Everyone's using XML, but does anyone care?

David **Megginson**

2005-11-17

Abstract

People are using XML in applications that are reshaping the technological world, like weblog syndication, recomposable web sites, enterprise messaging, and even cell phone information sharing. But what about XML itself? Does it matter? Is it just a bit of convenient plumbing, or a world-changing technology?

This presentation will look over XML's success stories so far, and will close the conference with a retrospective of some of the most compelling ideas that have come out of presentations, town halls, vendor displays, and hallway chit-chat over the past four days.

Table of Contents

1. The *Venn* of XML

I am going to start this evening by talking about Louis Rosenfeld. Rosenfeld is co-author of the very readable book *Information Architecture for the World Wide Web* (popularly known as the *polar bear book* due to the O'Reilly critter on its cover). I never know whether to applaud or jeer when I hear the phrase *Information Architecture* (IA). On the one hand, I truly do believe that most web sites are horribly designed; on the other hand, I have yet to work on a project that was actually made better by spending six months on metadata taxonomies — we simply end up with a badly-designed web sites with a nice taxonomy. Still, this book, and Rosenfeld's bloug [http://louisrosenfeld.com/home/], are required reading for anyone thinking of building an application online, if only as an excellent source of web information design patterns.

At the beginning of this decade, the dot.com crash smashed into the technology world like a meteor. Its fallout killed off technologies, ideas, companies, and even entire professions left and right. Information Architecture, as a profession, did not come out too well: its flagship consultancy, Argus Associates, went under, and IA people like Rosenfeld have regrouped around the banner of *User Experience* (UX), hoping to evolve fast enough to avoid extinction in this strange, new world — legend has it, though, that pure IA is still alive and well in certain deep crevasses where no light can penetrate, otherwise known as government contracts.

So let's take a moment to be smug. Our speciality, XML, *did* survive the meteor impact. If anything, the crash eventually made us stronger, even if our salaries of consulting fees had to go through some lean, hungry years first. There is orders of magnitude more XML now than there was in 1999 or 2000. I used to complain that it was hard to actually find XML on the Web, but that's no longer true, as hundreds of thousands (or millions) of RSS and Atom feeds demonstrate. All of the major office suites — proprietary and open source — have XML-based save formats. The collection of design practices used for new, super-interactive web sites like GMail is called AJAX, where the *X* stands for *XML*. O'Reilly is waving the banner for *Web 2.0*, a new paradigm where web sites share and recombine information in creative ways, mostly using public interfaces based on HTTP and XML. For better or for worse, *Service-Oriented Architecture* remains a big buzzword in the enterprise, and those services are almost invariably delivered using XML. News industry leaders like Reuters and Agence-France Presse are moving their wire services to XML, huge technology companies like Microsoft and IBM are building their product lines around XML, and governments are moving towards managing the very information of our lives, recording our births, marriages, and deaths using XML.

It is easy, then, to grant the first part of the title: everyone *is* using XML. Unfortunately, that leads to second part of the title: does anyone care?

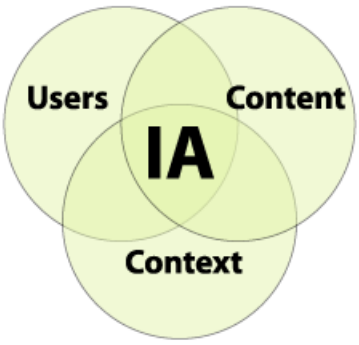

Figure 1. Rosenfeld's Venn diagram

One of Rosenfeld's best-known contributions to IA is a Venn diagram (Figure 1, “Rosenfeld's Venn diagram”) showing the areas of knowledge an information architect has to be able to deal with. The first circle, *Users*, represents the concerns of the people actually using technology (in Rosenfeld's case, a web site); the second, *Context*, represents the purpose

(or business case) of a site; the third, *Content*, represents the information that's actually available on the site. Rosenfeld admitted in his *bloug* that there really should be a fourth circle for *Technology*, but that four-circle Venn diagrams do not look attractive.

Information architecture is right at the centre of the three circles, taking in all concerns. But were is XML? It falls entirely inside *Content*, one subspecialty inside one of the areas of knowledge for an information architect. By this measure, almost nobody but obscure technicians need to care about XML (though it is nice for an information architect to know at least a bit about it). Do users need to care about XML? Do business managers need to care?

