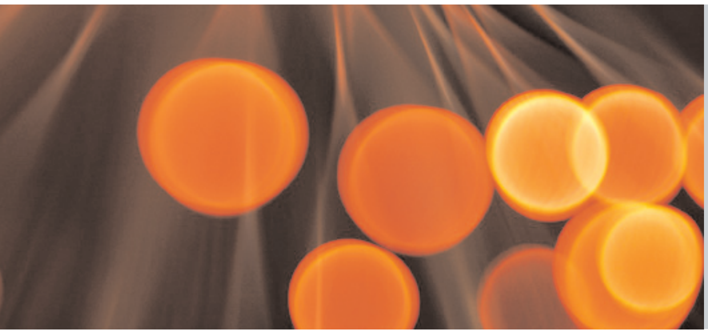
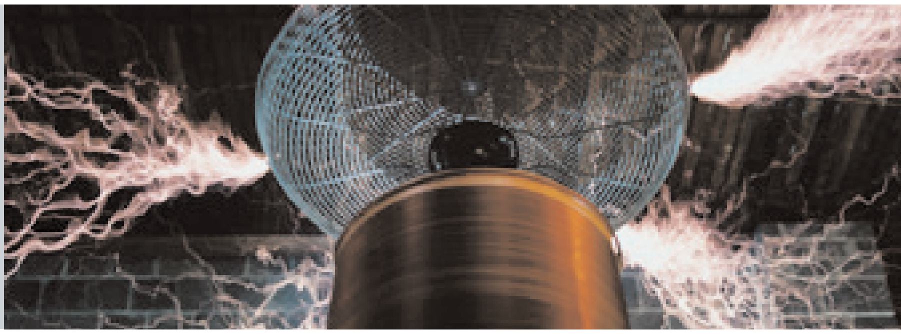



TECHNOLOGY SECTION

"A KNOWLEDGE COMMUNITY FOR THE SOCIETY OF ACTUARIES"

CompAct

CompAct Electronic Newsletter • Issue No. 23 • April 2007 • Published in Schaumburg, Ill. by the Society of Actuaries

Inside

Actuaries
Risk is Opportunity.sm

Technology Section Newsletter
Issue Number 23
April 2007

Published quarterly by the Technology Section of the Society of Actuaries

475 N. Martingale Road, Suite 600
Schaumburg, IL 60173
phone: 847.706.3500
fax: 847.706.3599

World Wide Web: www.soa.org

Letter from the Chair

by Paula M. Hodges

The Technology Section, as a special interest section, has the blessing and curse of being able to define what we mean to our members. We're not limited to a particular "practice;" we don't need to decipher new regulations, and we don't focus on a particular product or market. However, technology is present in virtually everything we do each day, not only as actuaries but as human beings, and we are charged with covering this broad set of topics. As such, our council has been listening to you, our members, on what is important to you, and what we can do as a section to give your membership value.

One such topic that has generated quite a bit of discussion is education on technology. Our formal actuarial studies focused on the mathematical and business aspects of our jobs. However, once we landed our first position in the "real world," we were suddenly confronted with our colleagues' spreadsheets, macros, programs written in Visual Basic, C++, Pascal, APL and other technologies that we'd never dealt with before. These foreign objects were put before us, and we were expected to figure them out, update them and make them easier to use.

We didn't need spreadsheets to solve our calculus or differential equation problems in college, so this was a very different world. If we were lucky, there was some training available to help us along the way. Many of us needed to quickly build an informal network of co-workers and peers. Those actuaries who were lucky enough to get hands-on experience with some of these technologies prior to entering the job market clearly had an advantage when coming in on the ground floor.

So … why aren't we teaching this?

Well, we should be. And we plan to.

Paula M. Hodges, FSA, MAAA, is manager of Modeling Strategy with Allstate Financial in Lincoln, NE. She can be reached at phodg@allstate.com

Our section understands the importance of learning both the "basic" and the "advanced" aspects of technology. We also have a great appreciation of how quickly these things change. We will be working to track the pulse of our membership and provide relevant and timely training opportunities via webcasts and sessions at our Spring and Annual Meetings.

We need to keep hearing from you. We're surveying our membership to make sure we're delivering what you need. Please help us by taking the few minutes to respond when we ask. This is a simple way you can help move our section in the right direction. We look forward to hearing from you!!

