2001 Valuation Actuary Symposium
November 29–30, 2001
Lake Buena Vista, Florida

Session 19PD
General 'Modeling' Techniques

Moderator: Rebecca L. Burton
Panelists: Rebecca L. Burton
Michael A. Harris
Patricia L. Renzi

Summary: Actuaries today use models in all but the most simple of applications. Such models must both accurately portray the assets and liabilities in question and produce relevant and meaningful projection results. The panel discusses important considerations in:

- *Proper organization of a model*
- *Structuring of a model for an intended purpose (i.e., pricing versus financial reporting, statutory versus GAAP accounting)*
- *Static and dynamic model validation techniques*
- *Data integrity*

MS. REBECCA L. BURTON: My co-presenters are Mike Harris and Pat Renzi. We will be presenting general modeling techniques. Mike Harris has nine years of modeling experience. He received his FSA in 1997 and started his career in product development at ManuLife. He then programmed for a commercial software company, SS&C, Inc. He dabbled in consulting and worked on various modeling projects involving cash-flow testing, financial forecasting, actuarial appraisals, and demutualizations. He is currently a product development actuary with ING's investment products group.

Note: The chart(s) referred to in the text can be found at the end of the manuscript.

Pat Renzi has been working in the insurance industry for more than 20 years. She started as an actuarial student and moved into actuarial systems, where she has been involved in designing, supporting, and building commercial modeling systems for more than 13 years. Her insurance company experience has been with GEICO, CIGNA, and Reliance Insurance. She spent about eight years at Chalke, Inc., and now she is a product manager of Alpha at Milliman U.S.A.

I've been working in the insurance industry about seven years and have recently received my FSA. I started as an actuarial student with Equitable, now AXA, and after a few years, I moved to Tillinghast–Towers Perrin and have been working on a variety of projects, mostly involving modeling. These projects have covered appraisal, demutualization, and embedded value work.

We will begin with our first speaker, Mike Harris who will tell us about model structure.

MR. MICHAEL A. HARRIS: I was a little perplexed when I first looked at the description of this session. I couldn't imagine the idea of actuaries with no modeling experience. I'd be surprised if anybody in the room hasn't had some exposure to modeling in one way, shape, or form over the course of their actuarial work. I hope I've struck a reasonable balance on the level of difficulty that I'm going to present to you. I hope you'll come away with something of value at the end.

I'm going to go through the model structure. I'm first going to talk about model purpose and the basic types of models that actuaries most commonly use. We'll then get into a little bit of how those different purposes will affect the structure of the model, and the way that you build and use those models. Finally, I'll touch briefly on the data structure, a topic for which Pat and Rebecca will give more detail.

The first type of model I want to talk about is a pricing model, which generally has one of the simpler structures in terms of common actuarial usages of financial models. Typically, it's a bunch of contracts that are being sold all at the same time. You don't have to deal with in-force

contracts and a lot of data. There are often simplified asset assumptions. It could be as simple as all cash flows earn 6% or something. Depending on the type of product, there might be more complicated assets in there.

The focus is typically on the details, policy mechanics, and the performance of the contract and policyholder selection. When pricing models, you're designing a product, tinkering with it, building it, and really stress testing it. You're looking for ways that policyholders might select against you. If there are holes in your pricing, you really want to make sure that the exact mechanics of the liability itself is functioning properly. So your focus is often on getting those mechanics to work properly. In light of that, the liability characteristics will determine where most of your effort is going to be warranted in terms of building that model.

Let's move on to financial models or asset/liability models. These generally are a little bit more complex than pricing models, and most of that complexity arises from the existence of in-force liabilities. You must extract huge amounts of data about the population of liabilities that you have on the books and somehow get that into your model. The same thing goes on the asset side. There is simply not as much data, but you will have to look at the assets that you're holding and model those as well. More often this is the case with pricing models, there will be multiple product lines. In the pricing model, you might price multiple product lines simultaneously, but I would think, in most pricing instances, there would be only one product line. In most financial models, there are probably multiple product lines. You might be modeling a whole company or a whole business unit or something of that sort.

Additional input they would require in a financial or ALM model would include items at the corporate level, which can be ignored in simplified pricing models. These include things like your GAAP/DAC balances, your investment maintenance reserve (IMR) or asset valuation reserve (AVR), the patterns over which they will amortize in the future, and tax positions that the company might have.

In the financial model, the focus is generally on the aggregate financial results. In a pricing model, you want to make sure you have all the proper mechanics on the contract. In a financial model, that might not be very important to you. In fact, you might take an entire product line and just map it into a different product, if the size of that product is insignificant. The focus is more on whether the aggregate results are reasonable? Do I have my revenue streams and magnitudes correct? In an ALM model, the focus is more on the interaction of the assets and liabilities and the range of possible outcomes that could result; it's the level of risk that you're taking on.

