RECORD, Volume 25, No. 3[*]

San Francisco Annual Meeting
October 17-20, 1999

Session 75PD
Actuarial Software: Build or Buy

Track: Computer Science
Key Words: Computer Systems

Moderator: STEPHEN J. STROMMEN
Panelists: MARK D. J. EVANS
SUSAN M. LEE
STEPHEN J. STROMMEN
Recorder: JEFFREY JEROME KRYGIEL

Summary: The panel analyzes the choices between purchasing actuarial software and developing the software in-house. The panel also offers contrasting views of the advantages and disadvantages of the two approaches, as well as a combination approach.

Topics include:
- *Determination of software needs*
- *Evaluation of software packages*
- *Combining in-house developed software with commercial software*
- *Modifying commercial software to the needs of the organization*
- *Staffing and training for in-house developments versus staffing and training for installation of purchased software*

Mr. Stephen J. Strommen: As actuaries we all know that we tend to use large volumes of data in our work and we apply very complex analyses to that data. The types of things that we do simply are not amendable to the old yellow-ruled paper, and pencil method. We rely on computers extensively. As a result, one of the issues facing the management of any actuarial function is the determination of how to obtain the computer software that is needed to carry out the work. The first question that needs to be addressed is whether to build or buy.

We have a panel of three people who have quite a bit of experience dealing with some of these issues. Mark Evans is second vice president and actuary at AEGON U.S.A. in Louisville. Mark has had pretty good success with the approach of building much of the software that he needs, so he'll take that point of view in his discussion. Susan Lee is an actuary at Allstate Life & Savings in Chicago, and she has had pretty good success with buying much of the software that is used in her actuarial function. She'll be leaning towards that point of view in her presentation.

I'm an associate actuary at Northwestern Mutual Life in Milwaukee, and I will be giving a case study of the procedure we went through at our company a couple of years ago when we were deciding whether to build or buy some of our actuarial systems.

Mr. Mark D. J. Evans: As Steve mentioned, I'll be talking about the advantages of building actuarial software systems. Let me first describe briefly the different parts of my presentation. First, I'm going to tell you a little bit about my background so you know where I'm coming from. Then I'm going to do a situation analysis of some of the background that leads to my conclusions. That will lead into a discussion of my philosophy about actuarial software development. Then I'm going to talk a little bit about how the actuarial and information services (IS) departments should interface. That will set the groundwork for the core part of my presentation, which is "Build It and They Will Come."

I've been involved mostly with individual life, and some with individual health and annuity products. I spend a lot of time on experience studies, mortality, lapse, and a little bit of time with morbidity. I put a lot of work into research, specifically different valuation issues as new things come up, mostly from a GAAP perspective. I spend quite a bit of time with corporate modeling and acquisition analysis also. I've also done work on the strategic analysis of distribution systems, and for the past two or three years, I've been heavily involved in product development.

With modern computer languages, you do not have to worry about a lot of the housekeeping that you did years ago. That enables you to develop the computer programs pretty quickly, especially those that actuaries use. For that reason there is a very large number of applications. For the actuary who is moderately proficient at programming and able to write, type into the computer, and completely debug the program, an application can be built faster than they could even explain it to a conventional programmer.

This leads to a philosophy of actuarial software development. Certain premises are necessary. First of all, all actuaries should be computer-literate; I am talking about going beyond knowing how to put formulas in a spreadsheet or knowing how to use Word. I am talking about a competency in a procedural language, whether it is C, Basic, Fortran, APL, JAVA, or what have you. Why is this? Let's face it. As Steve was alluding to, actuarial work involves a large number of calculations. It's only logical that we are going to use computer tools and skills to solve these problems. Another point I want to make is if you don't program, you are not going to have an appreciation for the kind of problems or the kind of areas you can get into when you have computer tools available to you and you do know how to use them correctly.

In addition, there is a high overlap between the actual thought process and the skills we have to have to be good actuaries, to be able to get through the exams, etc., and the type of thought processes, skills, and talents that are needed for developing programming proficiency.

Another consideration is that actuarial programs tend to be computationally complex and very specialized. As a result, there is a fairly narrow range of what is done compared to the universal software development environment, and because of that there are not a whole lot of programmers out there that are real comfortable working with actuarial software. Those are reasons that actuaries need to be computer-literate, and they need to be deeply involved in the development of actuarial software.

Along those lines, I feel strongly that actuarial programming should take place in the actuarial department, not in the information technology (IT) department. You would have greater accountability. You would tend to get faster turnaround on your projects. You are going to get better quality because the people doing the programming are closer to the business unit. You will also have a better understanding of the needs of the business unit. On the other hand, of course, there are those large systems that cut across many departments and those types of programming fall more naturally within the domain of your data processing department.

Of course, if you are doing programming within the actuarial area, you still need to cooperate with the IT department. You will work closely with them. Actuaries with programming expertise can help with that relationship. First of all, when you have something you need them to do for you, for whatever reason, it is going to allow you to better judge the difficulty of various IT projects. If they tell you that it will take two years to do something that you know can be done in two weeks, it's helpful if you know that. The other advantage is when IT is developing systems, if the actuarial department has programming proficiency, the actuaries will be able to write code, etc., that will help debug major systems.

