

CompAct

TECHNOLOGY SECTION

"A KNOWLEDGE COMMUNITY FOR THE SOCIETY OF ACTUARIES"

Inside

Letter from the Chair _____	2
<i>by Paula Hodges</i>	
Illustration Software Testing – Part I – Interface Testing _____	4
<i>by Joe Alaimo</i>	
Horses For Courses _____	8
<i>by Phil Gold</i>	
Scenario File Format Project Update _____	12
<i>by Carl Nauman</i>	
Microsoft Press Release _____	13
Join the Technology Section! _____	16
Articles Needed for <i>CompAct</i> _____	17

Actuaries

Risk is Opportunity.™

```
<?xml version="1.0" encoding="UTF-8" ?>
- <XTbML id="Table1" Version="XTbML2.9.91"
  xmlns="http://ACORD.org/Standards/Life/2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ACORD.org/Standards/Life/2
  XtbML2.9.91.xsd">
- <ContentClassification>
  <ContentType tc="45">Cash Value</ContentType>
  <TableName>Tabular Cash Value</TableName>
</ContentClassification>
- <Table>
  - <MetaData>
    <ScalingFactor>0</ScalingFactor>
    <DataType tc="45">Currency</DataType>
    <CurrencyType code="USD" />
    <TableDescription>Issue Age
    Duration</TableDescription>
    - <AxisDef id="A_1">
      <ScaleType tc="3">Age</ScaleType>
      <AxisName>Issue Age</AxisName>
      <MinScaleValue>35</MinScaleValue>
      <MaxScaleValue>45</MaxScaleValue>
      <Increment>1</Increment>
    </AxisDef>
    - <AxisDef id="A_2">
      <ScaleType tc="4">Duration</ScaleType>
      - <!--
        also known as Duration
      -->
      <AxisName>Duration</AxisName>
      <MinScaleValue>1</MinScaleValue>
      <MaxScaleValue>20</MaxScaleValue>
      <Increment>1</Increment>
    </AxisDef>
  </MetaData>
  - <Values>
    - <Axis AxisDefID="A_1" T="35">
      - <Axis AxisDefID="A_2" T="1">
        <V>0</V>
      </Axis>
    - <Axis AxisDefID="A_2" T="2">
      <V>0.39</V>
    </Axis>
    - <Axis AxisDefID="A_2" T="3">
      <V>14.81</V>
    </Axis>
    - <Axis AxisDefID="A_2" T="4">
      <V>29.81</V>
    </Axis>
    <Axis AxisDefID="A_2" T="6">
      <V>61.56</V>
    </Axis>
    - <Axis AxisDefID="A_2" T="7">
```

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

Nariankadu D. Shyamalkumar

CompAct Editor
Assistant Professor
Statistics and Actuarial Science
241 Schaeffer Hall
The University of Iowa
Iowa City, IA 52242-1409
phone: 319.335.1980
fax: 319.335.3017
e-mail: shyamal-kumar@uiowa.edu

Technology Section Council

Paula M. Hodges, Chairperson
Kevin J. Pledge, Vice-Chairperson
Joseph Liuzzo, Secretary/Treasurer

Council Members

Van Beach, Council Member
David Minches, Council Member
2007 Annual Meeting Coordinator
Carl J. Nauman, Council Member
Timothy L. Rozar, Council Member
*2007 Spring Meeting Program
Committee Coordinator*
N.D. Shyamalkumar, Council Member
Newsletter Editor
Dean K. Slyter, Council Member
Web Coordinator

BOG Partner: Mark Freedman

Staff Partner: Meg Weber
mweber@soa.org

Staff Support: Susan Martz
smartz@soa.org

DTP Coordinator: Joe Adduci
jadduci@soa.org

Facts and opinions contained in these pages are the responsibility of the persons who express them and should not be attributed to the Society of Actuaries, its committees, the Technology Section or the employers of the authors. Errors in fact, if brought to our attention, will be promptly corrected.

Letter from the Chair

by Paula M. Hodges

I am writing this fresh from the Society's Annual Meeting in Chicago this year. What a great experience! While I've been to several meetings over the years, in my opinion, they are just getting better and better. It's so great to walk through the foyer between sessions, at the reception and beyond. There are opportunities to talk with so many friends that I've made over the years. Beyond our actuarial credentials, we share so many common daily struggles in our jobs. Having the opportunity to discuss these with old friends, new acquaintances and finding new ways of looking at current issues is what makes the annual meeting such a positive place to be.