Another type of model that is quite a bit different than the first few models that I talked about is an actuarial contribution model to use in a demutualization. In this type of model, you have a mutual company that's going public. You have this huge pie and you have all these owners that own the pie. The job of this model is to cut that pie up into pieces in a fair and equitable way. It's essentially an asset share model. It goes back in history to the issue of each individual contract and tries to replicate what has happened in history. It then continues into the future. You have to go back and look at many historical events that have occurred. For example, let's say that nobody in your in-force population died or lapsed, so there's no mortality or no lapsation in the history. There are things like tax changes that have occurred and dividends that have been declared. It's details that might have changed, and you might have to go back 40, 50 or 60 years in some of these models.

Another thing that's interesting in these models is when you have an extreme outlier. For example, you might have one policy with a $1,000 face amount that was issued in 1915 and is still on the books. The next oldest policy might be issued in 1928, for example. In a financial model, you normally wouldn't care about that one contract in 1915; you would just map it into a different issue year or something. In this case, you probably have to set up a model point, and you'd have to model that point explicitly because the contribution of that cell could be a lot different than the contribution of the other cells around it. You don't want to have to extrapolate. You want to make sure that everybody's fair contribution is measured accurately.

The accounting framework that you're working in can also have an impact on the way you design your model. The accounting framework doesn't really affect the underlying economics, the way the liability reacts, or the way the cash flows work. The basic economics of the model are unchanged by the accounting framework. What the accounting framework changes is the way that you put information into your model, and the way that you get information out of your model. A couple of examples of that are, if you're doing a GAAP model, for example, you might have to model every issue year because of the different commission rates. The result of that is you have different DAC balances and amortization schedules. If you were looking on a purely economic basis, you might not care about some of the smaller issue years, for example.

Another impact is if you're doing a legal entity or a business unit within a larger corporation, you might have to separate your liabilities into groups in ways that you might not otherwise have done. This would be done just so you can get output out of the model that is broken down by business unit or by legal entity. The accounting changes might be done externally to the model. If you're using a commercial package, you might do a statutory run, and you could even bring those statutory values out and modify them and turn them into GAAP results or some other basis, if needed. Again, this just emphasizes the fact that the underlying projection itself is unchanged by the accounting framework.

Let's move on now to get into the model structure. I'm going to talk to you a little bit about what I'm calling a modular model structure. By modular, I mean that you essentially break the model up into pieces and build them separately. You put them back together to make the whole model. Anybody who has used some of the commercial packages that are available out there is probably familiar with this design because that's the way all the commercial packages (at least the ones that I've seen) set themselves up. There's a very good reason. There are a number of advantages to the structure. The model inputs are separated from the calculation itself or what you might call the calculation engine. It's important to keep that piece separate from your input so that you can limit the number of people that have their hands in the engine and the number of

times that calculation engine is being modified. You want to be able to control that and make those changes relatively infrequently. Having this type of design allows you to break up the work effort. If you've broken the model up into pieces, you can hand off one piece to one person and another piece to another person. One person can be responsible for getting all the product details correct. Another one could be responsible for getting the economic scenarios correct. It allows you to break the work up and share it across people. It also allows you some flexibility to expand the model down the road. You simply build more pieces and add them to the model. It also allows you to share the model more easily. A good example of that is how a product development department might design a new product, and, a few months down the road, the financial area has to forecast that plan because this new product is now being sold. You can take that new piece (the product information) out of the product development pricing model, and you can hand that piece to the financial area. They can incorporate it into their model, which might have other components that are different than the pricing model. This structure also gives you a tractable audit trail, and it allows you to reproduce your results down the road.

It's also easier to maintain, which might seem hard to believe when you're first setting it up because it is a little bit more work to get organized and establish this type of structure. What happens in terms of maintenance? If something changes or something needs to be changed, it's easy to identify where that change occurred or needs to be put because you have all these separate pieces. You can isolate it more easily.

Chart 1 is sort of a pictorial representation of a common structure for a pricing model. The calculation control is really the heart or the brains of the model itself. The calculation control is the piece that pulls everything together and makes it work. That's the place where you tell the model, "Hey, I want to run product A or I want to run product B. I want to project this thing out for 20 years," or "I want to project it for 30 years." It's really all the controls that you want it to run. Basically, the start button is in there, too. It's your interface for really running the model. All the pieces around the calculation control with arrows pointing into it are essentially data input. This is breaking down the data input into pieces. It is what I'm talking about in terms of modular design.

Investment strategy will also include disinvestment strategy or what assets to buy and sell. That's one set of input. That may point to another file that has in it the definitions of what those assets are. The investment strategy might say, "Buy corporate 10-year bonds." This definition file tells it what that bond is and what characteristics it has. The economic scenarios can tell you any variables you need, such as the yield curves, equity scenarios, or inflation rates. On the liability side, I've broken that down into a few pieces. The population that you're running is: Do I want to run a single age? Do I want to run ten different ages? What sales mix am I assuming those different ages are going to have? What product features are those people going to select? That file relies on product data so it would point to another file that describes exactly how that product works? What sort of loads does it have? What fees? What sort of bonuses and guarantees? What reserves does it have? All that information is stored in the product file. In the example I gave earlier, in which you had to give this information, the pricing area has developed a new product. They probably hand that piece over to the financial area down the road.