One comment you hear a lot when actuaries are trying to develop applications is that IT often does not want actuaries messing with a company's data. They feel actuaries could get in there and mess it up, so they can't write this program. They want to do it for you. They say they can't get to it for three years, but that is just the way it is. In many cases, that is unacceptable. Most operating systems you have now provide a workable alternative. It is very easy to restrict files to a read-only basis so that you can't change the data even when you are using it as input. You have it available on a read-only basis, and in the few situations where it doesn't work that way, you can simply get copies of production files.

As I see it, the advantages to building rather than buying software are as follows. First, a lot of the work we do is very experimental in nature. What we think we are going to have when we start out and what we end up with is quite a bit different. This takes a lot of reprogramming and changing it as we go along. Second, we tend to do a lot of prototyping where we will have a system up and running, and early in its life we will be making a lot of changes to it. We want to be able to do that quickly and easily.

Similarly, you might have applications where you are looking at frequent modifications, or where you are wanting to do things differently or trying different things. Once again, that's a situation where you want to be able to build your system. If you have custom or unusual products or other actuarial problems that are a little bit out of the mainstream, you are not going to be able to purchase software to do that. The same statement goes with respect to complex formulas.

If you have unique data storage in existence at your company, or if you are doing original research, don't expect to go out and be able to buy the research software package.

Now, to put these advantages in context, there are several real-life examples that I've either been involved with or familiar with where you needed to build. First, with mortality and lapse analysis, a lot of companies have their data stored in different types of formats and different types of file layouts. Different companies have different ideas about significant factors affecting mortality. Some companies say that policy size has a big impact on our mortality or underwriting classification or what underwriting rules we have for different blocks or whether we sell it on a monthly direct bill or annual. Since that is likely to vary from company to company, it makes it pretty difficult to go out and purchase the standard software that is going to handle everything.

Related to that is how you slice and dice your expected rates. This is going to vary quite a bit from company to company, and how you organize those expected rates within your computer files is going to vary quite a bit also. This leads to the point that it can be difficult to go out and buy software to do this as opposed to building something in house.

Similarly with ad hoc reporting, the very name itself tends to say that this is something that is going to be a little bit different from what you have done before. The needs, the various selection criteria, and the various subtotals, are likely to change with each application, and you are likely going to need to build that reporting functionality.

Let me convey one specific situation that I was involved in early in my career. We wanted to analyze a lease-back agreement and be able to test various payment patterns, look at present values, and tax impacts. Initially our financial department went out to different banks. They could not do anything with it because they didn't have programming capabilities, and there were not any available in the marketplace because this application is fairly original. There was no software you could buy. Within the actuarial department, we were able to quickly and easily develop a program to do this, which told them how they needed to analyze the different proposals they had and told them which proposal was the best.

There are quite a few packages out there that you can buy that will do reserves and cash values for traditional life insurance plans as opposed to universal life. But if you have unusual plans or unusual grading (minimum cash values for three years grading to Illinois modified preliminary term method after 17), you may have more than a little bit of trouble having purchased software do that. However, that is an application where if you are able to build and it is a problem, it is not that bad.

There are situations for valuation systems, and even administrative situations, where you have innovative products or complex products. You are not going to be able to go out there and buy something, or if you can buy something, it is something that you are going to have to spend a lot of time modifying.

Another consideration is when you are involved in original research. Don't expect to be able to go out and buy software that is going to help you. You are going to have to build it. When the FASB was struggling with how to do GAAP for universal life (UL), they were getting a lot of comments from actuaries. Those comments, frankly, were not all that effective because the actuaries were addressing it at a conceptual level. With my company's huge interest in this topic, we got involved in this discussion. We sent the FASB detailed numerical examples of how different approaches would work. We were able to do this because we had the computer programming capability and the infrastructure in place to be able to build these models to feed this information to the FASB. This helped guide them in their decision-making process. As a result, we were able to gain credibility with them that others were not.

Our company, being a big UL writer, was interested in early adoption of *FAS 97*. There was little or no software available to help us at that time. We had already developed extensive models of our UL, for both in-force and new business. It was a lot easier to leverage those existing models and just add *FAS 97* calculation capability as opposed to start over with new software.

Similarly, we had done a lot of research on deferred acquisition cost (DAC) for traditional business and had come up with some unique methods that we felt met our business needs and fairly reported our earnings to shareholders. Once again, we were treading on new ground. We were doing a lot of experimentation and a lot of stochastic scenarios to see how different DAC techniques would respond to different lapse patterns. Once again, there was not any software that we could buy that would allow us to do it, which is why it was important to be able to build it.

