CompAct

Inside

```
  </TableReference>
     JIA vol 101, Part II or TFA vol 34
  </TableReference>
  <ContentType tc="4">Mortality Insured</ContentType>
  <TableName>
     1967-70 UK A1967-70(5), Male, Age nearest
  <KeyWord>Select</KeyWord>
  <KeyWord>Insured mortality</KeyWord>
  <KeyWord>United Kingdom</KeyWord>
<ContentClassification>
<Table>
  <MetaData>
     <TableDescription>
           1967-70 UK A1967-70(5), Male,Age nearest select
     </TableDescription>
</TableReference>
     JIA vol 101, Part II or TFA vol 34
  </TableReference>
  <ContentType tc="4">Mortality Insured</ContentType>
  <TableName>
     1967-70 UK A1967-70(5), Male, Age nearest
  <KeyWord>Select</KeyWord>
  <KeyWord>Insured mortality</KeyWord>
  <KeyWord>United Kingdom</KeyWord>
<ContentClassification>
<Table>
  <MetaData>
     <TableDescription>
           1967-70 UK A1967-70(5), Male,Age nearest select
     </TableDescription>
```

Society of Actuaries

CompAct Electronic Newsletter • Issue No. 16 • August 2003 • Published in Schaumburg by the Society of Actuaries

Computer Science Section Newsletter
Issue Number 16 • August 2003

Published quarterly by the Computer Science 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

This newsletter is free to section members.

Charles S. Linn, FSA
Computer Science Section Editor
MG-Triton Actuary
Milliman USA
80 Lamberton Road
Windsor, CT 06095
phone: 860.687.0157
fax: 860.687.2111
charlie.linn@milliman.com

Lois Chinnock, Staff Liaison
lchinnock@soa.org

Clay Baznik, Publications Director
cbaznik@soa.org

Bryeanne Summers, Graphic Designer
bsummers@soa.org

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 the CompAct, several associate editors are needed to handle various specialty areas such as meetings, seminars, symposia, continuing education meetings, new research and studies by Society committees and so on. If you would like to submit an article or be an associate editor, please call Charlie Linn, editor, at 860.687.0157.

The CompAct is published as follows:

Publication Date	Submission Deadline
September 15	July 15
March 15	January 15

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 Bryeanne Summers, 847.706.3573, at the Society of Actuaries for assistance.

Please send electronic copies of the articles to:

Charlie Linn, FSA
Computer Science Section Editor
MG-Triton Actuary
Milliman USA
80 Lamberton Road
Windsor, CT 06095
phone: 860.687.0157
fax: 860.687.2111
charlie.linn@milliman.com

Thank you for your help.

Chairperson's Corner

by Charlie Linn

Yes, believe it or not, this is an issue of CompAct, the newsletter of the Computer Science Section of the Society of Actuaries! We have not published an official newsletter since May, 1997. Subsequent to that newsletter, a decision was made to move to an all-electronic format for the newsletter. Seemed like a no-brainer. Computer Science Section....paperless newsletter. We wanted to be on the leading edge of technological advances. The Section Council also realized that rather than relying on the SOA staff to compile and publish an organized newsletter, all we would have to do is get people to write articles and we could post them on the Web one-at-a-time, as they became available. Great idea, or so we thought. However, the reality was that without an editor, a set schedule and specific deadlines, it was difficult to get people to agree to write articles and then actually follow through and write them. So, the newsletter went dormant.

However, we are now reviving the newsletter, and have a good collection of articles covering a wide range of topics related to computer science of interest to actuaries. First, Cindy Jeness and Jacques Rioux, members of the Computer Science Section Council, have provided a summary of a major project undertaken by the Computer Science Section, in conjunction with ACORD, the standards setting body, to develop and maintain a standard for table data storage. This project grew out of work begun a number of years ago by Steve Strommen to develop a common library of mortality tables for use by all actuaries. The new project with ACORD has developed a standard for table storage of all types of insurance data, not just mortality data. In addition, the Computer Science Section has sponsored a project by GoldenCode to develop a sample implementation of the new standard. Information on this sample implementation is available on the SOA Web site at http://www.soa.org/sections/compac.html.

We have an article by Steve Welander, offering a methodology for using Microsoft Visual Basic to automate the creation of a Microsoft PowerPoint presentation. This can come in handy for anyone who regularly creates a report for others using data available in electronic format, allowing them to "work smarter, not harder."

In addition, Jay Brown provides a summary of MathML, an XML-based language for presenting and reading mathematical formulas on a Web page. This article gives a good introduction to how MathML works and the capabilities already built into it for actuarial expressions, and provides sources of more information for those looking for additional detail.

The Society of Actuaries is in the process of redesigning its Web site to improve navigation and content organization. Rob Hayashida gives a brief summary of the project, for which the Computer Science Section has provided funding assistance.

We also provide information on other activities of the section to keep members informed of what we're doing.

So, please read and enjoy what is here, and let us know what you think. We don't want to wait six more years for another newsletter, but we will need your help in telling us what you would like to see in the newsletter, or better yet, what you would like to contribute!

Charlie Linn, FSA, MAAA, is Chairperson of the Computer Science Section. In his spare time, he is an MG-Triton Actuary at Milliman USA in Windsor, CT, specializing in traditional and health valuation systems. He may be reached at charlie.linn@milliman.com.

The XTbML Standard: A Fresh Approach to Interchange of Table Data

by Cynthia Jeness and Jacques Rioux Ph. D

Cynthia Jeness, FSA, FCA, MAAA, is a project manager/ developer for Philibert Software Group in Atlanta, GA where she is working on a J2EE-based insurance illustration system. She can be reached at cjeness@bellsouth.net.

Jacques Rioux, ASA, Ph.D. is Senior Actuarial Scientist at SAS institute in Cary NC where he works on software to calculate and report operational risk for the banking industry. He can be reached at jacques.rioux@sas.com

Introduction

If you have been following the work of the Computer Science Section over the last few years, you might have heard of a standard for interchange of tabular data. You might even have attended the breakfast session of the last few annual meetings and seen a precursor of it in action. XTbML is the name of that standard and in this paper we will offer a preview of what it is, where it came from and what else may be coming to you from the section.

The original impetus for the creation of the XTbML standard may be traced back to the efforts of Steve Strommen and the creation of the Table Manager program and the binary format in which it stored the table data. Prior to the implementation of the XTbML standard, the technique for delivering table data from the SOA Web site was to utilize the Table Manager application and associated database. This required downloading and installing the Table Manager program under MS-Windows and then using it to extract the tables of interest. The Table Manager will export data in an ASCII format which is consistent across all tables of the same type and is described in the appendix of [1]. However, tables in this ASCII format are not available directly on the SOA Web site.

In 1998, the Computer Science Council of the Society of Actuaries revisited the task of defining a standard for the interchange of table data. The focus of this work was to be on the format of the data itself and not the tools to maintain or update the data. In 2000, a Data Standards Committee was formed. This Committee was composed of Jacques Rioux, chair (SAS Institute), Cynthia Jeness (Philibert Software Group) and Mark Horowitz (Towers Perrin). This Committee determined that there was already an open W3C standard, XML, which could be used for actuarial tabular data. Using XML as the standard, the Committee developed a preliminary dialect and presented the results to the Computer Science Council. Subsequent discussions with members of the SOA led to the submission of the XML standard for adoption by ACORD.

On March 21-22, 2002, members of the SOA and other insurance industry professionals met with ACORD in Westwood, MA. At this meeting the original standard was generalized and improved so that it would apply across a range of actuarial tabular data needs. The improved version, XTbML, went through the approval process at the ACORD meeting in June, 2002 and was adopted on a pilot basis by ACORD. The pilot period was intended to provide time for interested parties to use the standard and submit comments before it was officially adopted as a standard in November, 2002.

In September, 2002, another ACORD meeting was held to define the transactional interaction for XTbML. Utilizing this transactional standard, tabular data could be stored in a central repository and transferred to a requesting computer utilizing various industry standard protocols such as SOAP and HTTP GET/POST. This should extend the utility of the standard from the realm of human interaction to computer-computer interaction.