Paula Hodges
Chair - Technology Council

Illustration Software Testing–Part 2: Calculation Testing

by Joe Alaimo

In my first article I focused on automated interface testing of illustration systems. In my opinion, the implementation of automated interface testing is one of the most overlooked aspects of illustration system testing.

This article will focus on the issue of automated calculation testing. Although everyone tests the calculations in his or her illustration system, many companies don't use full testing automation or give the testing process the focus that it deserves.

Opponents of automated calculation testing use the same arguments against it as they do against interface testing. The biggest is "creating a test system will increase the testing time." Upon further investigation this statement proves to be incorrect. This article will describe the tools and procedures that need to be set up to perform proper automated calculation testing. It will offer a timeline analysis that will show how automation will improve the timing of the testing cycle and I will also outline the benefits of automated calculation testing.

Tools

Due to the unique and individual nature of life insurance calculations, an automated calculation testing tool will have to be created specifically for each project. The tool can be written in any computer language with the most common being Visual Basic, Microsoft .NET, C++ or inside an EXCEL spreadsheet (using VBA). This system is usually called the test system or shadow system.

The tool must basically perform the following functions for each case in a test deck of cases:

- Read-in policy information input from a file, database or spreadsheet
- Perform the calculations thus mirroring the actual calculation engine
- Write the resulting output to a file, database or spreadsheet
- Compare the output from the test system with that of the calculation engine using tolerance parameters (discussed more in the benefits section)
- Document the results of the comparison, usually outlining which columns did not match and in which years they did not match

When differences between the illustration system and the shadow system are found, the question of which system is producing the incorrect answer can arise. It is possible that the test system has the error while the calculation engine is correct. The solution to

(continued on page 4)

this problem is that a calculation specification document must be created at the start of the project. The calculation specification is used by the creators of both systems as their calculation blueprint. When a difference occurs, both systems must be checked against the specifications to find out where the error has occurred.

"As with interface testing, calculation testing does not begin at the end of the development cycle, but rather begins from the start of the project."

It is vitally important that an independent party create the test system. This party can be a second set of in-house developers, the actuarial department (as long as they were not directly involved in creating the calculation engine) or an outside consulting firm. There is no value in having the same people who were involved in developing the calculation engine involved in building the test system. They would just reproduce any mistakes or misinterpretation of the specifications in the test system.

Timing

As with interface testing, calculation testing does not begin at the end of the development cycle, but rather begins from the start of the project. There are two elements to calculation testing that begin right from the first day.

The first element of calculation testing is the development of test cases. This step is totally independent of the creation of the testing system. The test cases must be planned out and documented (a methodology that can be used to determine the test cases is outlined below). Once the cases have been planned out, they must be entered into a file, database or spreadsheet so that the testing program and the calculation engine can access them. The format of the test cases must be decided at the beginning of the project so that both the calculation engine and the testing system can be developed to read-in the test cases.

The second element of calculation testing involves the development of the test system. This system should not take as long as the calculation engine because it can be a lot more customized than the calculation engine. Since this system will only be used by the testers, a lot of error and business rule checking code does not need to be written into it. The development of this system should begin at the same time that development on the calculation engine begins. At this time the calculation specification will already have been developed.

Comparing Automated versus Manual Testing

To illustrate how automated calculation testing can reduce the length of the testing cycle I will use an example. Let's assume the worst-case scenario where the testing system was not built in advance. Therefore, at the time when testing is to begin, the testing system has yet to be built. We will also assume that we have allocated 12 weeks to perform our testing. Two separate teams have been hired to perform the testing. Team A will perform manual testing and Team B will create a test system and perform automated testing.

Team A requires two weeks to run every test case and manually check the columns to ensure that they match. During this time they document any mismatches that they find. Team B requires four weeks to build their test system. Once the test system is complete they only require one day to run the testing. Once the testing cycle is complete, we will assume that the development team needs one week to fix all reported problems. Let's keep track of the testing progress over the 12 weeks.

By the end of the second week, Team A has run through all of the test cases once and submitted their results to the developers while Team B is still building their test system.

By the end of the third week Team A receives a new system from the developers and begins testing.

By the end of the fourth week Team A is half way through their second testing cycle while Team B has just finished their test system. The next day Team B has run their first test cycle and submits their problems to the developers.

