reside in external files (which may be referenced from the user interface).

First, in this formalism, input includes raw data, assumptions and parameters. However, it is possible due to convincing reasons or just bad design that certain aspects of these are "hard-coded" in a given model's implementation code or set-up, that is, throughput.

Second, the use of *input structure* instead of *input* is because input consists of structure and content/values *leading to a distinction between input structure and input content/values*, which is a deliberate and important distinction in this framework. Input structure is the general "shape"/data structure of the inputs—for example, what type of inputs are expected and how the various elements are arranged—whereas input content refers to specific values for these inputs in the structure. This (input) structure can be represented abstractly as a collection of tables. Note that this choice is for convenience.

One natural motivation for the distinction between content and structure is for the purposes of defining what a model change is in this framework (see "Defining 'Model Change'"). In particular, we naturally have a situation where a unique model can have different input content and hence generate different output. In other words, changing an input value to a model does not result in a different model; it would be the same model producing different output.

Third, though our choice of tables as building blocks of input structure is for convenience, it is also general enough to support any form of input structure. This readily follows from the fact that for any set of inputs, one can construct a one-field table for each element of the input. One immediate outcome of this observation is that the representation of an input structure is not unique (e.g., one could organize the structure into two or three or more tables). Hence for a given set of inputs, different input structures could be used to support them. These structures could be different but capture the same information. Physically, the input structure could consist solely of one or multiple types of the following: relational database, text files, Excel files, special file format native to system,[6] and so on.

Finally, note that we consider the input structure as encompassing all that a collection of code, plus any other processing component constituting the model, operationally supports to process input values.

Throughput

The throughput is the second operational component of a model and refers to logic that transforms the input to provide estimates or output. In addition, we also consider as part of the throughput any other component of the model that is not considered as part of input or output structure. In other words, we include parts of modeling system that are responsible for generating and formatting output and performing modeling housekeeping activities such as validation of inputs as well; not just the business logic.

Output Structure

Similar to our highlighting of *structure* for inputs, we emphasize the structural aspects of the output. The output depends on the throughput. In addition, similar to the case of the input (structure), we will assume without loss of generality that the output has a structure of a collection of tables. Again, similar to the input structure, we consider the output structure as the union of all (table) structures supported by the code base via its point of entry.

Interrelationship Between the Three "Puts": Input Structure, Throughput and Output Structure

We first note that in this framework, software code that implements some logic, but has no functionality to provide output, does not qualify as a model. All three operational components must be present for a classification as a model.

Another aspect of the interrelationship between the operational components of a model is related to whether an assumption is part of the input or part of the methodology (throughput) of a model. For example, consider a (toy) model that projects stock prices under the Geometric Brownian Motion (GBM). The functional specification of this model is an assumption.[7] It is also correct to say that the volatility parameter is an assumption.

The challenge when talking about assumptions related to a model (e.g., appropriateness for a given modeling use) is to determine whether one considers the GBM specification as an assumption or only the volatility parameter. This is the motivation for the following two definitions.

Defining "Assumption Input"

An assumption that can be captured via the input structure is called an *assumption input*. Since input consists of both structure and content as noted earlier, we consider an assumption input as consisting of an *assumption input structure* as well as *assumption input content*.

Defining "Assumption Throughput/Implementation"

An assumption that is part of the implementation software code (processing logic/throughput) is called an *assumption implementation*.

Let us revisit the point earlier about different input contents to the same model in the light of our GBM asset projection system. Assume our input structure consists of four entries per stock: the number of time steps, length of time step, number of paths, and the volatility. Changing any of these input values does not result in a different model. An equivalent deduction is that the input content, while necessary to produce output content of a model, is not a component of the model. This makes the definition of a model in the framework an operational abstraction.

SOME APPLICATIONS OF FRAMEWORK

In this section we outline some applications of this framework. First, we answer the question of whether an assumption is part of a model or not. Next, we tackle the problem of determining if a component is part of a model. We then address the issue of determining the number of models represented under a given modeling setup for different products supporting different business processes/metrics. Finally, we consider in general some model management concepts of model design, change management and related activities.

When is an Assumption an Input or Part of a Model?

For example, consider the earlier simple model that projects stock prices under the GBM. Assume that the input structure supports a single (constant volatility) parameter (ignoring other input values supported by the input structure) per stock. Are the volatility parameters and GBM assumptions part of the model?

To answer this question, note that the volatility parameter value is an assumption input content and hence is not part of the model. In addition, recalling that input consists of input structure and input content, and that it is the input structure that is a constituent operational component of a model, we will say the *assumption input structure* is part of the model (though the *assumption input content* is not, as noted earlier). For readers who might struggle with the latter point, note that intuitively, one can change the content of the volatility input structure without creating a different model as a result (more on this in the upcoming formal definition of a model change). On the other hand, as the formulaic implementation of the GBM is fixed in the model processing code/logic, it is an *assumption implementation*; and since the throughput is an operational component of the model, it is part of the model.

However, another design could involve an input structure that captures the type of stock price evolution assumption specification as well as the parameters for the specification. For example, the user can specify either GBM or GBM with Jumps as input content/values in addition to the assumed input content/values of the parameters. In this case, the assumed functional specifications as well as parameters are part of the input, and we would conclude that the actual *choice of volatility input* and *model specification* used for the model is not part of the model—they are mere inputs into the model. This last design illustrates an important consideration for designing models that are flexible and operationally efficient. We hope to follow up with an article on elegant, efficient and flexible model designs with emphasis on user configuration of third-party projection platforms.

Determining Constituent Parts of a Model

Consider a vendor modeling platform[8] that has an operational unit used for ALM projection and has:

- Input structure supporting inputs like economic scenario input (projected yield curve for Treasurys and spreads over Treasurys) for all fixed income assets in the portfolio of assets backing general account liabilities, equity and dividend growth rates of indices mapped to separate account values (AV), liability policy data (AV, guarantee bases, age of policyholder, etc.), assumptions input (parameters for lapse formulae; GA reinvestment strategy, e.g., target allocation, reinvestment frequency, etc.) among others.

- Suppose also that this model projects the assets and liabilities and produces (via the throughput component) an output structure housing cash flows (assets and liability cash flows) and financials on a STAT basis by scenario and for each monthly time step for 40 years.

- The output structure consisting of at least one table with a field that captures scenario number and houses the monthly income statement output for 40 years.

  - At least one table because there may be other, lower-level information that constitutes the output structure, for example, debugging information that has intermediate calculations or calculation results at a lower level of granularity. These may be optional output that is supported by the throughput and hence is part of the output structure. For our purposes, it is the financial statement component of the output structure that is important in this example.

- After the results are generated in the output structure, an Excel Analytics tool calculates a conditional tail expectation (CTE) number among other analytics and graphs.

In this scenario, is the Excel Analytics tool part of the model?