We also spent a lot of work with my company on the home-service market, where profitability by field office can be very important because it can vary quite a bit from geographical area to geographical area. We developed a system to study profitability by field office. The particular approach we used was not an asset-share pricing-model approach. We were looking at deviations as the percentage of premium due to claims, persistency, and expenses. The formulas we used on this case were not particularly complex, but once again they were unique, and it was not something where we even had the option of going out and buying the software to do it.

Similarly, with expense analysis, approaches vary widely by company. The data and budget information tends to be stored in company-specific formats. Philosophy on this varies quite a bit from company to company. Once again, you are looking at probably having to be able to build this as opposed to being able to purchase software to do this.

Switching gears, at AEGON, we don't typically buy options for equity-indexed liabilities. We usually do option replication. If you are going to do that, you better understand that concept well. The only way to understand it well is to be able to model it and see how it would do under different stochastic scenarios and under actual historical environments. Once again, no known existing software package was able to do this. We had a very specialized custom need, and we wanted to be able to research different ways of hedging our liabilities. Many of those ideas were not in a textbook, and we weren't going to be able to find software out there to do the analysis.

I would also like to make some points about the use of purchased software packages, like TAS and PTS. You will have issues arising as far as getting data into and out of those systems. They are not going to do it for you. I suppose you could key it all in, and I suppose you could cut and paste a column at a time. But presumably you have more important things to do. The efficient way to address that data input and data output is to be able to build interfaces, both coming in and coming out, according to your needs.

Many purchased software systems let you go in and put your own custom code right into their program, or you even have access to their source code in case you want to change it. You are going to be much better equipped to do that efficiently if you are already familiar with the procedural programming language because you are going to have a better understanding of how your languages are going to interact with the system they already have.

There will be an article coming out from the Computer Science Section's newsletter talking about what courses the universities and colleges are recommending for actuarial students to prepare them to enter the actuarial world. We have good results from all of the different actuarial schools as far as responding to this survey. On average, they recommend about six credit hours with a good mix of spreadsheets, math packages, and procedural programming languages. I would say on average that is about six semester hours. That should give students about half of what is needed. But I do want to say that the vast majority of the colleges and the universities do see this as an important area.

Ms. Susan M. Lee: Like Mark, I am going to start off with a little bit of my background and give you an overview of my presentation before I get onto my "buy" soapbox. I am going to present a framework that we like to use whenever we come to the build/buy decision.

The past five years I have been mainly working with supporting and developing proprietary and vended software systems. These systems are used for cash-flow testing, economic value analysis, and asset/liability management. Prior to that, I spent nine years in product development. My computer experience includes a variety of languages and a variety of platforms. I am also the Y2K coordinator for our actuarial department, as well as an IT liaison. What I wanted to point out in regard to computer experience is that it is ever evolving. Once we were predominantly known for a programming language (APL) and that is changing. We constantly have to keep up our skills and move forward with the industry as it moves on.

What is your company's technology strategy? Do you want to be known for building software products? Managing information is an enormous challenge for everyone. It doesn't face just the life insurance industry; it doesn't face just the financial industry's market. Technology innovation has increased our capabilities for creating and storing data. The environment is complex, and it is ever expanding. Look at the state of the Web today. If you went back ten years ago, who would have thought we would be in the position we are now?

Application life cycles are shrinking. Before you might have had 7–10-year life cycles on some of the applications you would develop or buy. Today those are shrinking and are down around 2–3 years. What will happen in the next few years? Will we be looking at a 3- to 6-month life cycle for applications? Currently Microsoft is fitting itself within that 2- to 3-year range. They had Office 95, then 97, and now we are on 2000. Will they be upgrading every year? As we use those tools, we need to keep pace with that technology.

As I touched on before, the technology solutions should be developed through outlining your business drivers. Technology should be an enabler. It is not a driver. It should not be a goal. What is required to position your company to win? If building a software application will position you to win, then that is what you should do. But is that consistent with your corporate vision? Do you want to be in the position of maintaining these systems in the long run? Probably five out of six or seven would say, "No, I'm not in the computer science field. I'm in the insurance field; I'm in the banking field; I'm in the financial institution field." Keep that in mind when you are looking at this.

Another point is that you need to leverage, and leverage is very important whether you build or buy. It helps you to minimize your cost, eliminate duplicate work, and it ensures that it supports your company's strategic direction. This is strategic direction not from a software perspective, but from a business perspective. If it intends to expand into certain marketplaces, what is going to be the technology that allows it to do that? You need to weigh the department's needs versus the enterprise's needs. Will this be an application you can use across all business channels? You need to look at the cost versus the functionality. A good example of this is brokerage houses that made a large investment in the Web, but they are seeing wonderful paybacks today. They balance the cost versus the functionality.

Don't forget about the future. The solutions that you adopt today will need to work tomorrow. With our shortened life cycle (2-3 years out), will what you implement today position you well going forward? You need to identify re-use opportunities so that a particular application used in one part of your company has other uses in other parts of the company. You need to weigh tactical or short-term solutions versus the long-term strategy.

What I would like to do now is to review a typical decision-making cycle. I want to go through a process that we use for some of our large-scale endeavors. The idea is that you should constantly look at whether you should build or buy. You don't want to pigeon-hole yourself into one particular venue.