Later in the fall of 2002, the Computer Science Section decided to contract with Golden Code Development Corporation to provide a sample implementation of the standard at least for the tables already available under the Table Manager database format and to make those available from the SOA Web site. The resulting e may be best represented by a two-work contains more technical details about the standard itself and should be made available online very soon. In the meantime, we provide here a preview of what will be available for early potential implementers of the standard.

The XTbML Standard Itself

The primary documentation of the XTbML standard from ACORD consists of two documents:

- XML Schema
- Type Code Definition Document

The schema is the actual technical definition of the standard. It dictates what is and what is not a valid XTbML document . This note is not intended to dissect the standard or its schema in all its fine points and generality but rather to give an overview so that one can get the big picture.

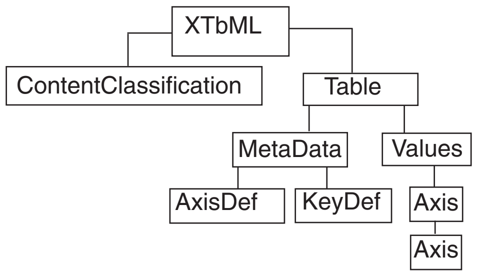
The type code definition document is available from the ACORD site and may be downloaded by anyone who is willing to register. In terms of the XTbML standard, the following type code groups are the most important:

- Content Type Codes
- Nation Type Codes
- Scale Type Codes

Beyond the documents cited above, ACORD does not provide information related to how to successfully implement the standard. They look to the users themselves to provide this type of information. Because the ACORD standard is very general and also because it takes a different approach to storing data than did Table Manager, some more specific standards need to be adopted in order to facilitate the exchange of tabular data.

ACORD Standard: Table Structure

The XML structure underlying the ACORD standard may be represented diagrammatically as follows:



Under this scheme, the XTbML element contains a content classification element and one or more table elements. Each table element contains both a metadata element and a values element. The values element contains one or more axis elements and these elements actually store the values. This assumes that the indexes to the values are numbers (e.g. age). Non-numeric indexes (e.g. gender) require the use of both KeyDef's and key's. For the purpose of the SOA table database the use of these elements is deemed unnecessary.

some more specific standards need to be adopted in order to facilitate the exchange of tabular data

The purpose of each of the elements shown in the diagram may be summarized as follows:

- Content Classification—Identifies the tabular data and provides general descriptive information.
- Table–This is the container for the tabular data. In some cases, it is more convenient to use multiple "tables" to represent a set of tabular data. For example, a select and ultimate table may be best represented by a two-dimensional select Table plus a one-dimensional aggregate Table.

• continued on page 6 •

XTbML continued from page 5

- Table: MetaData—This describes the attributes of the data stored within the Table; e.g., the minimum and maximum values for the indexes, the type of data, etc.

- Table—Values—This is the top-level container for the data itself.

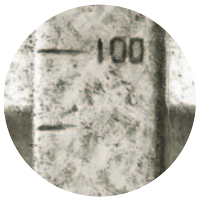
- Table—Values:Axis—This is the container for one dimension of the data. A one-dimensional table would have one axis while a two-dimensional table would have one axis which contains a second axis. The lowest level axis will contain a series of Y elements which actually house the data.

For the purpose of the SOA project, the types of tables stored by Table Manager will be structured in XTbML as follows:

- Aggregate Tables—A single table element with one Axis element.

- Select and Ultimate Tables—Two table elements. The select part of the table is represented by one of these. This table will have two axis elements. The outer most axis will be indexed on age and the innermost axis will be indexed based on the duration. The second table element will contain the ultimate data which will be handled just like an aggregate table.

Since the ACORD standard is so generic, other structures could be used to represent the SOA data. The proposed scheme is recommended because it is simple and consistent with the pre-cursor to the ACORD standard developed by the Computer Science Council. There was considerable discussion and development around the pre-cursor standard.



Sample XTbML documents

To make the structure of the tables more concrete for the reader, we supply two examples of actual implementation in appendixes 1 and 2. The first one is an aggregate mortality table and the second one is a select and ultimate table. Those two tables are available from the Table Manager database as table number 10 and table number 259 respectively. The vertical ellipsis obviously indicates a series of missing rows that were not included here to improve readability.

An interesting point to be made here is that most everyone with no knowledge of the XTbML standard itself but with familiarity with mortality tables could read the two samples provided and extract all the pertinent information available in them. That is part of the beauty of XML, it is quite verbose and therefore can carry a lot of semantics for the human reader.

Of course, most XML documents are not meant for human consumption, they are instead meant for consumption by computers. There again, XML with its panoply of related technologies such as XPATH, XSL/XSLT, DOM, SAX and the likes provides the programmer with a number of ways to access the information desired. It is clear however that if one is to write code to access a given probability of mortality for a certain age from any old table supplied in XML format, more knowledge is needed. This is precisely where the schema itself will be useful for the programmer of the program itself to make sense of the actual layout of the data.

What else should you expect

Those of you who attended the section's breakfast session of the last annual meeting will recall a few proof of concept demonstrations of what can be done with such a standard. Those demonstrations were all based on the fact that it is as easy to request a certain mortality table from a server providing it under the XTbML standard, as it is to request an HTML page from a browser. Initially, the Golden Code project sponsored by the section will provide just that, a way to download and browse tables directly from your web browser. However, once you have an XTbML document available in memory on the client side, the pos-

• continued on page 10 •

Smarter, Not Harder

An Example by Steve Welander, ASA, MAAA

By now I'm sure we've all been told at some point during our careers to work smarter, not harder. With this article, I will present one possible solution. As an example of working harder, let us assume that we need to create a presentation showing reinsurance cash flows and that we need to do it monthly. Since upper management will see part of this presentation, pictures are necessary. We are currently doing this by gathering all the data into Microsoft Access, building pivot tables in Microsoft Excel, and putting the results into a Microsoft PowerPoint presentation. This presentation consists of the following parts. A title slide, a summary slide showing monthly cash flows by line of business (LOB), a series of detail slides showing the monthly flows for each reinsurer by LOB, and a few bullet points.

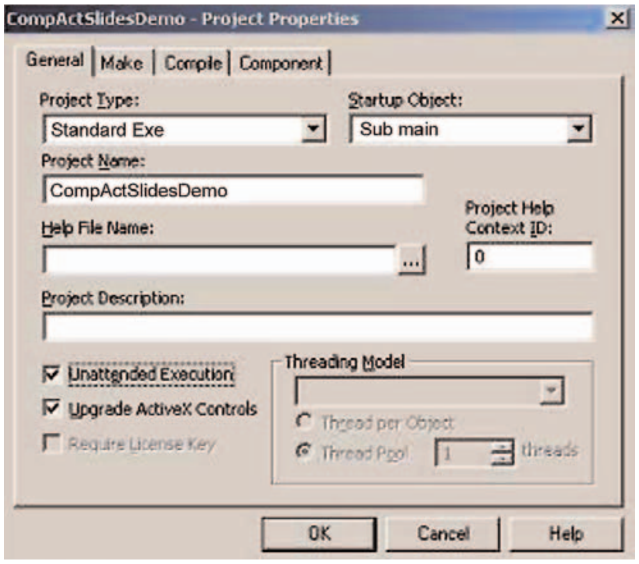
Now it is time to work smarter. What if once our database was populated, all it took was a double-click to create the presentation? Or better yet, it ran automatically? Using Microsoft Visual Basic (VB), we can do just that.

VB allows us to access various objects, including a PowerPoint object. We will take advantage of this PowerPoint object along with a connection to our Access database and some relatively straightforward code to accomplish our goal. The code will consist of a main "driver" routine and a couple specialized subroutines to create the various slide types.