Assumption tables allow you to pull your tables out of the product file, so it could be lapsation tables or mortality tables or any sort of standard tables, which can be stored separately. The integrity of those tables can be more easily maintained because there is only one copy of them and everybody uses the same copy. If they need to be updated, you can just update that one copy and automatically update all your models.

The box in the bottom left corner is policyholder behavior. I can't think of any commercial applications where I've seen that piece separately identified, but there's no reason why you couldn't break that out, if that's a particularly important component of the product that you're trying to price. Policyholder behavior means things like the tendency of people to lapse or not lapse, depending on how you set your credited rates or depending on the position of their guarantee relative to their account value. A variable annuity is an example. All these inputs are brought together by the calculation control in the middle. That information gets sent down into the calculation engine, and the calculation engine, in turn, produces the output that you use.

In a financial or ALM model (Chart 2), you essentially have the same structure, but it's more complex because there are more moving parts, and there are more data elements involved. In particular you have in-force liabilities and in-force assets. You probably have multiple product lines, too. I put in a second product in this example: data items. You're going to have to go out and get some more information. You're going to have to go to your mainframe system and get an extract of what your liabilities are and what your assets are. Then you'll have to talk to your financial people and find out what your IMR balances and AVR balances are for the corporate data.

I'd like to discuss some practical considerations as you go through the process of building models. Are there any steps that are kind of like recipes or that follow a distinct pattern? Say that you're doing a quarterly financial model. You have to do the same steps every quarter. If there are any pieces that you can automate, it's probably worth the time you spend to automate that process for a number of reasons. You *will* make input errors, or somebody working for you *will* make input errors. It's bound to happen sooner or later. If you automate that process, you reduce the chance of that substantially. Final results are seldom final. You probably have to do the process over again, so it's a lot better if you can just push a button to do it.

As you're building your model, you want to organize your output in a comprehensive manner, and really think about how you lay that output out and how you're going to use that information. A good idea is to include some of your inputs in your output. I don't know how many times I've looked at output, such as an income statement or something that you've just run from some model, and scratched my head wondering whether I flicked that switch in the calculation control model or turned on some feature. You wonder whether it was turned on or off. You don't know so you end up running it again. If you have all your input and output stored, then, even a year later, you can go back and see if that switch was turned on or off.

If you're doing stochastic modeling, that has another dimension. You can create reams and reams of output once you start doing scenarios. If you have 100 or 1,000 scenarios or even more in a stochastic model, there's a lot of data there. You really have to start thinking carefully about how you're going to manipulate that data or and how you're going to be able to use it in a

meaningful way. You might even want to write the information out in some sort of database format.

Another important part of the modeling is validating your results. Pat's going to talk a lot about that. You want to look at them both on a static basis and a dynamic basis.

Custom in-house systems. In-house systems are systems that you build yourself as opposed to taking a commercial package. It's kind of the age-old trade-off of having the flexibility that you want and being able to build things exactly the way you want versus the overhead and the maintenance of continuing to maintain that system. The most important piece of advice I can give you, if you're going to take on the task of building your own system, is that you should document. It's hard to do and actuaries are historically bad at doing documentation, but it is really an important part of it, especially in terms of the source code and the model structure. You want to be able to figure out what that program is doing, especially if one person is building it and he or she leaves the company. You'll have this black box, and you won't know how it works. You want to try to avoid giving all the knowledge to one person. It's kind of a bad situation to let yourself get into, so force yourself to document what's going on.

Another important thing is that you don't want any model parameters in the source code. In the first job that I worked in, we had the situation where we had these pricing programs. If you wanted to change to a lapse sensitivity, for example, change your lapses and reevaluate the product. You went into some function inside this APL program and dug down to line 372. You changed the one to a two in the vector, and then you saved the program and ran it again. Basically, we couldn't reproduce what we had run yesterday, let alone what we ran the year before because the calculation engine itself was changing from day to day. You could probably find ten different versions of that work space all doing slightly different things on each person's computer. There was really no audit control or anything. You want to take all those inputs out of the calculations and keep them in separate input files. You should be making changes to the calculation engine infrequently, and it should be controlled by one or two people.

If possible, it's good to set up a familiar interface in a modular design. We're currently using a custom in-house system in our department. We've built a calculation engine, which is pretty complex, and it can be pretty intimidating to new actuarial students that we bring to the department. We've put in an interface that's in Excel. It has this modular design so it's something that the students are comfortable with. They can go in and set the parameters in Excel. They know how to use Excel, and they don't have to know anything about the source code that it's written in. They don't have to look at that. It's just an engine that runs and executes what they tell it to do, but it gives them a comfortable way to work and the output comes out for them in Excel, too. It's a comfortable interface for them on both the front end (where they are setting the inputs and running the model) and on the back end. It takes a little bit more effort to build up that interface initially, but once it's there, it gives you a lot of flexibility and allows new or novice users to be effective and use your system.