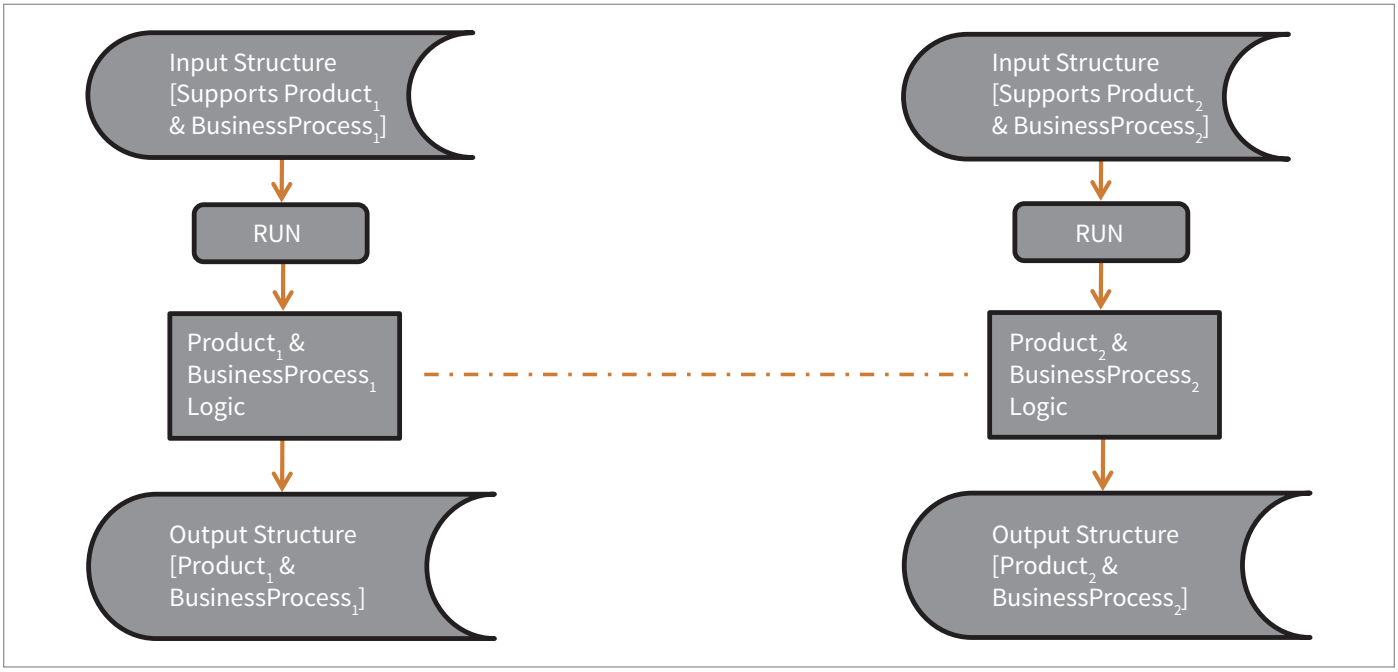
The framework gives a natural answer, which is it depends on whether that analytics functionality is part of the platform's processing logic (i.e., throughput). The reasoning follows naturally from throughput consisting of all logic that is reachable from the model run entry point. Consequently, in this example, if the analytics tool is a stand-alone tool that can only be activated by manually opening and using it without it being driven by the model throughput (via its entry point), it is not part of the model.

The natural follow-up is whether the Excel-based analytics tool can be made a de facto part of the model in this formal framework, and it can. (So fans of "model-as-a-process" paradigm can still operationally design a single model that touches all applicable processes.) To do that, it suffices to incorporate the analytics tool as part of the throughput. Operationally, one option is to add to the throughput some logic/functionality that triggers the working of the analytics tool directly in a way that is reachable from the entry point.

In other words, hitting the proverbial "RUN" button would run the model and trigger the analytics tool functionality. This does not have to involve removing the option to use the analytics tool independently on a stand-alone basis. The formal principle that is applicable is the so-called *enclosing/encompassing property of throughput*. This property posits that any *customized* (potentially independent/stand-alone) functionality (code, .dll, .exe, etc.) that is reachable (e.g., called) from the entry point of the throughput is a de facto part of the throughput.

This property is not just a purely abstract algebraic concept for its own sake.[9] Its importance derives from the fact that the operational model component of throughput usually consists of various (potentially independent, multi-technological)

Figure 2
Determining Number of Models



operational components. In addition, many customizations by users may involve adding .dlls, executables or scripts that add various functionality, including input validation, and analytics to the throughput.

Determining "Number of Models"

In this example we consider the case of a projection system that supports the projection of variable annuity liabilities for n products $Product_1,...,Product_n$. These products are all modeled on the same platform, say PlatformX. In addition, using this platform, m business processes (or metrics) $BusinessProcess_1,...,BusinessProcess_m$ are supported.[10] How many models are represented in this scenario? On one extreme, we have n × m models (one model for each product, business process (metric) combination). On the other extreme, we have one model that supports all products and metrics.

But what is the right answer? We show how to make this determination naturally (without resorting to subjective "judgment") using this formal model description framework. Indeed, the number of models in this case is determined by how many stand-alone operational units are represented in the modeling setup. In other words, it is determined by the *design/configuration of the model(s) on the platform*.

Let's delve deeper and show how to make the determination. Based on the formal definition, if there are n × m different operational units (consisting input structure, throughput and output structure) then there are n × m models. Without loss of generality, let's consider that there are two products, $Product_1$ and $Product_2$, and business purposes, $BusinessProcess_1$ and $BusinessProcess_2$. In one extreme, we could have four models (operational units) with the following model representation for each combination:

- A model for $Product_1$ and $BusinessProcess_1$
- A model for $Product_1$ and $BusinessProcess_2$
- A model for $Product_2$ and $BusinessProcess_1$
- A model for $Product_2$ and $BusinessProcess_2$

Diagrammatically, we illustrate any one of the four models in Figure 2. (We have shown the first and fourth models.)

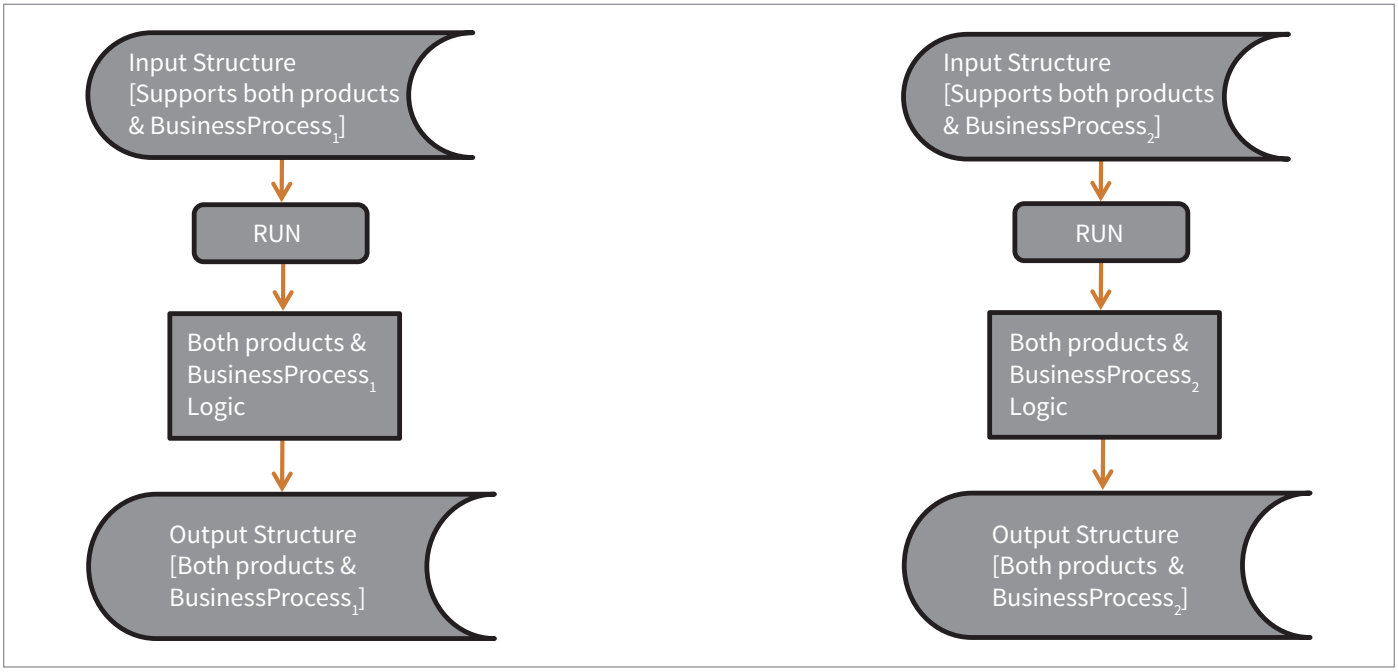
Another possible design would be a two-model design:

- A model for $Product_1$ & $Product_2$ and $BusinessProcess_1$
- A model for $Product_1$ & $Product_2$ and $BusinessProcess_2$

These are illustrated in Figure 3.

