RECORD, Volume 26, No. 3*

Chicago Annual Meeting
October 15-18, 2000

Session 90PD
XML for Actuaries: A Web of Data and Metadata

Track: Computer Science

Moderator: MARK D. HOROWITZ
Panelists: RON COLE†
JAWWAD A. FARID
MARK D. HOROWITZ
KEVIN KELLY‡
Recorder: MARK D. HOROWITZ

Summary: As the Internet grows, information must be more easily shared across time, space, and subject. This session introduces actuaries to XML. It discusses the origins of XML and its relation to HTML. It shows how to model with XML to achieve independence between platforms and systems. The speakers present XML applications, including online insurance, employee benefits and mathematics.

Mr. Kevin Kelly: For the last 15 years I've been in, what I call the belly of the beast, in the back room, and trying to help them wrangle with mainframe systems, PC networks, electronic data interchanges (EDIs), networking. In general, trying to automate the value chain in insurance end-to-end. So I'm going to try to cover three fundamental areas in my slice of the presentation. You're going to get a lot of data today because there are a lot of speakers, so we're going to try to go fast and dense. So forgive us if this is a little speedier than some of the other presentations.

I'll focus on what XML is. I'll give you some context around what this propeller head technology is. I say that lovingly because I'm half propeller head and half sales and marketing professional. I work on the financial services team at Microsoft. We deal with banking insurance and securities as a holistic group. I am responsible for our worldwide strategy for insurance, our vision, how our products should be marketed and used, and what partners we work with. I was invited to speak by the fine people from Essentium, who actually make products that connect your world and mine. They use our technology and XML to make your lives easier and help you prepare products and things like that. We're going to talk about the relevance of this new technology.

†Mr. Cole, not a member of the sponsoring organizations, is vice president and chief technology officer at Essentium Inc. in New York.

‡Mr. Kelly, not a member of the sponsoring organizations, is worldwide insurance industry manager at Microsoft Corporation in Redmond, WA.

After my presentation, this XML technology will be something that's a little more obvious to you. It is something that is readable and understandable. It is a global standard and it is one of what I call the Holy Trinity of global Internet standards. The first is Transfer Computer Protocol/Internet Protocol (TCP/IP), which is used to establish links between computers over the Internet. The second is HTML. The final part of that Holy Trinity is XML. XML is going to do for computers communicating over the Internet, what HTML did for humans communicating over the Internet. It's a standard lingua franca to allow computer databases and applications to pass data to one another and understand what the data is. It is similar to how HTML is used as a standard way to present data so humans can read it, and so they can see text and pictures and things like that. That's kind of a simple explanation. Like I said, I'll try to go through a lot of the general stuff very quickly.

The most important thing to understand right now is we have gone well past the use of the Web as an electronic brochure, with a picture of the chairman, a picture of headquarters, and the marketing text. That is the least attractive and least functional use of the Web. It is boring to the end user. It's even boring on a corporate Intranet to see cute graphics and some fundamental text. What's important is processing applications.

There needs to be functionality on Web sites. There needs to be functionality for consumers of e-commerce and business workers with a business application that is delivered over the Web. The use of the Web as a library and as a publishing medium for online reference material is wonderful, but we're well past that. We're not past the deployment of that because many companies are still lazy, way behind the curve, not paying attention, and haven't actually gotten most of their general reference content out there. The associations use the Web site to promote conferences like this. They can tell you session rooms and who's speaking. There's a lot of general use of the Internet and the Web by individuals. This XML technology is about taking the Web to the next generation.

What also is occurring is the way people use the Web today. It is traditionally on one type of device. There is this clumsy thing called a PC, where you have to learn how to type before you can really speak to the device. There are some ergonomic issues involved in learning how to make this a useful device in your life, but that's also evolving, too. More and more hand-held devices, cell phones, different types of technologies and devices are going to be inter-operating in this global network of information. As a result, it's breaking out of traditional application architecture. If you wrote a great application for the PC, how do you get it onto a cell phone? How do you get the application onto a hand-held device? The answer today is you don't. You do a lot of heavy lifting to custom create code to talk to that device.

That's great if you know there's only going to be five or six devices down the pike next year that might be relevant to your business workers or your consumers. In today's world, we have absolutely no idea which technologies are going to emerge. There's the wristwatch, the personal digital assistant (PDA), the Palm Pilot, the pocket PC, the cell phone, the lap phone, and the mall kiosk. You don't care how you expose your products. You want to get to a point, especially if you're in our

business, where the product is, ultimately, all digital. It's all information, whether it's the collection, the manipulation, or the return of a response.

I don't care what device we use or where my product has an opportunity to be exposed. I want to have an infrastructure that says my developers can very quickly expose the product and service out at that point of sale or service area; it could be a call center, a screen, a cell phone, or a booth somewhere in a mall. We want to be able to put our products, our services, and our calculations anywhere. That requires a change in architecture. That's where XML comes in. XML is a new standard for deploying application services out over the Internet, and this is a brand new revolution for computing. This is the next tier of the Internet.

At Microsoft, we've supposedly been threatened because everybody wants to move to dumb terminals again; that is, the Web browser as the client. All applications and power would reside on the server. We just know, statistically, from the way technology is going, no one's going to ignore the trillions of millions of instructions per second of processing power that have been put on desktops and hand-held devices. You need to have an architecture that takes advantage of both.

If you're going to have speech recognition be a way that you browse on a small device, or if you're going to use your cell phone to communicate in very elaborate ways, or use handwriting recognition on a hand-held device, you need local power and local processing. You need local storage. That will always be the case moving forward. Trust me. The way the messages will pass back and forth will not require that everybody be on one platform. That's the dream of this technology called Java from Sun. Everybody writes in one language and we're all connected. That's wonderful if you're Sun. It's not wonderful if you write in COBOL, or Visual C++, or Visual Basic and you write applications for other platforms.

XML has emerged as a way to pass messages back and forth from one application to another. I'll explain how it works for lay people. This is not going to be a propeller head session, although I might throw some buzz words out that you don't know. The way computer applications are written, the way your personal and your business applications will be deployed to you, and the way you'll have access to them is going to change forever. We're at the brink of that revolution right now.

Until now the revolution has been all about the personal computer as a stand-alone device. How many of you still remember when you first printed something in your cubical or in your office and were thrilled with this power? This $3,000 typewriter was a stunning device. Then we networked our computers and that brought another method of sharing documents and collaboration. It was a very crude way, but it did bring another level of interoperability to our corporations or our business use of the computer.

The Internet came along and suddenly networks were networking: my business to your business, our e-mail to your e-mail. It has changed the way we do business. It has streamlined communications. It has started to coalesce a lot of our communications away from the myriad of platforms we were using: the Post-It note, the fax machines, the telephone, the scribbled note in the meeting that has to

be retyped later. These are very dumb methods of doing work. These contribute to very poor workflow.

We have a global network now on the Internet. We have global standards for a presentation called HTML. There's no reason why, with all these new portable devices, we can't sync up all this information in a richer fashion. So that when you go to a meeting, you won't have to take your pad and write notes, and then go back to the office to look something up, which then results in another meeting. You will have access to the technology in the meeting, access to the database, access to the data, access to the calculations, and access to the Web. You'll make decisions faster and you'll have fewer meetings.