The Annual Meeting provided an opportunity for the Technology Section to meet and discuss our goals and objectives for the upcoming year. Each of the sections of the Society of Actuaries was formed to address five key areas of interest to its membership:

- Networking
- Information Sharing
- Publications
- Professional Education
- Research

We used this as a context to develop the goals for the Technology Section for the upcoming year:

Networking

We will try to have several face-to-face networking opportunities throughout the year. Possible activities may include:

- A reception at the annual meeting
- Local networking events in large metropolitan areas (New York, Chicago)
- Online communication forums for discussions on current topics and/or published articles

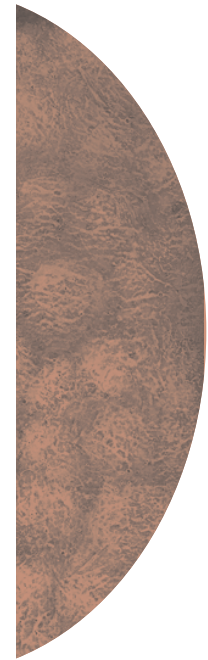
In doing so, we hope to spark interest in the section and promote growth as well as provide valuable interaction between our members.

Information Sharing

We want our membership to be able to find and share information regarding actuarial technology. To meet that need:

- We will explore creating a Web site where information and research can be uploaded by our membership for review and comment.

Spring Meetings Program	Tim Rozar
Annual Meeting Program	David Minches
Technology Webinars / Seminars	Van Beach
Research and Surveys	Carl Nauman
Communications Coordination	Kevin Pledge
Newsletter	Shyamal Kumar
Web site / Online Forums	Dean Slyter
Scenario Generator	Steve Strommen
Rate Manager	Joe Liuzzo
Speculative Fiction	Gary Lange
Membership and Recruiting	Paula Hodges



- Members will be able to post to discussion forums on topics of interest to our section.
- Our section's volunteers will also participate in the SOA's Web site redesign efforts.

Publications

We'll continue the publication schedule that we put in place this last year:

- Four newsletters (January, April, July and October).
- Bi-monthly Tech Update e-mails from the chair.
- The bi-annual Speculative Fiction Contest is also underway, and the winner will be announced on April 1, 2007.

Professional Education

Our section will become more active in the area of education by working with the Education and Research Section. There is interest in providing technology-based curriculum for the SOA's basic education requirements. In addition, we'll be providing:


- Sessions at the Spring and Annual Meetings.
- Other options for technology-related education to actuaries, such as webinars or 1-2 day seminars with a focus on technology topics.
- Links to relevant topics on our SOA technology Web site pages.

Research

In the area of research, we need to first understand the needs and makeup of our section. To this end, we plan to:

- Conduct surveys of our membership on technology topics.
- Explore trending of technology usage, and report to the section and SOA membership on emerging techniques and tools.
- Continue work on development of standard practices for scenario generation.

It is worthy of note that these goals will not be achieved by the Technology Council alone. This is your section. It is only with your help that we can achieve the goals. Each of the council members is leading up a critical effort to advance the goals of the section. Their roles for the upcoming year are listed above. Please contact one or more of them if you're interested in helping their cause.

With your enthusiasm and support, our section will be doing some great things this year. I am very much looking forward to being a part of it. 

Paula Hodges
Chair - Technology Council



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



Illustration Software Testing – Part I – Interface Testing

by Joe Alaimo

Illustration software testing frequently takes far too much effort and time to complete. Often, the field force finds bugs in the first week of the software's release. The cause of this is usually due to two major factors: the lack of proper test case development and the use of manual testing instead of automated test methods.

Creating automated testing programs and procedures are usually perceived as unnecessary and a waste of time and resources. Writing testing programs is usually only considered at the end of the project when testing is ready to begin and the common argument against them is that it would take too long to develop and delay the completion of the project. The truth is that automated procedures actually reduce the length of the testing cycle and allow for the delivery of a much more stable system.

Once a product is released to the field, the number of people testing the system will grow from a few to a few thousand! If a system is released that has not gone through proper testing procedures, any bugs the system may have will almost certainly be found by the agents within the first week of its release. If the field finds too many problems with the system they may lose all confidence in the software. This can result in the field not using the software at all or doubting every result that the software produces.

Using automated test procedures is a must if the goal is to deliver a stable system and reduce the testing cycle.

This article is the first of two parts. This first part will outline the benefits of automated

software testing procedures for the software interface. The second part of the article that will appear in the April issue will focus on the calculations as well as provide a methodology of when and how to create test cases.

Interface Testing

Even when companies use automation to test the calculations, they often overlook the benefits of using automation for interface, business rule and report testing. This article describes some of the interface testing tools available, some criteria on selecting testing staff and the benefits of using automated interface testing.

Tools

There are a number of tools that are used by quality assurance professionals to perform automated testing. Some of the more common tools are WinRunner, Test Complete and Rational Robot.