One the other hand, we could also design a single model to cover the products and business processes. To do that, it suffices to combine the processing components and hence the input (and output) structures. Without loss of generality, one need

Figure 3
Two-Model Design



only consider addition of fields that specify the product type and business process as part of the input structure. Naturally, this leads to a combined input structure that supports both products and business processes.

Note that this naturally satisfies the conditions for having a single model:

1. There is a single input structure that supports both products.[11]

2. There is a single processing unit that acts on the same input structure (that supports input contents representing both products and business processes).[12] Another way of saying this is that there is one processing code base for both products and business processes (metrics).

3. Typically, once the first two are satisfied, we have the same output structure for both products.[13] In other words, the input structure plus the throughput determines the output structure as well.

Figure 4 illustrates such a design.

A similar scenario is this: Given an asset modeling platform that supports the modeling of different asset classes $A_1,...,A_m$, does this represent m models or some n < m models? Using similar reasoning as before, if the underlying code framework supports all the different asset classes, then this constitutes one model. If, on the other hand, there are stand-alone code (base) units that support the individual assets and these units are not reachable via a single entry point, then they can only be run as individual units and each such individual unit is a separate model.

Finally, a top-down mechanism for determining if the setup represents a single model consists of answering the question: "Can one utilize the same input structure and singular entry point ("RUN" command) to generate results for both products and business processes?"

Model Design Implications for Model Governance and Control

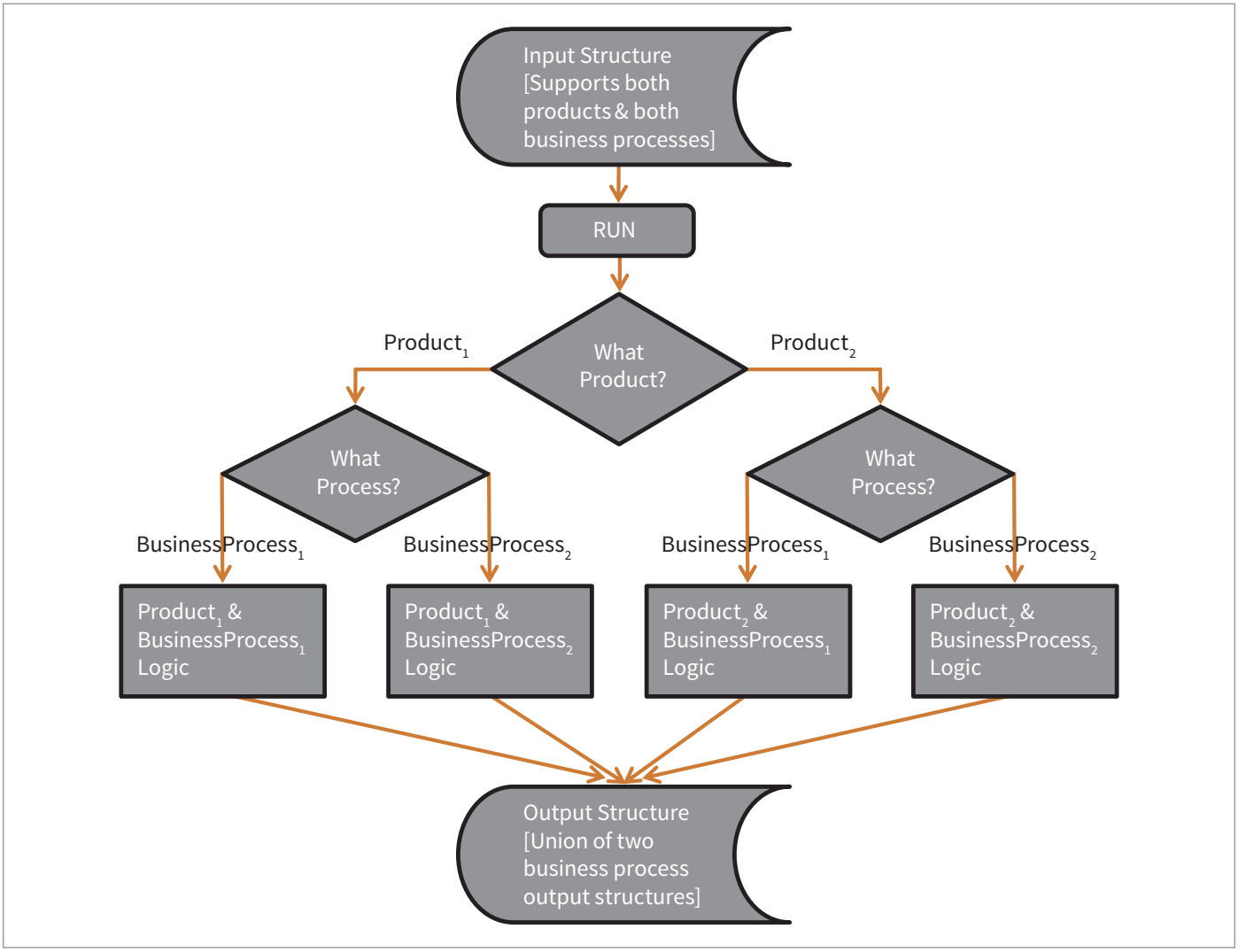
For a better appreciation of the implications for model design and controls, we propose the following definition of what a model change is using this framework.

Defining "Model Change"

A model change is defined as a change in either the input structure or throughput.

Now let's proceed by considering the example in the previous subsection. There would be one model that supports multiple processes if the model is designed such that its throughput can interact with an input structure that:

Figure 4
Single-Model Design



- Supports the modeling of different products; that is, the input structure supports different products

- Supports different types of business processes; that is, the input structure supports different business processes (e.g., AG43 output, C3-Phase 1 output)

In addition, as more assumptions and parameters are supported by the input structure (as assumption inputs), we have a situation where such assumption updates do not go through model change processes since they are input (content) changes, not model changes. This is a natural corollary of the definition and is consistent with one's intuition regarding "inputs."

The alternative is to make a determination of what constitutes a model change based on (subjective) judgment sans a framework. Interestingly, changing an input structure (as minor as that may be) is a model change, whereas changing an assumption such as the target allocation of a reinvestment strategy that could have major impacts on model results is not a model change if it is solely effected through the same input structure.

This does not imply that business users should not test and assess the validity of results coming from the model before putting the new assumptions into "production." It only means this is work that is outside *model change control* and is rather an *assumptions change control* process that can happen without triggering model change protocols.

