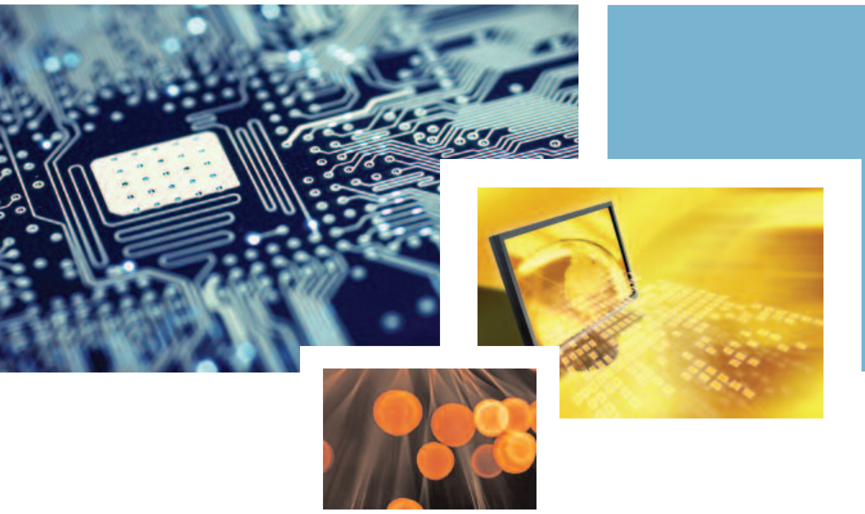
TECHNOLOGY SECTION

"A KNOWLEDGE COMMUNITY FOR THE SOCIETY OF ACTUARIES"

CompAct

Issue No. 25 • October 2007



Inside

Letter from the Chair 2
by Paula M. Hodges

Spreadsheets and Specifications 4
by Howard Callif

Illustration Software Testing—Part 1: Interface Testing 7
by Joe Alaimo

Illustration Software Testing—Part 2: Calculation Testing 11
by Joe Alaimo

High-Performance Computing: the Modeling & Trading Platform of the Future is Here Microsoft is Enabling Secure, Affordable HPC Solutions across the Enterprise 16

GPS—Don't Leave Home Without One!! 19
by Paula M. Hodges

Wireless Print Server: The Next Device on your Home Network? 22
by Shyamal Kumar

Book Review: Data Quality and Record Linkage Techniques 23
by Kevin Pledge

Actuaries
Risk is Opportunity.℠

Letter from the Chair

by Paula M. Hodges

It is with very mixed feelings that I write this, my last letter as Chair of the Technology Council. It's hard to believe that it's been three years since I was elected to my term on the Council. During that time, it has been so exciting to see the work done by the active members of our Section. I'm disappointed that my time on the Council is coming to an end, but very pleased with the new leadership that is coming on board. I leave the Council in the very capable hands of Kevin Pledge, who will be taking over as Chair during the 2007-2008 Council year.

If you've been following the section activities, you can feel the excitement that is growing in our area of the SOA. There are so many people who have contributed, I risk missing someone. However, I'd like to publish my thanks to some people by name. First, thanks go to the outgoing board members who have served on the Council with me over the last three years: Dean Slyter, who has contributed on the Web site and on the XtbML standard table format; and Nariankadu Shyamalkumar (Shyamal), who has helped breathe life back into our newsletter, and ensured its delivery on a quarterly basis over the last two years.

Also, thanks to our current Vice-Chair and incoming Chair, Kevin Pledge, for his endless enthusiasm and ideas on how our section can do more, and more, and more … Last year, Kevin put together a spectacular agenda of sessions for our annual meeting. This year, he's spearheading the development of a collaboration portal for our section.

Other Council members serving their last year during 2007 and 2008: Van Beach who is actively working on coordinating a webinar to highlight the benefits and approaches for record linkage; and Tim Rozar who was our Web coordinator last year and spring meeting coordinator this year.

Our Council members who began their terms last year at this time:
David Minches: Annual meeting coordinator for 2007.
Carl Nauman: Survey coordinator and Council representative on the Scenario File project.
Joe Liuzzo: Secretary/Treasurer and currently planning ongoing maintenance for the SOA's Table Manager.

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

Other significant contributions were made this year by:
Steve Strommen (lead), Joe Alaimo, Chris Clark, Erin Cole, Steve Craighead, Mark Horowitz, Carl Nauman, Anton Pineda, Michael Pustylnik, Tan Vu Nguyen—Scenario Generator file format team members.

Gary Lange—Speculative Fiction Contest.

Carol Marler, Steve Rubenstein, Shiela Silva—Education Committee.

Of course there were many others who contributed articles to our newsletter, were speakers at the SOA meetings, or helped the section in another way. My thanks go to you as well.

Thank you, one and all. So much progress has been made, and it couldn't have been done without some great teamwork. I'm looking forward to thanking some of you personally at the Annual Meeting in Washington, D.C. See you there!!