All of the tools have the same core functionality and offer the same basic features. The tools have the ability to record all actions performed on a software application including keystrokes and mouse clicks. These actions are recorded and saved as scripts. The scripts can then be automatically replayed to reproduce the same actions accurately and consistently.

The tester can create many scripts to reproduce many different scenarios. This bank of scripts can be run automatically by the test program and in any order desired by the tester. This allows the tester to run the complete bank of scripts unattended either during a daily run or overnight. The test program

records the results of the scripts and provides a log of any problems that occurred during the run.

Using a testing tool allows the same keystrokes to be tested in the same order in which they were originally entered. The testing tool can perform these keystrokes consistently and much quicker than a person performing the same task.

Selecting Testing Staff

When selecting testing staff, we have developed guidelines to assist us in our selections. We have found that following these guidelines allows us to choose people that find the most number of problems and provide the most unbiased testing.

- Do not choose a programmer: Although the tester will need some basic programming expertise to use the testing scripts, they should not be a developer who was involved with the system. It is also best to not choose a programmer at all. Programmers will usually have an in-depth knowledge of how Windows works and how a user interface works within Windows. They will be more likely to unconsciously interact with the interface the way it was intended. There is a lot more value to perform unexpected actions during testing, as this will more closely mirror what a real user will do.
- The tester should not know how the system works internally: It is important that the testers have as little knowledge as possible of how the system works internally. At ProComp, we ensure that the testers have no knowledge of our system architecture. The less the tester knows, the more unbiased they will be in their testing.

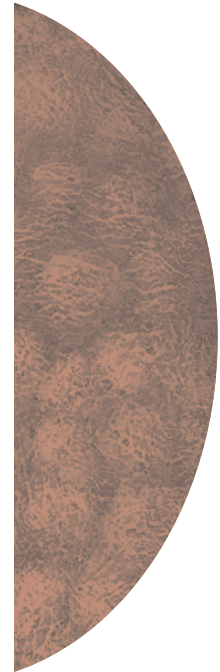
- The tester should not be an insurance expert: This seems like an odd requirement because we usually try to find people with the most insurance knowledge. It is important that the tester have insurance knowledge but if they know too much then they will unconsciously give the system all of the correct values. They may need to know as much as a typical insurance agent but not as much as your marketing personnel.

- The tester should have quality assurance training and experience: Some people believe that this is a step that can be overlooked. Nothing is further from the truth. It is important to use testers who are familiar with quality assurance procedures. Entering cases and randomly using the system are a small part of the quality assurance process. Proper quality assurance procedures include test case planning and development, entering the cases, running the test cases, test case reporting and regression testing throughout the project. Using a person who is unfamiliar with these procedures will usually result in many important steps being overlooked or forgotten.

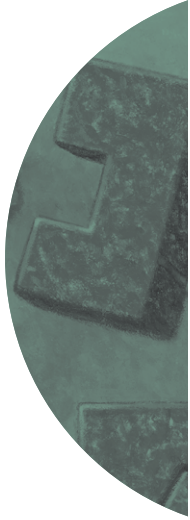
Timing

The first question that is usually asked about automated interface testing is when it should begin. The first step in testing is the planning. Before testing can begin a test plan must be created and scripts must be generated from this test plan. The plan can begin development on the first day the project begins. The plan includes the kind of scenarios that need to be addressed in the testing.

The actual script creation cannot begin until an initial version of the system is available to work with. When we develop a system at ProComp, we start by creating an interface



(continued on page 6)



specification. From this specification we create the user interface in our development environment. This interface has no functionality attached to it but allows the team to analyze the “look and feel” of the system. Once this interface is given final approval then it can be used to develop the test scripts. This interface will not have any logic built into it yet but the test scripts can be created and tested while the logic is being built into the system. When the system is ready to test, the scripts have already been generated and testing can begin immediately.

Benefits

The benefits of automated interface testing over manual testing are numerous and quite compelling.

- Problems are easily reproducible: When a system is tested manually, a person or a group of people usually will spend a lot of time entering data, clicking on different parts of the interface and trying many combinations of options and features until they produce a problem. Often when they find a problem, they cannot reproduce it because they don’t remember all of the steps they went through just prior to the error occurring.

Problems that occur in an interface are different than those that occur in calculations. Reproducing the problem does not only depend on the current state of the system but is dependent on how the system reached its current state. The exact order of steps performed prior to the error occurring is important because entering the same data in a different order may not produce the error.

Automated testing alleviates this problem because when an error occurs, the exact steps

are recorded. Furthermore, these steps can be reproduced every time. Thus when the problem is fixed, we can be confident that the exact problem we encountered has been resolved.