Still with me? If you are management, maybe you want to skip this next part, look at the sample slides, write "can you do this?" on a copy of this article and pass it down to the your resident computer whiz. OK, you are still with me and you aren't management, ☺ so let's get a little more technical and discuss how to actually get this to work. I'll review the program flow, provide a little detail on some of the key elements, and discuss some additional considerations.

Since the goal is to have this run as a scheduled task, we will create the project so that it calls a module on load rather than a form. Once running, we need to establish access to the PowerPoint object and to the database. Next we will pass control to a routine to build the slides. Finally we save the presentation, close our files, and exit.

In order to set the program to run a module instead of loading a form, we need to go to the Project Properties menu and change the startup object to "Sub Main" (see figure 1). The "Unattended Execution" box should also be checked if this will be a standalone job. Since the program is now expecting a Main subroutine, we better make sure we have one with that name. We should think of this subroutine Main as an introduction to our program. By doing so, it makes it easier for us, or for someone else, to look at the code and get a pretty good idea of what the program is going to do.


CompActSlidesDemo - Project Properties

General | Make | Compile | Component

Project Type: Standard Exe

Startup Object: Sub main

Project Name: CompActSlidesDemo

Help File Name:

Project Help Context ID: 0

Project Description:

[✓] Unattended Execution

[✓] Upgrade ActiveX Controls

[ ] Require License Key

Threading Model

( ) Thread per Object

(•) Thread Pool 1 threads

The key routine for this example is the Build_Slides subroutine. First we need to establish a link to PowerPoint. Then we will pass a link to the presentation to several specialized subroutines that will actually build our slides. Once our slides are built, we can add global headers and footers.

This is done via the SlideMaster.HeadersFooters object. Finally, the presentation is saved and control is passed back to the Main subroutine.

The title slide is a fairly straightforward slide to create.

```
oPres.Slides.Add Slide_number, ppLayoutTitle
oPres.Slides(Slide_number).Shapes
("Rectangle 2").TextFrame.TextRange.Text =
Title_text
oPres.Slides(Slide_number).Shapes
("Rectangle 3").TextFrame.TextRange.Text =
sub_title_text
```

We tell it to add a slide of type ppLayoutTitle (a predefined variable). This slide consists of two rectangle boxes, the first of which is the main title box and the second is the sub-title

• continued on page 8 •

Steve Welander, ASA, MAAA, is an actuary with AIG | American General in Springfield, IL. He can be reached at actuary@welander.com

• Smarter, Not Harder continued from page 7 •

box. We assign our text to the appropriate properties and we are done with that slide.

The bullet point slides are created using a type ppLayoutText. Like the title slide, it has two predefined rectangles on it. The first one is the slide title box and the second is for the bullets. We don't need to get too fancy with this slide since it probably will be manually updated each month. But it is good to have as a placeholder in the presentation.

The chart slides are the real meat of the program. I've attempted to make a fairly generic subroutine to accomplish this.

```
Sub Add_Chart_Slide(strChartType As String, oPres As PowerPoint.Presentation, lSlide As Long, _

strTitle As String, strQuery As String, strXAxis As String, strYAxis As String)
```

What we are passing here are the type of slide (Bar or Line), the PowerPoint presentation object, the most recent slide number, the title for the chart, a string containing the query necessary to create the slide, and titles for the X and Y axis. Once we start the routine, we bump up our slide number by one since we are adding one, then we add a totally blank slide using ppLayoutBlank. After adding a blank slide, we need to insert a chart object.

```
Set shpGraph = .Shapes.AddOLEObject(0, 0, oPres.PageSetup.SlideWidth, _
    oPres.PageSetup.SlideHeight * 0.9, ClassName:="msgraph.chart", Link:=msoFalse)
```

What this is doing is inserting a chart starting in the top left, for the width of the slide and leaving the bottom 10% of the slide blank. This is the area where the footers would go, so it makes sense not to overlap them. Now that we have a chart, we need to tell it what type of chart.

Can we work even smarter?

```
oGraph.ChartType = xlLine ' to make line graphs
```

The sample is set up to switch between a line chart and a bar chart depending on the value we provide as strChartType.

Within the chart object is a datasheet. This looks a lot like an Excel spreadsheet. We need to take data from our Access query and fill this datasheet. First we blank out some of the default values. This isn't necessary if you are sure that you will be using at least the same number of rows and columns as the default. Then we set up some formatting for the axis. Here we could define a minimum and/or maximum scale if we needed our slides to be consistent. In the sample I've commented these out, but you can see how it would be done if desired. Next we loop through the query setting up my row and column headings. These values would go into row one and column one on the datasheet.

```
oDataSheet.Cells(lRowCnt, 1).Value = Format(.Fields(0).Value, "0000-00")
```

Now we reset the query back to the beginning and fill in our actual datasheet values. We need to keep track of what row and column we are in, but with a couple temporary variables this isn't too difficult. Now that our table is populated, we can thicken up the weight of the lines if desired.

```
oGraph.SeriesCollection(x).Border.Weight = xlMedium
```

Then we need to update the changes we've made to the chart.

```
oGraph.Application.Update
```

If we don't update the changes, when we look at our presentation, all we will see is the default chart for each slide.

As mentioned at the beginning of the article, we also want to make a series of slides based on a single query. In this case, it is monthly cash flows by LOB with a separate slide for each reinsurer. Initially I set about doing this with a subroutine very similar to the one just described, but with an extra layer of looping involved. I quickly saw that this was going to be a lot of redundant code so I rethought my strategy. My new train of thought had me taking the query and running a query on it. This created a new query that had just the reinsurer's name in it. So now I could loop through this new query and call the single slide query using a slight modification of the original query. This slight modification was just to select off only the records for the reinsurer we want for this slide. A little testing, a little cursing, some more testing and I got it working.

What is left to do? Best I can tell, we just have to fire up PowerPoint, edit the bullet slides, and ship it off to management.

And there we have it, an automated slide generating program. Now for some considerations. If the database has a massive amount of data, the queries could run very slow. Since some of the queries get multiple hits, it might make sense to run a CreateTable query first to store the summary data in and run the queries off of that table. It is entirely possible that you don't have access to Visual Basic. In that case, it should be possible to modify this logic to work as a module in either Access or PowerPoint. It should also be fairly easy to make this a form-based application and allow inputs for file location, "as of" dates, custom titles, or whatever.

It should be noted that there were no "holes" in the sample data. That is to say every cell had a value. If that isn't the case, allowances need to be made. This can be accomplished in many different ways such as loading the data into an array and then populating the datasheet from the array or tweaking the queries to ensure that there are no empty cells. If you are trying to run the sample code, make sure you have the database placed in the same location that is hard-coded in the sample code.
Can we work even smarter? If we could automate the populating of the database, then we can answer "yes". If there is enough demand, then maybe another article will fully answer that question.

This project has been created using Office 97 and Visual Basic 5. I would hope that the modifications to newer versions of the software would be trivial. Some of the routines were found in some samples on the Microsoft website. Some web searches (using www.google.com) turned up surprisingly little data on this topic. The Object Browser also proved helpful. I was able to obtain some help on the Chart objects by recording Chart macros in Excel. I found recording them in PowerPoint didn't produce any useful information.

The following KnowledgeBase articles were used to help with this project: Q143038, Q172836, and Q176443.

The author has made a sample program and sample database available along with the article, and they can be found at: [insert web link here]. The author may be contacted at actuary@welander.com with questions and/or comments.

2003 Annual Meeting—Orlando, FL Computer Science Section Sessions

Data Mining—How to Find Gold (co-sponsored with E&R and NTM Sections)
Monday, October 27, 10:30 a.m.–12:00 p.m.

Are your customers profitable? Data mining lets you slice your current customer block in a variety of ways. This session shows you what techniques are available to find and maximize the additional value within in-force blocks of business. Real-life examples of when to use each technique and why are presented.