Paula M. Hodges
Chair—Technology Council

From Incoming Chair of the Technology Council, Kevin Pledge

At the Annual Meeting in October, Paula Hodges will be completing an outstanding year as chair of the Technology Council.

Over the last year the Technology Section has built on initiatives from previous years and started some new ones. On the networking and communication front we have provided networking opportunities at various meetings, met our objective to produce four newsletters over the year, updated our section of the SOA Web site and recently completed a membership survey. We have provided input on enhancing technology education for actuaries, contributed first-class sessions at SOA meetings and, thanks to efforts from Van Beach and Thomas Herzog, we will soon be sponsoring our first webinar. The Scenario Format Project has developed an XML format for sharing scenarios, as well as developing a utility reading, writing and viewing scenarios. We have also made progress to secure the long-term management of the Rate Manager—a tool, originally developed by Steve Strommen that so many actuaries rely on. Finally, we have started to investigate portal technology to improve our internal communication and management.

Thank you Paula for your leadership on all these projects. As Paula hands this responsibility over to me, my immediate goal is to ensure these projects stay on track. New initiatives will be based on the survey results and feedback from you. So please contact me or any council member with your thoughts.

Two other council members are also completing their term on the council—Dean Slyter and Shyamal Nariankadu. Over the past three years they have both made significant contributions to the section. Shyamal deserves special thanks and recognition for his role as newsletter editor; when he took on this role over two years ago, we hadn't published a newsletter for some time—maybe one or two over a couple of years. Under Shyamal's guidance, CompAct has developed into a high-quality, regularly published newsletter.

As we look forward to the coming year, I need to put a call out for volunteers. Our council and volunteers are a special group of people dedicated to furthering the goals of the section, but we are stretched thin from all the projects we have on the go. Shyamal has completed his term as newsletter editor, so we are especially keen to fill this role and to support this with an editorial committee.

You don't have to be elected to the council to get involved with the section by taking on roles such as newsletter editor, meeting coordinator or to volunteer in other ways. So, if you have an interest in any aspects of technology and you want to volunteer, please contact me (kevinpledge@insightdecision.com, 905.475.3282 x2#) or any council member. A full list of our council members can be found on our Web site.

Spreadsheets and Specifications

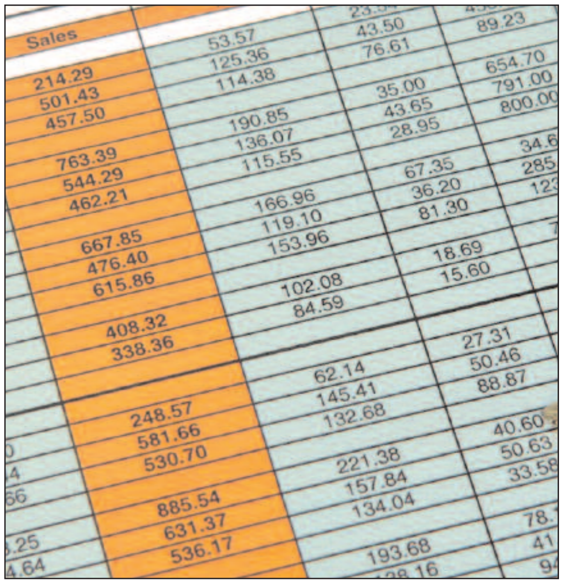
By Howard Callif *(with contributions from COSS staff members Kevin Cottrill, Kenneth Hanson, Thomas Rhode, David Schultz and Stewart Shay)*

Do you create or use spreadsheets? Do you need to provide specifications for your Information Systems department or external companies to implement? It is very common for documentation to be either ignored or done last when developing products or implementing policy changes. I believe actuaries would benefit from more training on creating specifications, and this is the first of a series of articles to help make this task easier. I would like to tailor the articles to your needs, so please provide comments and feedback on what you think would be most helpful.

This article will focus on spreadsheets as specifications, and will provide a wide range of tips and suggestions to make your spreadsheets easier to understand, and as self-documenting as possible. Follow-up articles will focus on what real specifications look like, and why they are so helpful.

All spreadsheets are specifications to a certain extent, since they are usually the benchmarks systems are tested to. Since this is often the first place calculations are created, specifications will often be created from the spreadsheet. In a worst-case scenario, the spreadsheet becomes the specifications document, which is handed to an internal systems area or an outside vendor!

For all these reasons, it is important to make the spreadsheet as clear and understandable as possible, and include documentation on usage and construction. This article will provide a few guidelines, focusing on the most important suggestions and best practices I've seen. In fact, following these tips will generally result in smaller spreadsheets, shorter recalculation times, and will make you (or someone else picking up your work) more productive.