- Ability to run regression tests more often: One complete testing cycle consists of running all of the test cases and documenting all of the problems found. These problems are then relayed to the developers who fix all of the known bugs. At this time the cycle begins again with the testers re-running all of the test cases to ensure that the known bugs have actually been fixed and to see if new bugs have been introduced. The fixed bugs are marked complete and the new bugs are added into the bug tracking system. The re-running of test cases is called regression testing. This cycle of test-fix continues until all of the known bugs have been fixed and no new bugs are found.

When the regression test is performed manually, the process can take a week or more to complete. At best the tester will follow a script that is outlined on paper. At worst they are left to attempt to re-test the problem from the list of documented bugs. This method has the built in risk that the tester will accidentally skip a step while testing and may assume that the bug is fixed when it really isn’t. Also, if the tester does realize that they have missed a step until later in the script, they will have to re-start at the beginning of the script.

When using automated testing, the regression test can be run over a number of hours, or run overnight. This reduces the test cycle to running the test overnight and documenting any problems the next day. The cycle is reduced from one week down to one day! Since the test-fix cycle usually consists of four or more

cycles, the total time reduced from delivering the final product can be one month or more.

- Automated tests will find problems that manual methods will not: There is no scientific basis for this statement, but my experience has shown automated testing has found problems that just were not found using manual methods. This may be due to the fact that testers usually create more test cases when using automated testing. After all, a few or even 100 extra test cases don't take any extra time to run when testing is performed overnight. The only extra effort used is when the cases are first created.
- The same tests can be run on multiple platforms: Manual testing requires one or more testers to sit in front of the computer running through test scripts or scenarios. If the mandate is to test the system on multiple platforms (i.e. Win '95, Win '98, Win XP, Win NT, Win 2000) then the same number of testers must perform the same tests on each platform. This can effectively double, triple or quadruple the testing time in person hours. This would

require you to hire more testers or increase the testing time as each platform is tested one at a time. When using automated testing, the same scripts can be run at the same time on each platform with minimal resources. The only extra time may be due to documenting the problems found on each platform, if the problems are platform specific. This offers incredible savings of time and resources.

Conclusion

Automated interface testing requires proper planning and extra initial effort; however, the overall benefit is a reduced testing cycle thus allowing the product to be shipped sooner. The product will also be more stable because more cases are tested and they are tested on more operating system platforms.

This article covered automated interface testing of illustration systems. My next article will discuss automated calculation testing. I will illustrate the savings in time to delivery using an automated calculation system, describe the benefits of automated calculation testing and provide some tips on test case development. 🖥️



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 joelaimo@rogers.com.

Online Dues/Section Membership Renewal

Now you can pay your annual dues and sign up for SOA and IAA professional interest sections with our new easy-to-use online payment system! Just visit <http://dues.soa.org>. Using your credit card, you can pay your dues, renew section memberships or sign up for new section memberships. Online dues payment is just one more way the Society of Actuaries is improving your membership services. Renew at <http://dues.soa.org> today!

Horses For Courses

by Phil Gold



Reprinted with permission by Albert Wilking, a contemporary artist. He can be reached at albert@crazystudios.com.

We live in an environment where the word *open* has positive connotations, while *closed* has a negative feeling to it. Linux is open source, and Windows is proprietary or closed. So Linux must be better, right? Well, for some people it is, and if you read the Internet blogs, there's no contest. Yet Windows has the bigger market share. How many of us are running on Linux today? Strangely, Apple's OS/X gets even better reviews and that is a closed system. So let's keep an *open* mind.

There is no universal answer that open code is better than closed code, or vice versa, although it seems almost an item of religion for some people. You have to look at the requirements of the application, the quality of the vendor, the size of the organization, initial and ongoing costs of each approach,

corporate governance control requirements, the rate of change in the environment and the availability of skilled resources.

I am a software developer and I have chosen an approach closer to the closed end of the spectrum than the open end. I thought you might like to know why I made that decision, and what I have done in my application to meet the requirements for flexibility that many claim can only be satisfied by open code.

The first problem is to define my target market. Let's say my target market is all life actuaries, everywhere. I am writing a system to perform all manner of actuarial calculations, first to support the requirements within my own country, then internationally. By far the easiest solution for me is to write an open code system. All I need to do is develop a nice framework, probably based on Excel or something that looks like it, add some database support and a report writer, then prototype some typical products and let the client or a consultant worry about adjusting the sample code to fit the real world products. Then I could look forward to a lucrative stream of consulting assignments to implement and maintain your systems. In fact, I must be nuts not to have followed this model. Why? Because the alternative is to code every possible combination of product features and regulatory requirements myself, as well as user-specific methodologies and approaches, and that would take forever.