We're all prisoners of the data we can get access to. We're prisoners to how much we're trying to get done versus how clumsy our automation is, versus how many different places we have to go to get an answer to be smart knowledge workers for our companies. We have an industry filled with task workers that don't really do knowledge work. The only way to do knowledge work is to get what we hear on the bottom. People are using the Web and the Internet for entertainment, for personal connectivity, pictures of the kids, viewing the day care center while you're at work, for their family photos, and for their family videos.

All of that leads to a need for corporations, large enterprises and small, to come up with what Bill Gates said is their digital nervous system. What is your electronic version of wiring all your business applications together, so your employees can use them, so your consumers or your business customers can get access to some of the data that brings us all together to use these global standards? What has emerged is the need to have a strategy for building a digital nervous system.

That's the revolution thus far. That's the opportunity that we have. If we move forward, mobility is what it's all about, wireless connectivity, ubiquitous access to your applications. You don't have to get back to the cubical and back to your PC to get your mail or your business application. But if you happen to be near another terminal in the building, near another PC on your network, and near a PC at the office of a customer or at a hotel, you should be able to get secure access to your information and applications. XML is the technology that's going to enable that kind of distributed computing.

The challenge with multiple devices now is also synchronization. The insurance industry has been given a brand new life to do what they used to call agency company interface, which is the ability to interface the point of sale and service professional's database with the carrier's policy administration system.

I've been intimately involved in all of the initiatives that have been surrounding this, and what is embarrassing is how little progress this industry has made in the last 25 years. We have been working with ACORD and with other standards bodies worldwide to put their efforts into XML to develop new methods of using the Internet. It would be a free or almost free network, compared to an expensive systems network architecture (SNA) or a private network that carriers pay a lot of money for. They want to use the Internet as the open network; use the open

protocols of HTML for displaying the information, or the forms, or the data and use XML to identify what the data elements are, so databases can pass information in and out and do business processing and pass results in a standard form. We've been trying to wire together the financial services industry, banking, securities, and insurance to take advantage of that. So the concept of synchronization becomes easy.

If you ask this industry silly questions, like how does your industry handle sending attachments over their private network, it will tell you, "We've worked five-and-a-half years on that, and we've spent $17 million. We have nothing to show for it. It's hard to send a picture of the house over the SNA network. Go ask a 14–year-old whether he or she can send attachments ? You have 11-year-olds doing better EDI than this grand insurance industry.

So we're hamstrung by our past technology. Our past technology providers have tried to screw Internet technology onto their old platforms and say that's e-commerce. What's really needed is a revolution in using the browser, e-mail, and XML technologies to streamline the work flow, so that the insured, the risk manager, the underwriter, the actuary, and everybody in the value chain in insurance has a common platform to communicate documents and information back and forth. That creates a new level of application design and, more importantly, a new level of workflow design. That is, real workflow on a global network versus workflow inside a department, or workflow inside a couple of departments, or workflow inside one company. We've got a global workflow engine called the Internet, and that's what this is all about. There is more mobility in workers, and a lot more people are moving in and out of different roles in their lives.

Take a look at the Internet. We've gone past phases one and two. Phase one was marketing and basic digital real estate to promote a product or service. Phase two was the experimentation of doing calculations on the Internet, charging for a credit card and conducting business. Have you purchased something on the Internet? Would you agree that it was a wonderfully stimulating experience—that it was easy to use and you didn't have any problems? How many of you have had to type your address in more than once to go to multiple Web sites? How many are getting kind of tired of that? Do you get impatient when it takes eight seconds for a Web page to show up? Do you believe the insurance industry sends its product to the customer fast? You usually get your product in ten weeks. If we cut it to five, we'd make a quantum leap, right? People are waiting. They're getting upset when they don't get sub-second response time from a Web server.

The world is changing, so we've now moved to a digital economy where people are safe purchasing things over the Internet: khakis from L.L. Bean, toys, insurance, and so on. It is a very slow process to purchase insurance over the Internet. People don't want to buy things today the way they bought them yesterday. They don't know what they're buying when they buy insurance, so this is a raw, cold experience and the Internet doesn't help them. They're certainly not going to read the 800 pages of text, the information you would have to put out on a Web site, to give them good insurance advice to help them learn how to make the decision for themselves.

So the psychology of buying comes into this. Sales agents will not go away. Insurance will not be sold on the Internet the way banking and securities are done on the Internet. People understand what an equities trade is. What's the price? How many shares? What price did I get it at? What price did I bid it at? We've got a small set of data elements. Banking is the same situation. You add, subtract, multiply, and divide, and you're done with banking. Forget insurance. There are very complex transactions. It is a complex industry. The reason it's an incestuous industry and no one ever leaves it is, because once you've acquired all this arcane knowledge, where else are you going to use it? You're stuck. You have this valued asset that can be used inside this industry only. Once you've gotten into the club and figured most of it out, you can only talk to people in the industry, because no one else understands what it's all about.

The ability to virtualize those experiences and to make the knowledge work better, to help make decisions better is exciting. I'm interested in manufacturing a financial instrument to do the calculations, to build a model, to deploy it quickly, and to tweak it, based upon regulatory compliance. It would also be exciting to be able to adjust those models in real time based on data mining.

We have a very clumsy process for the way we make our product and then get it out on the shelf and get it approved. This is a very silly industry. If you talk enough about it to the average person, they would hang themselves to understand what we have to do to get a product out. So we've come to this revolution. We've come to the digital economy moving data in and out of these different devices. This is the world that we're going to work in. There is a very elaborate world of different platforms and different devices. There needs to be some new technical architectures that will let these devices all participate in e-commerce and any personal entertainment.

The other aspect that's important at Microsoft is the understanding that we need to have a common platform. People shift their roles during the day from a family person at home to a business mode, when they hit the cubical and get on the network. You constantly have to turn devices on and off and log on and jump in and out of different technology tools to do all these things. With the Internet as a global network and with XML as a set of standards to make all of your information accessible no matter where you are (whether you are in your car, your home, or at your office), the shift of roles will be a lot easier. You'll have less human lifting to get in and out of these different roles.

The founding vision of Microsoft in 1975 was to see a computer on every desk and in every home. Thinking that there could be a computer in every home was lunacy in 1975. How many of you have two computers at home? That's a boring vision now So we have a new mantra: anytime, anywhere, any device computing. And that's going to be enabled, again, by this technology called XML. It's a new initiative at Microsoft called Microsoft.net. It's on the same platform that the PC was on: the Internet. We have a brand new platform for doing applications. Microsoft wrote the tools, the operating system, and some of the best applications for that platform. We're about to do the same for the Internet. We're going to give people a new user interface for the Internet, a new way to write applications and all of those things

will be based in XML. XML will be the technology that this company, the largest software company in the world, is using to rewrite all of its applications. That's how important this is.

There are some fundamental deliverables that come out of these technologies. The connectivity comes out of this TCP/IP standard you don't need to know about. The presentation of data and information comes out of this standard called HTML. The programmability of Web sites, of applications, the ability to do calculations on the Internet and the opportunity to pass results back in a very easy way will be done via XML.

It's a standard for self-described data and document types. XML is many things. One thing to remember from this session is it's a nascent technology. It isn't cooked yet. It is still emerging. It is a very powerful technology. It is changing every computer application developer.