Of course, you want to validate your calculations against some independent source, whether that's another commercial system that you have or other administrative systems or quotation systems. Use what systems you have available to you that you can use to independently verify what you're doing. That's always a wise thing to do. You want to revalidate it when you make changes. If you change that calculation engine, you probably want to have a standard set of cases that you run through and make sure that you didn't bust anything while you were making your improvements.

Policyholder behavior, which I touched on earlier, is probably the most difficult part of modeling for actuaries. It is for me. The reason is because you are trying to model decisions, and the decision function being made by policyholders is not a precise science. An actuary likes things that are precise. There's no right answer in this. When you build the model, you can't say I have the right answer and feel good about it, because you don't really know what the answer is. There's a whole range of outcomes. You can put two policyholders side by side, and they might make different decisions in the same situation.

Some examples of policyholder behavior are lapsation or segregated fund benefits or some of the living benefit guarantees in the U.S. You might have reset provisions. The policyholder might have to elect when to reset his benefit, or he might have to elect to withdrawal or annuitize. An important consideration here is that the formulas can have rational (from an actuarial point of view) and irrational components. It's a decision to lapse or not when you have a substantial guarantee in your product. If the policyholder lapses that product, that might seem irrational from our point of view. He might have perfectly good reasons for doing it. It might be perfectly rationale from his point of view, but in terms of our financial analysis and evaluation of that, it might seem irrational. You want to have a formula that kind of moves in the right direction and generally does rational things but also allows you some random noise. By the end of the day, you're probably going to have to explain this to your senior management. You can spend a zillion hours building formulas and trying to model the human mind. In the end, you could be sitting in front of your manager trying to explain why you have an arc tangent in your program. The reality is, you might build these very complex models of behavior and have a super rational model. You might then assume 50% of the people act this rationally, and the other people do something random. You're kind of building a lot of precision in there. In the end, you are making an arbitrary decision anyway, so why waste the effort from the beginning? Try to keep it simple and look at the outcomes from your model. In particular, you might do a stochastic model and look at it by percentile. That's the way I present this information to management in my company. I'll run through a whole set of stochastic scenarios, and I'll look at the lapse patterns by scenario. In the 95% worst case duration, our persistency is 25%, but in the best case, it's 50%. That's the way you kind of get comfortable with it. It seems like a reasonable amount of policyholders would leave.

An audit trail is an important thing to build into your model. You want to be able to reproduce your results over time. You also want your results to be reproducible by any team member. In order to do that, you probably want these models to be in a standard location or a standardized file path, including your data tables. I mentioned mortality tables. You can put those in a standard place. You can make one person responsible for the upkeep and maintenance in

updating those files. That way, any person on your team can reproduce results that were run last week or last year. If you have particularly important runs, like year-end results, or if you're doing an appraisal or something like that, you probably want to archive those exact models, including the calculation engine. If a question arises in a litigation, or in the case of an appraisal or something like that, you want to be able to go back and see exactly what you did, how you came to the conclusions that you did, and what were the exact calculations that were performed.

Let's talk about data structure. When you're wanting to set up a pricing population, you're really trying to make your best estimate of what your perspective sales distribution is. The first place you're going to probably start looking is your recent sales distributions. What information do I have about my existing in-force policyholders? You can't stop there though because you need to consider what's different about this product or about the environment that will change some of the data that I've seen in the past? For example, say my old contract was issued up to age 90, for example, and my new product is going to be issued up to age 80. I'm probably not going to expect an 85-year-old in the new contract, so you must take things like that into account. If you're going to change your compensation pattern, you must figure out how that is going to affect sales? That's kind of where actuarial science becomes more of an actuarial art. What is tweaking things going to do to the sales pattern?

When you're looking at your in-force data, you want to look at key assumptions both in isolation and combination. An example here is, if you look at your population, 60% of them are male and 30% of them are smokers. That doesn't necessarily mean that 18% of them are male smokers. You might go into your population and find out that all your smokers are female, which means there might have been an error in coding the sex on your quotation system. There's an explanation for that. You can find things out by sifting through the data and looking at them closely. Don't just presume that these things are multiplicative.

In a financial model, you're trying to build a set of model points that will produce a reasonable representation of your liabilities in total. In order to do that, you identify what the key characteristics are of those liabilities that are going to have a significant impact on your model. You isolate those characteristics, and you start grouping together the particular liability points into model points. This is called mapping or model building, and it is a phrase that's commonly used. The real trade-off here is between run time and model accuracy. You could just take your seriatim listing and run each policy through one at a time. It might take your model a week or a month to run if you did that, so the main reason is to try to make the model more efficient.

