The Statesman, The General, His Lieutenant, and Her Sentry

Mary Fernández

June 21, 2004

I want to thank Ioana and Yannis for giving me the opportunity to speak to you today. I really feel like I have won the lottery!

I've pondered what to talk about today. So much has already been written about XQuery – by those who embrace it and those who hate it – that I struggled to find something new to say. At university, I took a creative writing class, in which the professor told us "to write our own stories, because those are the ones you know best". Following that advice, I want to share with you what I have learned during four very demanding and exciting years.

In one sentence, this is what I have learned: Creating technology is like making sausage – it is a very messy business. Science and mathematics have tidy and ordered laws that transcend people and time. But technology is a slave to its creators and to the time and place of its creation. Without Bill Joy, there would be no Berkeley Unix and no Linux, either. Before James Gosling and Guy Steele, Java was just a lovely island in the South Pacific. People are what make technology messy and also what make it so exciting.

XQuery is no exception to this rule. But unlike many technologies that hinge on one person's genius, XQuery is the inspiration of many people, all who have devoted numerous man-years to its design and specification. I've learned so much from each of these people, that I want to introduce you to some of them. Along the way, I hope to share some of the historical and sociological forces that have influenced the creation of XQuery.

The story of XQuery has a large cast of characters. If I introduced you to every one, we would all miss our planes home, so instead, I have chosen people based on the role that they played within the XQuery

working group. The characters include a general, a lieutenant, and a statesman, among others.

Every story has a setting, and XQuery is set in the Internet era. Before the warp-speed of this era, standardization was the process of taking an existing, widely-used technology and normalizing its definition so that it could be implemented consistently by multiple vendors. SQL, C++, and Unix are all examples of technologies whose standardization followed many years of wide-spread used.

For better or worse, Internet time has turned standardization upside-down. Within the W3C, as soon as the requirement for a new technology emerges, a coordinated effort begins to standardize it. SOAP, XQuery, and Web service infrastructure are examples of technologies whose standardization preceded their wide-spread use. But standardizing technology that does not yet exist is a potentially ruinous task: The technology's slate is blank, and the Internet-time clock is ticking very, very loud.

This is the setting in which Paul Cotton, XQuery's general, was tasked to create a group that could fill in the blank-slate that was XQuery.

1 The General

Paul Cotton's leadership is the foundation of the XQuery group. His organizational and procedural skills are phenomenal, and he has a keen ability to identify each person's unique skills and tap them to work on problems to which they are well suited. Most importantly, his 15 years of work on database standards gave him the authority and experience to lead a diverse group.

So, who showed up to write XQuery's story? All the major database vendors: IBM, Oracle, Microsoft and Software A/G – I will return to these 900 pound gorillas shortly. XML middle-ware start-ups such as CrossGain, Enosys, and Nimble. Research organizations such as AT&T, Lucent, MITRE, IPSI-GMD from Germany, and DSTC from Australia. Not to mention numerous power users of XML from the finance and health-care industries. Each organization arrived with a unique technical agenda, and each person with their unique skills and personalities. Imagine trying to herd forty cats, and you have some idea of what Paul faced.

With all this diversity, the potential risk of divisiveness and competition was very high, especially given the weighty presence of the major database vendors. Unlike other group members, the vendors had, cumulatively,

billion-dollar product lines to protect. The vendors recognized XML and XQuery as "disruptive technologies" that could potentially eat into their market share if they did not embrace and influence the new technologies' development.

Relational vendors had faced disruptive technologies in the past – remember object-oriented databases? OODBs were rendered almost impotent when the relational vendors integrated just enough OO features into their products to satisfy users. Personally, I was concerned that the same fate awaited XQuery, but I was wrong. To their credit, the vendors recognized that querying XML documents was fundamentally different than querying relational tables, and they came prepared to create a language that could satisfy the requirements collectively defined by the working group. Nonetheless, it was Paul's leadership that guaranteed every member's voice, whether a gorilla or a mouse, was heard.

In addition to the full-time job of managing our group, Paul is responsible for rapprochement with many other W3C groups. The W3C mandates that mutually dependent W3C technologies inter-operate. Paul established and maintained close contact with those responsible for the XML, Namespace, Infoset, and DOM specifications.

But it was XQuery's relationship with the XSLT group, that was unclear and potentially contentious. I'm sure that to the XSLT group, the XQuery group a potential usurper of the technology that they had successfully brought to market. But it was also clear that, unless the two groups cooperated, XQuery might subsume much of XPath 1.0, and the cumulative impact of all three languages could be weakened.