You are using XML now and you don't even know it. You are hitting many Web sites, which are already using XML in background processing. Many commerce servers, and many commerce experiences that you go to are using XML already. Microsoft is leading a lot of standards charges in lots of different industries to create the vocabulary, if you will, of the data elements that will be passed back and forth.

Let's take a peek at what it is. Let's get an understanding. If we want to present my name in a browser, we use the HTML standard by tagging my name with the start bold tag and the end bold tag. HTML was derived from a fundamental technology called Standard Generalized Markup Language (SGML). It was in the document publishing business. It came up with the idea of tagging all the information about a document, so you could describe it, print it, and put information out to technology platforms to print and distribute documents. HTML is just a subset of that idea, and it's a very easy thing to understand.

If you use a character-based word processor, you also do this to bold or italicize text. What if that's the same way I'm presenting the data to the browser? I also pass this kind of contextual information behind the scenes. Start customer name, start first name Kevin, end first name, start middle initial S, end middle initial, start last name Kelly, end last name, end customer name. So I used the HTML commands to paint it on the screen for the user. I use the XML tags in the document to allow a database to suck the data out of the text file that passed over the Internet.

Remember the eight-second trip? The data comes to the browser. At the same time as it comes to you, it can be parsed out. That's the technology term. It is parsed by our browser or other browsers. It is handed out as raw data. It creates a hierarchical data tree of information that an application can use to suck the data in, do some processing, and perform some business logic. Again, with the instructions on that Web page for the buttons or the calculations, return the result back to the server, pass the work to another desktop or another IP address or computer address. This is a very simple, elegant technology. It's a way to describe data.

What we've been working on in the industry are standards for how you describe an insurance policy. What you're going to hear are other standards initiatives and other programs that are being built that describe mathematical calculations, that describe other issues around actuarial development of products. If we can come up with industry standards for doing this work, we can solve interoperability. I can pass insurance policy information from a Windows NT platform to Solaris, to a mainframe in a very easy fashion. In an industry that understands big text message cues, it knows how to do these kinds of systems. So this is nothing new, but it is a very elegant implementation of some very simple technology.

The other beauty of this is that I take one XML data stream with all the vocabulary hierarchically arranged the way it's supposed to be, and I point it to something called an Extensible Style Sheet Language (XSL). I can take one data stream and have it viewed one way. We have programs now that emit and consume these XML messages. We can very easily conform this same set of data for the executives, for the agents, for the actuaries, for report generators, and for the claims adjuster without constantly having to reprogram their sites or their view of an application. So this will streamline the internal application development in XML as well.

There are a number of key characteristics for XML that are going to make it a very powerful technology. It is simple. You've already gained more knowledge today by looking at it. You understand it. It's a tagged specification.

The X in XML stands for extensive. You don't have the problems that you did in the past with EDI. We agreed on an EDI standard, and it was a fixed file format. We counted positions to know where text was or to know where values were. If I change that text file, it breaks when it gets to the other, because that parser hasn't been reprogrammed.

This parsing is dumb parsing. It just pulls out all the tags and passes the data out. So if you add or subtract data to an XML file—and this is a very simplistic explanation—as long as the program on the other end still sees all the tags it knew it saw, it doesn't care that it saw other stuff. If somebody has decided to use some additional extended information, they'll catch it. They'll know where to look for it, and they'll actually be able to process it. So it makes life a lot easier in the grand scheme of EDI. It makes standards that don't break so easily. Standards that don't break so easily are much easier to adopt and more cost efficient to maintain.

The business relevance for all of this is because we've come up with some great learning with respect to the Web. The Web is a tremendous business presentation model that can be used to put a picture of the product, the price of the products, calculation fields, the form for ordering the product, and the form for requesting service. We have this great model on the Web for conducting business, whether you're a knowledge worker inside the company at the call center, or you're the actual consumer or risk manager working with the institution. Business is very forms driven and forms are very data-driven. XML is all about data—how to present data, how to structure it in a highly ordered fashion, and how to pass it in and out of different systems. So this Internet technology that's been evolving over the last few years has been tremendous.

This XML technology lets us get the data pulled out of the arguments over technology: it has to be a mainframe; it has to be Sun; it has to be Microsoft. As long as it will be able to emit and consume these XML messages, it doesn't matter what you wrote the logic in. You can go ahead and buy a piece of logic from this vendor and a component of logic from that vendor. It's easier to integrate them. There are some challenges. If you have very high demand, highly scaled models, then this will be kind of bulky. So there might be choices programmers make where XML won't always be a great solution. Compared to the fights we have today about technology, this is an elegant solution to get us out of that argument.

This is what the future will look like. It will be a set of Web sites and programming services that are available with a common language in XML. It is easy to consume from all of these different devices. There was a time, when the Internet began, when everybody thought it would all be JAVA. Everyone was going to run a JAVA virtual machine on each of their platforms. Programmers would write in one language, and that's how we'd get all this heterogeneity erased. But that's just not the way it's going to be. It's going to be a sea of XML messages being passed back and forth from devices, from platforms, from big Web servers to PCs, from PCs to cell phones, and then to smart cards. It's going to be a much different model that is much more flexible. We're talking about evolving the world to something we called software as a service.

If you think about a procurement order-processing model, it would basically be request and response XML architecture. The server pings out a piece of information to a back-end database. It knows the kind of response it's going to get, because it's a standard, whether it's an internal corporate standard or an industry standard. It knows what response to expect, how to parse the data out, and what to do with that data. We're seeing remarkable cost reductions in systems being built to order (pencils, Post-It notes, and so forth).

This is a very simple evolution that we've come to. In 1997, Microsoft released an architecture called DNA, which showed how to build an application that no longer just runs inside your corporation for a few thousand or a few hundred desktops. It might have millions of users, and it might have to scale to demands you never did before.

In 1998, we released Windows DNAFS. Some of you might have read an article or heard about this and didn't understand what it was. It was Microsoft doing XML in between application components. A customer object in a policy administration system could pass an XML message to a banking system, so it could share customer information with a banking system or integrate another application functionality. This was a very early precursor to what we released in 1999. It was a technology called Simple Object Access Protocol (SOAP). It was the ability for me to have a core object running on a mainframe or running on an AIX, an IBM UNIX system platform. Pass an XML message to a computer output microfilm (COM) object running on Windows NT. In the past, these were nightmarish ways to try to do business processing. But we can use XML to pass the requests back and forth between these platforms.

Now to the year 2000. What we've released is Microsoft.net. This is a new platform that uses XML as a foundation to make it easy for programmers to write applications that are going to live in all of these different places. I have another reason that demonstrates how critically important XML is going to be to the world of technology and not just the insurance industry or a few places. We are literally replacing all of our fundamental models for how we build software, which we're pretty good at and pretty well known for. We will replace them with this new kind of platform. This Internet mounted platform uses XML to pass all this information back and forth.

To wrap up here, XML is a mission critical technology at IBM and Microsoft. It's not a fad. This is not a lightweight technology. This is absolutely vital. What we're doing with this is creating a brand new user experience. There will be a brand new desktop, a brand new user interface for personal users of computing devices that will blend in all of the Internet functionality that you need where your applications and personal files are stored. It will make it easier for you, as I've said, to move in and out of the roles that you play. It will be a brand new world in which consumers can automate their lives, to interact with their families. It will be a brand new world inside businesses for IT professionals who have to quickly build applications and products and get them out to market. This changes the rules. It streamlines application development.