There is a higher initial cost in setting up and testing models in this way, however. For example, test strategies should cover different input choice combinations to ensure that the abstraction(s) inherent in the model design are valid.[14] On the other hand, the advantages in flexibility, maintainability and efficiency of this input-driven approach can be huge. Users have more flexibility to perform analysis or assumptions updates that conform to input structure on the (official, validated and tested) model without triggering a model change process.

In certain cases, models are designed with inadequate consideration for what should be hard-coded vs. what should be part of input structure. This leads to duplication of models that are logically the same except for a few differences in the throughput. Over time, because these "similar" models develop a life of their own, they tend to diverge in unintended ways leading to potential problems and inconsistencies down the line.

In fact, the considerations for good model design are enough for an entire article, and we end this subsection by noting that a lot of efficiencies can be derived by shifting stuff that is traditionally considered as throughput into the input structure. Finally, using this framework to design elegant and efficient models is not only possible with home-grown systems but also with models built on vendor-supported software platforms like GGY AXIS, MoSeS and Prophet. As noted earlier, we hope to pursue this in a follow-up article.

CONCLUSION

In this article we introduced a formal framework for representing a model that is consistent with the financial industry standard definition of a model as seen in SR Letter 11-7. This framework operationalizes the definition of a model and naturally answers questions such as what functionality constitutes a model or how many models are represented by different projection capabilities or business processes. Finally, this framework also provides a natural and succinct way of communicating model design and hence improvements in existing designs or entirely new ones for modeling capabilities. ■

Dodzi Attimu, FSA, CERA, CFA, MAAA, Ph.D., heads the model validation program at MassMutual. He can be reached at *dattimu06@massmutual.com*.

ENDNOTES

1 Board of Governors of the Federal Reserve System Office of the Comptroller of the Currency, Supervisory Guidance on Model Risk Management (SR Letter 11-7), April 2011, *https://www.federalreserve.gov/supervisionreg/srletters/sr1107a1.pdf.*

2 The exposition in this article is a synthesis of concepts from the rigorous (algebraic) development of the framework by the author.

3 *Supra*, note 1.

4 The main philosophical/semantic challenge is that a model and a process are not necessarily interchangeable. For instance, consider that a model is a tangible thing (operationally), whereas a process is not; it is a sequence of operational steps, some of which may involve running the (tangible) model(s).

5 This concept is easier to grasp in models developed on programming platforms but requires more work to formalize in purely spreadsheet-calculation-based models.

6 We strongly believe that third-party platforms should be able to communicate with well-known file formats. They could convert data to some underlying native file formats as needed, but forcing the user to convert to native file formats is a bad use of users' time.

7 In other words, the stock price will evolve under the specification $S_{t+\Delta t} = S_t e^{\left(\mu-\frac{1}{2}\sigma^2\right)\Delta t+\sqrt{\Delta t}N(0,1)}$.

8 For example, GGY AXIS, Prophet, MoSeS.

9 The technical (algebraic) development of framework provides more rigor and insight into this and other concepts introduced in this article.

10 For example, AG43, C3-Phase 2.

11 The input structure could be two different tables, one for each product, for example. The key is the structure (no matter the number of constituent parts) is acted upon by the same throughput via unique entry points.

12 Obviously, the throughput would have different logic than would be the case if it were to support only a single product.

13 Similar to the input structure, nothing requires the output structure to be one "integral" unit (i.e., one table).

14 Attimu, Dodzi, and Bryon Robidoux. 2016. Abstractions and Working Effectively Alongside Artificial Intellects. *Predictive Analytics and Futurism* 14:18–23, *www.soa.org/sections/pred-analytics-futurism/pred-analytics-futurism-newsletter* (accessed Sept. 19, 2018).

Long-Term Care Modeling, Part 3: Model Validation

By Linda Chow, Jeremy Levitt, Laura Donnelly Knab and Yuan Yuan

The increasing complexity of models and reliance on them has been accompanied by a wave of validations. This has occurred as a result of companies recognizing the inherent risk in relying on these models, in some cases having experienced model failures, and increased regulatory scrutiny from the Federal Reserve and Own Risk and Solvency Assessment (ORSA). In addition, the expanded role of modeling in valuation for principle-based reserving, International Financial Reporting Standard (IFRS) 17 and elsewhere has contributed to the wave of validations. The primary focus of risk management—which comprises model validation—is to increase the level of transparency of what's in the model. In practice, expenditure on model validation could exceed the cost of developing the model being validated, but that does not imply that some validation cannot be done for less. What are some of the best-practice techniques to validate a long-term care (LTC) model, as part of an organization's larger risk management framework? We touch upon these techniques and discuss the answers to other questions in this article.

In the first two installments of our three-part series (published in the December 2016 and November 2017 issues of *The Modeling Platform*[1]), we compared a claims cost approach with a first-principles approach and dove deeper into first-principles modeling for LTC. In this installment, the focus is on the validation of these models. In this article, we will describe Ernst & Young's (EY's) five-pillar approach that balances practicality with comprehensiveness, and how this applies to LTC model validation.

The insurance industry continues to enhance data analytics and management reporting capabilities that lead to significantly greater granularity in respect of actuarial projection and computational models. This is particularly apparent for LTC carriers amid the financial issues that the products have caused the industry. As management increases their effort to scrutinize the financial status of their LTC block, they recognize the need to have good risk management, governance and controls around their LTC models. An important model governance step is model validation. Due to the complexity of LTC products, the assumption structure and the evolution of understanding and handling industry-wide issues such as rate increases, validating LTC models is a very involved task and requires investment from management to ensure structured protocols are followed in performing the validation.

FIVE-PILLAR MODEL VALIDATION APPROACH

The five-pillar model validation approach combines conceptual soundness, documentation/governance considerations, model performance/integrity, implementation and data quality to challenge a model effectively. These "pillars" can be tailored to the type of model and needs of the organization undergoing a model validation exercise. For example, greater emphasis can be placed on the data quality pillar if data quality is known to be poor.

A visual depiction of the previously mentioned five-pillar approach is set out in Figure 1. We apply each of these pillars to LTC model validation in the following section.

Figure 1
The Five Pillars of Model Validation

Model Validation Key Elements and Procedures
Conceptual Soundness
• Review model concept against financial, economic and actuarial theory • Check for completeness and appropriateness of inputs used to develop key assumptions • Analyze the construction of the model from component pieces and identification of interdependencies across models
Documentation and Governance
• Model risk mitigation and management • Documentation completeness
Model Performance and Integrity
• Analytical review of model output to confirm consistency with expectations • Sensitivity analysis to evaluate the stability of the model • Testing of key model drivers through attribution analysis
Model Implementation
• Independent testing of sample or full results using parallel models • Assessment of process and controls • End-user computing controls
Data Quality
• Data sources and quality review • Data quality validation and testing (e.g., data anomaly testing)

LTC CONSIDERATIONS

Both product type and organizational needs should be taken into account when applying the five pillar approach. This section details the considerations the actuary validating the model needs to take into account within the context of LTC.

Conceptual Soundness

Any model, including actuarial models, should be theoretically sound and suitable for its intended use. The main examples of conceptual soundness include assessing the reasonableness and appropriateness of the model assumptions, methodology and modeling decisions made. As part of this process, consideration should also be made to model simplifications, limitations and materiality.

There are many uses of an LTC actuarial financial model, including reserving, cash-flow projection, pricing and capital adequacy analysis. Each of these cases may have its own unique assumptions and methodology. The conceptual soundness review includes both a subjective and an objective perspective. The objective perspective looks for elements such as whether or not the model strictly follows regulatory or accounting requirements. The subjective perspective takes into account the complexity of the product design, assumption structure, management decisions and governance process, financial impact and purpose of the model.