Again, I'll take a look at that same example where we have 60% males and 30% smokers. You could just assume that 18% of your population is male smokers. What happens if you just start multiplying things like that? You'll end up with a large number of data points, and you'll end up with a lot of data points that have a very insignificant weight. In this case, 18% are male smokers. If 1% of my population was issued in 1985, for example, I'll have less than 2% in the male smoker 1985 cell. As you multiply things, they'll keep getting smaller and smaller, so you might end up with a lot of these really insignificant model points that aren't going to have any real impact on your results; they're just going to waste run time.

A better way to go through your data and to do that mapping is an asymmetric mapping. There are some commercial packages out there that do that. I've used the one from Tillinghast, and it seems to work very well. You also learn things about your population as you're going through the process, too. It's pretty interesting. It allows you to go through your different plan codes and isolate your characteristics in terms of model points that have a significant weight. If I'm looking at all my male nonsmokers, it will show me how much weight I'm looking at. If that's only 1% of the population, I'll just lump all my issue years into that cell, for example. I won't spend a lot of time on it, but if it's 50% of the population, I'll break it down more. I'll break it down by other characteristics. So, by definition, what you end up with in that situation is every model point has a significant weight because you chose it that way. If it doesn't have a significant weight, you would have just lumped it in with something else. You'll end up with fewer model points, and going through that process will give you a better model validation and faster run times.

Asset modeling can be done seriatimly, which is typical. There's usually a smaller number of points there. There are less assets than there are individual contractholders. I think it's fair to say that they're generally more homogenous than liabilities. You can map assets in the same way by looking at the key characteristics, too, particularly if you had vanilla bonds that you just wanted to lump together. That's an obvious way you could lump things together and get reasonable results.

That concludes my section of the presentation. I'll hand the mic off to Pat.

MS. PATRICIA RENZI: Mike did a good job of discussing the model structure and model construction. He said that you're absolutely going to have errors in your model, and that's true. I have never seen anyone build a perfect model, and that's why validation is such a critical part of modeling. You really need to make sure that you have a good process in place within your organization to make sure that you are consistently and regularly validating the models that you're building.

Mike talked about model structure and different ways to build your model. He mentioned the importance of documentation and defining the process for grouping. All of that is important because it leads to an easier and better validation process.

What needs to be validated? One thing is the data integrity, and that's what Rebecca is going to talk about. Before you can even get to a point where you're validating a model, the data that will be feeding that model need to be validated. You need to make sure that you're not bringing garbage in, but for now, let's assume that this has already been taken care of. Once you have your model built, you first need to validate the model fit. If you have grouped your data together (for example, grouping liabilities by ages, plan, and so on), you need to make sure that in doing that, you still have a good model fit.

A second thing that you need to make sure that you validate is input values. There are two different kinds of input values that need to be validated. The first is the structural data. There are things like tabular data, cost of insurance charges, valuation interest rates, and so on. All of

that obviously needs to be validated. You need to make sure that it is readily accessible, and that it is easy to audit that information.

In addition to the structural data, you also need to make sure that you're validating your subjective data. This is a little bit more difficult. Information like policyholder behavior or just the basic lapse rates that you're using is not something that you can go to the contract or policy specifications to determine the lapse rate to use in your model. Validating that information is obviously a little bit more difficult and subjective, but it is also a very important part of the model validation process.

Finally, it's critical that you validate formulas and calculations. Whether you are using an in-house system or a vendor system, it is your responsibility to make sure that the calculations are doing what you expect them to do for the particular product that you're modeling.

What does the validation process involve? First, it is critical that you have reliable data to be validating against. You need to plan ahead and determine what information is accessible that you can validate against. When you are constructing your model, you should consider what you need to do your model fit test. If you have your company's internal financials and the breakdowns available for reserves, cash values, premiums, and so on, and they are available by legal entities line of business and plan code, you want to make sure that you construct your model to match your information. In other words, you want to construct your model to match your company result. Make sure that your model is going to match what you're validating to.

Obviously spot-checking inputs and outputs is necessary. Reasonableness tests are another tool that you have available to you for validation. Use your knowledge of the products being modeled by asking questions such as, "As I project forward, what does the ratio of the account value to the cash value look like? What do the reserves to the cash values look like?" Look at the progression that you have in the model, and make sure that makes sense on the asset side. Look at book value to market value to par value and make sure that those relationships, under different

scenarios, are fitting together and make sense. You need to make sure that you have thought about what those reasonableness checks are, and have designed your output so that you have access to the information to do the analyses.

Mike talked about documentation, and I can't stress that enough. There are a number of different things that need to be documented. Documentation and peer review go together because it's critical to have someone else look at your model. You can look at your model over and over again, but a fresh pair of eyes can catch things you will not. You might spend days validating your model. You feel confident that everything is correct. You might take it in to your boss and say, "Here it is," and in two seconds he'll say, "This doesn't make sense." When you're really involved in the project, you just don't see certain things, so a peer review is essential. To facilitate a productive review, documentation is critical. Documentation also certainly makes it a lot easier for someone else to pick up a model and to use it for another purpose. Mike talked about moving pricing models into the forecasting area. If you haven't documented why you made the decisions that you did, that's going to be very difficult to do. As much as people loathe documentation, that is a critical piece of a good validation process.