Participants gain an understanding of:
- Data mining techniques currently in use
- Privacy and regulatory issues

Computer Science Section Hot Breakfast: ACORD Data Standards
Tuesday, October 28, 8:00 a.m. - 10:00 a.m.
Computer Science Section members learn about current section activities and provide input to the Section Council. The session also teaches the workings of the XML-based ACORD data standard, including the far-reaching applications of ACORD. In addition, examples of representing tabular data in XTbML format and transmission of data are incorporated.

At the conclusion of this session, attendees:
- Learn about section activities
- Recognize opportunities to volunteer for section projects
- Learn about ACORD's XTbML standard and its applicability in some areas

Relating to Relational Databases
Tuesday, October 28, 10:30 a.m. - 12:00 p.m.
Sure, data are all part of one big database family, but are they properly related? Actuaries can feel intimidated when the information technology (IT) folks start talking about normalization, SQL, foreign keys, object-relational and entity relationship diagrams.

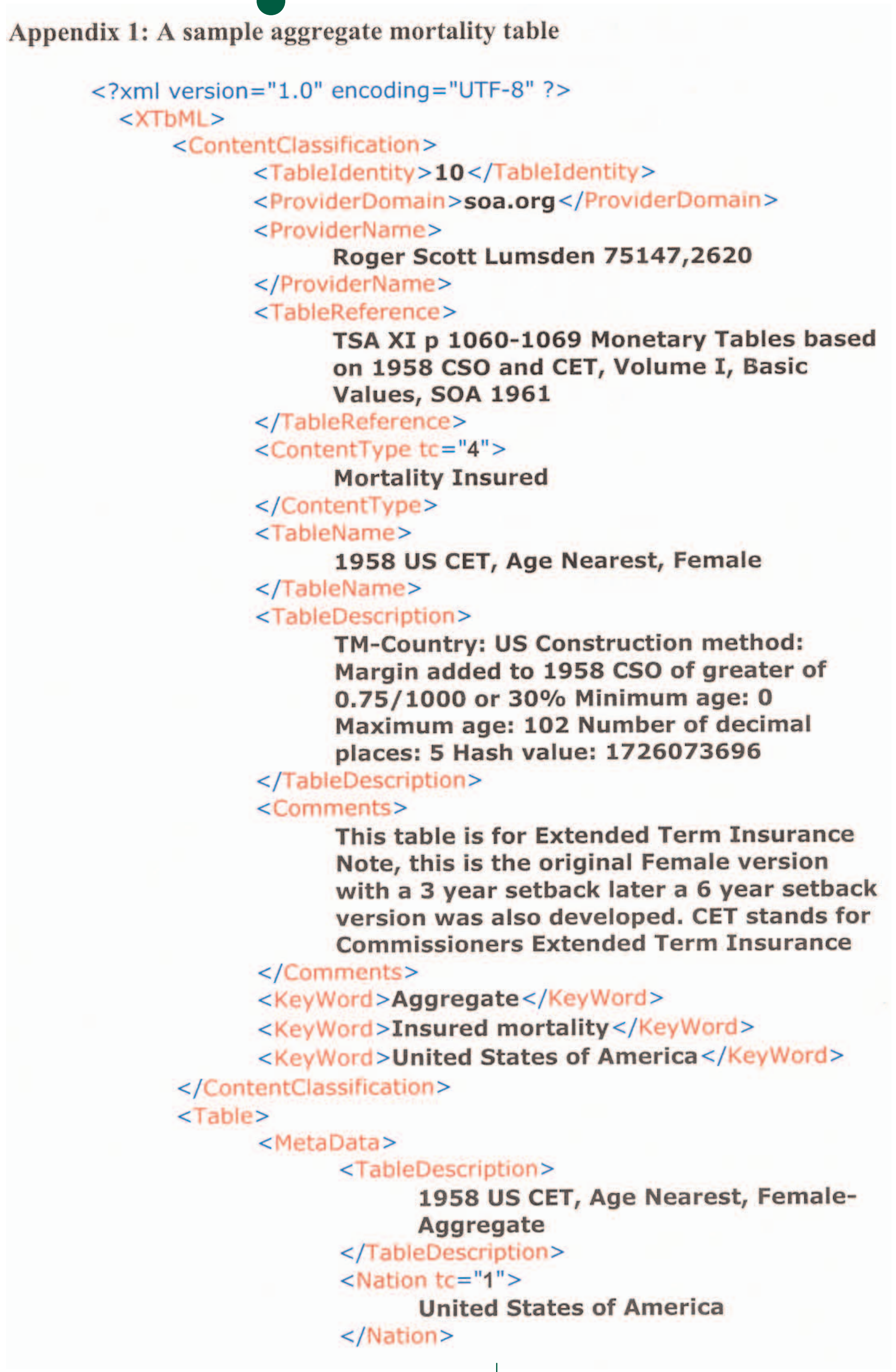
Attendees learn when to:
- Use spreadsheets
- Choose entry-level databases
- Consider the powerful large databases

This session gives attendees an understanding of how to arrange and store data in a manner that reduces duplication of effort and improves the ability to utilize it for more informed business decisions. Actuaries receive common-sense definitions to IT jargon.

XTbML ontinued from page 6

sibilities are limitless.

One could write custom clients in the language of their choice. One such custom client could be an Excel add-in that allows to access table data just by typing in a custom worksheet function taking the table number, age and duration as arguments for example. The second author demonstrated such a client based on a previous version of the standard and plans on distributing another one written in Visual Basic as open source later this summer. The first author has plans on providing a similar custom client implemented in Java.



Appendix 1: A sample aggregate mortality table

```
<?xml version="1.0" encoding="UTF-8" ?>
  <XTbML>
     <ContentClassification>
          <TableIdentity>10</TableIdentity>
          <ProviderDomain>soa.org</ProviderDomain>
          <ProviderName>
               Roger Scott Lumsden 75147,2620
          </ProviderName>
          <TableReference>
               TSA XI p 1060-1069 Monetary Tables based
               on 1958 CSO and CET, Volume I, Basic
               Values, SOA 1961
          </TableReference>
          <ContentType tc="4">
               Mortality Insured
          </ContentType>
          <TableName>
               1958 US CET, Age Nearest, Female
          </TableName>
          <TableDescription>
               TM-Country: US Construction method:
               Margin added to 1958 CSO of greater of
               0.75/1000 or 30% Minimum age: 0
               Maximum age: 102 Number of decimal
               places: 5 Hash value: 1726073696
          </TableDescription>
          <Comments>
               This table is for Extended Term Insurance
               Note, this is the original Female version
               with a 3 year setback later a 6 year setback
               version was also developed. CET stands for
               Commissioners Extended Term Insurance
          </Comments>
          <KeyWord>Aggregate</KeyWord>
          <KeyWord>Insured mortality</KeyWord>
          <KeyWord>United States of America</KeyWord>
     </ContentClassification>
     <Table>
          <MetaData>
               <TableDescription>
                    1958 US CET, Age Nearest, Female-
                    Aggregate
               </TableDescription>
               <Nation tc="1">
                    United States of America
               </Nation>
```