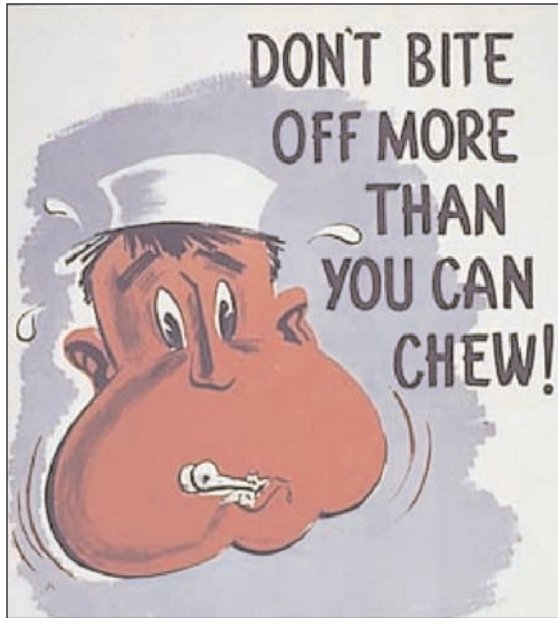
And yet, masochist that I am, that is indeed the path I chose. I'll tell you how later. Right now, I'll concentrate on why.

Why take the closed code route?

- Because I've been down the open code route before, in three companies. In every case each new product resulted in a new model with new source code, incompatible with other models in the company, and a variety of errors because of the lack of a proper testing environment. I want a reliable universal model, not a collection of independent models.
- Because I believe actuarial resources are scarce and expensive, and they should be employed on real actuarial problems, not developing software. Those actuaries that want to develop software should come and work with me – I'll need them for sure.
- Because despite the unique characteristics of each company and product, there is a lot more common ground between us than elements that separate us.
- Because I like a challenge. When someone tells me it can't be done, that's when I get interested.
- Because I was working in a reinsurance company, where we required a quick turnaround on each new product and the volume of such products prohibited the down time of developing a new model each time around.

Why persevere?

- Because the senior management of my company encouraged my efforts.
- Because my initial efforts were met with surprising success in the market.

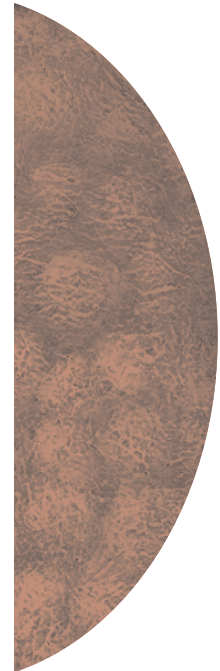


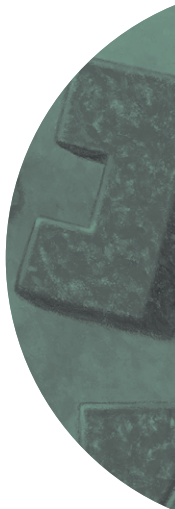
Reprinted with permission by Google Images. It is an old World War II poster and can be found on the Web site at: <http://images.google.com/imgres?imgurl=http://z.about.com/d/history1900s/1/0/t/S/wwip247.jpg&imgrefurl>

- Because I found partners and employees who shared my philosophy.
- Because our clients gave us encouraging feedback.

Now the golden rule in software is **do not bite off more than you can chew**. The whole industry is tarred by those who promise the world and don't deliver. We developed a different philosophy. We never promised something unless we were certain to achieve it, and we encouraged potential customers to simply try what we had right now, and see if it would be useful to them. I would urge all in the industry to follow this model. This way, you get clients who trust you, and recommend you to their friends. We have lost new business along the way by refusing to make

(continued on page 10)





aggressive promises, but I am convinced it is the right way to proceed.

We proceeded by concentrating on particular market segments. We could not be all things to all people, especially at first. So we built out our portfolio gradually - first conventional products, then UL, then Par, then Disability, then Assets and so on, at the rate of about one new product line or module per year. We started off with just one country, then two, until today we operate on five continents.

Most of all, we concentrated on keeping our current customers happy. When you write closed code, you are taking on the responsibility of providing good service and fast response time. In our system there is only one code base and everyone gets the same software. So every client gets the benefit of each new feature no matter where the request came from.

OK, there's a problem right there—what about secret new features you don't want your competition to know about? This does happen, although as you know, there are few secrets in this industry. Suffice it to say there are various ways to solve this problem, and we can build in enough flexibility so that proprietary product features will not be given away by the software.

This is where the closed code approach pays off in a big way. If you have just one code base, then you can have many users reviewing and validating the calculations. It helps enormously when regulators and consultants review your software and during acceptance testing at each new client. If you have something wrong or missing in your code, you're going to find out and have a chance to fix it. This simply is not the case for open code. Think for a minute about the SOX implications here, their importance simply cannot be overestimated. Actually this is not a consequence of closed code but of common code. By going open code you simply preclude this level of scrutiny.