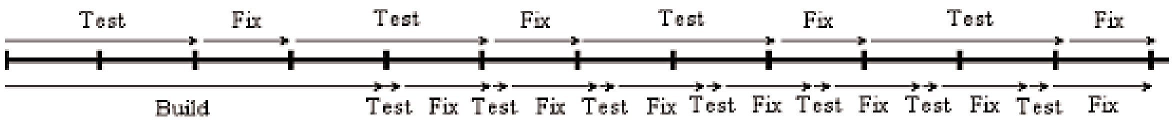
By the end of the fifth week, Team A has completed their second testing cycle and submits their problems to the developers. Team B is still awaiting the developer fixes from the first cycle.

By the end of the eighth week, Team A has just submitted the results of its third testing cycle. Team B, on the other hand, is halfway through getting the problems from their fourth testing cycle fixed.

By the end of the tenth week, Team A is muddling through their fourth testing cycle, while Team B has just completed their sixth testing cycle.

By the end of our 12-week period, Team A completed four testing cycles, while Team B completed seven.

The picture below graphically represents the above scenario. Team A is shown above the line and Team B below. Each tick represents one week.



Benefits

Automated calculation testing offers a number of benefits over manual testing. The two largest benefits are:

1) Ability to run regression tests more often: The usual test cycle consists of running a complete test, fixing all of the problems then re-running the complete test. When using automation, the complete test can be run so quickly that the developers can request a regression test of all or some of the cases more often. They may make a particular fix and be unsure of the effects on the rest of the system. In these circumstances, they can run the regression test immediately to find out if the fix had any ill effects without waiting until all of the other bugs are complete.

2) Cases can be run with tolerance specifications: Due to rounding errors that can occur, the calculations between the test system and the calculation engine may not match exactly. The testing tool, therefore, should be built to compare the numbers within a certain tolerance. The tolerance can be a single number or can increase as the policy matures (after all, if the policy is different by $0.50 in year one, then the difference will grow larger every year). The tolerance may also increase as a function of face value. These tolerance values should also be programmed so that they can be set on a column-by-column basis.

Comparing the calculations to this degree of precision is nearly impossible when performed manually, but a computer program is ideally suited to this type of precision.

Test Case Development

Development of test cases is an essential part of calculation testing. All test cases should be numbered and fully documented.

(continued on page 6)

This gives the testing team and the development team a common reference point when communicating bugs. It also makes the bugs reproducible.

Calculation test cases can be developed right at the start of the project. There is no need to wait until the calculation engine is complete. The testing team and development team should agree upon the documentation format of the tests in advance. If testing automation is used, then the documentation may also serve as the automation input file.

When developing a series of test cases the following guidelines should be followed to help ensure that a broad range of features are included in the test deck and to ensure that when a problem is found, its source can be easily deduced.

1) Document the test cases: The test cases should be planned and laid out in advance of when the actual testing will occur. The cases should be numbered to provide the testing team and the development team a common reference point. The cases can be entered into a spreadsheet or, if testing automation is used, can be entered into the format needed for the automation input file. Thus the documentation may serve two purposes and duplication of effort can be avoided.

2) Start with a basic case and build upon it: I have encountered the situation a number of times where the first case tested failed. This first case consisted of an insured that had mortality and flat extra ratings, had multiple riders and multiple benefits on the policy, paid above the maximum premium and multiple funds were selected. Finding the problem in this case is like finding a needle in a haystack.

   It is important to start with a basic case. The typical basic case is a male non-smoker, aged 40 with $100,000 of insurance paying a modest premium. No riders, benefits or ratings should be placed on this case. Use this basic case as a starting point for the next series of cases adding one feature at a time. The second case might add a rider, the third may add a benefit. When a problem occurs this will help pinpoint the problem.

If case two worked successfully but case three fails, then we can be confident that the problem was with the benefit that was not present in case two but was in case three.

Using this method will leave the test deck broken up into groups of cases, each case in the group building upon the last.

3) Use age ranges as criteria to differentiate cases: When creating test cases, a common mistake is to assume that each individual age creates a unique case. Three cases where the only difference is that the insured is age 20, 22 and 25 do not constitute three distinct cases. I would say that this is really just one case. It is a better practice to break the allowable age range into three categories: young, middle aged and older insureds. For example, you may use 18 to 35 for the younger, 36 to 59 as your middle aged and 60 to 80 as your older range. Your testing team can decide the actual breakup that will be used. This method should provide you with a better mix of cases and ensures that time is not wasted testing cases that are too similar.