- Start on the right foot! Name the spreadsheet appropriately, and include a version number in the name! Have a sheet dedicated to Release Notes, and mark changes as you make them (especially after there are several versions of the spreadsheet, or several people have a copy). If external references exist, note them, and include them when you send the spreadsheet to someone. If you start from an existing spreadsheet, add this if it is missing. There is no easy way to show differences between two different spreadsheets, so notes are critical, and can quickly help explain what has changed.

- Organize the spreadsheet! Keep fields that are associated together. Place inputs in one column, in a separate sheet, with a blue background for each input. This will make it easy to store a set of clients, to avoid reentering cases, and enable running a batch with a macro. Try to keep dependencies on the left, so the calculations flow, and formulas reference cells in an organized fashion.

Howard Callif is a senior system architect at COSS in the Illustrations unit. He can be reached at howardc@cross.com.

There are some other ways to organize information:
Use named fields as much as possible. This makes formulas much more readable! You can easily create a table of named fields, which documents the ranges names and cell references (In Excel 2003, you click on the menu item Insert, then Name, then Paste, and then click on Paste List, and a list of range names and reference cells will be created starting in the currently selected cell).

Use selection lists for items that have specific inputs (Yes/No, etc.). Validate data, or at least include notes on what is valid. Even if it is a text comment, it is much easier to be told Y/N in the input, than to have to figure out what the input should be. If possible, use the marketing names for fields, or note them next to the selection. Underwriting classes are notoriously difficult to identify, but there are other instances where not everyone uses the same terminology.

Numerical codes simplify lookups, and minimize "If" statements within the spreadsheet. The preferred approach is to use selection lists as input fields, and then convert that to a numerical value in a cell to the right. This is an example of the Do It Once rule we will discuss below. For example, underwriting classes are usually letters (or a selection list), and converting the class to a number (using a formula, table, etc.), will enable rate lookups to use the class number directly. Coding hundreds of rows of "If" statements looking for preferred takes up space, is difficult to read, and is harder to update.

Code all formula dependencies in the spreadsheet! It is common to have circular references that are not apparent, frequently leading to serious implementation problems for administrative or other systems. Many times features such as solves are not implemented in a spreadsheet, because they are too complex. However, many dependencies can be coded with a simple "If" statement. For example, Target and Minimum premiums are frequently calculated, but the formula to select them as a premium input is usually not entered, so premium on an illustration is usually not a dependent reference. Doing so would identify a problem with a UL policy that has a minimum premium depending on the charges in the first year (which depends on the premiums paid).

Don't be afraid to start a new sheet! Don't put calculations that don't fit in the bottom right corner, a separate sheet is easier to find and ignore, as the case may be. There is certainly a tradeoff between using a new sheet, and putting a large number of calculations in one place. The audit functionality cannot be used to track formulas on different sheets, and it is a shame to lose this very useful feature. However, many times there is too much information on one sheet, and it becomes almost impossible to find information. A sheet labeled COI is very easy to find, and the total can be gathered to another sheet where detail is not shown.

Use named fields as much as possible. This makes formulas much more readable!

- Two complementary rules: Avoid duplication, and not too much in one line. Usually breaking one of these rules leads to breaking the other. Have you ever seen a spreadsheet with nine rows of text in the formula? It is not easy to debug or evaluate. Many times, the same condition is tested repeatedly, or the same value is calculated in several conditions of the formula. Have one column test the condition (is the rider on, are we in corridor, etc.), and then create a second column to calculate the correct value. For example, instead of trying to calculate all of the riders in one cell, split them apart into separate columns, and put this detail in

(continued on page 6)

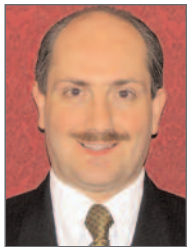
a sheet labeled Rider Detail. Refer to the total in the main sheet, and make a note where the detail can be found. Range names can make this clear, easy to navigate to, and easy to reference.

- Use comment fields, especially for input fields and numbers. Examples and formulas can be combined with range names, to provide a powerful and clear view of what is being calculated, and how.

- Show the precision used in the calculation or include a note if more precision is used than is shown. Rounding in Excel is different than most programming languages (and it occurs in intermediate calculations), so if you have a say in the formula, avoid rounding until the last step. Note that when summing, always sum the smallest to largest number.

- A few additional miscellaneous tips: Do not hide any cells, columns or sheets. Set the print area for each sheet, and try to limit it to letter size, large enough to read without a microscope. If a spreadsheet is not completed, please include notes on what isn't working.

I hope you find these hints and ideas helpful, and that this article will spur you to integrate additional documentation into your spreadsheets. This is certainly not an all-inclusive list, but if you have some ideas or hints that you think are critical, please forward them to me.



Winner of the CompAct Article of the Year Prize 06-07

Two-part series of articles on Illustration Software Testing by Joe Alaimo, president of ProComp Consulting, has been chosen by the Technology Section Council as the CompAct Article(s) of the Year 06-07. Joe receives an iPod Shuffle for his much appreciated and informative contribution to CompAct. Congratulations Joe!