Soumitra Sengupta, presenting for Dave Campbell, reminded us of the three stages that a technology goes through:

1. emergent;
2. transitional; and
3. refined.

If XML is currently at the refined stage — in other words, if it has become a technological commodity, necessary but predictable — then, by this measure, the perception/buzz associated with it should be *obscurity*, and the mood of proponents should be *depression*. In other words, people will have stopped caring about XML because of its very success: if the low-level markup issues are solved, they can move onto higher-level problems.

2. The new ASCII?

A well-known XML specialist once predicted that when XML because as ubiquitous as ASCII, no one would care about it. Because this paper needs a straw man, I'm going to deliberately misunderstand that statement, however, and talk about plain text in general rather than a specific encoding of it. So, is XML as unremarkable to users and business managers as plain text?

XML itself is just syntax, but the syntax has an implicit mental data model — even if the *XML Infoset* did not exist, people looking at and working with XML would be thinking in terms of elements, attributes, content, and other (less important) objects. Aside from the occasional academic or other nutcase, however, people generally do not write XML for the sheer joy of it; instead, the XML *represents* something else. For example, the XML elements, attributes, and content might actually might represent *business objects* like customers and invoices, sections and paragraphs, RDF triples, or blog postings. Many IT specialists would like to be able to think about information exclusively on this level, without having to worry about the details of XML.

But even that's not the end of the discussion. Business objects are still IT details; as Sharon Adler pointed out earlier in this conference, Service Oriented Architecture is all about integration on the *business process* level. Managers would prefer to be able to deal with process directly, without having to worry about even abstract business objects, much less XML. And for the CEO and Board of Directors, business processes themselves are simply details on the way to world domination, but that discussion belongs in a separate presentation.

A cleanly-layered model like this looks good on paper, but can it really work? Can XML really be abstracted away completely, hidden under higher-level layers which are, in turn, hidden under even higher layers? Or is it possible that XML will keep forcing itself on our attention, swimming up through the layers for the occasional breath of air?

To discover whether there are places that XML really matters to non-specialists — places people are forced to care about it even when abstraction layers are piled on top — we can use two criteria:

1. Users and business people, not just technicians, have to be aware of XML; and
2. XML has to cut across higher-level specifications, so that the same XML knowledge is useful in more than one vertical application.

XML does not matter to a user simply because his spreadsheet happens to save in an XML format, it does not matter to a manager because the web services in her section happen to use XML, and it does not matter to a university student because her podcast happens to use XML for transport. None of these people has to start thinking in terms of elements, attributes, and content, because all of that is hidden. What areas might provide good starting points?

3. Search

During the conference's opening keynote on Tuesday morning, Jim Hendler identified what is probably the biggest opportunity for general XML to become relevant to users: search. Jim was talking specifically about RDF and other Semantic Web specifications; however, consider the range of popular, generalised XML data formats currently used on the web:

- RSS (in its many forms)
- Atom
- XHTML
- DocBook
- TEI
- RDF (and derivatives)

And this list does not even start on domain-specific XML formats for industries such as news and finance, or the thousands of local, custom-rolled XML formats.

Looking at companies like Google, Yahoo!, MSN, and Technorati, it would be hard to argue that search is not still one of the killer apps of the web. There is a lot of XML on the web right now, and there will soon be a lot more, but how should we search it?

One option is to search for the semantic items that XML documents represent. That's what Jim was proposing in his keynote — the ability to search for RDF triples — but that model has serious problems. Search engines that can search only for blog postings, RDF triples, DocBook sections, XBRL items, and so one quickly break the search market into a series of tiny niches, each with its own search silo. While some (like the blogosphere) can be large, most niche markets are small and expensive to serve.

As an example, consider the potential market for a DocBook search engine. In an attempt to locate DocBook documents online, I search Google for files that have the extension `xml` and contain the strings `itemizedlist` and `listitem`, two fairly characteristic DocBook element names. Google reported just over 500 matches. Even assuming that there are 10, 100, or 1,000 times as many DocBook documents available online, could there possibly be a robust market for DocBook-specific search engines? If any kind of sophisticated search is going to arrive, it will need to work across *all* XML document types to gain any kind of economy of scale, and to do that, it will have to require users to be aware of the XML way of thinking, using elements, attributes, and content.

