From: Yuval Neeman (Exchange)
Sent: Fri, 12/25/1998 11:33 PM
To: Adam Bosworth; Bill Gates
Cc: David Vaskevitch; Anders Hejlsberg (Exchange); Edward Jung; Eric Rudder; Paul Maritz; Brad Lovering (Exchange); John Shewchuk (Exchange)
Subject: RE: Thinking about XSL

There has been important progress made in integrating data access and specificaly xml directly into VB langauge (& easily cool). I asked eric to find time soon for bardlo & johnshew to show this to you.

As for the how logic is expressed, the last thing developers need is a new language. if we provide a great way for our languages to interact with data, schema and transformations we play to our strenghts.

—Original Message—
From: Adam Bosworth
Sent: Thursday, December 24, 1998 12:18 PM
To: Bill Gates
Cc: David Vaskevitch; Yuval Neeman (Exchange); Anders Hejlsberg (Exchange); Edward Jung, Eric Rudder; Paul Maritz
Subject: RE: Thinking about XSL

I'm happy to schedule the time of course. My only constraints is that I'm out of the country chatting with partners 1/4 through 1/14. And even that could be slid a bit.

The question about logic is, of course, a swift stab to the heart of the main XSL weakness today. its model for procedural extensibility is poorly done aind needs to be fixed. Today, the logic is in the XSL itself (with some kludgy script extensibility). This model works well for a large set of cases because of the richness of the transforms that XSL understands, but it certainly needs to gracefully allow for procedural extensibility rather than itself sliding into yet another messy procedural language. This was first pointed out by Victor Stone about 9 months ago. It is worth noting that SQL is also declarative as are ReportWriting languages and StyleSheets and XSL can easily be made far richer a transform language that those collectively, but the point stands.

The "constraints" today are expressed as what are known as "XSL patterns". I'm including in this response a document I wrote proposing extensions to XSL that begins with a mini-tutorial on XSL.

It is important to note that Brad Lovering and John Shewchuk and I all agree that the programmer model for manipulating the XML should be far more integrated into our languages and I hope to have a specific proposal from the 3 of us by the end of February. However, having the XSL language itself be an XML grammar has significant advnatages. All tools and programming models that take advantage of the XML API (the DOM) automatically also can process and modify and display XSL. Secondly, XSL is already innovating in the direction of data navigation in a way I think our languages can follow and learn from. Again, I've tried to work fairly closely with Yuval's team in these efforts. But the short answer to the question about whether it needs to be another language is yes. it isn't a procedural language. It is a declarative one expressed as XML and this lets us easily analyze it, optimize its execution, much better optimize for handling of asynchronous processing of incoming data, and much more easily construct and deconstruct it. What I do believe, and this is what I hope Yuval's team and I can present to you at the beginning of March/End of February, is that we can do parallel innovation in our languages for transformation, link traversal, and so on. I

Date 7-11-02 Exhibit # 2
Case Sun Microsystems vs. Microsoft
Deponent Eric Rudder
Reporter Marta M. Rice
Naegeli Reporting Corporation
(800) 528-3335 FAX (503) 227-7123

Plaintiff's Exhibit
6521
Comes V. Microsoft

MSSunII 000000047020
CONFIDENTIAL

have an XSL transform on my desktop which is about 8 lines and does the following:
Enumerates each class offered by a school sorted by Class Name.
For each class, enumerates all the teachers who teach this class
For each class enumerates all the students who attend this class, and for each of them, enumerates all the classes that they attend.
And I can filter and sort any part of this graph. We should be able to do that on graphs of objects in our languages. We should be able to do that where the graphs of objects are "views" into other data stores living in different parts of the world, be they IMS through CICS or SQL Server or some MTS application server without having to know anything about the nature of these data providers. And that is the real promise of XML and XSL and the work we're doing on extending language to manage and traverse these graphs. This isn't object persistence. This is rich graph-based traversal and transformation into existing but foreign data. And I completely agree that our languages need to build in these capabilities, that XSL should merely be a useful declararative model, not the only way to do this.