For knowledge workers, it will give them richer and richer integrated applications at their desktops so they can do their jobs better. It will also allow you to collaborate more with the people downstream from the products that you develop in a more collaborative fashion. So real feedback can come back digitally from data mining, or real results in sales or in claims. It will actually be usable through the kind of tools these gentlemen are going to talk about to tweak the product, to adjust the products without so much of the human intervention that goes on today.

From a developer's standpoint, it is going to change the way they build applications, because the world is different. They're not only writing for the PC; they are also writing applications for the Internet.

From The Floor: Do you have to learn a whole new language for this program, or can you start using XML and integrate it somehow with all the programs that you already have?

Mr. Kelly: That's a good question. XML is not a replacement for existing technologies. If you have a set of programs in COBOL, and you have something that's written in VC++ on a different platform at another enterprise, all we're asking is that people begin to adhere to the ability to pass data in and out of that existing application with an XML interface. XML will easily pass through all over the Internet and participate through a service like Business Logic on this particular platform. So it's going to allow that. You'll get an immediate impact. It looks great in the chart, but someone has to define what the messages are to go back and forth between those business applications, so that the process can be done. XML is now the global standard. It's already been agreed upon in the last few years that it's the flavor that will be used to pass the message. It's just a matter of getting the

industry to do two things. Will we use the industry standard or will two parties map their two standards together? Those tools are coming out together very soon.

From The Floor: Is it just investment of the data or is it the actual manipulation of that data?

Mr. Kelly: As these gentlemen will tell you, it's going to be a combination of both. There will be a scheme identifying the actual data, the data type, and what you can and can't do with the data. You can also express business logic as a function of passing an XML parameter or tag and it can inter-operate with software or business logic on a particular machine. As I said, it's such amazing technology. The document type can be each of these things, it can actually be a programming model or language.

From The Floor: It could also be encoded?

Mr. Kelly: Yes. It quickly gets very technical and very complex when you start to look at that. My role here was just to try to tell you that this is a very important technology. I wanted to give you a layperson's view of what XML is. These gentlemen are going to get much more specific as to some of the business functions that are relevant.

Mr. Jawwad A. Farid: That's a tough act to follow. I come from a computer science, business technology, and actuarial background. I've spent six or seven years in consulting. I just came out of business school in May and started this educational venture. I am going to take a very simplistic view and talk about what XML is and is not. I went through a one-year period, during which I read maybe 15 to 20 books on the subject. I had no clue what the subject was, what it did, what it did not do, and what were all the issues. Still, we actually got down to building the application and that's where a lot of problems and issues result.

Let's get some terms down. A client is a PC or a machine that's running an independent browser. A server is a machine that sends Web pages to clients. When you are attached to a Web site, the machine that you're working on is the client and the machine that the Web site is on is a server and they talk to each other. An XML document is a document that's organized in a specific structure with a specific format. Java is a programming language.

So what is XML? It's a collection of tools and it's a collection of standards. The tools work on the browser (the client) and the server side, and they allow both of them to read and process XML documents. The standard defines what a document is. How is it structured? What goes in it? What belongs and what doesn't belong? The key use, as Kevin pointed out, is exchange of information.

There is a collection of tools and a collection of standards. So what's the big deal? What's the fuss about? The fuss is that it's extensible. There are three things that extensibility means. It means the ability to define new structures, the ability to define new languages, and the ability to establish standards. Each of these items by itself might not mean a lot, but when you put them together, what are languages?

Languages are a compilation of a collection of structures. What are standards? Standards are structures or languages that have been accepted by a lot of speakers.

How would define a standard for the insurance industry? One of the things that you could look at is policyholder information. I have a structure for policyholder information that has age, gender, identity, geographic location, profession, and other information. I have another structure for underwriting data. I have another structure for claims data and another structure for terms of contract. These structures are like Russian dolls. You would have additional structures between them.

I would start off by using them for internal use where I would be just using them to talk to my own systems. I would soon have customers and suppliers who would be using these things. I might even find another insurance company that wants to talk to me about exchange of information and it has its own structures.

You start off internally. You move to external users. Then you might have six to ten in an industry group. Once the standard is adopted by the whole industry, you have a new method of exchanging information about whatever you want to exchange information about—it could be about policyholders, terms of contracts, claims, or whatever.

Is there anyone who can take a shot at a one-line explanation of what XML is?

From The Floor: XML uses program language to interface between different applications.

Mr. Farid: Perfect. You should be sitting here because you know that. I'd like to discuss what XML is not. I think that will help clear some of the confusion. It's not a standard programming language. A standard programming language allows you to define structures, to display structures, and to play with structures. XML is very good on defining structures, but you do need a programming language to play with those structures and a programming language to display those structures. XML is not going to take your IT problems away. You still have issues and problems, but it's a very powerful tool. It's not a database or a medium for storage. It's primarily a mechanism for the exchange and transfer of information.

Let's move on to the case study. We're an online education company, and, for us, there were a number of issues that stood out when we were designing our systems. The first issue was that we wanted flexibility and people to have the ability to do whatever they wanted without compromising performance. We also wanted to customize individual experiences, and we wanted dynamic content. The big challenge for us was that we needed a method for transferring really large chunks of information. Formatted and real time have no limits on the size, or length, or depth of that information and displayed in such a format that could be used by people. All of this was driven by our desire to empower customers to generate their own courses and their own concepts in real time. Third, was the ability to exchange information with third-party solutions on the back end as well as on the front end.

So what did XML do for us? What was the benefit? What was the advantage? Our first step was to define a structure, the structure that we would work with on the education side. We defined a course, which is something that teaches you something in this format. A course had more than one session. Each session had more than one concept, and each concept had an explanation, an example, a sample problem, and a practice test.

This is what the system did. I will do two things. Let me first walk you across the flow and then tell you what it is that the flow did for us. It was very simply a client, which is a machine with a browser. Send a request out that says I want to take this course, but in that course, I want to have these specific sessions and these specific concepts with these specific examples. That goes out to an XML engine. That converts it into an XML document. The XML document goes to the server. The server decides where the knowledge base is, where the items that are needed are, and it dumps out data that goes back to the XML engine. The XML engine converts that into another XML document that goes back to the client.

So why did we take such a torturous route? We are converting stuff into an XML document, and then converting the stuff back into XML document, and then going through the cycle again and again. The key thing for us was that we wanted the ability to integrate with third-party solutions on the front end. We wanted to talk to external systems on the front end where you will be working with trained solutions companies. We wanted the ability to work with third party back ends, where you'd be working with the knowledge bases the company already had. What did XML do for us? Choosing this format, and using this step-by-step approach, it was very easy for us to go into a client site and say, "Tell us what you do on the knowledge and training side. We could have a solution that would work with it."

Beyond the third party integration side, the real important thing was having two ways that you could steer the course. If you're on the Web, a client would come in and say, "I want to take this course." You could go back to the server. You could get the first concept, and then bring it back, give it to the client, go back again, and get the second and third concept. There's a regular process of going back and forth. It creates a tremendous amount of network and client-side issues. XML went to the server just once. It grabbed everything that it wanted for this specific client. It brought it back to the client site and the stuff resided on the client. There was a major performance gain. We avoided the going back and forth to the site, too.