CompAct Article of the Year Prize 07-08

The Technology Section Council will choose a CompAct article published between June 2007 and May 2008 for the award of the CompAct Article of the Year Prize 07-08. The author of the chosen article will either receive an iPod Nano 4GB or equivalent MP3 player. In the case of multiple authors, each author will receive the stated prize.

Illustration Software Testing—Part 1: 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

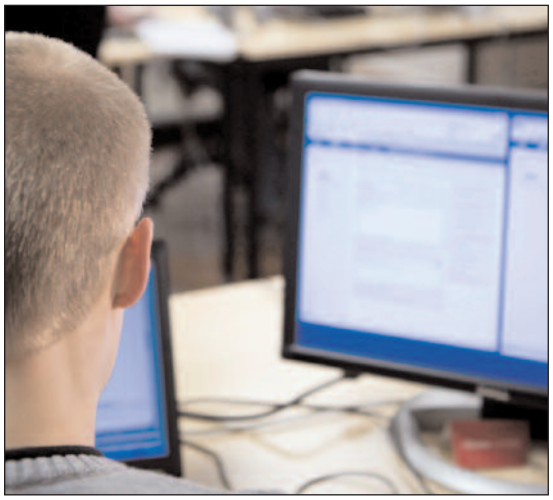
There are a number of tools that are used by quality assurance professionals to perform automated testing. Some of the more common tools are WinRunner, Test Complete and Rational Robot.

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.

(continued on page 8)

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 in performing 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 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

(continued on page 10)

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.

• continued from page 9

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.

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!

Article of the Year 06-07

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 calcu-

(continued on page 12)

lation engine is correct. The solution to 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.

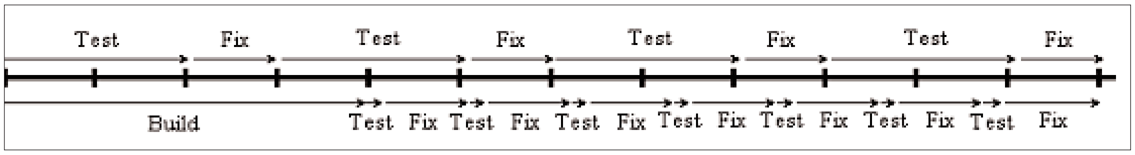
By the end of the fifth week, Team A has completed their second testing cycle and submits their problems to the developers. Team B is still awaiting the developer fixes from the first cycle.

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 10TH 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 illustration 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. 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

(continued on page 14)

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 nonsmoker, 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."

High-Performance Computing: The Modeling & Trading Platform of the Future is Here

Microsoft is Enabling Secure, Affordable HPC Solutions across the Enterprise

Reprinted from Windows in Financial Services • www.windowsfs.com

Neil A. Cowit HPC solutions specialist, Microsoft.

According to Neil Cowit, New York-based HPC solution specialist for Microsoft Corporation, Microsoft Windows Compute Cluster Server 2003 provides a secure, cost-effective solution for the compute-intensive requirements of financial services.

In a recent interview with Wall Street & Technology, Microsoft CEO Steve Ballmer said, "The way in which performance gains are achieved on computer chips has changed, creating a new challenge for HPC software. For years, Intel just kept doubling, doubling, doubling clock speeds. Now they don't double clock speeds anymore—they give us twice as many cores or processors. Figuring out how to take any application and parallelize it so it can run across multiple cores will be a key, not just in high-performance computing and not just in financial scenarios, but in all applications figuring out how to exploit the increase in power that physics is giving us."

We checked in with Cowit for an update.

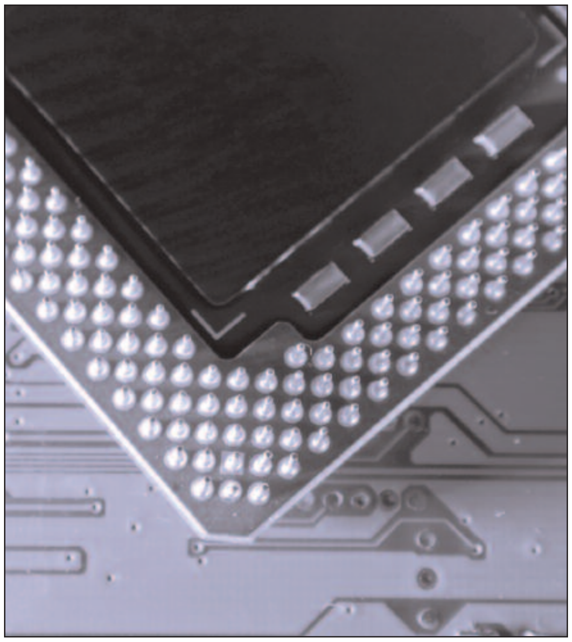
WFS: How has Microsoft's HPC offering been accepted since its launch last August?
NC: Quite nicely, thank you. Then again, running compute-intensive work in a Microsoft-based environment isn't new in financial services. A large Microsoft user base has existed for several years with some of the largest banks using Microsoft for their dedicated HPC-computing requirements.