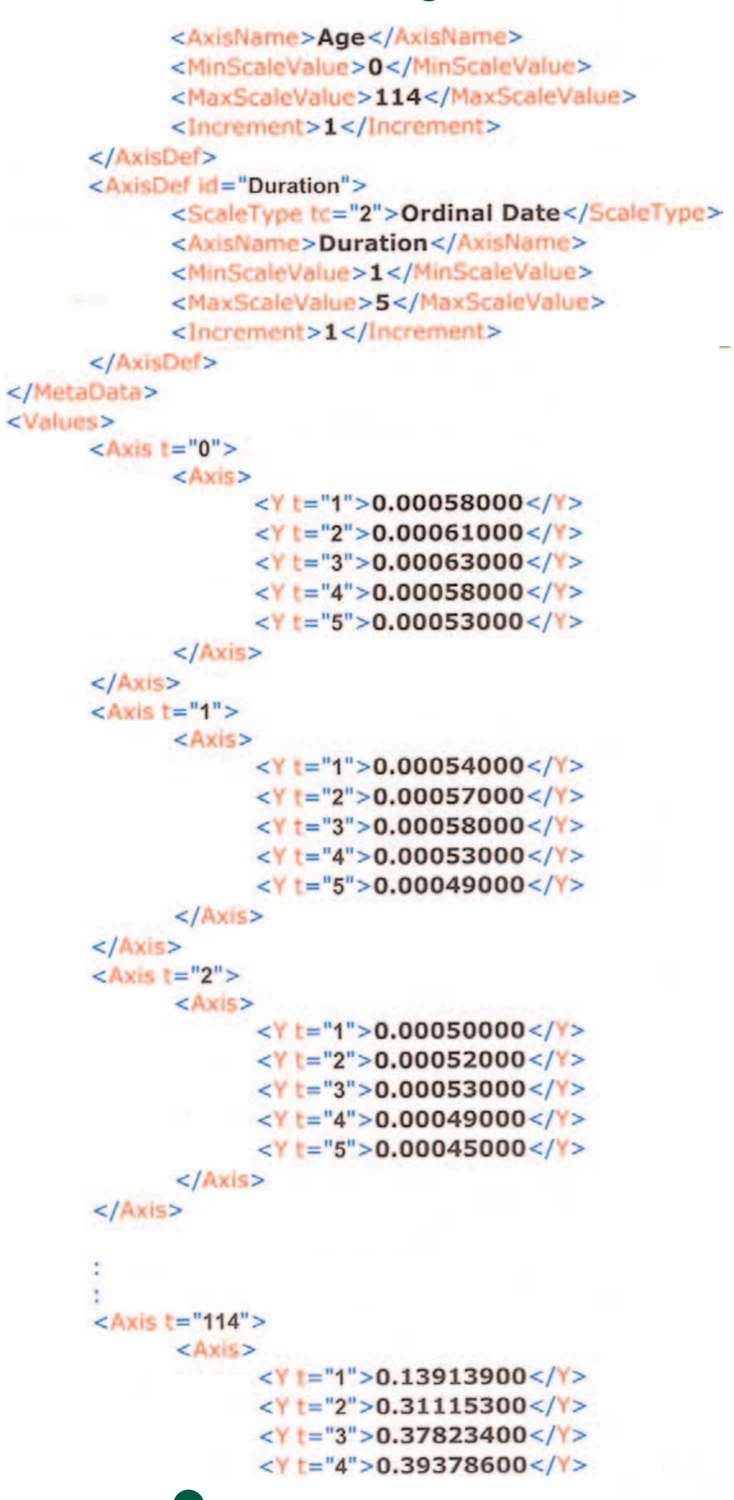
```
            <ScalingFactor>0</ScalingFactor>
            <DataType tc="2">Floating Point</DataType>
            <AxisDef id="Age">
                <ScaleType tc="3">Age</ScaleType>
                <AxisName>Age</AxisName>
                <MinScaleValue>0</MinScaleValue>
                <MaxScaleValue>102</MaxScaleValue>
                <Increment>1</Increment>
            </AxisDef>
        </MetaData>
        <Values>
            <Axis>
                <Y t="0">0.00695</Y>
                <Y t="1">0.00242</Y>
                <Y t="2">0.00216</Y>
                <Y t="3">0.00210</Y>
                <Y t="4">0.00204</Y>
                <Y t="5">0.00199</Y>
                :
                :
                :
                <Y t="100">0.63500</Y>
                <Y t="101">0.86877</Y>
                <Y t="102">1.00000</Y>
            </Axis>
        </Values>
    </Table>
</XTbML>
```

Appendix 2: A sample select and ultimate mortality table

```
<?xml version="1.0" encoding="UTF-8" ?>
<XTbML>
  <ContentClassification>
        <TableIdentity>259</TableIdentity>
        <ProviderDomain>soa.org</ProviderDomain>
        <ProviderName>
              Roger Scott Lumsden 75147,2620
        </ProviderName>
        <TableReference>
              JIA vol 101, Part II or TFA vol 34
        </TableReference>
        <ContentType tc="4">Mortality Insured</ContentType>
        <TableName>
              1967-70 UK A1967-70(5), Male, Age nearest
        </TableName>
        <TableDescription>
              TM-Country: UK Source of data: CMIR
              Intercompany mortality study on Ordinary issues
              in the UK Observation period: 1967 to 1970
              Construction method: Fitted to formula separately
              for duration 1, 2, 3-5 and ult of form qx/px =
              B.c^y - Hy where y=x-Y There is also a 2 year
              select A1967-70(2) more commonly used
              Minimum age: 0 Maximum age: 119 Select period:
              5 Maximum select age: 114 Number of decimal
              places: 8 Hash value: -187989173
        </TableDescription>
        <Comments>
              I generated these from the formulae, occasionally
              giving a slight difference in 8th decimal place
              Please note that the published parameters for Ult
              are incorrect A1967-70 stands for Assured Lives
        </Comments>
        <KeyWord>Select</KeyWord>
        <KeyWord>Insured mortality</KeyWord>
        <KeyWord>United Kingdom</KeyWord>
  </ContentClassification>
  <Table>
        <MetaData>
              <TableDescription>
                    1967-70 UK A1967-70(5), Male, Age
                    nearest-Select
              </TableDescription>
              <Nation tc="44">United Kingdom</Nation>
              <ScalingFactor>0</ScalingFactor>
              <DataType tc="2">Floating Point</DataType>
              <AxisDef id="Age">
                    <ScaleType tc="3">Age</ScaleType>
```