Of course, we do not know if users will ever want this level of search sophistication, and with so many different linking mechanisms used in XML documents, building a web crawler is a significant challenge; nevertheless, as this paper will discuss shortly, people in general have shown a remarkable ability to find the benefits of new information technology, once they've had time to play with it long enough. If that happens, then there's little doubt that XML searching will meet the criteria for XML mattering to non-specialists.

4. Services

Services, on the other hand, must be a different case — mustn't they? After all, the whole idea of the WS-* Web Services stack is to hide XML under a huge layer of more abstract specifications, like dozens of mattresses covering a pea so that it can be found only by a true XML princess.

In fact, there is a variety of different approaches. The first widely-deployed service specification, XML-RPC, was designed to hide XML completely, so that applications could simply make remote procedure calls without worrying about the XML representation. A popular current practice, REST, involves simply hurling XML at a user or server using standard HTTP methods, and letting them deal with it without any safety mattresses. SOAP exists somewhere in the middle, providing an XML-RPC-like standard encoding that could allow XML to be hidden, but also allowing custom XML payloads.

By anecdotal evidence, this SOAP section five encoding is rarely used. It turns out that most businesses *do* want to pass around raw XML and deal with it as XML, rather than using XML as an invisible RPC payload. David Nielson of PayPal told this conference that plain XML (using REST) is much easier for PHP developers and others outside the firewall to work with — even the SOAP envelope wrapper causes problems, much less the whole WS-* stack. Parand Darugar of Yahoo told us that while Yahoo uses SOAP inside the firewall to take advantage of its management features, it uses REST and raw XML outside, because that is what developers can work with. So who are we protecting? It turns out that people can deal with basic XML, but that the technologies we propose to hide the XML actually make their lives more difficult.

Passing around raw XML payloads, however, brings the XML information model into the foreground. Now, a business manager has to start worrying about problems on the XML element/attribute/content level, rather than on the abstract business-object level. Should the company use an industry-standard vertical vocabulary or a company-optimised horizontal vocabulary? What are the interoperability issues (internally and externally)? What information will be lost with each approach, and how will that affect the business's productivity and revenue?

These are general information problems, but the XML data model provides the tools to think about them, so both managers and developers often do best when XML is staring them right in the face. In fact, developers are demanding not libraries to hide XML from them, but new programming-language support to make XML even *more* visible. Mukund Ragavachari told us about how IBM's XJ extension to Java is making XML into a first-class programming construction, like character strings, and Erik Meijer demonstrated the same approach at Microsoft with the LINQ initiative for C# and Visual Basic. The more successful XML gets, the more we seem to see of it.

5. Text

Example 1. Truly Plain Text

```
ITISATRUTHUNIVERSALLYACKNOWLEDGEDTHATASINGLEMANINPOSSESSIONOFAGO
ODFORTUNEMUSTBEINWANTOFAWIFEHOWEVERLITTLEKNOWNTHEFEELINGSORVIEWS
OFSUCHAMANMAYBEONHISFIRSTENTERINGANEIGHBOURHOODTHISTRUTHISSOWELL
FIXEDINTHEMINDSOFTHESURROUNDINGFAMILIESTHATHEISCONSIDEREDASTHERI
GHTFULPROPERTYOFSOMEONEOROTHEROFTHEIRDAUGHTERSMYDEARMRBENNETSAID
HISLADYTOHIMONEDAYHAVEYOUHEARDTHATNETHERFIELDPARKISLETATLASTMRBE
NNETREPLIEDTHATHEHADNOTBUTITISRETURNEDSHEFORMRSLONGHASJUSTBEENHE
READNSHETOLDMEALLABOUTITMRBENNETMADENOANSWERDONOTYOUWANTTOKNOWWH
OHASTAKENITCRIEDHISWIFEIMPATIENTLYYOUWANTTOTELLMEANDIHAVENOOBJEC
TIONTOHEARINGITTHISWASINVITATIONENOUGH
```

Example 2. Not So Plain Text

```
It is a truth universally acknowledged, that a single man in
possession of a good fortune, must be in want of a wife.

  However little known the feelings or views of such a man may be on
his first entering a neighbourhood, this truth is so well fixed in
the minds of the surrounding families that he is considered as the
rightful property of some one or other of their daughters.

  "My dear Mr. Bennet," said his lady to him one day, "have you heard
that Netherfield Park is let at last?"

  Mr. Bennet replied that he had not.

  "But it is," returned she; "for Mrs. Long has just been here, and
she told me all about it."

  Mr. Bennet made no answer.

  "Do not you want to know who has taken it?" cried his wife
impatiently.

  "You want to tell me, and I have no objection to hearing it."

  This was invitation enough.
```

It appears, then that people need to care more about XML than they do about plain text, but perhaps we have not been giving text its proper credit. An example of truly plain text — without whitespace, punctuation, or capitalisation — shows that the text we are used to is actually far from plain. Example 1, “Truly Plain Text” contains the text of opening of the novel *Pride and Prejudice* completely unadorned, while Example 2, “Not So Plain Text” contains the same text in the form we are accustomed to.