Model fit is really a static validation. You're trying to make sure that the modeling or grouping assumptions that you have made are going to produce appropriate results. Some of the things that you're going to look at are: reserves, account values, cash values, premiums, book and market values, policy count, face amount, and loan balances on different kinds of products. Those are all pieces of information that are available from your valuation system and administration systems. You want to make sure that once you've grouped things together that you're able to validate against them. Don't do your validation in aggregate only. You want to be able to look at individual cells, plan level validation, and legal entity validation. You also want to be able to look at the final aggregation to make sure that everything has been included.

Valuation of input values is self-explanatory, but the big key here is that part of your documentation needs to be complete documentation of all of the input assumptions. You must know what they are and where they came from. Somebody needs to go through and spot-check

for both accuracy and appropriateness. It's tedious, but it's obviously a necessary part of the process. You want to make sure that you have a good process in place to be able to review the data and make sure that you understand and can reconcile the information that has been input.

Part of the data validation process is a little more complex than validation of data entry, and this is assumption validation. First, you're going to want to do a static validation, and after you finish the static validation, you will need to do a dynamic validation. There are a couple of things you can do for the static validation. One is to validate cash flows to recent quarterly earnings. Another option that you might want to explore for *FAS 97* products is validating against the GAAP valuation system. Details on COIs collected, surrender charges, interest credited, and so on will be available from the GAAP valuation system. Look at your model to determine what you're generating for those items in your model and validate against the GAAP valuation system. Make sure that you have a good model fit of the assumptions for the progression of information.

Another thing to validate against is illustration systems. Look at individual policies and generate an illustration through your model. Validate this against your illustration system. That's going to give you a good validation that can be used to make sure that all of the mechanics are correct as well as the data inputs for the product features.

Reasonability ratios are valuable when looking at the progression of values beyond time zero. For asset holdings, you should make sure to look at the book and market values that are calculated by the model, and check those against actual values over the past several months. You can also look at these values going forward to make sure that, in a static interest rate environment, the relationship is holding up the way that you would expect it to.

The next piece is the dynamic validation. This is needed to make sure that the assumptions that you have in place for things like policyholder behavior, debtor behavior on the asset side, and corporate behavior for interest crediting strategies are reasonable. The difficulty here is that there's nothing to validate against. As Mike said, it's really something you get more of a feel for.

Determining what policyholder behavior is going to be is far from an exact science. You want to make sure that the assumptions and the calculations in your model are holding together and making sense under different economic environments, if interest rates are driving the dynamism.

There are two different things that you're going to want to look at. First, just pick a set of deterministic scenarios so you know what should happen. For example, if you have a scenario where rates go up, the callable bonds should not get called. But in an interest scenario where rates go down, you should see more calls. You want to make sure that the basics are holding together. You have to have a sense of what the magnitude of those changes should be. The same thing is true for policyholder behavior. The other thing with policyholder behavior is to just look at some ranges.

You're going to want to do a lot of sensitivity testing on behavioral types of assumptions. Even though it's not an exact science, you should know what your expectations are and verify that this is what happens, but make sure to test boundaries. For example, you might decide a particular UL product is not very sensitive, but you should test it to see what happens if it is extremely sensitive or completely insensitive. It is critical to know the possible range of results to understand how critical that assumption really is to that particular product.

Finally, stochastic scenarios are another option to understand the financial implications of particular assumptions. Run your model across a set of stochastic scenarios and look at the mean and the variance. This is another way to understand the key drivers and which assumptions you really need to focus on. There are some things that are not going to have a big impact on the financial results, and there are others that are going to be driving it. You want to make sure that you have a good understanding of what the key assumptions are that are driving the financial results of the models that you're looking at and focus your validation efforts on those issues.

Validation of formulas and calculations is the final piece. Checking against a policy illustration system or an independent model is the best option for this. You want to make sure that the policy mechanics are being determined correctly. If there's a product feature on this product that is new, and if it's something that you haven't ever modeled before, you really need to step back

and make sure that whatever the model is generating for those results is appropriate and that the calculations are all being done correctly. These values need to be verified with something like a spreadsheet that can reproduce those calculations or by hand-checking values. You need to validate all the calculations and make sure that the model is doing what you expect it to do. I would suggest a full audit of a couple of the model points, making sure that you can reproduce all of those calculations. And, of course, including these audits in your documentation is a necessity.

Rebecca is going to talk about data integrity.

MS. BURTON: The best part is about to begin—the topic of data integrity.

Seriously, you might say, "Data—how boring and tedious." What do we mean by data? What are some examples?

- In-force policy data
- New business data
- Data on policy loan experience, death and surrender experience, premium payment experience
- Geographic data
- Demographic data
- Data on assets—portfolio assets, earned rates, investment expenses, default costs, quality ratings
- Data regarding agent/manager compensation

I guess we have established that data are everywhere, and without data, there would be no reason to build the model. I want you to take away from this presentation that data are core.