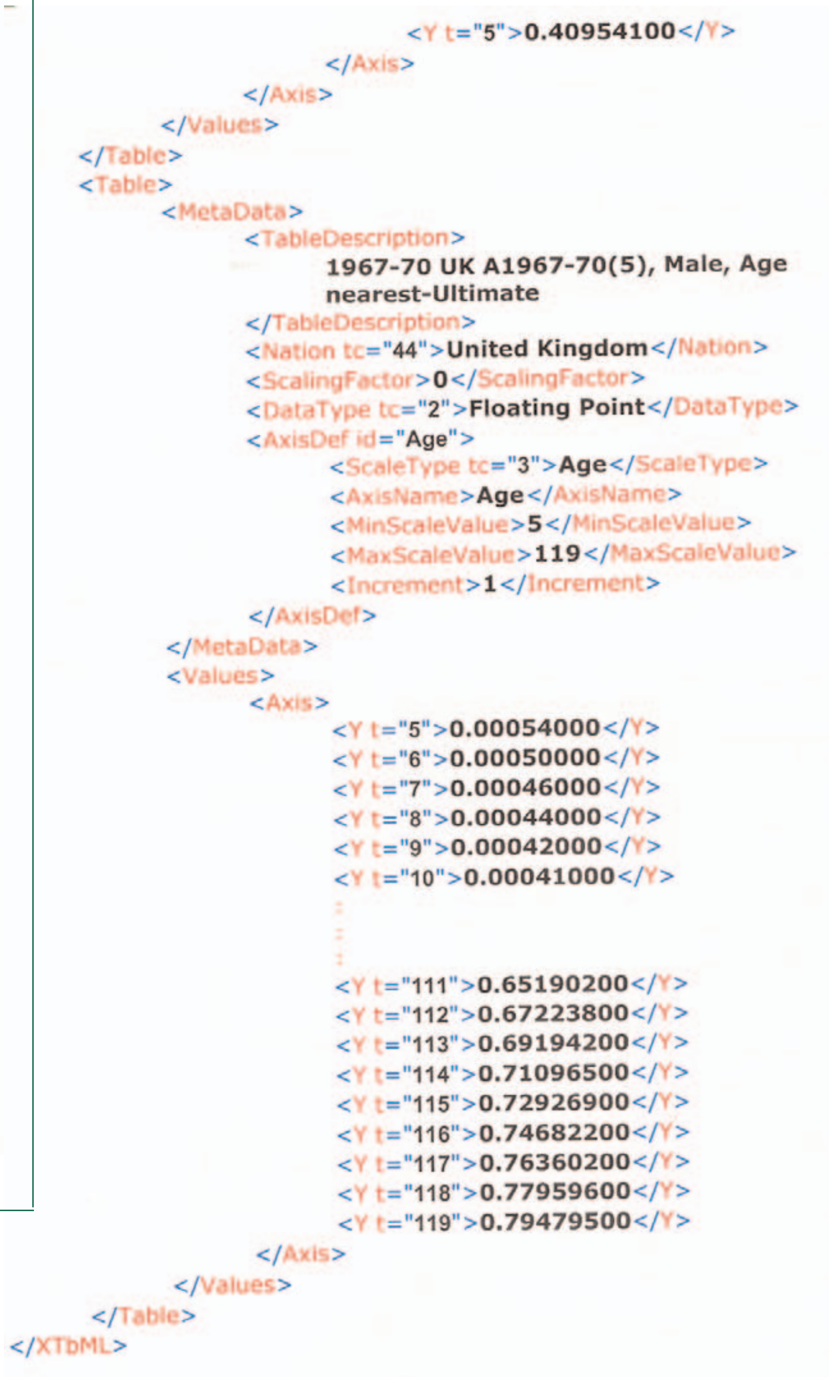
• continued on page 12 •

XTbML continued from page 11



```
                <AxisName>Age</AxisName>
                <MinScaleValue>0</MinScaleValue>
                <MaxScaleValue>114</MaxScaleValue>
                <Increment>1</Increment>
        </AxisDef>
        <AxisDef id="Duration">
                <ScaleType tc="2">Ordinal Date</ScaleType>
                <AxisName>Duration</AxisName>
                <MinScaleValue>1</MinScaleValue>
                <MaxScaleValue>5</MaxScaleValue>
                <Increment>1</Increment>
        </AxisDef>
</MetaData>
<Values>
        <Axis t="0">
                <Axis>
                        <Y t="1">0.00058000</Y>
                        <Y t="2">0.00061000</Y>
                        <Y t="3">0.00063000</Y>
                        <Y t="4">0.00058000</Y>
                        <Y t="5">0.00053000</Y>
                </Axis>
        </Axis>
        <Axis t="1">
                <Axis>
                        <Y t="1">0.00054000</Y>
                        <Y t="2">0.00057000</Y>
                        <Y t="3">0.00058000</Y>
                        <Y t="4">0.00053000</Y>
                        <Y t="5">0.00049000</Y>
                </Axis>
        </Axis>
        <Axis t="2">
                <Axis>
                        <Y t="1">0.00050000</Y>
                        <Y t="2">0.00052000</Y>
                        <Y t="3">0.00053000</Y>
                        <Y t="4">0.00049000</Y>
                        <Y t="5">0.00045000</Y>
                </Axis>
        </Axis>

        :
        :
        <Axis t="114">
                <Axis>
                        <Y t="1">0.13913900</Y>
                        <Y t="2">0.31115300</Y>
                        <Y t="3">0.37823400</Y>
                        <Y t="4">0.39378600</Y>
```



```
                                        <Y t="5">0.40954100</Y>
                                </Axis>
                        </Axis>
                </Values>
        </Table>
        <Table>
                <MetaData>
                        <TableDescription>
                                1967-70 UK A1967-70(5), Male, Age
                                nearest-Ultimate
                        </TableDescription>
                        <Nation tc="44">United Kingdom</Nation>
                        <ScalingFactor>0</ScalingFactor>
                        <DataType tc="2">Floating Point</DataType>
                        <AxisDef id="Age">
                                <ScaleType tc="3">Age</ScaleType>
                                <AxisName>Age</AxisName>
                                <MinScaleValue>5</MinScaleValue>
                                <MaxScaleValue>119</MaxScaleValue>
                                <Increment>1</Increment>
                        </AxisDef>
                </MetaData>
                <Values>
                        <Axis>
                                <Y t="5">0.00054000</Y>
                                <Y t="6">0.00050000</Y>
                                <Y t="7">0.00046000</Y>
                                <Y t="8">0.00044000</Y>
                                <Y t="9">0.00042000</Y>
                                <Y t="10">0.00041000</Y>
                                :
                                :
                                <Y t="111">0.65190200</Y>
                                <Y t="112">0.67223800</Y>
                                <Y t="113">0.69194200</Y>
                                <Y t="114">0.71096500</Y>
                                <Y t="115">0.72926900</Y>
                                <Y t="116">0.74682200</Y>
                                <Y t="117">0.76360200</Y>
                                <Y t="118">0.77959600</Y>
                                <Y t="119">0.79479500</Y>
                        </Axis>
                </Values>
        </Table>
</XTbML>
```

Bibliography

[1] Proposed Data Standards for Table Data in the Insurance and Pension Industries, http://www.soa.org/sections/data_standards.pdf.

Introducing MathML: The Solution to Mathmatics on the Web

by Jay Brown

Haven't you always wanted to put

$$\beta^{\text{com}} = P + \frac{{}_{19}P_{x+1} - c_x}{\ddot{a}_{x:\overline{n|}}}$$

on your web page? Who hasn't? Unfortunately, HTML was not made for mathematical expressions. With it, there is no way to mark up mathematical formulas so that they will be properly displayed. Fortunately, there is an answer: MathML.

Standard World Wide Web mark-up languages do not allow for the accurate representation of mathematical formulas, even though one of the early intents of the Web was for it to be a pooling of scientific knowledge. Previously, the only way to include mathematical expressions in HTML Web documents was to link to images stored in JPG or GIF format. The quality of such documents is often very poor. The images are usually hard to create, very primitive in appearance and cause a marked increase in document load time. Additionally, once the images are created, they are fixed. They cannot adapt to changes in font size or background color, which can cause anti-aliasing halos (blurred, or fuzzy edges and shadows). It is also difficult, if not impossible, to correctly align formulas horizontally, so that they appear appropriately when used in the middle of a line of text. Once these documents are finally created, they typically print poorly, because of the limited pixel depth (70 dpi) of the printed images. Beyond simply being displayed, formulas should ideally react to events, like mouse-over events for portions of formulas, with functionality beyond the limited capabilities of image maps. For convenience, formulas should also be able to be folded and unfolded as long equations are being viewed. When formulas are stored as images, they are unavailable for searching, and cannot have portions selected for cut-and-paste operations. Nor, can they be transferred into other software applications for manipulation, or be exchanged between software applications. Enter MathML.

MathML is a World Wide Web Consortium (W3C) recommendation, released on February 21, 2001 (see http://www.w3.org/Math/), and having a last call on the Working Draft of its second edition on May 9, 2003. It has its origins in a 1994 proposal by Dave Raggett for HTML Math in the HTML 3.0 Working Draft. This led to a discussion panel on mathematical markup at the April 1995 WWW Conference in Darmstadt. In May of 1995, Wolfran Research then submitted a proposal for mathematical markup. A year later, the Digital Library Initiative Meeting in Champaign-Urbana, IL brought interested parties together, leading to the formation of the HTML Math Editorial Review Board. This Board became the W3C Math Working Group in March of 1997, and was given a longer charter as the 2nd W3C Math Working Group in July of 1998 and again in June of 2001 (for a list of members see http://www.w3.org/Math/workingGroup).

MathML was developed with the goals of encoding mathematics suitable for teaching and scientific communication. The language was to encode both notation and meaning and be sufficient for use as an exchange medium for both software applications and scientists.

MathML is an application of XML (eXtensible Markup Language), which was created to surpass the limitation of pre-defined HTML tags. In XML, authors define their own tags

• continued on page 14 •

Jay Brown is an MG-Triton client support manager at Milliman USA in Windsor, CT. He specializes in traditional and health valuation systems, and holds a Masters degree in Computer Science from Rensselaer Polytechnic Institute. He may be reached at jay.brown@milliman.com.

as needed. MathML is further supported by style sheets, including a published XSL style sheet called the Universal MathML Style Sheet (UMSS) that can adjust to the current environment, to be used in editors, translators and native renderors (browsers, plugins, applets or ActiveX components). It can be used to mark up both the presentation of mathematical expressions and their semantics, or underlying meaning. It has been designed to support mathematics up to and including the high school/early university level, and is intended to be primarily written by software, such as authoring tools, and not by human hand—the assumption being that an author lay out their expression using a WYSIWYG tool and let the tool generate the corresponding MathML.