Let's go back to my conclusion: what is it that we are seeing as developers or as technology people that's happening in the industry? We expect to see a lot happening on the information exchange side, and not just in the insurance industry, but across the board on the financial services side. We see XML used with and without EDI for transferring information between businesses. There's a lot that's happening on the integration of financial information side. It's a very powerful tool that people have just started to notice. I noticed it a year ago and Microsoft wrote one in 1997, so there's a lot that still has to happen across the board.

Mr. Mark D. Horowitz: Our next speaker is going to be Ron Cole who is vice president and chief technology officer at Essentium. Ron's going to be talking about some of the work that's being done in the insurance industry.

Mr. Ron Cole: I've been in the insurance or financial services industry for more than 20 years. Before that, I was database administrator for another company for five years. I have a pretty good knowledge about what has been going on here. The way this session is billed is we're to assume that the people that come to this session don't have any particular knowledge about XML. I'm going to give a little background and foundation about why we might want to use and embrace a technology like this and how we might apply it to what we're doing as far as our architecture and platforms.

I think the most important thing to realize is that any standard will do as long as it does some fundamental things. If it improves your communication, reduces implementation time, and eliminates overhead, then you can take it and run with it. In other words, it's like motherhood and apple pie.

Communication is important because if you don't have that communication, then you really can't interchange information. A standard is an excellent way to create communication. If you can predict what I'm going to send you, and you're able to interpret it, then you can understand the information that I want to transmit, and you can deal with the environment that you're in.

I'd like to give you a very simple idea about why standards are important. Let's look at these different environments that we have to operate in. We'll use an automobile as an example. There are certain standard things in automobiles, and it's extremely important that these things are understood by the user.

In all countries the steering wheel is nearest to the lane where you can see around the car in front of you. It's always nearest to the middle of the line. It's on the same side of the car on every continent, except the British Virgin Islands, where they have the steering wheel on one side and they drive on the opposite side of the road. In case you ever go there, it makes it very hard to see around the corner. The accelerator is very important. If the accelerator is not in the same place, you might make a fatal mistake. But if you get down to the trunk switch, it really doesn't matter where that switch is. It's not going to affect your ability to survive in that automobile. So you might find it under the steering column, in the glovebox, or wherever. It's kind of like an Easter egg hunt.

Another thing that has some standard components that we work with everyday is a house. You have to be able to find that door handle in the dark or you're not going to be able to get in. You'd better be able to find the light switch or you're not going to be able to find anything else in the house at dark. So these things are sort of ideas about why standards are important.

About every five years, someone comes up with this new, greatest standard that's going to revolutionize everything, save everybody a whole lot of time and everything. What usually happens is management discovers a radical work method

improvement, introduces the new standard across the organization, and everybody receives tremendous benefits. That's the plan.

What happens is management establishes a standard. Workers resist the new work. It changes what they do. They don't get any extra budget for it and everyone receives marginal benefits. There's also this little shell game that goes on where somebody in the organization thinks of this new idea and how it's going to save the company one million dollars this year. This work appears under somebody else's shell in this little shell game in the organization. Actually, it doesn't save the company any money. They just wind up doing work in a different department.

When we get to something like XML, we find that we suddenly have an opportunity to really blend improvements in the way people operate. There are some really nice things about this new XML gadget. First, it does one thing for everyone and that allows us to define data clearly. In the XML structure, even somebody that doesn't know programming languages can look at a well-formed document and dig information out of it just by reading it.

Then we have another part of this little scenario called Extensible Style-sheet Language Transformation (XSLT). It's the way we dress it up. It's analogous to going to the gym, getting our muscles in shape, and then putting on a really nice suit. You can think of it like that. We have to show it to someone. We get down to HTML. We have a well-formed data structure, and then we dress it up with XSLT with nice little neat formatting and things, and then we show it with HTML.

Another standard that follows a similar structure is something like a Microsoft Mail Merge. You might come up with a form letter in Microsoft Word. It has nice fonts and everything's spelled correctly. The grammar is correct, done by some wordsmith. The salesman comes in, gets his contact list of 20 people, and says, "Print 20 letters putting the name in there." You can transpose that vision onto what we're trying to come up with in the way of a platform.

Is XML going to be our new holy grail? Is it the answer to all our problems? Probably not, but it does provide a lot. It's based on a very sound technology. It has critical mass of support, which is imperative for a standard, and it's relatively easy to use.

It's based on a subset of SGML. SGML has been around since the 1970s. It has critical mass of support. It's all over the Internet, and it's free. If you want to parse XML, you can download the parser from some Web site and you can open any XML document and parse it as long as it's well formed. We can parse Kevin's document, I'm sure. It's a flat programming model. It is very easy to understand, with very few operators, and very few commands.

So we have a technology, like Kevin was describing, that can be here, there, and yonder. It can be in a client-server sort of environment, a server-client environment, thick or thin. Basically, we just have to make the decision, and there are a lot of decisions to make. The best thing about those decisions are that this environment is very forgiving. If it's on a server today, it can be on a client

tomorrow or it can be split between the client and the server. Whereas, in the past once you made those decisions, you were stuck with them.

One thing I want to add here is that, in this environment, these different constituents that play in this environment can actually identify themselves to the environment. So if I'm a server sitting over here, and I'm trying to interact with a client browser, I might ask, "Do you understand XML?" It says, "No." Do you understand XSL? No. I know what I can do with you and what I can't. This is very important, because, in the previous version of Web sites, that wasn't there. In fact, I used to do Web sites, and we used to have standards. We used to have a 4,000-byte picture. It has to appear in less than three seconds. It has support from HTML 2.0 to 4.0, because you didn't know what was running on the client.

There is a funny story about that. I was contacted by a company that wanted to do a Web site for me and wanted to do some sample pages. I said, "Okay, but here are our standards. We need a 4,000-byte picture, and so on." So, it put a sample Web page up there. It told me to go there. I went there and after eight seconds I just clicked off the site, because I still hadn't downloaded the first graphic art picture that was on their Web site. So it's really nice to have those capabilities built in and to be able to organize your environment.

What if you have a really sophisticated environment? On the client, you would have the XSLT, which is sort of like your Microsoft Word template. On the Web server, you have the ability to respond to data requests. It would send down this XML data structure. It would get merged with the template, and it would appear in front of the user in the form of HTML.

On the other hand, if your client didn't understand very much and was still running Internet Explorer 3.01, your server could then make the decision to use the XSLT, merge it with the XML, produce the HTML and send it to the browser. The browser would look the same, because the HTML is being produced for the client.

There is another thing about this. Everybody thought this was going to be done in Java with Java applets. There's a real neat expression going around that says XML gives Java something to do. It really is true, because without that data structure, without that rich content of stuff moving around the Web, you can basically make these pretty little logos and flash and things like that, but you can't get to the core of the business without the XML structure.