If you ask actuaries what is the biggest problem they face with their software, the most common answer will probably be the problem of keeping it up to date, the problem of conversions. Let's examine how these work on a completely closed system and a completely open system. On a completely open system, the vendor can really only provide changes to the framework and some new sample code. Changes to the framework must be limited or they will disrupt the current implementations. So the onus is on the developer to come up with the perfect framework for all time on day one. Let's be honest here. How many system architectures from 15 years ago are currently state of the art? If you got something wrong on day one, you can't always fix it later because users will have built their application around your architecture—change it and it breaks their applications. Now in the closed code context, we can provide automatic conversions of user models no matter what changes we make to the architecture. We have changed from DOS to Windows, from Basic to C++, from FoxPro to JET, and from a separate system for each country to a unified system, all without breaking the users' applications. Sure it takes a lot of work on our part, but the advantages are overwhelming. Users get a system that can be kept up-to-date with all the latest technology and functionality without massive conversion problems. They can upgrade in hours and not months. This is one of the biggest justifications for the closed code approach.

Now I promised to tell you how we tackle the need for flexibility, given we don't have infinite resources. Actually, it really does take an enormous effort to build in all the features clients require, and we have a very large and highly skilled workforce here we would not need if we were an open code shop. We have found that in most areas of the actuarial system we can provide enormous flexibility through the switches, scalars, tables and



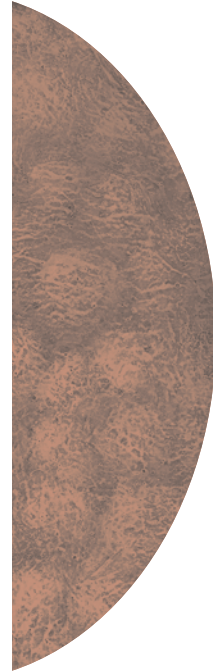
Reprinted with permission by Google Images. It is an old painting by John Stubbs sometime between 1543-1591.

objects we have built up over the years, some at our own instigation and some to meet user request (about a 50:50 split). But there are some types of logic that are very hard to accommodate in this way. Take polycholder behavior for example, or crediting rate strategies or experience refunds.

Some closed code systems are really closed down tight. Others allow you insertion points, and with the aid of a compiler or by using a non-compiled language, they allow you to change the source code. The first option seems too rigid to me, although it is what we offered for a number of years because we felt the second option simply negates most of the advantages of closed code systems. Then one day our developers came up with a third way that provides the advantages of open code wherever you need it most, but preserves the integrity of the source code and allows for full automatic conversions between software releases. I won't go into too much detail, but the breakthrough involves treating user code as data input to the system, and some very sophisticated use of the .NET framework. The cost of having this type of expertise in house may be prohibitive for the end user.

If I were starting over today, would I do the same again? I'd have to say that from a financial and marketing point of view maybe not. It is cheaper to develop an open code system and probably easier to sell. But when I think of the client's best interest, I would have to say yes.

For less ambitious software projects where the scope is more focused, the balance may be completely different. Each case should be taken on its own merits. So the golden rule is **there is no golden rule**. Choose instead the right horse for your particular course. 🐾



Phil Gold, ASA, FIA, MAAA, is a founding partner of GGY and past chairperson of the Technology Section. He can be reached at Phil.Gold@ggy.com.

Scenario File Format Project Update

by Carl Nauman

Earlier in the year the section formed the Standard Scenario Format Working Group. The purpose was to provide a tool for actuaries to store and exchange economic scenarios using a standard format.

The working group has been meeting about twice a month since July and has made steady progress. The first step of the project was to decide on the format to store scenario data and then design the layout.


Although nothing is written in stone, this is the direction the group is heading.

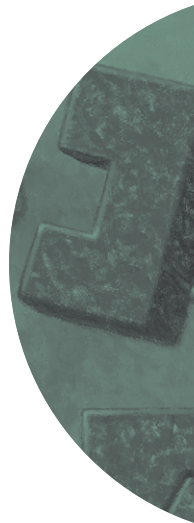
The data will be stored using XML format. XML was chosen for its flexibility. Due to the wide variety of scenario formats in use, a data storage format able to accommodate the diversity is required.

With XML, it will be possible to handle not only yield curves that are completely specified but also curves that are specified with sample points or parametrically. As for equity returns, various types are able to be stored including those defined by the user.

A "Scenario" is taken to be a collection of economic conditions. The collection would be multiple economic conditions keyed by calendar date and by country. Economic conditions comprise various yield curves (bond/spot/forward), equity return rates, inflation rates and other economic indicators.