Typical approaches to LTC modeling include first-principles and claims cost. However, there are many variations to these two approaches, including healthy claims cost vs. total claims cost, various forms of semi–first-principles models and other hybrid approaches. It's important to assess whether or not the modeling methodology is appropriate for its intended use and to make sure that the assumptions developed are appropriate for the selected model. For example, if a claims cost model was used, the validator should confirm the exposure basis for the claims cost and confirm whether or not the claims cost was appropriately applied to the right exposure basis in the model.

Typically LTC policies offer multiple contractual options for even an immaterial benefit feature. Assessments should be made to confirm whether the model adequately covers all variations of the contractual language. LTC-specific product design (base coverage, riders and special features) should be considered in validating a model.

In addition to the original contractual terms, attention should also be paid to any endorsement, amendment or modification triggered by a rate increase. If a company has been approved to increase rates, one should consider the potential implications to both assumptions and reserving under the various accounting approaches. For example, one of the rate increase options currently available to certain insurers is referred to as the "landing spot" option, where an insured life is allowed to choose an actuarially equivalent reduced inflation option instead of taking a premium increase. This option is a newly approved concept by regulators. There is a generally accepted (and regulator-approved) treatment that applies in this situation. For example, when validating statutory reserve models, one should assess whether or not the net valuation premium and benefits were properly modified to reflect the reduced inflationary benefits.

Other conceptual LTC model considerations include appropriateness of the projection period and the explanatory parameters (e.g., whether internal rate of return is an appropriate measure of profit) to make sure that the right modeling decision is made for the intended purpose.

Any model, including actuarial models, should be theoretically sound and suitable for its intended use.

Documentation and Governance

Model documentation sets out technical details to facilitate knowledge transfer and improve model transparency. It supports the proper use of model results through understanding of intended uses and model limitations, and allows independent model validators to review the model. Documentation that contains a sufficient level of details allows parties unfamiliar with the model to reproduce the documented progress successfully and thus reduces key-person risk.

Documentation should be created during model development, and reviewed and updated periodically. Although the documentation is usually tailored toward the product, business use case and modeling platform, it usually contains the following sections:

- Model purpose and use
- Key data/inputs, assumptions, outputs and process flows
- Known model limitations and risk areas
- Evidence of model validations and peer review
- User manuals, including procedures, for model use
- Upstream and downstream model and model users
- Model and documentation versioning

With these considerations, the complexity of the documentation should align with the risk and complexity inherent in the model. For LTC models, model documentation should cover

product-specific topics such as development of the claims cost tables and the procedure to update incidence rate assumptions.

Model governance plays the role of model risk mitigation and management through effective change control procedures and assumptions governance. The model owner/steward is responsible for updating the model documentation when there are significant model changes. Older versions of the model and documentation should be archived for auditability. Various stakeholders, including the end users, model development team and model steward, should develop rigorous standards with regard to documentation while leveraging actuarial standards such as Actuarial Standard of Practice (ASOP) No. 23, Data Quality and ASOP No. 41, Actuarial Communications. To align with the enterprise-level risk framework, the developed model should be evaluated against company model development standards and model governance policies.

Model Performance and Integrity

The model should be assessed for performance quality and robustness relative to expectations. This component of the validation process should assess whether the model is capable of providing timely and accurate information to the relevant stakeholders, and to gauge the level of approximations used in the model. Additional areas that should be assessed are usability of the model, the ability of the model to provide actionable analysis, the level of model automation and whether the model still performs as expected under different sensitivities and scenarios. Of importance is a review of the final output produced by the model, how these outputs are signed off and reported upon, and subsequent processes used for generating any final reports or analyses required by stakeholders.

Owing to the complexity typically inherent in LTC models, it is essential that performance quality and robustness are assessed. Sample assessment techniques include the following activities:

- Compare historical data against projected model output calibrated to the period over which the historical data applies. Dynamic validation on key LTC model cash flows may be elected, too. This entails assessing whether the LTC model results—such as premiums, morbidity outflow, lapses and recoveries—follow a reasonable trend in line with historical data.

- Vary one variable at a time (stress testing) or multiple variables at a time (scenario testing) to assess the robustness of the model.

- Stress test morbidity rates, duration of disablement and lapses, as these variables play an important role in driving claim outflow. Similarly, scenario testing on variable combinations, such as a downward interest rate and upward lapse stress, should be performed.

Implementation and Testing

The purpose of this step is to assess the efficiency and sustainability of the model implementation. This could be accomplished by reviewing implementation controls performed by the model owners, through independent recalculation of the model output. The validator will have to identify a sampling process for selecting policies to test and determine a test plan, which includes intervals to test (e.g., time zero only, every 10th projection year) and testing thresholds (i.e., allowable differences). Considerations should be given to materiality, current risk exposure and potential future risk exposure when deciding on the sample policies. While recalculation could be performed in many ways, best practice is the use of an independent "challenger" model, which is an industry-vetted modeling platform different from the platform currently used. Different independent calculators will be needed for disabled lives vs. active lives, for different accounting bases (statutory, GAAP, tax, etc.) and for different purposes (e.g., gross premium valuation reserves vs. cash flow testing). This pillar is often the most time-consuming, as the independent recalculation process should cover appropriateness of the data input and methodology, implementation of the assumptions, and having matching results. Often, model differences will likely exist between the model being tested and the challenger model. It is the role of the validator to determine if those differences are acceptable approaches in the industry or if the model being tested needs refinement.

In addition to independent calculation, the validator should also manually review formulae in the models to see if they reflect underlying theory and methodology. If the model doesn't have open source codes, the validator should at least check the organization of the models.

Sensitivity testing should also be performed to review reasonableness of the model results. LTC models typically include a wide range of modeling assumptions, especially under a first-principles approach. Separate sensitivity should be performed on at least all of the key assumptions (e.g., incidence, disabled life mortality, recovery, cost of care, utilization, active life mortality, lapses and economic scenarios). Sensitivity analysis could also be considered at a granular level so that it varies by segment and by policy or claim duration. Careful considerations should be given to the interdependencies among the assumptions when analyzing results.

Other testing techniques should also be considered and used. Similar to the techniques already discussed in an earlier section,

they may include static and dynamic validation, and retrospective testing.

In addition to reviewing the core model calculation engine, the validator should also look at controls and governance around the model. These are typically done through reviewing model documentation and management testing/review evidence, checking for user access restrictions, maintenance of change log and historical archive history. Further details in respect of documentation and governance are described in the "Documentation and Governance" section.

Data Quality

Having quality input and output data is critical, yet it is an area where many companies struggle. Throughout the data flow, starting from in-force file creation to model results compilation, there are many areas where data could be mismanaged if proper controls are not in place. Reconciliation steps should be performed during in-force file creation, and the data creation itself should be as automated as possible. Similarly, assumption updates should be automated and checklists of changes should be created and verified with department heads. Post-processing procedures are typically more manual and thus leave more room for user error. Model output edits and overwrites are sometimes necessary, but they should be clearly documented and well-understood by all essential parties. Putting appropriate controls in place around data quality helps to make the model output more reliable and meaningful.

Specifically for LTC models, data dictionaries should be available to model users. The dictionary should allow users to understand the components and complex assumptions that make up a first-principles model. Additionally, having a thorough data dictionary helps to reduce key-person risk. This data dictionary should be reviewed and approved by all relevant functional units to ensure data accuracy.