So this brings us to integrity. What does that mean? Everyone just throw out some ideas of what you think integrity means.

FROM THE FLOOR: Honesty.

MS. BURTON: Honesty, yes. Accuracy, completeness.

FROM THE FLOOR: Internal consistencies.

MS. BURTON: I went on the Internet, and I discovered all kinds of definitions of integrity. I found the following: wholeness and unity; entireness; unbroken state; being honest; the standard of doing job and determination not to lower it; purity; unreduced or unbroken completeness. All are good definitions. So integrity seems like a hefty word for data, but, quite frankly, if we decided that it's core to the modeling process, then you better hope that you have some good data with integrity.

Dealing with data is where you begin in any modeling project. It is the starting spot. You begin by taking your real world data and summarizing them, much like Mike spoke of in his section on model structure. Then you perform the analyses. You come up with mathematical conclusions. You interpret these conclusions so you can predict future behavior and perhaps explain current behavior.

Let's go back to real world data. We might start with a huge amount of data. For example, those of you who work with insurance companies, give me an example of how many life insurance policies you think the company has?

FROM THE FLOOR: Two hundred thousand.

FROM THE FLOOR: Several hundred thousand.

MS. BURTON: Five hundred thousand or a million. It could be two million for some of the really large companies out there. Say we wanted to make some conclusions about the various lines of your business and how they're going to perform in the future. What if you have two million policies, a million policies, or even 200,000 policies to work with? How in the world

would we manage to work with this? It's not possible, so we might consider fitting the data into the model points as Mike has discussed already. As he pointed out, our goal is to maximize accuracy, which would promote more model points while working under the constraints of budget and time, which would promote fewer model points.

Let's go back to the integrity of the data. In my experience, there has never been enough time spent with the data. I know this; the people I work with know this; and, we are taught this concept in Course 7. There's never enough time or budget money allocated towards the data work. If you take that away from this presentation, take away that you have to put the time in. So much can go wrong with your data. For example, after you have been working for a month or two on your projects, you might realize what a field was supposed to be used for. You've already done a month of work, and you have to repeat three weeks of it. Spend more time with your data.

Let's back up one step to data collection and hit the high points of what can go wrong with data collection. Realize, of course, that all of these things that can go wrong would surrender the integrity of the data. First, what if the data are not according to your specifications? Maybe you requested a field length of 15, and you were given a field length of ten. When you read the data into Microsoft Access, or whatever database manipulation tool you use, you have the first ten characters of one item, plus five characters of the next one. That would really mess you up.

There are other problems. Requested fields are missing. You have duplicates, blanks, or no fields. You might not understand what you have. For example, you might have one word headings on all of the fields that are given to you, and it might be easy to gender and know what that means or what the issue year or plan code is. There are going to be other fields for which you'll need to know what exactly is included. You might have inappropriate text instead of numbers. It would not work if you were trying to sum up a bunch of premiums and the data read "27,000," for example.

Perhaps you've just flat out been given incorrect data. You've requested information for this one subset of policyholders, and you received it for another group of policyholders. What if critical elements are missing? I was working on a recent project, and as I was looking through the data, I noticed I had nonsmokers, preferred nonsmokers, and aggregate. I didn't have any smokers though. That clearly meant that I needed to go back and speak to the programmer who compiled the data extract and find out where the smokers went.

You've got to watch out for bias in data. You should watch out for inconsistencies or unreasonable items. If 70% of your population is female smoker, you might correctly think something has gone wrong in the data collection process.

Another problem would be a communication breakdown between the actuarial model and the collector of the data. This isn't always going to be the programmer's fault. You need to make sure that you have communicated very well and specifically what you want in the data. Know exactly what you want, and have an expectation for what you are to receive.

Because we are very concerned about the integrity of our data, there are several things we should do once we have received a data extract from the programmer. First, you want to make sure that you understand it. Where did it originate? What was its original use? Say you are collecting data from two departments. Perhaps one set of data came from your valuation system, and one came from your administration system. The one from the valuation system might have fields A, B, C, and D. The one from the administration system might have fields C, D, E, and F. You want to combine the two and have A through F. A couple of problems you might encounter is there is no unique connector to enable you to get the record together. Alternatively, Field C might be reportedly the same in each set of data. Maybe it's a premium, but maybe one set has it as annualized premium and the other has paid premium, premiums paid monthly or quarterly. These are things to watch out for.

You want to make sure, as I said earlier, to inquire about the elements of each field. Make sure you understand what you have. Of course, items like date and gender are going to be obvious, but there will be fields of information for which you won't know what is included. One thing I've run into is implied decimal points in your data. You just need to make sure that you are aware of these things.