What is new is Microsoft's focus in this area in terms of personnel, product and partners. Third-party firms DataSynapse, Digipede and Platform Computing, all Microsoft partners, have large groups of customers. More recently, HP, Dell and IBM announced specialized sales teams and configurations that best fit their products, plus a number of third-party software firms have begun porting their products to Windows Compute Cluster Server.

WFS: What is Microsoft's value to customers building their HPC environment with Microsoft?
NC: Running an HPC environment on Windows isn't simply about technology. It's about the ROI through what I call the economies of familiarity.

WFS: Meaning?
NC: Customers can leverage the skill sets of their established Microsoft operations and development teams to build, run and manage the applications of an HPC environment. Employees are able to access, manage and report on HPC results via a familiar, Windows-based environment, making them more productive by performing jobs and reporting the results via Microsoft Office programs.

We've also simplified deployment and management of an HPC infrastructure so you can come to one place and get an integrated environment of tools for HPC. It's all about high performance and high productivity. For example, customers familiar with using Microsoft Operations Manager and related management packs to monitor the health of their IT systems can use the same tools for Windows CCS.



We've added familiar management tools, such as a parallel shell, to the Compute Cluster Pack to make management of Microsoft-based HPC environments even easier. Plus, developers can leverage the familiar environment of Microsoft Visual Studio 2005, building and deploying parallel, Microsoft .NET-based apps to meet changing business needs.

WFS: Does all of this apply only to capital markets?

NC: No, insurance and banking embrace HPC to refine their models to reduce risk, shorten time-to-market when creating new products and improve overall operations performance.

WFS: What types of workloads do you see people running on Windows-based HPC environments?

NC: We see attention focused on real-time derivatives pricing, risk management modeling and the like. For example, a European customer's structured derivatives business built its second-generation HPC cluster using Windows CCS to speed pricing and risk management applications and simplify development and management—improvements that allow customers to increase the scale of their business and shift the focus of IT resources on solutions for that business.

WFS: Any other examples?

NC: We also partner with software and hardware vendors to get latency down as far as it can go. Let's not forget all the behind-the-scenes batch work completed before the next business day, as well as those firms using HPC to migrate work off mainframes.

WFS: What are some of the HPC challenges customers face today?

NC: Well, parallel programming continues to be somewhat of a black art in financial services. The amount of heavy-lifting necessary to integrate an existing application to HPC can be considerable, but the rewards are high.

Microsoft added tools such as a parallel debugger into Visual Studio 2005 to speed the development process, proof that the industry must continue to develop software development tools to ease the task of parallel programming and Microsoft is making significant strides in this area. Firms like Digipede also have some rather exciting .NET development and deployment tools that speed integration.

WFS: Doesn't Microsoft Compute Cluster Pack compete with existing vendors, including DataSynapse, Platform Computing and Digipede?

NC: No, they're cream-of-the-crop in schedulers, Microsoft supports them and they run very well on Windows Server 2003 Compute Cluster Edition today. Interoperability is something that has been part of the design goal from the beginning. Looking back to Supercomputing 2005, Bill Gates spoke and demonstrated the sharing of work between different scheduler vendors and different operating systems in different parts of the country. At Supercomputing 2006, we demonstrated job scheduler interoperability with Platform Computing using a new industry specification.

And yes, interoperability also means sharing work between Linux and Microsoft HPC environments.

(continued on page 18)

WFS: Is there a business model for systems integrators to be successful with HPC?

NC: HPC benefits every business in the Microsoft value chain. Our model's always been built around the system integrator and application developer who bring a specialized capability to the table. Whether it's departmental- or application-based deployment or a firm-wide utility-based deployment, applications do not magically integrate into HPC environments. Skilled developers are necessary, which is why we're working with integrators Infusion Development, Vast Computing, Avanade and Polaris and other certified partners.

WFS: What's most difficult in deploying Microsoft grids?

NC: The difficulties are not in deploying Windows. It's in how you deploy and manage an HPC infrastructure within your company. You must understand the political nuances, some of which revolve around deploying a utility versus a silo approach, and in defining a charge-back mechanism that doesn't run counter to your goal of cost-effective computing.

Customers also need to strike a balance between tight controls to achieve desired SLAs and providing flexibility for innovation.

WFS: Where do Excel 2007, Excel Services and HPC come into play?

Excel Services opens up a whole other world of opportunity.

NC: Excel's the most well known, most utilized application in financial services. Thanks to the advancements in Excel 2007, multi-threading enables you to put Excel on steroids. The limits on a spreadsheets' size has been greatly expanded while multi-threading allows you to set the number of threads you need to perform calculations in parallel.

Excel 2007 also identifies the parallelizable code areas the first time it runs through the sheet. The second time the sheet is run, those areas that can run in parallel will then run in parallel.