The first thing you need to do is to gather your requirements for the particular application that you're looking at. You have some type of financial process. Perhaps it's some new product that you're coming out with, and you want to know how to get it to market or you've got the valuation of a particular product and you want to know what type of system to put that on. Not only do you need to solicit feedback from people who work with the product, but what about people outside. For example, you're working with a business unit that only sells term products. Why not also include in the conversation systems supporting those term products—your UL lines and your annuity lines. There might be some synergies there. Again, you want to be able to leverage your technology solutions. Then you need to consolidate all this feedback. You really need to be careful in isolating what are the "nice to haves" and what are the "need to haves."

Then you need to go out and find the available systems. Here's where you get into that build/buy decision-making. Who are the vendors? Could you upgrade a current system you have or should you build a new system? Once you reach that point you're going to break off into two different branches. Bring in the vendors; see what they can offer. Mark mentioned that sometimes vendors don't support some of the unique bells and whistles you have. Find out what is missing, and what it would cost to get those things put in. You also need to do feasibility studies of upgrading and building your systems. Do you have a department that can maintain these applications once they build them? When you need an upgrade, who do you go to if you have a system that's built in-house?

You need to solicit the feedback from both vendors and clients. If you have vendors that come in and do demos, get the reaction from people. Also, interview their clients. This can be very important. It will allow you to determine how responsive the vendors are, and what problems they have had implementing the system. Be aware of who you are going to get into a relationship with potentially.

A very valuable tool that we use is the preparation of a cost-benefit analysis (CBA). This is the tool that we use to allow us to determine the appropriate tract for us. This can be a very valuable tool to you, and you should do it regardless of whether you decide to build or buy. It does not have to be something that is elaborate and takes weeks to put together. You can put some of these together in 10-20 minutes. It does not have to be complex.

Finally, you need to score your vendor systems as well as the capabilities you have from building it in house in terms of the cost and in terms of the functionality. That will lead you to your decision. My experience at Allstate over the past five years is that 8 out of 10 times we follow this process and we have resulted in the buy decision.

Now I mentioned I would go over how to do a cost-benefit analysis. The first thing you have to look at is what are the associated costs. Again, this applies whether you are going to build your application or whether you're going to buy it. You need to look at your employee compensation. How many programmers, consultants, and clerical people is it going to take? Don't forget to include inflation adjustments and overhead backers for the management of that development staff, its employee benefits, and its office space. A lot of times these are costs that go unnoticed when you build your applications. You need to consider these.

The other thing is contract resources. If you are with a small company, you may not have the actuarial staff or IT staff to build applications. It doesn't mean you can't hire consultants to build applications for you. Don't forget to include those or any other professional services. Here again you have to include management time. The management will need to set the direction, the requirements, the time line, and assist in the testing. Don't forget a management overhead factor.

Another big thing is the hardware. Allstate was affected by this a few years ago, and we looked at leaving a mainframe environment and going to a PC-based environment for a variety of our systems. That meant a hardware investment: PCs and servers. As costs come down, PC-based applications become a viable option. Don't forget to include those in your analysis. If you go to a PC-based vender system, make sure you find out from them what the optimal configurations are. Don't just assume you can load it to your standard PCs that are in your workstations. Chances are you can't and you might have to go out and buy additional hardware. The same would hold for when you're building applications. Do you want to stick with the same technology you are using? Obviously, you have identified a process that needs to be done. Does your current technology address it? What is the optimal solution? It isn't just writing the program. It is also the operating system and the hardware. You have to take all those factors into account.

Don't forget peripheral software. Something Mark mentioned in his discussion was that when you do buy software, you often have to provide links from the variety of sources you have for data. If you are using Microsoft Office tools, don't forget about those licenses as well. There may also be travel, education, and training expenses.

Now we get on to the benefits. It looks like there are a lot of costs with very few benefits. That is a function of the way that we have designed our CBA process. You look at reductions in head count. When we go from this system to that system, whether we build it or buy it, what type of cost savings are we going to see in terms of reduction in head count? How much automation were we able to achieve? If we used to have a staff of five actuaries working on our monthly close, this new system might bring that down to three actuaries working on our close. Those are the types of benefits that you can realize, but there are also soft benefits. You may be able to get your numbers sooner, and you will be able to do more analysis. These benefits are difficult to factor in.

You should also consider potential reductions in resources. If you used to hire a company to do your cash-flow testing for you; if you purchased some software, you may no longer have to hire them because you can do it yourself. So those are the types of benefits you can use in the CBA.

The other thing to keep in mind about this evaluation cycle is don't sweat the small stuff. You are not going to do it for nickel-and-dime applications. You are going to put those in a spreadsheet. You're going to put them in a real simple visual basic application. What we are talking about here is the company investing significant dollars in either building an application or buying an application. You might want to set a threshold for anything under a certain amount. You want to use reason, and you want to prototype. If there's something new, some regulatory ruling for which you want to see the impact on your company, you are not going to go out and buy a vended system. You are going to prototype first, see the impact, and then you will go through a complete CBA process to determine whether you want to program this into your systems or just buy a system that can do this? The idea of a cycle is that it circles back around.