Another concern for LTC models relates to assumptions. Given that there is limited industry data for LTC products, it is often up to the company to use their own experience in developing assumptions. Many other products are able to use industry tables already built into the model, but LTC products often do not have this luxury. There is a risk that updates to company-specific assumptions are not made correctly, thus causing the model to use poor quality data. Therefore, checklists should be created for assumption updates, and controls should be in place to review the updates.

CONCLUSION

Model validation is crucial to ensure the LTC model developed is conceptually sound, fit for purposes, produces technically accurate results, and has business requirements and constraints adequately allowed for. In light of this, LTC providers should consider best practices in respect to their model validation efforts—one of which is the five-pillar approach we've highlighted. A well-documented model that undergoes sufficiently thorough testing enables users of the model to place greater reliance on the output and make more informed decisions. The benefits of ensuring a comprehensive model validation structure is in place are significant, and the implications should not be underestimated. ■

The views expressed are those of the authors and do not necessarily reflect the views of Ernst & Young LLP or any other member firm of the global EY organization.

Linda Chow, FSA, MAAA, is a senior manager at Ernst and Young. She can be reached at *lo.chow@ey.com*.

Jeremy Levitt, FSA, FIA, is a senior consultant at Ernst and Young. He can be reached at *jeremy.levitt@ey.com*.

Laura Donnelly Knab, ASA, MAAA, is a senior consultant at Ernst and Young. She can be reached at *laura.knab@ey.com*.

Yuan Yuan, FSA, MAAA, is a senior consultant at Ernst and Young. She can be reached at *yuan.yuan@ey.com*.

ENDNOTE

1 See *https://www.soa.org/sections/modeling/modeling-newsletter/* for these and other recent archives of the newsletter.

Modeler Q&A With Ryan Krisac

By Uri Sobel

Ryan Krisac, FSA, MAAA

Ryan Krisac, FSA, MAAA, is director of model controls at Penn Mutual Life Insurance Company in Horsham, Pennsylvania. He graduated from Penn State University in 2008 and attained his FSA in 2013. Ryan has spent his entire professional career at Penn Mutual, including roles in valuation and corporate modeling before assuming his current role. He co-wrote the article "Model Governance: Controls and Culture" that was published in the April 2018 issue of *The Modeling Platform* (*SOA.org/sections/modeling/modeling-newsletter/*).

Q: What kind of modeling work do you do, and what software platform(s) do you use?

A: I lead the Model Controls area at Penn Mutual. I typically do not perform a published modeling function, but instead review existing processes for improvements related to reliability, accuracy, transparency and efficiency. I also manage the company's model inventory, which rates all our models across consistent dimensions of risk and performance.

My company primarily uses MG-ALFA and PolySystems, but we still have a few Excel-based models. Almost all of my coding work is in MG-ALFA.

Q: How do you plan or prepare for a modeling change?

A: In my experience, the most effective approach has been to define the purpose of the change in plain, simple terms. With our model inventory in place, we may find opportunities to leverage existing code in order to address the desired change. Even if a modeling change is a new endeavor, a clear purpose allows more people to participate.

Q: When you find a bug that has an immaterial impact on results, can you let it go or not?

A: Timing has to be considered with any fix, no matter how immaterial. The production modeling environment should be stable and reproducible. Applying changes whenever they come up can make model results volatile and erode trust with management, auditors or anyone else relying on model stability. Because of that, I have learned to let immaterial differences go, at least until the time to apply a fix is appropriate.

Q: Documentation—as you go along or after you're done?

A: The honest answer is that the documentation tends to come at the end. At minimum, if I am changing an existing model, I try to publish a "roll-forward" of impacts that clarify my steps.

Q: Do you have a modeling pet peeve?

A: Yes, when coding lacks enough white space or indentations to make it legible. Also, when large sections of code are commented out, but never actually deleted.

> The most effective approach has been to define the purpose of the change in plain, simple terms.

Q: What's the most frequent piece of code that you can never remember the syntax for?

A: I get crossed up when reviewing commutation functions, usually within formulaic reserves. I never trust my memory on what C, D, L, M and N mean.

Q: What was the last problem you encountered that had an easier-than-expected solution?

A: Our old text-file-based model reports were read into Excel and reformatted with long, clunky formulas. We have replaced these with MG-ALFA report templates, which have been easy to set up and have simplified our processes. These reports are pulled into Excel via an add-in that can be more easily reviewed.

Q: What's something new you picked up recently that you'd like to share?

A: Externalizing valuation reserves into factor files that are fed as inputs into projection models. We have also started using this in pricing activities where we have one run to set the reserves and create these factors, and separate runs for profitability.

This has dramatically increased the speed of our projection runs, because there is no need to calculate reserves directly in those projections anymore. They simply multiply the given factor for a particular cell or policy against projected volumes. ■

Uri Sobel, FSA, MAAA, is principal and consulting actuary at Milliman in Little Falls, New Jersey. He can be reached at *uri.sobel@milliman.com*.

2017 SOA Modeling Sessions, Part 2: SOA Annual Meeting & Exhibit

By Jennifer Wang

Recorded webcasts and virtual sessions are available for a fee, but Society of Actuaries (SOA) members have free access to audio recordings synchronized with slide presentations of many of the major 2017 SOA meetings: Life & Annuity Symposium, Health Meeting, Valuation Actuary Symposium and the SOA Annual Meeting & Exhibit. Slides from meeting presentations are downloadable and free to all online.

In addition, Modeling Section members have free access to Modeling Section–sponsored webcast recordings one year after they were produced. Check them out!

SESSION 20 WORKSHOP: MULTIVARIATE SCENARIO REDUCTION TOOL USING R

Moderator: Mark M. Yu, FSA, MAAA

Presenters: Chin-Mei Yvonne Chueh, ASA, Ph.D.; Donald Davendra, Ph.D.; Tobias Gummersbach

"The technology to amass data exceeds our abilities to make use of it." People all over the globe are turning to R, an open source coding language for statistical computing. The R studio

may be applied to actuarial research and implementation with immediate applications to the current company model testing, as well as auditing tasks by insurance regulators. One example is various multivariate scenario reduction techniques being coded in R and C++ in our project to support the project sponsor and actuarial community (to empower their tool building and integrating research). The performance of R packages was compared with equivalent software packages. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-020.pdf*)

SESSION 26 INTERACTIVE FORUM: MODEL RISK MANAGEMENT

Moderator: Yimin Yang

Presenters: George Alvites; Charlie Anderson, Ph.D.; Gang Ma, FSA

One important lesson learned from the recent financial crisis was that overreliance on financial models could lead to unforeseen and harmful consequences. Model risk management (MRM) is becoming a core and mandatory component in enterprise risk management for many financial institutions, especially Federal Reserve/Office of the Comptroller of Currency (OCC) Comprehensive Capital Analysis and Review (CCAR)–compliant firms. In recent years, insurance companies are also being required to have model risk management programs by Solvency II, Own Risk and Solvency Assessment (ORSA) and other supervisory guidance. MRM is not just model validation. It consists of governance and policies regarding a model's life cycle including data, model development, model test and implementation, model validation, model monitoring and documentation. It also provides guidelines and standards for risk assessment and model inventory management. The presenters discussed current and best practice MRM, approach for model validation and auditing, as well as topics for technical issues such as types of model risks. They also discussed the differences between Fed/OCC standards and Solvency II standards. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-026.pdf*)