WFS: Take that a step further.

NC: User-defined functions can be created. Then these computations can be distributed outside of the desktop to Windows CCS.

Excel Services opens up a whole other world of opportunity. Being able to protect your IP, deliver a single version of the truth through a solution built on Excel using HTML and deliver performance by distributing computationally intensive work through user-defined functions to Windows CCS is a solution satisfying a lot of customers. If you work with parallel computing, incredible performance is within reach, and in many cases that speed can translate into profitability.

WFS: That's a lot to absorb.

NC: There's a lot to share …

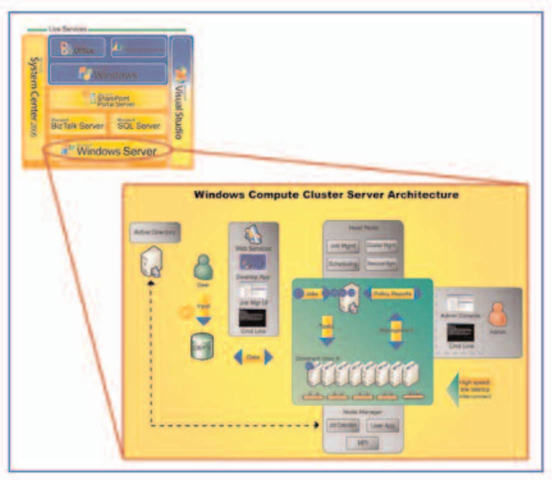
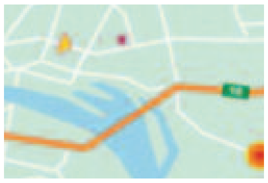


Illustration caption: High Performance and High Productivity – Windows CCS is designed for HPC environments. The existing corporate infrastructure and Active Directory are used for security, account management and operations management. Windows CCS frees developers and administrators to provide value through their domain expertise versus spending time building and maintaining HPC clusters.



GPS—Don't Leave Home Without One!!

By Paula M. Hodges

The GPS is one of those revolutionary tools that are changing the way the world operates. If you haven't yet personally explored the power of one of these cool gadgets, read on, and find out more about what these little gems can do for you.

The GPS is no longer being used by just the "early adopters," but is truly becoming a mainstream purchase as the prices have plummeted over the last two to three years. I expect GPS units to be a very popular item during the holiday shopping season this year. Just in case you're a bit behind the curve, a definition might be in order. GPS is an acronym for Global Positioning System—which allows ordinary people to use satellites to know exactly where they are, right now. Depending on the sophistication and purpose of the particular GPS device, your position is integrated with mapping software, allowing you to not only know where you are, but how to get to where you want to be.

I was first introduced to the world of GPS from my husband, who is an amateur pilot. He used GPS navigation systems while flying. So, we've had GPS units come and go through our household for about six years now. We are down to two primary units now—our TomTom unit that we use in the car, and my GPS running watch.

The TomTom is an example of the most popular type of GPS unit on the market: a self-contained unit that helps you to navigate in your car. Garmin and TomTom are the leading brand names to watch. The TomTom won out in our household, with its easy-to-use touch-screen. If you do any significant driving in unfamiliar areas, or in a large city, this is a tool that can save you time and frustration, and it provides a strong sense of comfort that you know where you're going.

Situations where a GPS unit has helped us out:

- Taking a detour because of construction—the GPS maps a new route to the chosen destination in a matter of seconds.
- When traveling for leisure, an unplanned stop can be calculated into the existing itinerary. The GPS calculates the extra mileage, and how to get back to the originally calculated destination.
- The GPS can scan the route ahead for gas stations, restaurants and other points of interest (POIs) on the planned route ahead.

Cool Features I like

- Tells you how far until your next turn, and what direction it will be.
- Displays how many miles to your destination.
- Gives estimated arrival time. This uses parameters such as: the types of roads, whether it is interstate driving, city driving, mountainous terrain. It does a very good job of answering the question "Are we there yet?" or at least "When are we going to get there?"
- Our unit has its database of roads and POIs stored on an SD card, which we can plug into our computer and update from the Internet to include new roads, and new POIs.
- Options to change the voice—everything from a "New York cab driver" to "Lori" who sounds like the girl next door. Additional voices are available for purchase from the vendor's Web site.

What To look For When Getting Your GPS

- Play around with a sample model. See if you're able to just pick it up and use it. You want it to be easy to use in those situations where you're driving and navigating at the

(continued on page 20)

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

same time. If you get a GPS that is too complicated, it's not just frustrating, it's dangerous.

- Look for the accessories that are included in the base price of the various units, or how much additional cost for the items you want. In my opinion, the "must haves" are:
    - A carrying case, to transport it in your suitcase without scratching the touchscreen.
    - A bracket to mount it to your dashboard—you don't want to juggle with it when driving.
    - A cable to plug into your car's AC power so that you can use it on long trips without losing the charge.