I know this mail is long, but last point is important. Our customers are starting to do this. They are looking to us for help. I was in NYC two weeks ago working with Prudential Securities and Merrill Lynch on help they needed on how to use the XML to let their NT boxes interoperate with their mainframes. And here we face a terrible threat. These customers, as loyal to us as any, are moving towards Java because of the promise that their server code is portable and easy to develop and IBM is starting to steal a lead from us by delivering XML tools in Java while we are not and at least promising that java will run on MVS. Merrill Lynch is coming into town to talk to us about this in early January as well as to talk about what they need from an "XML database" (my next project), but my real and deep concern is that we show leadership here. I view IBM as the deep threat, not Sun, because IBM understands the mainframe, has invested massively in Java, and is rapidly catching up in XML. Their site on it is already better than ours.
<< File: IBM XML Web Site, Home Page.url >>

We have consistantly been the ones pioneering here and leading in the implementations and the innovations, but IBM may yet steal the thunder if we are not careful.
Thanks for listening. Here is the document on XSL and extensions (Eric Rudder hates the syntax!). << File: XSL Extensions.doc >>. Another useful document to read is one Andrew Layman and I wrote on how to map any graph (e.g. a relational database) to a canonical XML grammar. It is on my web-site at the top (see URL that follows).
<< File: Adam's Articles, Reports, and Presentations.url >> or http://xmlweb/adam/s.htm.

Adam Bosworth

-----Original Message-----
From: Bill Gates
Sent: Thursday, December 24, 1998 11:40 AM
To: Adam Bosworth
Cc: David Vaskevitch, Yuval Neeman (Exchange); Anders Hejlsberg (Exchange); Edward Jung; Eric Rudder; Paul Maritz
Subject: RE: Thinking about XSL

Its clear I have a lot more to learn here. I would like to schedule time to discuss this.

When mapping from one format to another involves "logic" I am unclear where that logic is created. Does the XSL call out to another language or do you do the logic in XSL itself? If you are checking a "constraint" of some kind when you do the mapping where is that expressed?

Is it necessary for XSL to be a new language separate from VBscript/VBA, Cool, SQL or anything else?

When you look at XSL code it looks like SNOBOL to me - string traversal. I have always thought we should innovate in our languages for data navigation and transformation.

I am also interested to know what people thing of the Latinum work that emerged from the ecommerce team. This was just presented to me last week.

To understand the "high level strategy" thinking behind this - I am trying to figure out if Microsoft could resume having the leadership message position that we had from 1990-96 with the PC and lost during 97-98 to Java cross platform with a new message.

The new message would be about being able to use our tools/platforms to easily interoperate with data of all types - without having to rewrite all of your applications in a new language. We would have to provide the best tools in Visual Studio or elsewhere. We would have to extend our languages to make them do this better. We would have to get third parties to agree with us. It is ok that SUN is already somewhat on the XML bandwagon if we distinguish this message. We would have to actually develop examples of customers using our stuff. We would have to get third party consultants excited about using datatranformation tools to help customers. I am not being as articulate on this as I need to be. We would have to show how this applies to verticals, systems management,... We would have to get ahead of IBM's San Fransisco. We would have to have some open innovation and some things around this that work best in our tools environment.

I was purusing SUNs web site and was impressed with some of the XML dialog I found there. In particular:
http://www.sun.com/981201/xml/;$sessionid$5BGBHHIAAG2JLAMUVFZE3NUBSSUXEUDQ

-----Original Message-----
From: Adam Bosworth
Sent: Wednesday, December 23, 1998 1:41 PM
To: Bill Gates; Eric Rudder; Paul Maritz
Cc: David Vaskevitch; Yuval Neeman (Exchange); Anders Hejlsberg (Exchange)
Subject: RE: Thinking about XSL

We should discuss this in person.

Some key points until we do.