I can't stress enough that once you either build or buy an application, you have to continually address whether that still meets your business needs. Today you might build; tomorrow you might buy. You might be back to building in another 2-3 years.

I have quite a bit of experience with buying, so now I'll get on my buying soapbox. I'd like to share with you some of our experience at Allstate. The things you need to keep in mind are contract and legal issues when you do buy vendor software. Of particular interest is the scope of your license. Are there any restrictions on the number of users or the site? You need to be wary of your maintenance fees. Will they increase perpetually? Can you lock in your maintenance fees? You also have to consider the training and customer support services they provide. The problems that occur with upgrades to Microsoft Office for those who not only build their own applications but also buy them were mentioned recently at the Computer Science Council Section. What is the position of your vendor when Microsoft makes an upgrade? If your company goes from Office 97 up to Office 2000 will your vendor system work? You need to address that up-front, and that is a legal issue, believe it or not. You also need to know about the warranties. How do you enforce quality after acceptance?

A lot of times actuarial vendors will put out software in kind of a beta version. They rely on the users to test the software. If you have it in your contract what the warranties are regarding their distribution of software and their testing, you have a better chance of understanding and using and working within that system to get things corrected. You should not have to pay for getting things corrected. It should not be an enhancement to the system. If you have unique features to the product lines, you sell enhancements need to be addressed separately.

Confidentiality is a big one with Allstate. Take caution to prevent disclosure. If you have a lot of proprietary methods, and if you want a vendor software application to support those, you need to share information with the vendor. Make sure you have proprietary caveats put in your contracts to ensure that that information doesn't go into the general system or is not shared inadvertently with other users.

Vendor relationships are very important. This is a partnership that is entered into. You have mutual goals and you need to enroll your vendors in the bridge-building process. Mark mentioned that in his presentation as well. If you get your vendors to be cognizant of what your needs are, then you will have less worry on the back end in getting your data systems to speak to each other. The process of getting the data from one system into another will be less complicated.

There are also fringe benefits to buying. The biggest benefit is portability of skills. Turnover in the actuarial profession exists. If you know a particular application at your company and I have that application at my company, it makes it very easy for me to hire someone from your company because they bring prior skills. I don't have to train them on my system because my system is a vendor package. Let's say I'm a builder of actuarial applications. Once you have been at my company, where are you going to go? You are less marketable because you don't have the same background as someone coming from a company that has the same system.

There is enhanced confidence from management. The idea of safety in numbers. If a particular tool is regarded as best in class in the industry, it is going to sit well with your management when you give the numbers from that system. It doesn't get past the issue of garbage in, garbage out, but at least you know the garbage was processed the same way.

Consistency is important. When you use one application for a variety of sources, you don't have to reconcile. When you build there will always be the issue of business unit A that built an application for pricing; business unit B built an application for pricing. They both sell the same product; they just have different expenses because they have different distribution channels. Then you get the question of consistency and reconciling their results don't seem to mesh. Vendor systems avoid that by buying a single system that is used across the board.

Here are some of the examples of where Allstate has decided to buy software applications. We have bought the systems that underlie our financial projections for cash-flow testing, economic-value analysis, strategic planning, GAAP unlocking, and duration studies. This has been significant for us in the consistency and reconciliation stage. We can use one tool to do all of these applications, and it has cut down our development time in terms of model set-up and model construction. You build one model, and it moves from one process right into the other. We have purchased product illustration software used by our field offices and our reserve and nonforfeiture value rates and calculation systems are all vendor packages. Our policy administration systems are all vendor packages. Even the tools we use for competitive analysis work the databases and the query tools are all vendor products. Although Mark has decided to build an application that does their mortality and expense experience studies, we have actually purchased products that enable us to do that.

In conclusion, if you do buy products, focus on those that offer components. This way you can assemble them such that they meet your own needs. Lastly, I would like to leave you with something I found in an IT article. The biggest imperative facing IT shops is the ability to increase their success rates. IT projects have a 30% chance of not being delivered at all; a 70% chance of being over-budget, off-specifications, or significantly late. Mark says, "Build it and they will come." I say, "Build it, and with these kind of odds, they're not staying." What's more important is, can you, as a company, absorb this type of cost? When you go with a vendor system, that cost is spread over all its client base. If you're building, the cost is all your own.

Mr. Strommen: I'd like to run through a case study of the decision that my company had to make about two years ago when we were dealing with this decision of building or buying actuarial software for pricing and modeling systems.

We had a problem situation. We had been in the habit of in-house development for many of our systems, and virtually all of our actuarial systems. We had pricing systems that were built in DOS APL by our pricing actuaries over a period of time, and there was a separate pricing system for each of our major product lines. But given that they were developed by actuaries and not people with systems training, there was poor documentation and back-up. Maintenance was a little bit weak, and in some cases, it was hard to reproduce some of our old results, like, for example, an old dividend scale or a pricing for a prior series.