Features I'd Change

- The menu system is intuitive, but there is no "easy out." After navigating to five levels deep in the menus, it's necessary to hit the "back" button five times to get back to the main screen.
- Add a display for altitude. When taking a road trip through the mountains, it's nice to know where we're at on a vertical, as well as a horizontal basis.

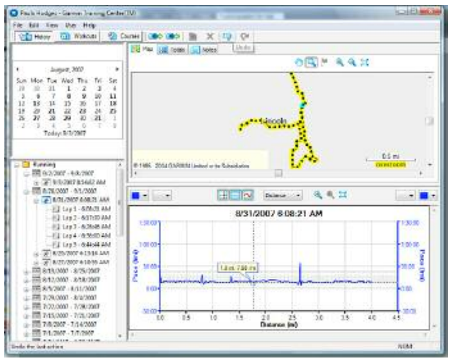
Watch It

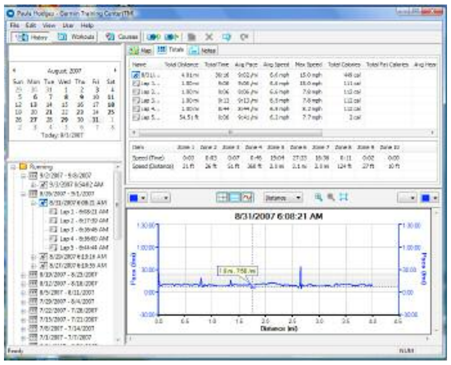
The other type of GPS that I'd recommend for hikers, bikers, runners, or other types of athletes is the GPS wrist watch. I received one as a gift last December and really enjoy it. I've run a few marathons before getting this gadget. As an actuary, I'm interested in watching statistically how my running performance is deteriorating now that I'm in my mid 40s, and this is just the tool to help me in that review.

The unit I own is the Garmin 205 trainer. While running, I'm able to monitor many metrics: my current speed, distance covered so far, elevation, and time elapsed. It also has all the features a good running watch has—lap counters, stop/start buttons, and a lighted display.

Cool Features:

- There are many options to customize when you want the watch to notify you when you're working out (these are all optional, and can be turned off):
    - you're going too slow.
    - you're going faster than a specified speed.
    - you've covered a specified distance (mine is set to beep every mile).
- The watch can be set to automatically "pause" when you're moving slower than a specified speed, then resume when you start moving again. This is nice when you stop at traffic lights—it stops the timer if you're moving very slowly or not moving at all. This enables me to just turn it on at the beginning of my run, and shut it off at the end. It does all the "pausing" automatically for me.
- After a workout, the information can be uploaded to your computer, and the ability to review performance graphically or numerically is built right into the program.





- Data and options for different sports can be kept separately on the unit. The built-in sports available are running, biking and other.

Features I'd Change

- I haven't found a way to export the data from the application (to Excel for example) so I can do analysis on my performance.
- The software needs to be updated for the Vista operating system.
- On overcast days, it takes quite a while to find the satellites, up to 20 minutes. When I want to hit the road, this delay is frustrating, and I have needed to start my run without taking the time to wait for it.
- The size needs to be reduced. I expect this will happen as the technology advances.
- It's not always easy to figure out how to change some of the options. It's one of those tools where you need to read the user's manual to find out what it can do.

The discovery of GPS technology hasn't changed my life, but these gadgets have added a layer of security when I'm traveling out of town. I have the comfort of knowing that I will be able to find my destination. My GPS watch has allowed me to combine my love for running with my innate desire to analyze everything. What could be a better gift?

If you're looking for a new gadget during your holiday shopping season, a GPS might be just the thing to buy (or ask for!)

Wireless Print Server: The Next Device on your Home Network?

by Shyamal Kumar

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

Searching for wireless networks in a small neighborhood in Iowa with less than 20 homes nets me about half a dozen (with two unsecured!). This empirical fact along with the popularity of portable computers makes me conjecture that a significant percentage of actuaries would have a wireless networked home. Assuming that you have a wireless network at your home, should you consider adding a wireless print server to your home network?

In our home we have a workstation (two years old) and two portable computers (one is six years old and the other is a month old), the recent addition running Vista and the others running Windows XP. And with the addition of the new portable, the need to be able to print from any computer was much felt. A quick fix was to make the workstation network share the printer, and hence make the workstation also a print server.[1] This approach would require the workstation to be switched on while printing, and switching it on every time you wish to print is a chore one surely wants to avoid. A solution is to leave the workstation working 24/7, which while not very energy efficient was not also feasible as I am too much of a paranoiac to leave my Windows box connected to the Internet. Besides, the best way to avoid security breach is to keep the machine off! This led me to search for a stand alone print server.