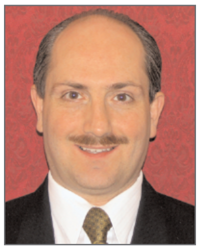
4) Add to your test deck as unique cases arise: Following the above techniques will provide you with a robust set of cases that will represent most of the situations that will arise. There are, however, unique circumstances that may be thought of when the cases are being developed or may be encountered during the testing. It is important that these special cases be added into your test deck whenever they are discovered. Examples of some special cases are developing a case where the exempt test may fail at a specific age or testing a very specific combination of multiple insureds, riders and benefits that caused problems in your previous system.

Whatever the situation may be, it is important to add it to your test deck, so that it can be part of your testing cycle.

Conclusion

Although it is often perceived that building a calculation test system will slow a project down, the truth is that the timelines can actually decrease when automation is used. Automation allows the testing to be performed more frequently and can quickly allow the developers to determine if a new fix has caused any other problems. This level of testing gives the developers more confidence that their calculations are correct and reduces the number of errors that are found once the system is released.

"Although it is often perceived that building a calculation test system will slow a project down, the truth is that the timelines can actually decrease when automation is used."



Joe Alaimo, B.Sc., ASA, is the president of ProComp Consulting. ProComp specializes in illustration system consulting including system development, concept development, calculation engine development and of course, interface and calculation automated testing. Joe can be reached at (416) 949-2667 or joealaimo@rogers.com.

Emulating A Random Surfer Is Not That Bad!

by N. D. Shyamalkumar, ASA

A Review of:
Google's PageRank and Beyond: The Science of Search Engine Rankings by Amy N. Langville and Carl D. Meyer

Most of us have heard of Google being a play on Googol (10100) and many of us have followed Google's Dutch auction IPO. But perhaps not many of us have heard of PageRank, an algorithm to rank search results, which lies at the very heart of Google and the focus of this book.

" The key idea behind PageRank (and HITS) is to view a hyperlink from Web page A to Web page B."

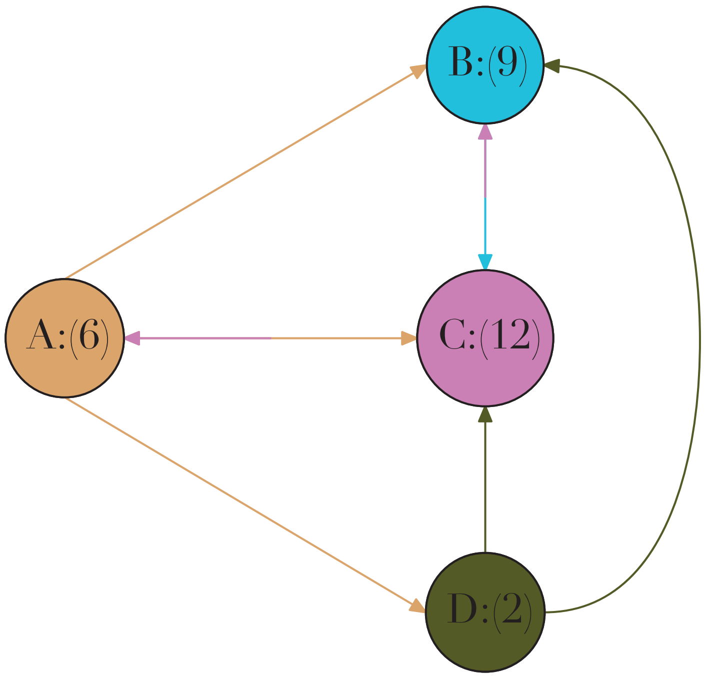
A typical user of a search engine accesses only the top few search results displayed, and hence, beyond indexing a certain proportion of the Web, it is the ordering or ranking of the search results that is vital. Given a query, the earlier search engines rated Web pages solely on their content, and the ranking of the search results was based on these ratings. This method is, by its very nature, subject to tamper as the author, by suitably modifying the contents of a page (with some modifications even invisible to the reader), can favorably affect its ranking. With the rapid expansion of the Web in the late '90s, solely content-based rankings increasingly delivered poor search results. It is in this setting that two research groups independently were focusing their attention on link analysis models. Jon Klienberg's HITS algorithm and the algorithm PageRank by Sergey Brin and Larry Page were the earliest of the models to use the hyperlink structure of the Web to augment content-based ratings. PageRank is the most dominant model for reasons including its commercial success.