We also had modeling systems that had been developed in a completely different department. They were in a financial planning department separate from the actuarial department. Those systems were partly in DOS, some parts were in Windows or the mainframe, and there was a small staff working on it. They were having a hard time keeping up with changes in accounting and products and everything else and that staff included only one actuary. There was one actuary that knew it all.

We had a little bit of a problem. We wanted to set future direction for the software supporting both pricing and modeling. We needed to address both of those, and we wanted some kind of connection between the two. We wanted to assure that all the related needs were met. For example, we're not just talking about forecasting. We have cash-flow testing, GAAP reporting under *FAS 97* and *120*, etc. We wanted to assure consistency between the pricing and modeling efforts. We wanted appropriate controls to be put in place for documentation and version control. We needed to keep an eye on costs. We are a large mutual company, but we don't have an unlimited budget. We went through an evaluation process that was very similar to one that Susie talked about. These are some of the high points of it.

First, we developed a list of criteria to be satisfied. Second, we reviewed various vendor systems. We looked at three for a cursory review and then picked one vendor system to evaluate in great detail. We determined a little bit of how to deal with some of the limitations we found in that system. Then we estimated the cost of in-house development if we were to continue our old approach. We laid the options and prepared a proposal. The first criterion was that the system had to handle all of our existing products without substantial system customization. Now the key word here is substantial. That is in the eye of the beholder, but we had our own criteria for what substantial meant.

The system had to be able to produce the required pricing measures that we use. Every company has it’s own. We focus on return on equity in some cases and we have other measures in other cases. The system had to be easy to use because we have a rotation system among our younger actuaries, whereby they spend about a year in each product line or in each area. We didn't want a system where an actuary needed a whole year to learn to use because we'd never get any productivity. We needed adequate documentation and we had a high standard for that. We needed to know exactly how every number is calculated that comes out of whatever system is used. We needed a modeling and pricing connection, and as you know, many of the vendor systems automatically provide both pricing and modeling connections. That was more important if we went the internal route.

In our case, we also needed to incorporate an external investment projection. We have an investment department that has its own model. We are very happy with that. Our task was to be able to use their projection of the existing portfolio and not to develop a new system to project our investments.

We first evaluated our vendor system and discovered that it was very good for many of the more modern product designs like universal life, term, and annuities. We discovered that it require substantial customization to handle some of our older product designs, especially some of our unique participating products and disability insurance. We also discovered it was a little weak on its support for GAAP reporting and the process of projecting margin by year of issue. We also had some concerns about the capacity and run time. We had been in the habit of running models with thousands of cells. You can do that when you have a custom-built model because it doesn't have a lot of overhead. Sometimes it's more difficult in a vendor system that has overhead to handle almost any situation. It can't be as streamline in that fashion.

Then we estimated the cost of in-house development. The cost is fairly high to do this kind of thing. Remember, we already had an existing model, but it was having trouble keeping up so we had a backlog of things we needed to do. We estimated it would take six-to-eight person years of development to get our modeling system to where we wanted it to be. We needed all of the parts to use a consistent language, have a desktop user interface, eliminate all hard-coded assumptions, and make it handle GAAP for every product line, and have full GAAP projection capability. That doesn't include pricing systems. We didn't even bother to make an estimate of the cost of improving our pricing systems because six-to-eight person years was far over the cost of any vendor system we were considering, at least in terms of the purchase price and even considering some of the training that had to go on.

We had to weigh these options and write a proposal. Doing everything in-house was too expensive. Six-to-eight-person years was just too long. We had some immediate needs, and we couldn't wait for this to be ready so we had to do something. On the other hand, the vendor system was not a complete solution in our case. It didn't handle all of our products without substantial customization, so we decided to combine the two. We connected our in-house and vendor models with a user interface shell and a set of file translation utilities so that the data that are used in one system could easily be used in another system. We also, with regard to those lines of business that we decided to build in-house software for, connected our in-house pricing and modeling with the use of some common modules.

There are a number of advantages of this combined approach. First, we got faster product development using the vendor system. As I said, we had an immediate need to develop a new product. The vendor system was there; it could handle this new product off the shelf, so we got it out the door very quickly. We were also able to leverage our existing Legacy model of the old in-force business, and that was a big time-saver. We also are able to maintain an in-house development capability, and this to me is very important because we never have to say that we can't model that. If our management wants us to do equity-indexed annuities or some new kind of product, we can develop a model for it even if there is no vendor support.

There are some disadvantages to doing it this way. First, the overall system is more complex. You know how complex vendor systems are; if you layer an internal system on top of that, you have another layer of complexity. A user must master both the vendor system and an in-house system. You also have to employ extra support resources to maintain both systems and any connections between them: the file translation utilities and whatever.