To control presentation and meaning, MathML specifies two new sets of XML tags: Presentation MathML and Content MathML. For the notation of mathematical formulas in markup (Presentation MathML), there are 28 new tags and 50 attributes covering 2000 symbols, which take advantage of the proposed extensions for mathematics in Unicode 3.1 and 3.2, with the ability to add others through the MathML mglyph element. To convey semantic meaning (Content MathML), there are 75 new tags and 12 attributes, with methods to extend them. An author of a mathematical expression may chose to use either or both Presentation MathML and Content MathML, based on that expression's intended use.

To see how MathML works, let's look at the mathematical expression for the quadratic equation (the solution to the basic quadratic formula ax^2 + bx + c = 0):

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figure 1: The quadratic equation

In Presentation MathML, the expression would look like the following:

```
<mrow>
  <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mrow>
        <mo>-</mo>
        <mi>b</mi>
      </mrow>
      <mo>&PlusMinus;</mo>
      <msqrt>
        <mrow>
          <msup>
            <mi>b</mi>
            <mn>2</mn>
          </msup>
          <mo>-</mo>
          <mrow>
            <mn>4</mn>
            <mo>&InvisibleTimes;</mo>
            <mi>a</mi>
            <mo>&InvisibleTimes;</mo>
            <mi>c</mi>
          </mrow>
        </mrow>
      </msqrt>
    </mrow>
    <mrow>
      <mn>2</mn>
      <mo>&InvisibleTimes;</mo>
      <mi>a</mi>
    </mrow>
  </mfrac>
</mrow>
```

Note that the marked-up expression is essentially nested XML, in line with the recursive nature of mathematics. Elements like mrow (indicating a horizontal row of elements) and msup (indicating a base element and its superscript) help define the typical two-dimensional layout of a mathematical expression. Other tags used are mfrac (a fraction with the first child as the numerator, and the second as the denominator), msqrt (the square root of the contained expression), mi (variables) and mo (operators). In an HTML document, the marked-up expression would be contained within <math xmlns= "http:// www.w3.org/1998/Math/MathML"> and </math> tags.

In Content MathML, the same expression would look as follows:

```
<mrow>
<apply>
    <eq/>
    <ci>x</ci>
    <apply>
        <divide/>
        <apply>
            <mo>&PlusMinus;</mo>
            <apply>
                <minus/>
                <ci>b</ci>
            </apply>
            <apply>
                <root/>
                <apply>
                    <minus/>
                    <apply>
                        <power/>
                        <ci>b</ci>
                        <cn>2</cn>
                    </apply>
                    <apply>
                        <times/>
                        <cn>4</cn>
                        <ci>a</ci>
                        <ci>c</ci>
                    </apply>
                </apply>
                <cn>2</cn>
            </apply>
        </apply>
        <apply>
            <times/>
            <cn>2</cn>
            <ci>a</ci>
        </apply>
    </apply>
</apply>
</mrow>
```

Note that from a semantic standpoint the formula looks like:

$$x = \pm\left(-b, \sqrt{b^2 - 4xaxc}\right) \div 2xa$$

The Content MathML is expressing the underlying meaning, and not the layout or presentation. This requires more expressive tags like apply (construct, using prefix notation), ci (child variable), minus (subtraction) and times (multiplication).

The Content MathML is expressing the underlying meaning...

It is also possible to use Presentation and Content MathML together.

For example, to render $\int_1^t \frac{dx}{x}$

The standard Content markup would be:

```
<mrow>
<semantics>
    <mrow>
        <msubsup>
            <mo>&int;</mo>
            <mn>1</mn>
            <mi>t</mi>
        </msubsup>
        <mfrac>
            <mrow>
                <mo>&dd;</mo>
                <mi>x</mi>
            </mrow>
            <mi>x</mi>
        </mfrac>
  </mrow>
  <annotation-xml encoding="MathML-
Content">
```

• continued on page 16 •

MathML continued from page 15

```
        <apply>
          <int/>
          <bvar><ci>x</ci></bvar>
           <lowlimit><cn>1</cn></lowlimit>
          <uplimit><ci>t</ci></uplimit>
          <apply>
            <divide/>
            <cn>1</cn>
            <ci>x</ci>
          </apply>
        </apply>
      </annotation-xml>
    </semantics>
    </mrow>
```

However, some renderors might present the integrand as (1/x)dx, when the author would like to ensure it is displayed as dx/x. The combination of Presentation and Content shown below will solve that problem, by defining both display rules and meaning concurrently:

```
<semantics>
      <apply>
        <int/>
        <bvar><ci>x</ci></bvar>
        <lowlimit><cn>1</cn></lowlimit>
        <uplimit><ci>t</ci></uplimit>
        <apply>
          <divide/>
          <cn>1</cn>
          <ci>x</ci>
        </apply>
      </apply>
     <annotation-xml encoding=
     "MathM-Presentation">
        <mrow>
          <msubsup>
            <mo>&int;</mo>
            <mn>1</mn>
            <mi>t</mi>
          </msubsup>
          <mfrac>
            <mrow>
              <mo>&dd;</mo>
              <mi>x</mi>
            </mrow>
            <mi>x</mi>
          </mfrac>
        </mrow>
      </annotation-xml>
    </semantics>
```

For the actuarial community, that latest version of MathML supports actuarial notation. For example, the MathML markup for:

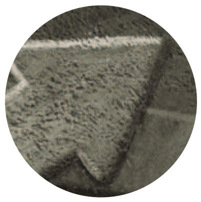
$$\beta^{\text{com}} = P + \frac{{}_{19}P_{x+1} - c_x}{\ddot{a}_{x:\overline{n}|}}$$

Figure 2: CRVM Beta equationis as follows:

```
<semantics>
  <mrow>
    <msup>
      <mi>&#x03B2;</mi>
       <mrow>
         <mtext>Com</mtext>
      </mrow>
</msup>
<mo>=</mo><mi>P</mi><mo>+</mo><mfrac>
  <mrow>
    <mmultiscripts>
      <mi>P</mi>
        <mrow>
          <mi>x</mi><mo>+</mo><mn>1</mn>
        </mrow>
        <none />
        <mprescripts/>
        <mrow>
          <mn>19</mn>
         </mrow>
         <none />
      </mmultiscripts>
      <mo>&#x2212;</mo><msub>
          <mi>c</mi>
          <mi>x</mi>
        </msub>
</mrow>
```

```
<mrow>
  <msub>
    <mover accent='true'>
      <mi>a</mi>
      <mo>&#x00A8;</mo>
    </mover>
    <mrow>
                    <mi>x</mi>
                    <mo>:</mo>
                    <menclose notation='actuarial'>
                      <mi>n</mi>
                    </menclose>
                  </mrow>
                </msub>
              </mrow>
            </mfrac>
          </mrow>
</semantics>
```

Netscape, Internet Explorer and Mozilla now have very good native browser support for MathML (see Table 1). Currently, tools exist to help with the creation and viewing of mathematical expressions on Web pages. For example, MathType (www.mathtype.com) has augmented their popular equation editor, to include MathML as an output option. They have also developed the WebEQ toolkit to aid in developing mathematics for the Web and the MathPlayer display engine for Internet Explorer. So, get your mouse and modem ready, because MathML, the solution to mathematics on the Web, is coming your way.