The key idea behind PageRank (and HITS) is to view a hyperlink from Web page A to Web page B as a recommendation by A of B. In order to limit the influence of A, it is allocated a certain amount of endorsement weight that is equally transferred through each of the hyperlinks it carries. The rating given to Web page B is then equal to the sum of endorsement weights it collects via each of the hyperlinks that point to B, and moreover a Web page's endorsement weight is defined to be its rating. Even though this results in a circular definition, under certain conditions it has a unique solution and this determines the ranking of the Web pages. For an alternate view, and one more to my personal taste, consider a random surfer who at every time epoch traverses from the current page by choosing one of its hyperlinks at random. Then the rating of Web page B can be shown to be the probability that a random surfer, who has been surfing for a very long time, is found visiting page B.

Illustrative Example: Consider a hypothetical Web site with four pages—A-D, each with hyperlinks as depicted in Figure 1 (e.g., A has hyperlinks to each of the other three Web pages). If the pages A-D are assigned endorsement weights of six, nine, 12 and two, respectively, it is easy to check that the ratings are equal to the endorsement weights. For example, Web page C receives endorsement weights of two, nine and one from A, B and C yielding a rating of 12, its endorsement weight. Hence the ranking of pages A, B, C and D are three, two, one and four, respectively. Alternatively, the Web pages visited at each time epoch by the random surfer can be viewed as a finite state Markov chain (part of the syllabus for Course M) with stationary transition probability matrix given by the following equation:

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix},$$

Figure 1 An Illustrative Web Site with Four Pages



Of course in the real World Wide Web, there would be pages with no hyperlinks, the so called dangling nodes (the term reminds me of hanging chads); there would be a cluster of sites with no link from inside the cluster to the outside, the so called rank sinks; and the number of pages would be more than 8 billion and rising at an ever increasing pace. How these and some other imperfections, or real-world complexities, can be dealt with is covered in chapters 5-10. Chapters 11-12 discuss other algorithms like HITS and SALSA. In chapter 13 the authors discuss the future of Web information retrieval. While chapter 14 lists some useful resources for the researcher in information retrieval, chapter 15 briefly covers mathematical prerequisites. And of course, the first four chapters introduce Web search and discuss its various components.

One of the enjoyable features of this book is its numerous asides—interesting digressions spread throughout the book. I particularly liked those on the Great Firewall of China (the effort of the Chinese government to control Web content), G-Day and Google Bombs (successful attempts at manipulating PageRank), how librarians have to deal on a daily basis with Googleopoly (Google's dominance of Web search), the discussion of the lawsuit of SearchKing (a search engine optimization company) versus Google, and Google Dance (the oscillations in the rankings during their monthly updates) and the Google Dance Syndrome.

I greatly enjoyed reading the book, and to me its only shortcoming is the lack of heuristics (probabilistic) alongside the algebraic proofs. Even by skipping most of the mathematical details, I am quite certain that an actuary will find it a worthwhile read.

N.D. Shyamalkumar ASA, an assistant professor of statistics & actuarial science at the University of Iowa, is a member of the technology section council. He can be contacted at shyamal-kumar @uiowa.edu

The Real Costs of Actuarial Software

by Trevor Howes

Is that aging actuarial system no longer doing the job? Are you worried about the future implications of the principles-based approach? If so, then sooner or later you may have to face the prospect of a software replacement project.

Being an actuary, it may help you to prepare an analysis of comparative costs before making your decision. After a little reflection you will realize your investment in a new system goes well beyond the sticker price! The true costs of actuarial software include not just the initial costs of acquiring and implementing the software, but also the ongoing expenses of operating and maintaining it. Furthermore, you must consider hard dollar costs, soft dollar costs and contingent costs that depend on the overall performance of the software and its vendor.

Hard Dollar Costs

The hard dollar costs of new software should be the easiest place to start. Ask the vendor for a proposal based on the number of users and the modules of the system you want.

The vendor's proposal typically will include initial license fees and their current schedule of ongoing maintenance fees. You should ask about their policy of revising fee schedules, and what the recent history of fee increases has been.

Nailing down license and maintenance fees from the vendor is a waste of time, unless you also have a clear understanding of what those fees give you in terms of software that's ready to use and training and support for your users.