My second bullet point is to reconcile, reconcile, reconcile. I think it's probably the most important point. After you receive the data extracts, you want to make sure that you can tie back to the most checked and most official source that is reviewed by regulators. I'm thinking about the Blue Book here. You want to make sure that you get your reserves to tie back to the Blue Book. You might run into the situation where the Blue Book isn't 100% correct, and then you'll have a situation where both sources are consistently incorrect.

You can also tie to reports that are internal to your company; perhaps they are published, or perhaps they are not. You can compare in-force count, volume, or premium amount to these reports. If you do see that you're missing data, you want to understand it. You might not necessarily need to go get a new extract. You would need to explain the disconnect between the extract and what is published in the Blue Book.

You are probably always going to have some errors in the data collection process, but it really boils down to a matter of materiality. You don't want to spend all of your precious time trying to dig out the little minute issues, but you want to spend enough time so that you expose the potentially larger ones.

There are many reasonableness checks that you will also want to perform. Print out and look at just a few records. Does the information make sense? Do a quick query to search for duplicates or blanks. If you do have blanks, you want to know whether they are truly blanks or they are meant to be a zero. Is anything actually missing out of those fields?

You want to have a preexisting expectation for the distribution of the data, a male/female distribution or an issue-age distribution. One time I managed to turn all of the one-to-nine-year-olds into 10-, 20-, 30- on up to 90-year-olds. I did a distribution of the data, and I had no one less than 10 years old and little humps at the 10-, 20-, 30-year olds and so on. That kind of clued me into the mistake I had made.

Again, calculate your totals. Tie them to published information, but also see if they look reasonable to each other. Do tax and statutory reserves have a reasonable relationship to each other, for example?

Lastly, after you have gone through the reconciliation and reasonableness checking processes, and you've come up with your list of questions, be sure to speak to the source of the extract. Remember that there is going to be a definite disconnect between you and the compiler of the extract. You are the actuary, and you know the actuarial elements of what you're trying to do. He or she knows the computer and had to compile the data. You might have been the one to request information directly from this person, or you may be one, two, or three people removed. But keep in mind, when you go back with your questions, you are going to be questioning this person's work. You need to be sure you let them know you're clarifying items, and that you're not questioning their work. You don't want to put them on the defensive. It might be a good idea, if you are one, two, or three people away from the data compiler, to get that intermediary person (the one who understands the actuary and the one who understands the technical end) to play an intermediary role and speak to both of you.

Another area in which your data are really important is for assumption setting. You're going to use not only your current data, but you're experience data. Unfortunately, you're going to need to do these reconciliations and verifications on the experience of data as well. You're going to need to make sure that the experience of data are consistent. Do they cover the same subset of people, and the same type of business. Bear in mind, for asset data, you will again need to perform the same reasonableness checks. You're going to need to question how the data was gathered. Also be sure to reconcile it to the Blue Book.

I'm going to harp on documentation, too. We say that actuaries never document, but we all get up here and we talk about it endlessly. Documentation is not an option. You will produce your report and have a specific section on data. Perhaps your work doesn't require a report, but it does require written documentation. It's not an option.

For example, let's say that your data did not have that much integrity, and against my advice, you went ahead and created your model. You're going to need to disclose this in your report or in your written documentation. In fact, your report will have a section of data including sources of data, the degree to which you reviewed it, any issues you discovered, how they were resolved, and the tests and various checks you did on the data. If you did have problems, you'll need to report on the materiality of those problems. Finally, you need a section on limitations and reliances. What are the limitations of the data, and what have you chosen to just rely on? Why have you done this?

Let me leave you with this. The most important aspects are: be sure to spend time with your data; do not just believe that it is without problems; be sure to reconcile and disclose.

CHART 1
Pricing Model

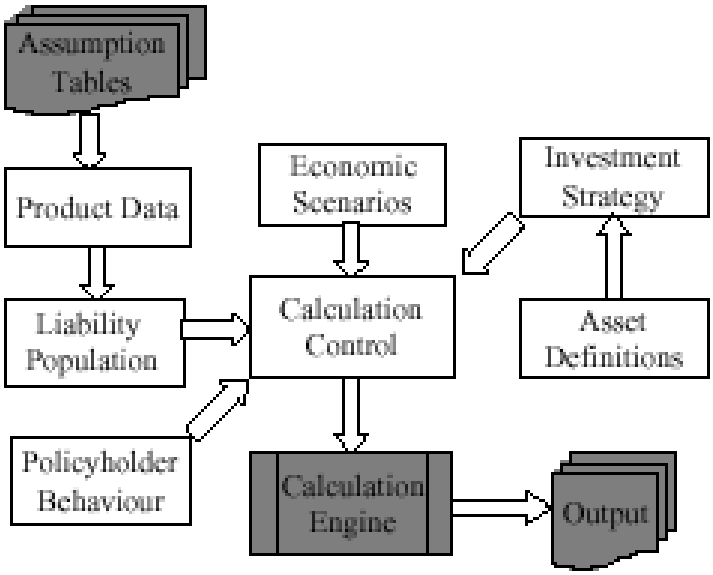

CHART 2
Financial/ALM Model