A stand alone print server could be either wired or wireless. Choosing wireless allowed the access point to be in a different location from the printer, and the one we bought is the DLink DPR1260 (RangeBooster G Multifunction Print Server). In the following I describe some of the nice features it has which led me to choose this over other competing products. First, it has four USB 2.0 ports. USB 2.0 transfer speed of up to 60 MB/s is about 40 times faster than USB 1.1, and hence much preferable even though most current devices do not use above 20 MB/s. The speed becomes more important if your printer, like ours, does not support postscript. One of the downsides is that it does not have a parallel port, but as our printer supports USB, and USB 2.0 will replace the legacy parallel port, it was a non-issue for us.[2] Second, it can also serve as an Ethernet bridge providing wireless connectivity to any Ethernet enabled device. In our case, the old laptop benefits from this feature as its wireless adapter supports only 802.11 a/b and not 802.11g standard. Of course, we sacrifice portability for the higher speed. Third, it supports many wireless security features which include WPA-PSK (WPA Home), WPA-EAP (WPA Enterprise) and up to 128-bit WEP encryption. We use WPA-PSK with TKIP encryption on our network (Linksys WRT54G) and required the print server to support the same standard. Fourth, it is a multifunction print server in the sense that it supports the scan functionality of multifunction printers. A short coming is that it does not support fax, which to us was a non-issue, but I can easily imagine this being an issue for some others. Also, the multifunction feature works currently only with HP products, and the scanning feature does not use the native scanning utilities.

The setup was easy. The only issue was it required a little extra work to get it to tango with our HP 1022 Laserjet. And we have been using it for a month now with no problems at all. If sharing a printer is becoming an issue at your home, perhaps its time for a dedicated print server!

[1] An article which provides the steps for doing this in XP can be found at http://www.microsoft.com/windowsxp/using/networking/expert/honeycutt_july2.mspx

[2] Parallel port has transfer rates about 2MB/s and is hence about thirty times slower than USB 2.0.

Book Review: Data Quality and Record Linkage Techniques

Authors: Thomas Herzog, Fritz Scheuren, William Winkler
Reviewed by Kevin Pledge, Vice-Chair of the Technology Section

Multi-million dollar reserves, pricing assumptions and the very survival of companies and pension schemes depend on the quality of the data used for analysis. However, little in our actuarial training prepares us for real-world data issues. As a result, data quality checks are often limited to testing averages, trends and spotting outliers, yet significant issues exist between the multiple systems that are used within an organization.

Finally, a resource is available to guide you through the data quality maze—Data Quality and Record Linkage Techniques by Thomas Herzog, et al. As you would guess from the title, is a comprehensive anthology of methods for joining records to improve the quality of the data. This book covers methods and models used to link records, supported with examples and case studies. The mix of mathematics, practical examples and case studies make for an interesting cover-to-cover read; the book is equally effective for readers who need to dive in to solve a particular problem.

Suppose you have more than one source of data; for example:

- employee records and claim records for an employee benefit plan that need to be joined, or
- an insurance company that has more than one administration system and the need to identify clusters of risk,
- you plan to enhance internal data with external data records, as discussed at the Technology Section session "Competing on Analytics" at the last annual meeting.

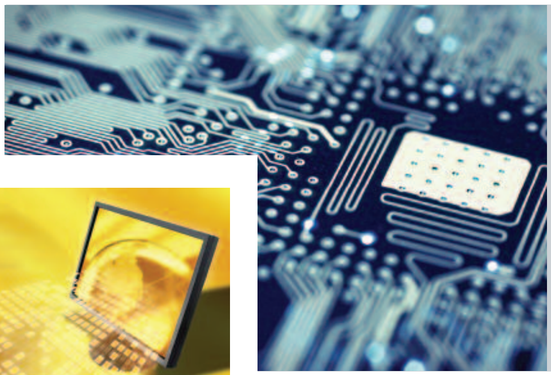
Data Quality and Record Linkage Techniques explains various techniques that can be used to join these records; not only does it explain the methodology behind the techniques, but it also discusses when each approach is most applicable.

The book finishes with a roundup of data quality software available. The discussion of software ties back to the techniques explained earlier in the book—this is critical as there is too often the tendency to naively implement software and trust it as appropriate.

Who should read this book? The short answer is everyone who is concerned about data quality and what can be done to improve it. Buy a copy for yourself; buy another copy for your IT support.

The Technology Section has negotiated a discount on the regular price of this book; please see the coupon included with this issue of CompAct. This offer is available for a short time only, so act quickly. If you are reading this online, after the offer has expired, you can still buy this book from Amazon.com or other book sellers.

Kevin Pledge, FIA, FSA, is president and CEO of Insight Decision Solutions in Markham, Ontario. He can be contacted at kpledge@insightdecision.com.



Technology Section Newsletter
Issue Number 25
October 2007

Published quarterly by the Technology Section of the Society of Actuaries

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

SOCIETY OF ACTUARIES

475 N. Martingale Road Suite 600
Schaumburg, Illinois 60173
www.soa.org