If the software needs customization, then the complexity and risk of the exercise just went up. If you need the vendor or third-party consultants to modify the system as part of the initial implementation, the payments for the customizations required can often exceed the initial license fees. If you take this on yourself, be realistic about the expertise and efficiency of your programming resources working with unfamiliar code. And don't forget to price-in the additional costs of validating, testing and documenting the customizations as well as developing the necessary reporting tools.

Of course, you can't really compare system A to system B unless you also budget for the complete IT installation required for each system to meet the anticipated volume and run times. This requires performance benchmarks from each vendor plus consideration of whether any distributed processing or grid processing capability will be needed. Plan for the costs of additional processors, networking, enabling software and initial and ongoing IT support sufficient to comfortably meet your projected demands. Concrete evidence backing performance claims, from benchmarking or from actual experience of current users, will go a long way to increasing your comfort with your cost estimates.

Soft Dollar Costs

Soft dollar costs for a new system can be significant, and may require an even greater element of judgment to quantify. These soft costs relate to the overall effort required by your staff to learn to use the software, to implement it successfully and then maintain it under projected business demands.

Actuarial staff productivity depends directly on the functionality and ease of use of the software and the quality of vendor support.

Some software is ready to go, straight from the box, but all systems presumably require some expert hands-on training in addition to built-in help and reference materials, so check the availability and quality of both.

While the best strategy is to involve current staff in the implementation of a new system, you must also carry on business as usual. If you are relying on a vendor or consultant to implement your new software, ask for references to confirm their ability to perform similar work on time and within budget, and add on the additional learning curve costs as your staff take over the maintenance.

Critical financial reporting processes impose additional costs in setting up and managing production environments. Sarbanes-Oxley demands more detailed documentation and new control procedures, and may force replacement of those ubiquitous spreadsheet calculations with more robust processes. Will your new software increase or reduce these costs? Will your vendor provide IT consulting assistance for the design of a controlled production environment at an extra cost, or consider it part of the regular client support?

Good Software Increases Efficiency and Effectiveness

Actuarial practice is increasingly turning to stochastic risk analysis requiring thousands of scenarios. The risks being analyzed are often more policy and insured specific, increasing the costs of building and validating compressed models. But if your software can avoid models by using full seriatim data, consider the significant reduction in time and effort, as well as the increase in comfort with the results that will be gained.

Better yet, look for additional increases in efficiency possible by using the same business data or model for another actuarial application. How much time is spent now in performing reconciliations of reserve calculators and modeled representations of in-force business between applications?

Look Beyond the Short Term

There is a lot to consider in evaluating the costs and benefits of new software. The challenge of that evaluation may make the idea of looking down the road to the next software replacement seem fanciful if not masochistic. Yet, changing actuarial and accounting standards and emerging technology will place huge demands on your software to keep up. If it can't, another system replacement project may have to be planned before the paint has dried on the currently planned purchase.

The key questions are:

- Will your software vendor be willing and able to make continued timely enhancements to the software to keep up with environmental changes while exploiting new technology effectively?
- Are the required system enhancements likely to be delivered as part of the system maintenance fees you are paying, or will they be chargeable projects?
- Will the vendor have to develop a new system and then charge you new license fees to cover their costs?

(continued on page 12)

- If the vendor upgrades the software, how difficult and time consuming will the installation of system upgrades or new system versions be and what assistance will the vendor provide, at what costs?
- If the software requires user customization or coding, what are the implications on installing future vendor updates and on the costs of ongoing maintenance?

If the answers to these questions are not comforting, then maybe you should increase your current software budget to cover pre-funding of the next required software upgrade and conversion and the collateral costs of replacing your actuarial staff who have quit in frustration.

Consider the Risks

Of course relying on a vendor's intentions or his track record may not be sufficient. If the vendor becomes unwilling to commit the resources to support and grow a software product that has become obsolete, uncompetitive or unprofitable, intentions will count for nothing. Vendors, like insurers, may fail financially, change ownership and business direction, or lose key personnel. These are hard risks to price-in to your purchase, but you can at least look at them and ask the vendor what protection you have.

You can also look at the business practices of the vendor relating to actuarial software and determine if it is being run as a viable, stand-alone business or if it is a sideline to a more profitable activity or software offering. As a last resort, you, or a consortium of users, might want contractual access to the source code.

How Much Work Should You Do?