	Internet Explorer 5.0	Internet Explorer 5.5	Internet Explorer 6.0	Netscape 6.1	Netscape 7.0 PR1	Amaya	Mozilla 0.9.9
Windows	• 1	• 1 or 2	• 3	• 1	•	• 4	•
Macintosh	• 1						•
Linux/UNIX				• 1	•	• 4	•

Table 1: Browser Support for MathML

1 = with Techexplorer plug-in
2 = with MathPlayer plug-in
3 = optionally, with Techexplorer or MathPlayer plug-ins
4 = presentation MathML only

Society of Actuaries Web Redesign

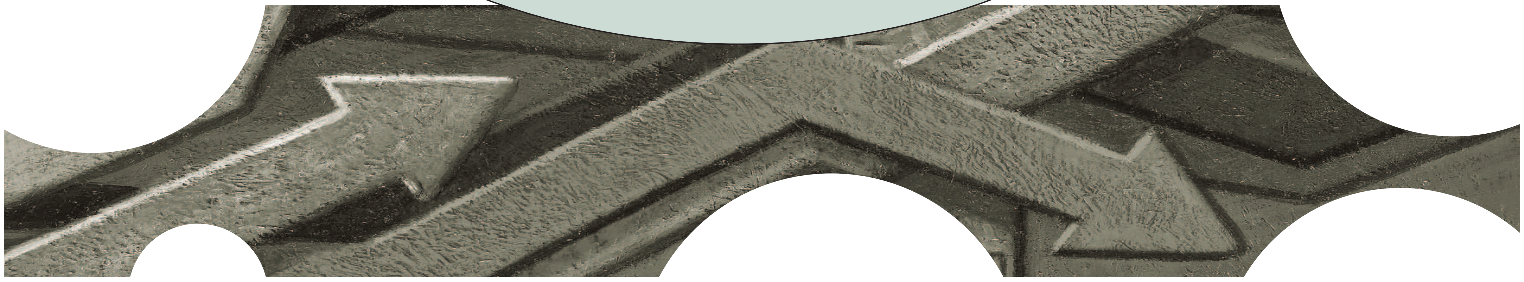
by Rob Hayashida

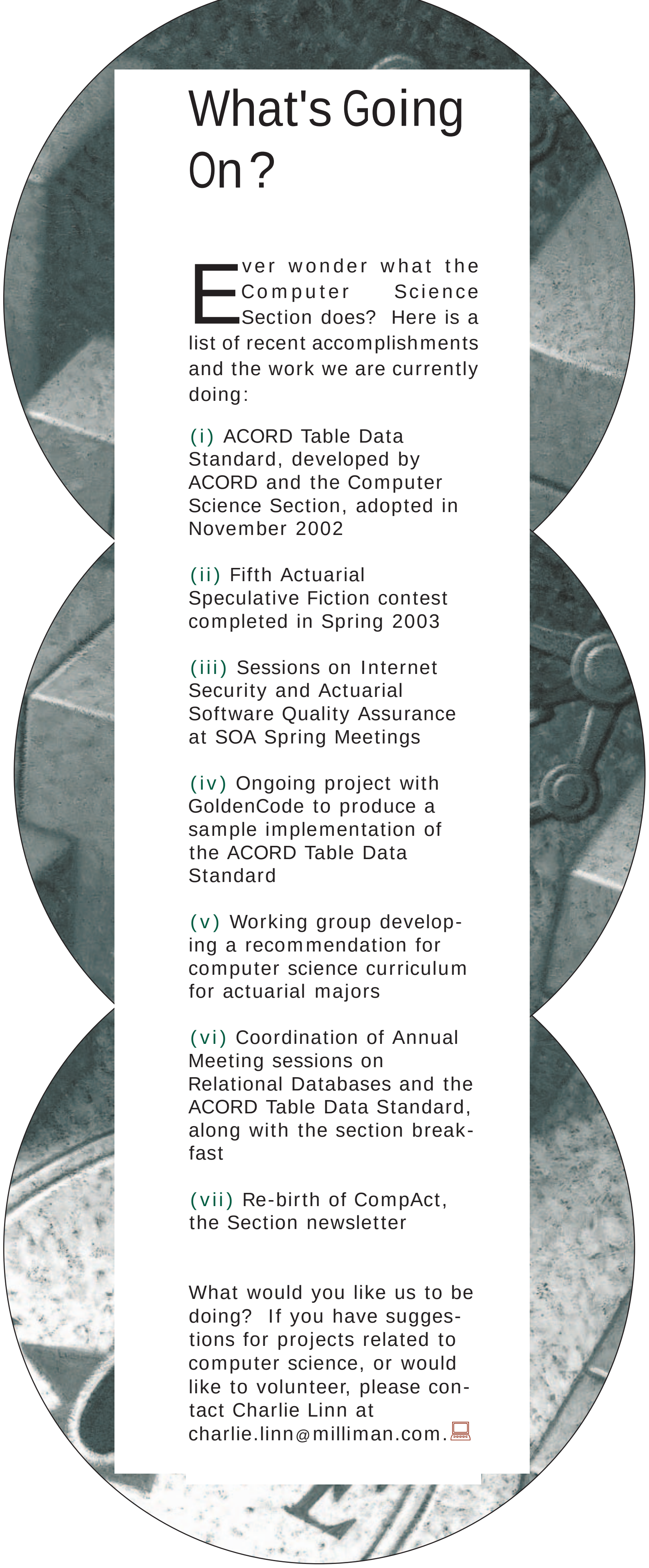
In May, the Society of Actuaries began the first phase of the SOA Web site redesign. The first phase focuses on making navigation more intuitive and easier to use by organizing the site's content in a logical manner. Other features that will be included in the first phase of the redesign will be a new and improved online library search, integrated site search and the possibility of new discussion forums and knowledge base of frequently asked questions.

By redesigning our Web site, the SOA will be providing members and candidates easy access to information, research tools, study notes, online education, interactivity and community. The aim is to create a user experience that reflects the SOA as the premier source of information about the actuarial profession.

The next few months will be a very exciting time for the SOA as we launch the new site in the beginning of 2004.

Rob Hayashida is the web manager with the Society of Actuaries and has been active in web-application development for close to seven years. He can be reached at rhayashida@soa.org.





What's Going On?

Ever wonder what the Computer Science Section does? Here is a list of recent accomplishments and the work we are currently doing:

(i) ACORD Table Data Standard, developed by ACORD and the Computer Science Section, adopted in November 2002

(ii) Fifth Actuarial Speculative Fiction contest completed in Spring 2003

(iii) Sessions on Internet Security and Actuarial Software Quality Assurance at SOA Spring Meetings

(iv) Ongoing project with GoldenCode to produce a sample implementation of the ACORD Table Data Standard

(v) Working group developing a recommendation for computer science curriculum for actuarial majors

(vi) Coordination of Annual Meeting sessions on Relational Databases and the ACORD Table Data Standard, along with the section breakfast

(vii) Re-birth of CompAct, the Section newsletter

What would you like us to be doing? If you have suggestions for projects related to computer science, or would like to volunteer, please contact Charlie Linn at charlie.linn@milliman.com.

2003 Actuarial Speculative Fiction Contest: The Outer Limits of Actuarial Thoughts

Take a walk through the outer limits of the minds of some of your actuarial associates! Contributors to the fifth Actuarial Speculative Fiction Contest, sponsored by the Computer Science Section, have provided a variety of views of the future of the world from an actuarial perspective, sharing those inner most thoughts with you. Come learn, marvel, chuckle and cry at what possibly might be, but do be careful. Once you have finished these stories, you will never look at your world, or the actuaries in it, in the same way. The thoughts and ideas shared here will change the way you think.

As part of the contest, the Computer Science Section presents prizes in various categories. This year's winners were:

First place
Alan Shulman wins $200 for his contribution "God's Actuary"

Second place
Gregory A. Dreher wins $100 for his contribution "Actuarial Certainty."

Best use of computers
Steve Mathys wins $50 for his contribution "Antiquity in Their Midst."

Best use of actuarial science
Joe Kincaid wins $50 for his contribution "Worth the Risk."

To access all of the stories, please go to http://www.soa.org/sections/scifi/fiction_version5.html.

Society of Actuaries

475 N. Martingale • Schaumburg, IL • 60173 • www.soa.org