There are a number of conditions that I think are necessary to make this kind of thing work. First, you need sufficient know-how and resources to do in-house development of models of this sort. You need a pretty good-sized actuarial staff to have these kinds of resources. You also need good relations between the actuaries and systems developers so they can communicate well. You need management respect for the actuaries that work with systems or otherwise the actuaries are not going to want to work on this stuff at all. If you have a situation where your senior actuaries learned their work before computers existed, and they feel anybody sitting at a computer screen is doing work analogous to filling in a yellow ruled pad, that's not the right situation. You need to have your actuarial management understand the benefit of having actuaries work on some of those kinds of things.

A couple of things to look out for. You need to keep your focus on the business needs and not cute systems. I'll give one example. I heard a story of an actuary who was given an assignment to work on a pricing system, and a year later, when his management touched base, he had developed a very beautiful user-interface and graphics but there was no pricing system. It just looked real pretty. You need to keep the focus on the functionality and not on making systems look cute. You need to ensure that the actuaries leverage the expertise of the systems professionals so that the actuaries are not trying to do it all. Otherwise, you can wind up with the situation we had before with our APL systems.

So build or buy? Sometimes you just have to do both. That's what we've done and we've been very happy with the results.

Mr. Paul A. Hekman: I guess you could say I have a vested interest here. I have worked for PolySystems about ten years, but prior to that I did a lot of financial reporting, so I would say I get my kicks more out of seeing people develop efficient reporting procedures than I do out of selling software. I would just like to share some experiences that I've had with people that have dealt with these issues.

First, I would encourage that you get involved with your IS people when you're making these kinds of decisions. This is important for a couple of reasons. Probably the major one is, if you're going to do valuation systems, you are going to need to connect your server or your hardware into the mainframe in some way or other to get the data that you need. I know of one case where this was not done and the user of the system was not able to hook into the mainframe system or even into the company local area network (LAN). He had to develop a very crude way of getting the data onto the PC for valuation purposes. It is essential to work out these details in advance.

Another issue is to not underestimate the value of having multiple users of a system to check out bugs. I would say, in many cases, when we replace in-house systems, we discover that there were embedded errors that nobody caught because there was only one user. That isn't to say that the vendor systems are bug-free as well. Everybody has them and any system that gets modified frequently is going to have bugs in it.

I think both sides of the issue were presented here in terms of your own personal career path. You have to decide what kind of skills you want to take to your next employer. Do you want to bring a program skill, or do you want to bring an analytical skill with you? I tend to think that actuaries better spend their time on analysis rather than programming, but that's a personal choice and that's up to you.

From the Floor: I spent many years both building systems for companies and buying systems. I think the panel did an extremely good job in pointing out both aspects of this issue. My bias is towards buying. In the case of the two systems I represent, not only do I sell for them, I bought the systems myself just so that I could have enough fire power to compete with some of the bigger consulting firms. But one of the points that was not raised is cross-fertilization of a vendor package. I think Mark said that the vendor packages have more products covered and more people have used them so there are fewer errors. There are errors in both, but generally a vendor product is going to have more people looking at it and there will be fewer errors in it, which also gives you a little higher confidence. Now you could work that both sides. Many actuaries say, "Well, if we didn't develop it, how could we have confidence in it?" In addition, I think it's easier to modify a vendor package than it is to get people to do it in-house. Usually the vendors are more responsive.

Another consideration, before a company goes to buy, particularly in new regulatory situations, is that regulations can change several times before a particular application is developed. I found this on almost all the GAAP issues that have come out.

One other thing is actuaries are lousy formatters. I've never seen an in-house system that has good reports and when you deal with management today you have to have good reports. The vendor systems usually do good reports. In addition, the documentation by actuaries is poor. You won't buy a package unless they have good documentation. I put my particular preference to a vendor system.

Mr. S. Michael McLaughlin: My firm is very interested in watching what happens here, understanding the different systems that are out there, and how our clients use them. I guess I enjoyed the presentation because it made me think through the wide range of actuarial software that is out there. You have research jobs that need spreadsheets or other original tools to do your research or prototyping; and you have experience studies, pricing, product illustrations, reserve valuation, DAC valuation for GAAP, modeling appraisals, and projections. There is a need for integration of all these systems with the rest of the organization because we are not working in isolation from the investment department, the accounting department, or the policy administration department. In fact, we sometimes need to do policy administration tasks ourselves for group contracts. We have options such as building our own software. We have the option to buy in many cases. What you have to keep in mind is what you are trying to achieve. We're not trying to build or buy software systems. We're trying to meet some ultimate objective. It just really seems that the picture is a little more complex. It also seems like we are spending a lot of money and spending a lot of time with this.

I would ask the question of the panelists and maybe others who are present, does it not seem that there ought to be some kind of integrated approach here? Do you see companies taking integrated approaches to evaluating what is their best way to create, build, buy, and subcontract actuarial software?

Ms. Lee: Companies are seeing the need for standardization. Many companies have opted for standardization of the operating system, most of the time that being the Microsoft product line. I think you are going to see this in the insurance and financial industries as well. Companies realize the efficiency, the ability to leverage when they adopt standard tools across the enterprise to solve their technology solutions.