So Paul and Sharon Adler, the chair of the XSLT group, realized that the two must join forces to define XPath 2.0, and they needed a neutral person to lead the joint group. Which brings me to our lieutenant – me.

2 The Lieutenant

The two large groups are divided into smaller joint task forces, which are aligned with specific documents. A joint task force exists for the data model, XPath 2.0, the functions and operators, the formal semantics, and the full-text extensions. Each task force makes recommendations on design that are reviewed, debated, and usually adopted by the plenary groups.

As chair of the XPath 2.0 task force, my first task was to meld members of the XSLT and XQuery groups into one cohesive unit. This, as you might imagine, was not easy.

Functionally, the two groups were polar opposites.

The XSLT group was concerned primarily with document-oriented applications of XML. The XQuery group was not.

The XSLT members came from a cohesive group with a long history and a well-established group dynamic. The XQuery members did not.

The XSLT group had already defined and deployed a successful technology. The XQuery group had not.

My contribution was helping the groups to find a common ground, to define a shared vocabulary, and to educate each other about the other's technical requirements and priorities. Initially, when we were debating issues, people would be aligned with their original groups. But over time, the tribal lines became blurred, and people re-aligned themselves based on their own technical opinions and aesthetics.

The XPath task force tackled many difficult issues. The most fundamental was meeting the requirement that XPath 2.0 be aware of schema-validated documents, but there was long and contentious debate about what "schema-aware" meant. The task force was divided between XSLT members who felt XPath 2.0 should be a loosely typed scripting language, and XQuery members who felt it should be strongly typed. Like other scripting languages, XPath 1.0 and XSLT rarely raise errors, and never raise type errors, which makes them particularly robust for use within a browser. I will return to the details of XML Schema types shortly, and for the moment, focus on atomic types.

Many XSLT members felt that XPath 2.0 expressions, like those in XPath 1.0, should support implicit casting between atomic-typed values. In XPath 1.0, for example, arithmetic operators implicitly cast their string arguments to doubles. Implicit casting simplifies ease of use and is trivial in XPath 1.0, which supports only three atomic types: string, double, and boolean.

XML Schema, however, provides much richer type information. The XQuery members were adamantly opposed to implicit casting, because they believed it under-minded validation. Consider a document containing a string value validated as an instance of an inventory-code type, which is derived from string. Applying an arithmetic expression to an inventory code makes little sense and should result in a dynamic type error.

Our joint solution was to distinguish untyped values in well-formed documents from typed values in schema-validated documents, in both the static type semantics and in the dynamic evaluation semantics of the language. XPath 2.0 expressions implicitly cast untyped values to required types, which makes it easy to use XPath 2.0 with well-formed documents. With the exception of implicit promotion between numeric types, typed values are never implicitly cast, which guarantees that the type information obtained during validation is respected.

The result is that XPath 2.0 behaves both like a loosely typed and a strongly typed language. Critics claim that this characteristic makes XPath unnecessarily complex, but I believe it reflects the fundamental ability of XML to simultaneously represent strongly-typed, structured data and loosely-typed, unstructured text. This characteristic of XPath 2.0 is a feature, not a bug.

XPath 2.0, in my opinion, will be success. It took three years, 129 phone conferences, and thirteen face-to-face meetings, but in the end, XPath 2.0 is the *core* of XQuery. The two languages share a common syntax and semantics and are so closely aligned that both specifications are generated from the same source document. This alignment will increase acceptance of XQuery by existing XSLT users, because the learning curve is relatively low.

Making XPath and XQuery schema-aware was a requirement, but several XQuery members took that requirement one step further and advocated that XQuery be statically typed. The champion of the static type system is Phil Wadler.

3 The Champion

Phil Wadler is best known for his foundational work on Haskell and on GJava, but it is his breadth and depth of knowledge in logic and type systems that contributed so much to XQuery.

Early on, Phil encouraged Jérôme and myself to work with him on defining a static type system for XQuery. We agreed that defining a type system based on XML Schema would be foolish. XML Schema is horrendously complex and, more significantly, was never intended to be a type system. Schema validation and type checking are closely related, but substantially different processes. Validation assigns types to values, whereas type checking assigns types to expressions and requires comparing types.

So we based XQuery's first type system on regular tree grammars, for which well-known techniques exist for checking containment and subsumption. It is impossible to define a type system without using a formal notation, so we wrote the formal type-inference rules for the core language of XQuery, and these rules became the first formal-semantics document.