SESSION 31 PANEL DISCUSSION: HURRICANE HARVEY: FLOOD CATASTROPHE RISK FINANCING AND MODELING

Moderator: R. Dale Hall, FSA, CERA, MAAA

Presenters: John Elbl; Carolyn Kousky

Presenters provided an update on the impact of Hurricane Harvey on insurance markets and government programs, the management of flood catastrophe risk through the U.S. National Flood Insurance Program, and an overview of the modeling of

Modeling Section members have free access to Modeling Section–sponsored webcast recordings one year after they were produced.

catastrophic risk from industry researchers. With new reinsurance techniques and cat models evolving to manage this risk, the presenters described how modeling of the risk can be used in estimating loss costs and how public/private financing programs are evolving. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-031.pdf*)

SESSION 49 PANEL DISCUSSION: PREDICTIVE ANALYTICS ASOPS: MODELING AND SETTING ASSUMPTIONS

Moderator: David N. Karo, ASA, MAAA

Presenters: Jason Jeffrey Altieri, ASA, MAAA; Eileen Sheila Burns, FSA, MAAA; Christine Hofbeck, FSA, MAAA

Predictive analytics is increasingly used to select, design, build and modify models intended to be used to set assumptions. This is a new area of practice for actuaries, with guidance only starting to evolve, and always in tandem with more traditional actuarial applications. The proposed Assumption Setting Actuarial Standard of Practice (ASOP) will "apply to actuaries performing actuarial services which include setting and/or assessing the reasonableness of assumptions." The proposed Modeling ASOP will "apply to actuaries in all practice areas performing actuarial services when selecting, designing, building, modifying, developing, using, reviewing, or evaluating all types of models that are not simple models." The presenters discussed these two important ASOP exposure drafts and their implications for practitioners in predictive analytics. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-049.pdf*)

SESSION 58 PANEL DISCUSSION: VALIDATION OF ASSET MODELS

Moderator: Rebecca Margaret Emily Kovach, FSA, MAAA

Presenters: Daniel B. Finn, FCAS; Scott D. Houghton, FSA, MAAA; Thomas V. Reedy, FSA, FIA, MAAA

For applications like principle-based reserving (PBR) and cash-flow testing, actuaries often validate and review models that contain projections of both the asset and liability sides of the balance sheet. Actuaries are often insurance product and liability experts, but experience levels with assets vary. The presenters discussed model validation techniques that an actuary can apply to the asset side of the balance sheet. Specific topics included:

- Validation of asset cash flows, market values, and statement values
- Review of common assumptions for modeling assets
- Common modeling issues and their prevention
- Validation techniques to ensure that primary risks of different types of assets are captured
- Validation of interaction of assets and related balance sheet items
- Validation of interaction of assets and liabilities, including dividends and portfolio crediting
- Case studies with input and output

(See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-058.pdf*)

SESSION 79 PANEL DISCUSSION: NAVIGATING ASSUMPTION SETTING ACROSS VALUATION BASES

Moderator: Lisa Marie Veldman Domonkos, FSA, CERA, MAAA

Presenters: Sebastian Joseph Kleber, FSA, MAAA; Leonard Mangini, FSA, MAAA; Kevin Piotrowski, FSA, CERA, MAAA

View this session recording to understand the convergence of assumption setting across different valuation bases, including statutory (PBR), GAAP and embedded value. The presenters provided an overview of the role of assumption setting within these bases while highlighting similarities and differences. The role of credibility was discussed, with a focus on considerations and trade-offs of the two measures for PBR mortality assumption setting (Buhlmann and limited fluctuation). Monitoring and unlocking of assumptions and how this fits into the management of the business were explored. Key governance considerations were also highlighted. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-079.pdf*)

SESSION 82 LECTURE: THE ROAD TO NORMALIZATION

Moderator: Hal Warren Pedersen, ASA

Presenters: Hal Warren Pedersen, ASA; Liang Yin, FSA, MAAA

The financial crisis led to global market conditions with few parallels in the historical record. Ultra-low and negative long-term interest rates have placed tremendous stress on insurers' ability to generate yield on investments. It now seems that markets are primed to return to more normal conditions. What might this road to normalization look like, and over what time period? What are the probable events that will lead to a normalization of interest rates? What might happen if the normalization process stalls?

The presenters discussed what we can learn from the historical data about periods of normalization and what risk modeling strategies can be employed to explore the ways in which the timing and magnitude of future rate increases might affect an insurance business. Specific topics included historical experience during periods of normalization, developing views on what our current road to normalization might look like, creating robust economic scenarios for measuring the risk in the normalization process, and allowing for an economic scenario generator (ESG) to accommodate scenarios that correspond to an arrested normalization of interest rates. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-082.pdf*)

SESSION 102 PANEL DISCUSSION: IMPACT OF VM-20 ON LIFE INSURANCE PRICING

Moderator: Trevor D. Huseman, FSA, MAAA

Presenters: Carrie Lee Kelley, FSA, MAAA; William Gus Mehilos, FSA, MAAA

VM-20 has created an array of new issues and challenges for companies in pricing life insurance products. The presenters covered considerations, challenges and benefits of pricing new products under VM-20. The development of prudent estimate assumptions, including when you have limited experience, was discussed. Other topics included modeling and projecting VM-20 reserves, changes to the pricing process, emerging best practices to pricing under VM-20, and ways to address additional reserve volatility under VM-20. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-102.pdf*)

SESSION 117 BUZZ GROUP: PREDICTIVE ANALYTICS: TECHNICAL TOOLS

Moderator: Eileen Sheila Burns, FSA, MAAA

Presenters: Peter J. Horman, FSA, MAAA; Christian Klose; Benjamin Williams

Practitioners from health, life and annuity product areas gave short presentations on a range of technical tools used to perform predictive analytics. Tools covered mathematical models, data management and agile workflow. The presenters adopted an Open Spaces format for the remainder of the session, and broke out into discussion groups to allow attendees to ask questions and share knowledge about the tools in which they are most interested. This was an opportunity to share use cases, best practices, frustrations, ideas and interact with others facing similar situations. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-117.pdf*)

SESSION 122 PANEL DISCUSSION: LIVING TO 100: MORTALITY MODELING

Moderator: R. Dale Hall, FSA, CERA, MAAA

Presenters: Andrew Cairns; Stephen C. Goss, ASA, MAAA

The presenters gave attendees an educational overview of mortality modeling and an update on new mortality modeling techniques being researched, developed and implemented. Highlights included results emerging from research projects using a variety of familiar population and insured mortality data from the United States and around the world. Specific focus was given to how these techniques and findings can be implemented in actuarial practice. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-122.pdf*)

SESSION 131 PANEL DISCUSSION: TRANSFORMING MANAGEMENT INFORMATION

Moderator: Timothy P. Noble, FSA

Presenters: Hunt M. Blatz, CFA; Mustafa B. Dinani, FSA, MAAA; Gavin Lubbe

Insurance companies produce a wide variety of data on a continuous basis to support business decisions and produce required regulatory reports. The intense level of change in the financial management of insurers in recent years has led to a demand for intelligent interpretation of the large amount of data and reports available. The presenters explored value-added functionality (data visualization, drill-down capabilities, etc.) to aid in the review of results, highlight what is driving the movement of results, and enhance the ability to communicate and explain results. In addition, the presenters discussed best practices for the platform to control and store the underlying data. Actuaries who review or communicate results gained insight into available tools in the marketplace and how to determine what underlying data requirements may be necessary.