The working group is close to finalizing the layout of the XML file. The next step will be to design and create the application programming interface (API) to the data. This will be a set of publicly available program modules that other software can incorporate to gain access to the scenario data. Besides just reading and writing the file, the API would provide routines to provide, for example, any point on a bond, spot, or forward yield curve for a specific calendar date.

There is still much to be done but the hope is to have a first version of the tool ready by mid-year 2007. 



Carl J. Nauman, ASA, is a consulting actuary with GGY AXIS in Toronto, Ontario. He can be reached at Carl.Nauman@ggy.com.

Noteworthy!

Kevin Pledge (Vice-Chair, Technology Section) was featured on World "Business Review". The show, hosted by General Alexander Haig was broadcast in October on CNBC and Bravo. John McGarry and Kevin Pledge, interviewed in the studio, discuss some of the information challenges facing insurance companies and how these are addressed with business intelligence systems. The show also featured on location footage with two other actuaries: Neil Lund from UAFC (Florida) and Lyne Francoeur from Standard Life (Quebec).



You may also have the opportunity to see the show as it will be shown on United and Delta Airlines as in-flight programming in April or May.

Microsoft Launches Experience Insurance Initiative, Aimed at Improving Insurance Customer, Employee and Operations Experience

More than 40 Microsoft Insurance Value Chain partners are developing solutions supporting the experience Insurance initiative

Earlier this year, at the ACORD LOMA Insurance Systems Forum, Microsoft Corp. furthered its commitment to the insurance industry by launching its experience insurance initiative, offering a comprehensive approach for addressing market pressures facing the insurance enterprise. The initiative is focused on significantly improving three key areas: the insurance companies' customer, employee and operations experience.

Microsoft's experience insurance initiative features a wide variety of activities including sales training, partner development programs, customer advisory councils, experience insurance business and technical briefings, and in-market and Web educational events.

The initiative integrates business value and architectural vision into new programs and new methods of interacting with Microsoft that are critical in its continuing efforts to become the preferred enterprise provider for the insurance industry. Founded in ever-increasing customer evidence of the business return on investment derived from the Microsoft® platform and Microsoft's industry-leading vision for service-oriented architecture (SOA), the initiative will enable customers to better determine which Microsoft technologies and partner solutions are best for them by understanding specifically which areas of business improvement they are best suited for. Its primary focus is to enable Microsoft to change the "experiences" that its enterprise customers are trying to create or achieve.

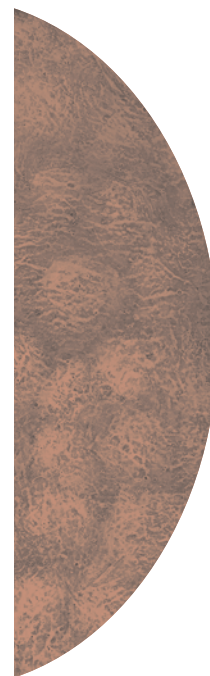
"Operating in an environment as complex as insurance demands an approach that can streamline workflow, improve operations, increase productivity and reduce costs," said Kevin Kelly, managing director of the U.S. insurance industry for the Financial Services Group at Microsoft. "The experience insurance initiative is designed to help insurers better understand and access software solutions that can reduce the complexity of the insurance industry and fundamentally change the customer experience, the employee experience and the operations experience.

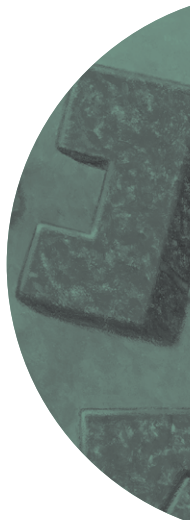
"Instead of dealing strictly in technical decisions at the end of a business analysis, we will now be able to engage the customer in joint decision-making and planning regarding the business value to be derived from Microsoft technologies and partner solutions up front," Kelly said. "This program allows us to provide better advice and counsel to our customers on a wide array of assets at their disposal for improving their business results."

Microsoft's Partner Program: The Insurance Value Chain

The experience insurance initiative builds on Microsoft's ever-growing ecosystem of Insurance Value Chain (IVC) partners that address key areas across the insurance industry—from point of sales and service to policy administration and claims processing—by leveraging Microsoft's core strengths in consumer, industry-specific and enterprise technologies.

(continued on page 14)





The IVC has grown from a handful of insurance application partners to more than 40 member insurance solution and service providers, with the common goal of providing innovative straight-through processing for insurance on the Microsoft platform using Web services, Extensible Markup Language (XML) and industry-specific standards.

The Insurance Value Chain initiative is composed of these simple yet powerful activities:

- Sourcing third-party applications that are best-of-breed examples of insurance business processing, ranging from point-of-sales-and-service all the way through re-insurance.
- Providing those partners with a framework for integrating their applications based on Web services and industry standards for forms and EDI transactions.
- Pairing and pre-integrating these applications over time, in an effort to reduce millions of dollars of post-purchase integration pain on the part of insurance customers.