We worked hard – really, really hard – on the first static type system, in part, because we naively believed that the other group members would also appreciate the benefits of static typing – early detection of errors and inference of a query's result type, among others.

We were so wrong. Static typing was regarded by many as XQuery's wicked step-mother and the formal semantics, its ugly step-sister. The three most common complaints were: one, the type system is not XML Schema; two, it was not XML Schema, and three, it was not XML Schema. Related complaints were that the formal semantics was too hard to understand and impossible to implement.

But Phil was undeterred. He and Jérôme, with help from Peter Fankhauser, went back to the drawing board and worked tirelessly to formalize all of XML Schema – including named types, derivation by extension and restriction, substitution groups – into a rational type language for XQuery. This tour de force is probably one of the most technically significant contributions to the group's work.

Not surprisingly, formalizing XML Schema identified many semantic inconsistencies, which the XQuery group raised as issues with the Schema group. For several reasons, however, our interactions with the Schema group were not as productive as those with the XSLT group. Most significantly, by the time XQuery came along, the Schema group was already entrenched in public review of their specification, so there was little time for the two groups to establish the rapport that proved so productive for XQuery and XSLT.

Despite the obvious value of formalizing XML Schema and the static semantics, some members adamantly opposed to having the formal semantics be the normative definition of the language. More than once, the formal semantics was blamed for slowing down the specification process even though each time we formalized a language construct, we typically identified inconsistencies or incompleteness in the informal definition. In the end, the XQuery and XPath documents are normative for the dynamic semantics and the formal semantics document is normative for the static semantics.

The vision championed by Phil that XQuery have a static type system and a complete formal definition

has been largely realized, due in large part to the efforts of Jérôme Siméon, our explorer.

4 The Explorer

Jérôme is head of the formal semantics task force, the principal editor of the formal semantics specification, and the creator of Galax.

Although all the documents have multiple editors who contribute text and handle issues, the principal editor is responsible for the day-to-day editing of each document. Although the other documents are more widely read, the formal semantics has demanded substantially more design time and required more major revisions than all the other documents combined.

Denise Draper had the inspiration to transform the formal semantics from a terse, notation-dense document written in an academic style, into a tutorial on formal language definition targeted to implementors of XQuery. Denise made the first major revision, but Jérôme refined and restructured the document numerous times, until the formal semantics document was largely aligned, section by section, with the XQuery document. I believe that the final document is the first formal specification of a commercial language written concurrently with the language itself.

I haven't explained why Jérôme is the explorer. One reason is technical and the other personal.

Galax, our implementation of XQuery, began as a platform for debugging the formal semantics, and it has been invaluable during XQuery's design. The formal definitions of the parsing, normalization, and static-typing phases of the XQuery processing model are implemented literally in Galax. Every time the formal semantics changed, Galax changed, and we found bugs in both at each step.

With time, Jérôme envisioned making Galax into a well-designed, public-domain implementation of XQuery that would permit other researchers to evaluate their results within a complete XQuery implementation. With Galax, we are beginning to explore many dimensions of implementing XQuery, including evaluation strategies, document storage and indexing, interactions between optimizations and more. Jérôme has been committed to making Galax a platform for researchers, and I certainly hope that some of you will consider using Galax for your exploration as well.

The other reason that Jérôme is our explorer is personal. Working on XQuery has been exhausting for

everyone, but Jérôme and I have been unusually lucky to work as a team from the very beginning. I've been reading about Lewis and Clark, the American explorers who discovered the westward passage to the Pacific. They are glorified in American lore, but in reality, they basically dragged each other across the continent, and when they were really in trouble, their Native American guides dragged them both. When I've just about given up on XQuery and can't go another step, Jérôme lets me ride in the wagon and pulls the horse, which is a great characteristic to have in a colleague.

Both Jérôme and I found that being an editor is demanding, often tedious, and fairly lonely work. All the editors have worked very hard, but no one more so than Don Chamberlin, who is the principal editor of the XPath and XQuery specifications and is XQuery's statesman.

5 The Statesman

Don Chamberlin requires no introduction, but to briefly review, Don is the father of SQL, the author of the definitive reference on IBM DB2, and an IBM and ACM fellow.

As principal editor of the XPath and XQuery specifications, Don has left his mark on literally every language feature. For each feature added to XQuery, one member would write a proposal outlining the feature's syntax and semantics and include examples to illustrate its use. The group would then review and debate the proposal, sometimes raising issues, and finally vote on it. Once a proposal was adopted, Don would transform it into concrete text and integrate it into the specification.