Fortunately or unfortunately, as the case might be, a typical XSL support environment looks like this, because we have all kinds of pieces out there on the planet. We have browsers with XSL support. You use scripts at the client site. You have XSL and XML coming down from the Web server. You might also have some data being pulled off the database server merged with the XML and sent down to the client to then be displayed in a nice pretty format. Because XSLT, XML, and HTML are all well-formed documents and all valid XML structures, (because they're all based on SGML), the operators that can operate on these things can often interact with the other one. What this means is, if you're trying to interact with an application from company A and an application from company B, you can go

through an intermediate transformation through an XML parser that will convert both of those into a like structure that your common HTML page can display. This can come in very handy if you're trying to write an application that does e-commerce business-to-business.

Our approach is to basically operate very closely in the XML world. We have a facility called a calculation server. This calculation server will take a product definition, will accept requests from an application, calculate the reserve, and it will give back the results. It takes XML in, and XML out. That's the only application programming interface (API) there is. So we have a product repository that gets exported into what we call IPML. IPML is nothing but a well-formed XML document, but everybody likes to give it his or her own name. We have an XML template, an XSL style sheet, and an Essentium parser. With those, you can make all sorts of output displays, reports, and interact with other systems, or whatever you can imagine in an XML environment.

The other day, we were talking to an insurance life administration vendor that happened to run on an AS/400 environment. We were on a conference call. So their technical, outside consultant was asking questions about how we were going to interact with this calculation server when we're on the AS/400. I said, "Our system actually is the calculation server, and it takes XML parameters in and XML parameters out." He said, "I know. It's a SOAP application, next question." Before this came along, if you were trying to interact with an NT 4.0 environment connected to an AS/400 application, you might have had a few more obstacles.

In a simplistic example similar to the sample of an XML document that Kevin was talking about, you can make up your own key words, and you can value them. Then, inside of your program, there are methods and parameters that you can use to ask what the value is of this thing coming in. In the XSLT, you can say put the value for this key word here on the page. That's basically what it looks like.

From The Floor: What language is the calculation actually being done in? Does it take a programming language or is one even necessary?

Mr. Cole: Our calculation is actually done in C. We have a parser, and it packs it in. Then we store the actual formula in XML. You can paint your formula like this, and that is actually what you use to do the calculation.

Mr. Mark D. Horowitz: I've been at Towers Perrin for the last ten years. I've primarily been involved in pension valuation software for the last 25 years. What I'd like to do is to talk about things built at both a high level and a low level and to give you some examples. It was last year that I went to a session similar to this on data standards, and I didn't really understand what the topic was. It turned out that it had to do with the representation of mortality tables on the actuary's Web site and a way of being able to distribute them. It was obvious to me, since I had been reading about XML for a couple years at that point, that XML was the solution. But XML was not too widely known at the SOA at that time. Hence, we scheduled this session to introduce this technology.

One of the things I'd like to talk about is the degree to which other professions have adopted XML. In 1998 the World Wide Web Consortium adopted the definition of XML as a recommendation. The World Wide Web Consortium is an autonomous body. It happens to be located in Tokyo, in Paris, and at MIT. The World Wide Web Consortium has a group of people who think about things on a global level. For example: What should the Internet consist of, and what kinds of recommendations can they make to software vendors or to people who are going to be using different aspects of the Internet and the World Wide Web? It has standards for things like XML, HTML, and so forth.

XML has been evolving tremendously over the last couple of years. XSL, itself, is a recommendation like XSLT. There's something called XSchema, which, as far as I'm aware, actually began with an initiative of Microsoft at least three years ago, called XML Data. Let me get a little bit into a couple of the other industries that are using XML in very serious ways in four different areas: insurance, accounting, human resources, and finance.

A couple of people made reference to the Agent-Company Operations Research and Development (ACORD) Organization. ACORD is a non-profit organization that develops standards for the insurance industry. Among the standards that it has developed is something called OLife. OLife is an organization of components of policyholder information. One of the things that ACORD recently did is translate their definition, OLife, into an XML representation. What this means is that it will be possible for different insurance organizations to exchange information if they have created that stream of information according to the OLife definition. It means that it can create a stream representing a group of policyholders, send it to another organization, and the other organization can understand the content of that data, without actually having to use the old style of data, where the first byte means something, the second byte of data means something else, and so on. The XML organization of the policyholder data allows things to be very flexible and to contain varying amounts of information and intelligently organize information. If you're interested in actually reading some of the white papers of ACORD and its definition of the OLife standard, as well as their property and casualty initiatives, you can go to its Web site: *http://www.acord.org*.

The accounting profession has come up with something called, the Extensible Business Reporting Language (XBRL). It's a framework for creating, exchanging, and analyzing financial reporting information, such as annual and quarterly financial statements, but it's done in XML. I mention this for a couple of reasons. Some of the members that I've included are the American Institute of Certified Public Accountants (AICPA), Arthur Andersen, the Canadian Institute of Chartered Accountants, Dow Jones, Fidelity, and IBM. This is a serious development using XML. Although most people here probably never heard of XML before, it's something that's taking the world by storm. I mention some of these organizations because they're very serious about this stuff.

What this kind of standard allows them to do is package data. They define a particular piece of information that they need. They create tags, which are

essentially names for the data. They are then able to communicate the data between different platforms and between different manufacturers.

I was talking to somebody yesterday at the NAIC booth, and I asked if he is doing the same thing for insurance industry information. As far as they're aware, the NAIC is not doing anything with this. I wonder if Ron could comment on this.

Mr. Cole: Actually, Essentium and Microsoft have had discussions with the NAIC about defining a standardized XML expression for them to get regulatory compliance and documents filed and a number of things like that. We're trying and there are people that would like to do that.

Mr. Horowitz: Right, and it's very important. It allows for a much easier way to exchange information between machines and organizations. I'd like to discuss another area that's tangentially of interest to actuaries: human resources. When I started doing a search about a year ago to see what people were doing with human resources and XML, I came across the HRXML Consortium (Human Resources XML).

For the time being, most of their thrust is being able to do job posting, candidate profiles, and resumes on the Internet. They've created their own standards for defining what a resume is. A resume consists of a job history, education, and things of that nature. Each of those things has sub-components.

As soon as you start thinking about what these data are actually composed of, logically, you can start creating an XML expression for these things and store the information. By using XSL, you can display the information independent of how the data are actually represented or stored. It essentially creates a data model. That's part of the reason this session is called XML Data and Metadata. The metadata is the overall description of your data as opposed to the actual data, themselves. I know the HRXML Consortium has intentions of being able to include employee benefits and other kinds of personnel related information.

I wanted to mention at least three different initiatives in the finance area. We must show how important other organizations consider the ability to use XML to contain their data. For financial products, there's the standard Financial Products Markup Language (FpML). There's something called the Financial Information Exchange (FIX), protocol that is primarily for financial information. For a capital market data exchange, there's something called FinXML.

Financial Products Markup Language (FpML) is a standard, primarily for financial derivatives and business-to-business e-commerce. It has a Web site, which is *http://www.fpml.org*. Their goal is to streamline the creation and maintenance of e-commerce for describing these products. FIX is a public domain specification. It's not specific to any one commercial organization, and it's primarily a messaging standard for being able to exchange real-time securities transactions.

The third that I wanted to mention is FinXML, which actually came about after FIX, because they actually have been able to work with other financial protocols, including FIX. FinXML is a framework for being able to support a universal standard

for data interchange within the capital markets so that different organizations can communicate details of different kinds of financial transactions.