The session did not endorse a business intelligence tool but rather shed light on what is available and how companies have put these tools into practice. Many companies are performing transformative activities, and the goal of the presenters was to critically think about and discuss the architecture of the end reporting stage. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-131.pdf*)

SESSION 157 PANEL DISCUSSION: "ONE MODEL CONCEPT"—END GOAL OR GUIDING PRINCIPLE?

Moderator: Dave Czernicki, FSA, MAAA

Presenters: Hunt M. Blatz, CFA; Dave Czernicki, FSA, MAAA; Joseph P. Peterson, ASA, MAAA; Stephen T. Verhagen, FSA, MAAA

The presenters renewed discussion on the prospect of the "one model concept," or the ability of an insurer to consolidate their model set down to a single set, platform or even a single model. Presenters defined the one model concept and discussed industry trends in the space, including efforts that are making the vision a possibility (such as vendor-provided platform developments, the transformation agenda, operating model evolution), and challenges standing in the way of the vision (such as regulatory changes, expanding demands on financial models). The presenters shared lessons learned from industry participants. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-157.pdf*)

SESSION 158 PANEL DISCUSSION: LIVING TO 100: MODELING OF MORTALITY IMPROVEMENT

Moderator: Andrew J. Peterson, FSA, EA, FCA, MAAA

Presenters: Elena V. Black, FSA, EA, FCA, MAAA; Marianne C. Purushotham, FSA, MAAA

The techniques used to project mortality improvement have changed dramatically over the past two decades, evolving from simplistic "age-only" scales to today's complex multidimensional models. In addition to providing an overview of mortality improvement models currently used by actuaries in the United States, United Kingdom and Canada, the presenters discussed some of the challenges faced by those trying to assess the effectiveness of the most widely used models. As available, findings related to the most recent updates to the Retirement Plans Experience Committee (RPEC) 2014 model, and its successive iterations of corresponding mortality improvement scales were discussed. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-158.pdf*)

SESSION 173 PANEL DISCUSSION: PENSION MODELING AND SCENARIO ANALYSIS

Moderator: Brett Brooks Dutton, FSA, EA, FCA

Presenters: Matthew W. McDaniel, FSA, EA, FCA, MAAA; Thomas William McNab, FSA, EA

Pension modeling, the development of potential future financial outcomes for pension plans and their sponsors, is a critical component of the value that actuaries provide to their pension clients. Pension modeling is used for many purposes, including budgeting, strategic planning and risk management. The presenters sought to survey the wide and evolving range of approaches to pension modeling, addressing such questions as:

- For what purposes should models be constructed? What range of modeling approaches exists, and what factors should be considered when selecting a modeling approach?
- What are key considerations in model construction (e.g., assumption setting, liability forecasting, and asset forecasting)?
- How can an actuary effectively communicate key model outcomes to a client?

Although the presenters' primary work experience is with U.S. corporate defined-benefit plans, the content of the session was general in nature in order to offer relevant insight across multiple regulatory frameworks (private/public sector, U.S./Canada, etc.). (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-173.pdf*)

SESSION 174 PANEL DISCUSSION: NEWLY PROPOSED ASOPS: PRICING, MODELING AND SETTING ASSUMPTIONS

Moderator: David C. Armstrong, FSA, MAAA

Presenters: Nick Fiechter, FSA, MAAA; Maria Rose Itteilag; Donna Christine Megregian, FSA, MAAA

The panel discussion covered three important Actuarial Standard of Practice (ASOP) exposure drafts titled Pricing of Life Insurance and Annuity Products; Modeling; and Setting Assumptions; recently proposed by the Actuarial Standards Board. The draft Pricing of Life Insurance and Annuity Products ASOP "applies to actuaries when performing actuarial services with respect to the pricing of life insurance and annuity products, including riders, that will be sold in the future." The proposed Assumption Setting ASOP will "apply to actuaries performing actuarial services which include setting and/or assessing the reasonableness of assumptions." The proposed Modeling ASOP will "apply to actuaries in all practice areas performing actuarial services when selecting, designing, building, modifying, developing, using, reviewing, or evaluating all types of models that are not simple models." Actuaries use numerous models that have various applications (e.g., economic capital, GAAP reporting, pricing, etc.). It's important that the use of assumptions is appropriate in light of the model's intended purpose. Focused topics of discussion addressed what these newly proposed ASOPs mean for the actuary. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-174.pdf*)

SESSION 176 PANEL DISCUSSION: EMERGING TRENDS IN MODEL RISK MANAGEMENT FOR SMALL COMPANIES

Moderator: Vikas Sharan, FSA, FIA, MAAA

Presenters: Brody D. Lipperman, FSA, CERA, MAAA; Stefanie J. Porta, ASA, MAAA; Vikas Sharan, FSA, FIA, MAAA

The presenters discussed some of the latest trends in model risk management (MRM), with a focus on companies that are not required by regulators to validate their models. The session started with an overview of typical MRM approaches used by Fed-regulated organizations (which are required to undertake frequent validation exercises). The presenters then discussed constraints and challenges faced by small to mid-sized companies when it comes to MRM. The session offered perspectives on various aspects of MRM, such as resourcing, validation approaches and independence of roles, that these companies can utilize in their organization to build upon a successful MRM function. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-176.pdf*)

SESSION 178 PANEL DISCUSSION: INDIVIDUAL LIFE MORTALITY EXPERIENCE STUDY

Moderator: Mervyn Kopinsky, FSA, EA

Presenters: Brian D. Holland, FSA, MAAA; Kevin P. Larsen, ASA, MAAA; Tony R. Phipps, FSA, MAAA

The SOA Individual Life Experience Committee presented the results of its latest mortality experience study. This study covers experience between 2003 and 2013 on fully underwritten life insurance policies. The results included experience split along multiple dimensions including duration, face amount bands, smoking status and underwriting class. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-178.pdf*)

SESSION 191 LECTURE: USE OF CATASTROPHE MODELS BY GENERAL INSURANCE COMPANIES

Moderator: Anthony E. Cappelletti, FSA, FCAS, FCIA

Presenters: Alan Frith, ARe, CCM, CPCU

There exist catastrophe models for a number of natural and man-made catastrophes. Primarily, the output produced by these models is used by general insurers to determine their exposure to catastrophic events and assist the insurer in managing this risk (e.g., optimizing reinsurance). Reinsurers use the output to price reinsurance that covers catastrophes. However, the output from catastrophe models can be used for other purposes. During this session presenters explored the use of catastrophe models by general insurance companies beyond their use as a tool for catastrophe reinsurance. (See session slides at *https://www.soa.org/pd/events/2017/annual-meeting/pd-2017-10-annual-session-191.pdf*) ■

Jennifer Wang, FSA, CERA, MAAA, is an actuary at Milliman. She can be reached at *jennifer.wang@milliman.com*.

SOCIETY OF ACTUARIES®

SAVE THE DATE

ReFocus Conference
March 10–13, 2019 • Las Vegas, NV

Life and Annuity Symposium
May 20–21, 2019 • Tampa, FL

Health Meeting
June 24–26, 2019 • Phoenix, AZ

Valuation Actuary Symposium
Aug. 26–27, 2019 • Denver, CO

SOA Annual Meeting & Exhibit
Oct. 27–30, 2019 • Toronto

Learn more at ***SOA.org/Calendar***

SOCIETY OF ACTUARIES®

475 N. Martingale Road, Suite 600
Schaumburg, Illinois 60173
p: 847.706.3500 f: 847.706.3599
w: www.soa.org

NONPROFIT
ORGANIZATION
U.S. POSTAGE
PAID
SAINT JOSEPH, MI
PERMIT NO. 263