Inevitably, once the proposal was in hard text, a flurry of emails would arrive saying, for example, "that is *not* what I meant" and "did we really vote for that?" Don always handled complaints with aplomb. He patiently incorporated good suggestions and diplomatically rejected bad ones.

Don has a high regard for his reader – for all of you, actually. He writes in a clear and plain-spoken voice that disguises his deep understanding of the content. He genuinely follows Albert Einstein's advice to "Make everything as simple as possible, but no simpler."

At this point in his career, Don does not need to be XQuery's principal editor, but his commitment to the working group and the technology speaks volumes about who he is as a computer scientist and as a person.

Because defining and standardizing XQuery occurred simultaneously, we constantly relied on the feedback from those companies that were tracking our work in their own implementations. Dana Florescu was the principal embassadress from the implementation community to XQuery.

6 The Embassadress

Dana leads the XQuery implementation team at BEA, is an editor of the XQuery specification, and, of course, has made important research contributions in the areas of query evaluation and data integration.

Embassadors are known to travel the world. Dana has traveled the technical world starting at INRIA to AT&T Research, and during her tenure with XQuery from CrossGain to Propel, then to her own company, XQRL, and most recently to BEA. Dana is not alone in these travels – many other group members have worked for multiple companies while working on XQuery, but Dana has traveled the farthest.

As the embassadress from these companies to XQuery, Dana has provided continuous feedback on the language features most needed by users and on those that are most difficult to implement. Dana's own XQuery implementation incorporated a streamed representation of validated XML and streaming evaluation operators long before everyone else was even talking about them. She frequently reminded us that a variety of implementation strategies were being pursued by vendors and that XQuery's design should be aware of these strategies.

Dana has also been our group's embassadress to the research community. Early in our work, she gave numerous talks and tutorials on XQuery to database researchers. The astounding amount of research activity surrounding XQuery is due in part to Dana's message that XML merited the database community's attention.

Dana's XQuery implementation gave us insight into entirely new techniques for evaluating XQuery, but we also benefited from the feedback of the major vendors. Michael Rys, our sentry, was most devoted to seeing that XQuery fit within existing database systems.

7 The Sentry

Michael Rys is a product manager for Microsoft SQL Server. Of all the major vendors, the SQL Server developers have provided the most consistent and detailed feedback on XQuery's design.

During the early language-design process, we were continuously expanding and re-shaping XQuery to make sure that it met the many requirements that we identified when we first convened. This phase was the most creative – we debated syntax ad nauseum, wrote the informal and formal definitions of every language construct, found bugs and inconsistencies, then iterated.

During this period, Michael's development team, better known in the group as Michael's "constituency", were implementing the specification. Michael never failed to remind us that each feature added to XQuery must be tested in isolation and in situ, must be documented, and would increase the learning curve for each new XQuery user. Michael stood at the door to XQuery and prevented many unnecessary features from flying in.

We are late in the design process, and now Michael stands at XQuery's door and gleefully tosses unnecessary features out on the street. He leads the effort to consolidate and simplify XQuery wherever possible. Most recently, he has worked on clarifying the semantics of sequence types and on separating element validation from element construction. Both these changes will substantially simplify the specifications.

One of my favorite features never even made it over XQuery's threshold, and Jim Melton, our historian, was the person to slam the door shut.

8 The Historian

Jim Melton is the Oracle representative to XQuery, an editor of the function and operators document, and most significantly, the principal editor of the SQL-3 standard. In support of SQL and XQuery, Jim has flown millions of miles all over the globe. If there were a uranium status for frequent fliers, Jim would have it.

In any case, in the middle of XQuery's design, we considered the features of user-defined exceptions and exception handling. XQuery already had exceptions raised by built-in functions and operators. User-defined

exceptions were an obvious extension and required by many application domains, so they were adopted quickly.

A good principle of language design is duality: for every constructor, there must be a deconstructor, so if there are exceptions, then there should be exception handlers. I was the advocate for this feature, so I did my homework, studied exception handling in ML, Module-3, and Java, and wrote a proposal for a try-with expression in XQuery. Since it was such an elegant proposal, I expected it to be adopted with little debate.

I was wrong. Numerous issues were raised against the proposal, the most significant by Jim. He recounted the history of exceptions in SQL-99, recalling that the SQL group struggled to understand the complex interactions between exception handling and transactions. He pointed out that this feature could delay us unnecessarily and might prevent future support for transactions. This tidbit of history tabled my proposal, and reminded me of the adage that "those who ignore history are doomed to repeat it."