Mr. McLaughlin: The question is also whether actuarial departments somehow integrate or plan ahead for the types of software that they want to build, buy, or subcontract. Is there any sort of organized approach within the actuarial function to this problem?

Ms. Lee: Related to what I have said before is how our actuarial department needs to cooperate and coordinate with the other technology areas. If we would decide to build a particular application or even buy a particular application, it has to be approved by a group of individuals, which represent the technical architecture that our company wishes to pursue. For at least Allstate, the actuaries are bound by the company's direction in terms of technology solutions.

Mr. Strommen: I'd have to say at Northwestern Mutual we have a history of almost always building everything, and we are beginning to go into the buy arena. I think we are just beginning with regard to the actuarial function to think in these terms. But in terms of the overall corporate standardization, there is a lot of that in place. We have Windows on the desktop everywhere, so we have to fit into the LAN structure that the IS department provides to us in whatever solution we come up with. But as I say, with regard to these kinds of actuarial software, we're just beginning to address a lot of these issues in my opinion.

Mr. Edward C. Jarrett: I have two comments to bring up. First, a lot of our actuarial tools are just that—actuarial tools. They may have a switch that allows you to supposedly do a certain amount of functionality, but it's our responsibility as actuaries to evaluate that particular tool to make sure it is doing what we want.

Just because your particular program has a tool to do reserves or switch to do it in a particular manner, it is your responsibility as an actuary to figure out that it is actually doing that. In many cases that falls down to an entry-level actuarial student because that student puts in these factors, and that person comes back to you. Yes, the system can do that but if you are the actuary that is reporting or helping your manager make the decisions, it's your responsibility to ensure that it is doing what you think it is doing. That may involve bringing some data out of the system and evaluating it with other programs, but it is still an actuarial tool and I know our particular company has vendor software and we are actuarial consultants. Many times it is done on a financial reporting basis. You have to put on your actuarial hat even when using these tools.

The second comment I wanted to bring up is again these are actuarial tools and we are not in the job of replicating the administrative systems. When you are dealing with a vendor or dealing with your in-house/purchase decision, be sure you keep it in the framework of building an actuarial tool and not trying to replace the IS department. If we try to attract young actuaries into an area where all of a sudden they are doing administrative IS type work, we are not going to attract them into that area.

Mr. Harold H. Summer: At Teachers Insurance, we are involved in a buy-and-build decision right now in terms of our short-term planning. It has been said, and I've heard from others, that vendor systems such as PTS, TAS, and Alpha are not adequate in terms of the specificity, in terms of the accuracy that management requires for the planning process. As a result, you would need to either upgrade our existing in-house systems or build from scratch our own in-house system. There are people in our company who feel we can go either way. I was wondering what experiences others have had.

Mr. Strommen: I outlined my experience with that in my part of the discussion. In terms of specificity, there are a lot of things that those packages do well, but there are also things that they do not do. My impression has been that buying a vendor package can be very helpful. But it will not be a complete solution for virtually anybody.

Mr. Evans: Mr. Jarrett had made a comment that the actuary is still responsible for the output, regardless of what system it comes from. I could not agree more and actually, I would like to go along those same lines and take a little bit of issue at the comment made earlier regarding vendor systems and the actuary being less likely to know the guts of the system. It's a little easier for the actuary to understand the models with a home-grown system. I would like to touch on a couple of other things real quick. The mere fact that some of these vendor-supplied systems do cover a broad scope is good. A lot of these are really good systems, but that, in and of itself, does not decrease the opportunity for errors as compared to what would be the case if you were building a specialized system. The less scope you are trying to cover, the lower the probabilities for error, so I think I would take issue with just a flat out statement that vendor systems will have fewer errors. That may be a little bit of an exaggeration.

The last point I want to make is another gentleman was talking about actuaries being analysts versus programmers. I don't see any dichotomy there. A lot of my work obviously has been in the programming areas, but that includes running programs to give me results that I can analyze as an actuary. The fact that I don't have to go back to a programmer every time I want to look at the numbers sliced and diced in a little bit different fashion improves my ability to work as an actuary. I don't see those as career paths that are somehow a dichotomy. They actually complement one another.

Mr. Harry Potent*: We first built software in order to do consulting but now we sell software. I think the thing that's germane here is not our software products, but our experience and on the build side, when we didn't know what we were doing, we were doctors and actuaries. We were using Microsoft products and we managed to get ourselves into a mess where the system was almost impossible to upgrade and only one person knew how to upgrade it. When that person left to take another job, we were fundamentally in a situation where we had to reprogram almost from scratch. Another important thing to think about is the idea of modular architecture and four-tiered architectures. I have found that the number of people who can program is vastly larger than the number of people who are architects. Architecture can make something upgradeable if you make it cheap to maintain and it can make it useful whether you are building it or buying it. I would just like to put in a plug for thinking about architecture before you build or buy.

Mr. Strommen: I agree. That is a very good point.

*Mr. Potent, not a member of the sponsoring organizations, is with Medical Scientists Inc., in Boston, MA.