The reason I'm mentioning some of these different things is to show that there are other industries: business, in general, human resources, and many different aspects of the finance-related industries. The Society of Actuaries, in some sense, has an opportunity to start to at least catch up to the 1998 and 1999 technical standards by using XML.

I want to shift gears a little bit and give you an example of some efforts by the World Wide Web Consortium. It has to do with mathematics. Anybody who's an actuary, at least at some point, had some interest in working with mathematics. There's a group called the Math Working Group, which is associated with the Worldwide Web Consortium. What it wanted to do was be able to put mathematics on the Web.

If you ever go to a Web site that happens to have a paper that has any kind of mathematical content, 99 times out of a 100, any expressions that you see that represent mathematics have been done with some kind of software to create a picture. The picture is actually stored in something like a GIF file that's referred to by the HTML. It's not actually a mathematical object, which can be manipulated in any meaningful way.

The Math Working Group came up with a bunch of goals for mathematics on the Web. Some of its goals were the ability to use mathematics on the Internet for both teaching and communication, but it wanted to be able to support mathematical notation in a fairly robust way. It is as robust as any kind of papers that you'll see in any of the mathematical, physics, or engineering journals. It not only wants to be able to represent and display mathematical notation, it also wants to be able to have something that could actually represent the meaning behind the mathematics. I'll give you a couple of simple examples a little later. There are a number of ways of representing mathematics. There's something called TEX, created by Donald Knuth for formatting mathematical notation. It's one of the standards used at universities. Mathematica is another tool for creating mathematics.

Additional goals of the Math Working Group were the desire to make the mathematical representation readable by people and analogous to XML, simple for software to both process the mathematical objects and to create them. Finally, there is a level of minimal functionality, which is to display the mathematics and to be able to print the mathematics in fairly high resolution.

An example of Math ML is shown below. Math ML is a vocabulary and it's an application of XML. When I say an application, I mean a set of XML tags that are going to have certain meanings. For Math ML, the kinds of tags that have meanings are things like msup, mrow and so forth.

TABLE 1
MATHML EXAMPLE: POLYNOMIAL

- Image: $(a + b)^2$
- Presentation markup:
- <msup> <mfenced> <mrow> <mi>a</mi> <mo>+</mo> <mi>b<i> </mrow> </mfenced> <mn>2</mn> </msup>
- Content markup:

```
<apply>
<power/>
<apply> <plus/><ci>a</ci> <ci>b</ci>
</apply>
<cn>2</cn>
</apply>
```

Math ML comes in two different flavors—Presentation Markup and Content Markup. Presentation markup tells your browser how to display an expression, but it doesn't tell you what the expression means. It tells it how to display fractions, roots, integrals, and so forth. So there's another kind of markup in Math ML, called Content Markup. If you look at the Math ML under Content Markup, this actually tells you what the expression means. It says that we're applying a power, which is going to be the number 2. The bottom, <cn>2</cn, means that we have a number. The n stands for number, which is 2. We're applying it to something which, itself, is an application of a plus to the sum of a and b.

The nice thing about the content markup is that you can actually have a program, which, after decoding the XML, can actually manipulate this. The program can either do calculations, if you supply values for a and b, or, if you had something that was going to be doing symbolic arithmetic, like taking derivatives with respect to a or b, you could actually ask for the derivative of this kind of expression and produce another piece of Math ML, which you could then display somewhere else. I have a couple of other examples here that represent an array (Table 2). I have the presentation markup for that.

TABLE 2
MATHML EXAMPLE: ARRAY

- Image:

$$A = \begin{pmatrix} x & y \\ z & w \end{pmatrix}$$

- Presentation Markup: <mrow> <mi>A</mi> <mo>=</mo> <mfenced open="[" close="]"> <mtable> <mtr> <mtd><mi>x</mi></mtd> <mtd><mi>y</mi></mtd> </mtr> <mtr> <mtd><mi>z</mi></mtd> <mtd><mi>w</mi></mtd> </mtr> </mtable> </mfenced></mrow>

The last two are pretty interesting; they show an integral (Table 3). The Presentation Markup isn't too interesting. The Content Markup actually tells you that you have an integral here. You're doing the integral from zero to *t* of 1 over *X*.

TABLE 3
MATH ML EXAMPLE: INTEGRAL

- Image

$$\int_0^t \frac{dx}{x}$$

- Presentation Markup: <mrow> <msubsup> <mo>∫</mo> <mn>0</mn> <mi>t</mi> </msubsup> <mfrac> <mrow> <mo>ⅆ</mo> <mi>x</mi> </mrow> <mi>x</mi> </mfrac> </mrow>

- Image:

$$\int_0^t \frac{dx}{x}$$

- Content Markup: <apply> <int/> <bvar><ci>x</ci></bvar> <lowlimit><cn>0</cn></lowlimit> <uplimit><ci>t</ci></uplimit> <apply> <divide/> <cn>1</cn> <ci>x</ci> </apply> </apply>

I actually created these images using an experimental browser available free from the Worldwide Web Consortium Web site. The name of the browser is Amaya. I think Amaya is an Indian goddess. It's a browser, which is intended to show capabilities that other browsers might eventually have, one of which is the ability to do Math ML. You can actually go into the browser and create graphic mathematical expressions. If you save the document, what you're left with is Math ML. Conversely, if you create a file that contains Math ML, and you bring it up in Amaya, you can get these expressions displayed. There aren't too many other systems that can create Math ML this way.

Tim Berner Lee said, in his book on the Web, that when he created the World Wide Web concept, he wanted to be able to edit HTML documents. There aren't a lot of systems that allow you to create HTML documents in a format other than straight text. Amaya is one of them.

Let me just briefly mention there are a few commercially available tools that do support Math ML. One of them is called Math Type. Math Type is actually the same engine that runs inside Microsoft Word for producing and manipulating mathematical equations.

Mr. Kelly: Yes it is.

Mr. Horowitz: Mathematica, I believe, is also going to be supporting Math ML. I just got an e-mail the other day from Math Type saying that they're going to be able to create Math ML as well as read Math ML, so they also consider it be fairly important.

Now I'll switch gears a bit and talk about a couple of applications for which I think the Society of Actuaries can start to create its own standard in XML.

We have something called a Table Data Standard, which is a way of organizing mortality tables. It's possible to go to our Web site and download a program that will let you view mortality tables as well as the tabular data. There has been an initiative to try to get various organizations to understand how the mortality table data are organized, so that organizations unrelated to the Society of Actuaries can actually read that.

It occurred to me at least a year ago when I first learned about this Table Data Standard that XML was the way to do it. This is not a criticism per se of what we did because what we still have in place is probably at least five years old, and it works pretty well.

However, if we want to be able to make mortality tables available to people and organizations, other than actuarial and insurance organizations, then it would be helpful if we can actually describe what a mortality table is in terms of its components. We would have a much better hope of organizations being able to use our mortality tables in a more robust way than the way that the data are currently organized. So there's going to be a small task force looking into coming up with some organization of mortality tables, possibly as XML. It's possible to think of an organization of other kinds of actuarial assumptions, besides just mortality tables. For example: morbidity, withdrawal, other kinds of economic assumptions, salary scales, continuance scales, annuity values, and conversion factors. All of these assumptions are ripe for a definition in XML. It's possible to create a single place where these tables exist and application programs are able to read them and know what kind of data they have and be able to process them.