9 The Majordomo

Jonathan is the XML architect at DataDirect technologies, co-designer of Quilt (a pre-cursor to XQuery), and an evangelist for XQuery within the large and often vociferous community of professional XML developers.

In the group, Jonathan serves many roles. Most importantly, he is the editors' editor. Currently, there are six principal editors and at least a dozen other editors that follow active discussions. Jonathan's job is to keep the editors moving our documents towards completion, resolving issues, maintaining alignment across documents, and making the next publication deadline. As the editors' editor, Jonathan embodies the institutional memory of XQuery – he can often recall with little effort debates that occurred two years ago and, when his memory fails, he is the fastest to find the relevant email thread in our archives.

In his spare time, Jonathan writes and criticizes proposals, attends virtually every telcon and face-to-face meeting, and gives enthusiastic tutorials on XQuery at IT conferences.

When I think of Jonathan, I am reminded of an old American television commercial for Eveready batteries. Battery-operated toy bunnies are playing their drums and one-by-one, they start to run out of juice. But the bunny with the Eveready batteries just keeps on going, beating his drum relentlessly. Jonathan is the Eveready bunny of XQuery. He simply never stops and, by his example, admonishes the rest of us to keep

going too.

10 The Cast of Thousands

Woody Allen once said, "Eighty percent of success is showing up". If true, then the XQuery group is successful beyond belief. Dozens of people, all whose roles were significant, showed up week after week at hundreds of teleconferences and meetings. Although I cannot mention each individually, I am thankful for them all.

The people with whom I worked closely include Peter Fankhauser, Denise Draper, and Kristofer Rose, Ashok Malhotra, Norm Walsh, Martin Nagy, Jonathan Marsh, and Scott Boag. Besides being wonderful colleagues, several have become good friends, which is an unexpected fringe benefit.

Before concluding, there is one more person that I would like to mention – Michael Kay.

11 The Guru

Michael Kay is the principal editor of the XSLT 2.0 specification, the author of the best reference on XSLT, the creator of Saxon, and XQuery's "guru".

I'm sure Michael would be utterly horrified by this characterization, and I admit to getting some perverse pleasure for thinking of it. But guru captures his contribution well.

Michael is a master at viewing technology holistically. When the group is wandering aimlessly through the trees, Michael knows which forest we are in and finds the path out. Michael looks at every technical problem from all perspectives: from the view of an experienced XSLT user, a new XQuery user, and an experienced implementor who knows where the potential performance bottlenecks and semantic complexities hide.

Early in the talk, I mentioned that XQuery must interoperate with related XML technologies. Of all, Namespaces take the prize for causing the most confusion and I bet are the source of the most bugs.

Namespace declarations are simply a scoping mechanism for binding prefixes to URIs – how hard can that be? But their correct use is arcane and we have struggled to use them consistently – when resolving QNames in queries and documents, when constructing new QName values dynamically, and when serializing

documents. Michael has patiently taught and re-taught the group about how namespaces are commonly used in XSLT, what users' expections are with respect to namespaces, and how we can specify XQuery to meet those expectations. Only recently, did his message sink in.

In every discussion, Michael implores us to see XQuery's forest through its trees. And this brings me to the end of our story and the beginning of yours.

12 A Challenge. A Wish.

By now, you can see that I have the highest regard for all these people. I hope that by meeting them, you have gained some insight into XQuery and better appreciate the challenges that we faced defining an entirely new language from the ground up.

The XQuery working group's chapter in this story is coming to a rapid close, and many of us are already moving on to new ventures. But as our chapter closes, yours is just beginning. It is your story – the creative and innovative research that you are all pursuing – that will determine how XQuery's story ends.

As researchers, our natural tendency is to deconstruct problems into their smallest parts, and focus on those parts for which we can find elegant and reproducable results. This is good computer science and, as researchers, we should continue to do good science.

But for the technology of XQuery to have an impact, we have to keep the forest, not just the trees, in sight. My challenge to you is to keep looking at the forest – in your universitites and companies, seek out real applications of XML and XQuery, discover the real barriers and opportunities for applying them, and assess how your research is enabling these new technologies in practice.

Seeing that we are in France, it seems fitting to leave you with a quote by the French novelist, Gustave Flaubert – "Success is a consequence and must not be a goal". My wish is that that a consequence of both our work and of your work is that XQuery is indeed a success. Merci.