Microsoft's Commitment to the Insurance Industry

Microsoft's Insurance group has grown from a U.S.-focused team to a globally focused team in under 10 years. Microsoft announced that it would increase the size of its U.S. sales force and its commitment across all vertical industries, including insurance, adding up to 10 percent more industry sales and product specialist positions nationwide.

Microsoft plans to invest almost \$7 billion (U.S.) in research and development this year on technologies that fulfill its evolving vision for the future of computing. The company also has created an internal group focused on understanding the long-term vision of its companywide R&D efforts and deciding how these

efforts apply to vertical industries, such as insurance. The group is playing a key role in demonstrating how innovations based on Microsoft's current and future technologies can come together to meet evolving business needs. More information can be found on Microsoft's Web site.

Google Announces Google Docs & Spreadsheets

Ever found yourself trading e-mail attachments with several colleagues, trying to collaborate on a document, only to have someone chime in at the last moment with corrections to an outdated version? Or e-mailing yourself a document just so you can move it from one computer to another? What about trying to manage a large guest list—say, for a wedding—when you have updates coming at you from all different directions and at all different times of the day, with other people trying to make their own changes?

In October, at the Office 2.0 Conference in San Francisco, Google launched a solution to these collaborative and document-management challenges. Google Docs & Spreadsheets is a web-based word-processing and spreadsheet product that makes it easier for people to create, manage, and share documents and spreadsheets online. Google Docs & Spreadsheets integrates Writely and Google Spreadsheets into a single, easy-to-use product that takes an innovative approach to a very specific problem in the productivity-software space: enabling people to manage and collaborate on the documents and spreadsheets they rely on in their personal and professional lives, no matter where they are or when they need to access them.

With Google Docs & Spreadsheets, Google is taking a set of important tasks and offering an online solution to completing them individually or with a broader group. With a Google Account, a compatible web browser and an Internet connection, users will now easily be able to:

- Create documents and spreadsheets, and then manage and access them in a single, secure location
- Easily collaborate with others, online and in real time
- Export to and import from a wide variety of file formats
- Share them with others as view-only
- Publish them to a blog or as an HTML page

Simply put, Google Docs & Spreadsheets is focused on providing users with an innovative and efficient way to create and share information on the Web.

There are many products in the productivity-software space, addressing a broad range of user needs, and Google Docs & Spreadsheets is intended to both complement these existing solutions and introduce new ones, adding its collaboration and document-management features to the productivity options people already enjoy.

Google Docs & Spreadsheets is currently in beta, available for free and open for sign-ups. To learn more about the product, visit <http://docs.google.com>.


Happy Birthday XML

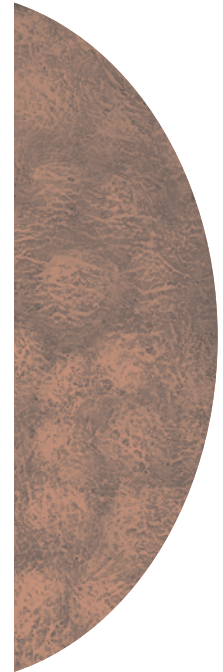
Extensible Markup Language (XML) is arguably the lingua franca of the Internet. And even as it marks its 10th birthday, it continues to advance the exchange of digital information both on and off the Web.

XML owes its success to a variety of factors. Because its message format is independent of the processing software, XML allows heterogeneous computer systems to communicate with each other. Thus, one can add systems or change their function without having to change the messaging mechanism. And

because XML is based on open standards and backed by a set of active standards groups, it is stable, interoperable and extensible.

The summer issue of the *IBM Systems Journal* (available online at <http://www.research.ibm.com/journal/sj45-2.html>) is dedicated to celebrating XML's first decade. The journal's keynote paper, "Technical context and cultural consequences of XML," traces the evolution of XML and puts the XML phenomenon in its technical context. The authors posit that the development of XML, which they refer to as a "code of integration," is a significant milestone in computer science that will have substantial economic, political, and cultural impact.

Noting that XML "gained notoriety in the hey-day of the 'dot-com' frenzy," the authors say that the language "continues because it worked and created vast new opportunities." It enabled information reuse by integrating text and data from different sources and by searching and linking across these sources, thereby breaking down traditional silos, which were barriers to information sharing. The Web became a vortex for this confluence of forces and allowed people to get a glimpse of the tremendous potential of universal access to information." 



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.

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.

Actuaries

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
April 1, 2007	January 15, 2006
July 1, 2007	April 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 Joe Adduci, 847.706.3548 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.