Many of the costs and considerations outlined above are impossible to quantify with precision. Clearly, the first question is whether these soft–dollar costs and business risks are material to the buying decision? To get a feel for that:

a) ballpark your ongoing direct and indirect expenses of performing the affected actuarial functions, in staffing, equipment and outside services,
b) compare the software alternatives using a rough grading scale as to the anticipated relative impact on ongoing costs and expenses,
c) use (a) and (b) to produce an estimate of the comparable cost differentials with each alternative.

If that analysis produces a basis for a clear business decision, then no further work is necessary. If it raises serious questions about the alternative costs, then additional attention to these questions would be well advised.

If you make the right decision now, you may not have to face one again for a long time. And your actuarial staff will thank you.

Trevor Howes BMath, FSA, FCIA, MAAA, is vice president & actuary with GGY AXIS, an actuarial software firm located in Toronto, Canada. Trevor has over 30 years experience in the life insurance industry including 12 years with GGY AXIS. Trevor can be reached at Trevor.Howes@ggy.com.

ACTUARIAL AND TECHNOLOGY PROFESSIONALS:
Increase your understanding of actuarial applications of technology by joining ...

Technology SECTION

A knowledge community for the Society of Actuaries

WHO WE ARE:
A broad-based community of actuaries and technology professionals.

WHAT WE DO:
Help actuaries understand and get the most out of technology.

BENEFITS OF MEMBERSHIP:

- **Networking** – Interact with other professionals who share similar interests and challenges.
- **Information Sharing** – Identify and communicate information and ideas on emerging technologies and their implementation.
- **Publications** – Keep up with current professional and industry trends through publications. Read or contribute articles to the online Technology Section newsletters.
- **Conference Sessions** – Participate in section-sponsored sessions at Society of Actuaries meetings and seminars.
- **Participation** – Gather information to enhance your work or volunteer to write, present or lead.

The Society of Actuaries, an educational, research and professional organization, sponsors a wide range of professional interest sections. Each section is a unique knowledge community formed around common professional issues related to an area of practice or a special interest. For more information about this section and others, go to www.soa.org and click on sections and practice areas.

Actuaries
Risk is Opportunity.sm

To join, detach the form below and mail it, along with the membership fee of $20, payable to ***Society of Actuaries, P.O. Box 95668, Chicago, IL 60694*** *or fax it to* ***847-273-8552.***

Section: Technology

Name: ____________________

Title: __________ Company: __________

Address: ____________________
(Street)

____
(City) (State) (ZIP) (Country)

Phone: ____________________

E-mail: ____________________

Payment Type: ☐ Check Payable to Society of Actuaries

Credit Card: ☐ *Visa* ☐ *American Express* ☐ *MasterCard*

Number: ____________________

Expiration date: __________

THE LAST THREE DIGITS ON THE BACK OF YOUR CARD

CVV2 Number*: ____________________

Cardholder's printed name (if different from above):

____

Cardholder's signature: ____________________

Cardholder's billing address (if different from above):

____
(Street)

____
(City) (State) (ZIP) (Country)

* American Express cards show the CVV2 printed above and to the right of the imprinted card number on the front of the card.
If your European or Asian credit card does not have a CVV2 number, you may enter 000 as your CVV2.

Articles Needed for the *CompAct Electronic Newsletter*

Your help and participation is needed and welcomed. All articles will include a byline to give you full credit for your effort. *CompAct* is pleased to publish articles in a second language if a translation is provided by the author. For those of you interested in working on *CompAct*, several associate editors are needed to handle various specialty areas such as meetings, seminars, symposia, continuing education meetings, new research and studies by SOA committees and so on. If you would like to submit an article or be an associate editor, please call Nariankadu Shyamalkumar, editor, at 319.335.1980.

CompAct is published as follows:

Publication Date	**Submission Deadline**
July 1, 2007	April 15, 2007
October 1, 2007	July 15, 2007

Preferred Format

In order to efficiently handle articles, please use the following format when submitting material:

Please e-mail your articles as attachments in either MS Word (.doc) or Simple Text (.txt) files. We are able to convert most PC-compatible software packages. Headlines are typed upper and lower case. Please use a 10-point Times New Roman font for the body text. Carriage returns are put in only at the end of paragraphs. The right-hand margin is not justified.

If you must submit articles in another manner, please call Susie Ayala, 847.706.3573 at the Society of Actuaries for assistance.

Please send electronic copies of the articles to:

N.D. Shyamalkumar
Technology Section Editor
e-mail: *shyamal-kumar@uiowa.edu*

Thank you for your help.