From The Floor: Can someone explain getting information from an XML document?

Mr. Cole: If you happen to get input to your program in XML structure, you can ask the Document Object Model through a method to give you a list of all the tags in that document. They'll show you every single item that's in there. You just tell them you'd like the value of the first one, the value of the second one, and so on. You can write a generalized report writer that will expose all the data in any kind of

format you want, and you don't even have to know what's in the document. You just have to be able to ask the methods to explain it to you and then you can format it. So it's more a common way or a protocol for talking to each other, than content. Once you get that protocol down, the sky's the limit. You can tell your server applications to do something and you can interpret it in some way.

From The Floor: Do you consider data transmission rates the major barriers to having more universal actuarial modeling because we're talking about megabytes upon megabytes?

Mr. Cole: Interestingly enough, the XML structure is dynamic, so it sends what it needs to send in the context of the conversation. You might say that a picture costs as much as a 100,000 XML tags. Nobody cares that you're going to send down a 20K GIF file to be the logo on the left corner of the HTML page. We create a product definition that will allow you to do accumulated value calculations and to get all the tables that go with it. You get everything you need to do that and it fits on a floppy diskette. It's half a meg, and it's pretty compact when you think about it, compared to all the sloshing you have to go back and forth with conventional LU65.2 data format and things like that. So it's not necessarily so bad.

From The Floor: I've been hearing about UML, and a little about XML. Is there any information?

Mr. Kelly: UML is more of an object modeling standard that allows people to define business processes or application processes as kind of work flows. Rationalization is a classic tool that programmers use to model business processes and application development. UML is the Universal Modeling Language that allows multiple applications and multiple users of this technique to express models back and forth to one another.

It's very interesting because they are closely related in that Rational Rose which is a developer's tool, allows you to draw boxes, lines, and arrows to define a business process, or a work flow, or a model for an application flow. By hitting a button, it will express it all out as XML as a hierarchy, so XML can now be used to define business processes or application functionality.

There is a tool called Visio. We bought Visio about a year ago because the user interface (UI) for Visio for defining workflow and structures can be used to configure network structure and it can actually configure your network, so we're going to use it for that UI. Our XML server is called the Biztalk Server, and our XML server uses the Visio interface tool allowing our business professional to drag work flow around boxes and generate the XML document translations behind the scenes automatically. There is no programming involved. So you can actually take the XML document that the Essentium application creates, as well as the XML documents that the Applied Systems application creates, and some that you've done internally. That represents a financial instrument being created, calculated, underwritten, and some marketing information. All this XML document flow is inside your enterprise with the Biztalk Server interface over that. If you need to change your routing destination or you need to add a new document, you can, literally, just drag the

boxes around, and it will generate the source code on the server that will do that. So we're kind of shrinkwrapping the enterprise. Just look at what Microsoft did with compound documents on the desktop—you can drag and drop Excel into Word.

From The Floor: I have to ask if you have something to say about the technology. I'm going to send you some information. It might be in ASCII. Then I'm going to send you my program. That might be on an AS/400. So I tell you what I'm going to do. Let's say I have some salary data—say it is $21,000. Then can you take that information? Is that XML used so I can just send you the data and then your Java, or Delphi, or C++ calculates it and sends it back? What if it's in PowerPoint? Is XML just a definition? Are you telling me that this is exactly how I want this to be defined, if I tell you right now what I'm going to be sending you?

Mr. Horowitz: The answer is, not necessarily. There are other ways of doing that. It seems like the example is sort of simple enough. It's not clear that XML per se has any advantage over other ways of communicating that information.

From The Floor: Let's say I have a mortality table with a five-by-five array or something like that.

Mr. Horowitz: Now you're starting to talk about an instance in which XML is much more valuable. If your Internet Explorer 4 or 5 parser is able to pull apart the information you receive and it knows that these values correspond to, for instance, select and ultimate mortality rates, it knows why this table of values corresponds to a particular duration.

From The Floor: Right. Can you tell me that at this point or do you not know that?

Mr. Horowitz: I don't already know that.

From The Floor: So the programs we enter in will have to match something at my end ahead of time.

Mr. Horowitz: In a word the issue is standard. A standard is a community over some space and time agreeing to what something is.

From The Floor: That's what I would have at this point in time. Do you agree what the standard is right now?

Mr. Kelly: There are two ways to look at it. Again, like he said, that is not necessarily true, because there's a lot of flexibility in your question. In the scenario that you described, a piece of information was missing. It is not that you made a mistake, but it's missing. Did you have a client application that was preinstalled to do that, or did you just have a raw browser that hit my Web site?

From The Floor: What I'm trying to get at is XML—is it doing the calculation or is it sending the data to someone else in the process?

Mr. Kelly: XML is kind of steering the transaction at the system to which you're deploying the data. It's participating and creating the result that gets passed down to the browser. When you first say, "I'd like to apply for a policy, or I'd like to begin a transaction," XML is used to structure the page or assist in structuring the page. It's also used to begin to define the data that will now be created on the server. It is a data form, passed to your browser as a dumb device, right? You are filling in data. As it comes back from the browser, the XML tags it being used to envelope the data you typed in. Then, you pass to a business logic that happens on the back end.

This is an extremely flexible conversation that we're having right now in that the answer is kind of all of the above. It can be used in a lot of different ways, depending on architecture. One common thing that it solves is, in the past, even prior to the Internet, if we weren't all on the same machine or on the same client server network using installed applications that we had, in order to do a lot of heavy lifting, we couldn't get computing done. The concept of the Web browser as a thin client and being able to just run all over the world over different servers and actually do business processing will be enabled from a plumbing perspective by these kinds of standards. It could be the standard is a proprietary standard needed by the company just to get in and out of the database to a Web page or, better still, it could be a Society of Actuaries' defined technical standard, so that companies adhere to all of these mortality tables and all industry standard data. Therefore, like an agency company interface today, the carrier has to agree to map to the ACORD standards and do that coding. The agency vendor has to agree to map to the ACORD standards and that's not enough. It then has to test because everybody implements the standard differently.

In the future, if the client continues to tell you to adhere to ACORD 1.0 of XML and the server absolutely does, you've got a foregone conclusion that there's a legal process that communication can occur. Again, it's just one little example of the kind of thing it will streamline. It will eliminate much more of the heavy lifting to map these things. It is this introductory kind of concept for the lay person. It demonstrates that there are XML applications for mathematical calculations. You can tag and express certain formulas that can then be referenced in a transaction because we have these I.D. tags, that are then employed in the transaction; the result is exposed to the browser.

In the past, we had to do a lot of very linear coding to pull all that together to just perform one small calculation. There was way too much agreement that had to go on beforehand. We still have some standards agreement, but we're going to have a lot more flexibility when using a cell phone, a lap phone, a browser on a PC, or a hand-held device. Again, it depends on the business application. Those platforms can be ridiculous.

I don't want to do e-mail on my cell phone whether or not you can let me do it. I'm not interested in trying to fight like this. I might want to make a billing query or an electronic payment because I only have a few pieces of data that have to go back and forth, which approve the digital signature, the amount of approval, and those kinds of things.