1) We do manage XML as an object tree complete with data types for all well-known data types. The typing model is engineered to allow extensible classes, not just the types we agree on. It could be any set of COM classes. Thus, even today, XML actually is more of an in-memory database that "strings". We even index all the objects in a hash table on ID. It is merely one that loads (very fast, 2-4MB/second) from text. We have a binary format we've prototyped which will increase this speed, but not as much as you think. Our text tokenizing costs are only about 50% of our costs. What is more, the grammar that describes the "types" of these objects is indeed itself just another XML object graph just as you suggest.

2) This logic is largely separate from XSL. XSL is a processing engine for transforming from one XML graph into another. As such it subsumes a really startling amount of classic transforms from forms rendering engines through classic SQL queries through Report Writers through rich text processing. One can write another XML provider to our XSL engine even today, but for speed and efficiency DLL reasons, we wired them together in the IE 5.0 deliverable. But you shouldn't think of XSL as creating views on strings. It creates graphs from object graphs (and based on

what I plan to do next, on databases). It doesn't create updatable views today apart from the work the Netdocs guys have done. It should, but we are also driving to make it an engine that can produce literally 100's of pages per second on a server.

3) I use the term "graph" advisedly. I have some demos on my machine where Andrew Layman and I have taken the sample Access databases (fully normalized), built XML documents for these graphs (in other words the entire Access Databases as XML documents), and then I have written XSL queries against them which start where SQL leaves off (n-way joins) and then goes where SQL never did to construct rich output graphs, not just tables, in short the object graphs that our UI code and application code will want to consume. What is particularly impressive is that if I put some rich structured documents into the database as "fields" the query can seamlessly ask all the classic relational questions and yet also filter/transform the document with the same engine and syntax. Now, today, none of this is optimized. It scales acceptable to 1MB of data, but not even to 1GB let alone one 1TB. It is my goal to work with the database folks over the next year to fix that by never using text and storing the XML directly inside of our databases and merely using the XML model as the "schema" for the data.

4) The issue with using XML to build an "arbitrary" object graph (meaning that the types are dynamically extensible) isn't with the XML technology. It is with deployment. It is the well-known problems with COM deployment. If those problems are solved, using XML and XSL to build really powerful rich object graphs will be trivial. Indeed in the Java world today, there are several projects where each "element" type is mapped to a Java Class and they are building their graph out of Java Classes. In order for us to scale to this gracefully, I would strongly argue that the inherent COM+ runtime object should be one that can live in these graphs (in other words can support the DOM) and that certainly isn't the case today.

All the best
Adam Bosworth

-----Original Message-----
From: Bill Gates
Sent: Wednesday, December 23, 1998 11:29 AM
To: Eric Rudder; Paul Maritz
Cc: David Vaskevitch; Adam Bosworth; Yuval Neeman (Exchange); Anders Hejlsberg (Exchange)
Subject: Thinking about XSL

I think I can learn what I want to know if I can study about a dozen XSL programs.

I am worried about us basing everything on XSL for a few reasons.

I have an alternate proposal to the way we are doing XML XSL that I want to discuss.

I think it is ideal in many ways:

a) It allows us to bless the public standards truthfully but have something that is better - faster and simpler. We still have XSL as a public standard. However we use XSL and more declarative tools just to map the string tree into an object tree.
b) It allows us to combine our belief in data exchange everywhere (our parade to trump Java) with language innovation in VB/Cool and COM+

The basic idea is that instead of the XML just being a set of strings WHEN you

have schema (object definitions) around the tree has another form which is a tree of objects. You have have the same flexibility to transform the tree and map it into objects in the language.

When the XML tree is just strings you get the blessing of having flexibility and no schema needed. In lots of cases where "strangers" want to exchange this is good.

However it is BEST to have the logic that converts the tree from a string tree to an object tree SEPARATE from the program that takes the tree and works with it. The reason is threefold: a) the speed of the program running b) if the string format changes you just have one place where you have to change a transform - this is the most important and c) the program is a lot easier to understand.

If I can study some XSL I can give an example of how I seperate out the mapping from string to object and then let programs create views on the objects.

At least I think so.

All this stuff where we are tranforming trees makes me wish someone would help me figure out what part of IP (simonyi's stuff) might fit into this.