Two thousand years ago, European manuscripts — typically written on papyrus scrolls — looked a lot like the first example. They were essentially scripts meant for reading aloud, not for silent reading, and contained few visual guides to help the reader (who was assumed to be a native speaker who simply needed help to jog his or her memory). Over the middle ages, whitespace, punctuation, and eventually, a distinction between upper and lower case typography all entered as a kind of markup to make manuscripts easier to read, both aloud and (another innovation) silently. These are all kinds of markup — a capital letter might mean *start sentence* or *start proper noun*, while a period might mean *end sentence* or *end abbreviation* (as in “Mr.”). Horizontal whitespace marks the start of a paragraph (at the beginning of a line), or the end of one word and start of another. Vertical whitespace also marks paragraph boundaries. Quotation marks indicate the start and end of direct speech, commas indicate the boundaries of child syntactic units inside sentences, a question mark tags a sentence as interrogative, and so on.

This is not a simple markup system to learn. English-speaking countries devote many years, both before and during university, to teaching their children to use this system correctly, and English speakers are likely to judge a writer's social status and even intelligence based on his or her use of this system. XML training is trivial in comparison. Surely, people capable of learning this system are capable of understanding the basics of XML structure, if there's a strong enough incentive.

In addition to being hard to learn, this markup represents an horrendous waste of parchment bandwidth, when this bandwidth was very expensive (consider the difference in length between the two examples). However, various com-

pression schemes (such as heavy use of abbreviations) were eventually abandoned, and *binary text* never caught on. The extra clarity and usability more than made up for the verbosity.

There is no Plain Text 2006 conference (as far as I know), but plain text is not something that anyone takes for granted, and it is something that we do not always, or even often, hide under more abstract layers. Many people in the audience have used the Unix **grep** utility during the past few weeks, and even more have visited the Google web site to search through plain text. Decades after the introduction of graphical programming tools, most of us still write our code in text editors, and would be furious if we could not put spaces exactly where we wanted to. As Erik Meijer reminded the conference, text is still the universal data type, the payload passed to SQL servers and XML messaging systems, not to mention the foundation of nearly all Internet protocols.

6. Conclusions

During late classical times, a seemingly minor innovation in book production took hold: gradually, the scroll — a continuous roll of papyrus, which had to be read sequentially — was replaced by the codex (or book), a group of cut pages bound together, allowing random access. Cutting the pages was a simply idea — and a necessity when writing was done on vellum, which could not be produced in long rolls — but it had profound and unexpected results, many of which laid the groundwork for the modern information age. It is fairly simply to open a codex to a specific page, or canto of a poem, or verse of the bible; as a result, people gradually began to introduce cross references into book margins, creating the first hypertext. References could be collected into alphabetical lists, or indices, creating the first search engines. Reference works like dictionaries and encyclopedias followed, and people soon were able to find information with speed and ease that the Greeks and Romans would never have thought possible.

These innovations took time, however. Traditionally, writing was considered one-dimensional, simply a rendition of speech. These innovations belong to a world of silent reading, something that had no place in the old thought models, and it took many centuries for them to develop. Even something as seemingly simple as alphabetical order developed slowly over the Middle Ages, beginning with only the first letter of each word, then the first two, and so on. People coming from a world of one-dimensional information did not learn to think about information in two dimensions quickly or easily, but once they made the transition, they were able to lay the foundations not only of the web but of science and the modern world.

XML's model is trying to add yet another dimension to information, and the real implications of this change are just beginning to sink in, even for the people who work most closely with markup. Like the shift to two-dimensional, random-access information during the European middle ages, this change will not be fast or easy, but eventually, people will find benefits to the model that we have not yet begun to consider.

Markup itself is not hard for people to grasp. Consider Wikipedia, which currently contains over 1.5 million articles, all written by volunteers using markup that is considerably more difficult (though less verbose) than XML. It is unfortunate, then, that XML specialists like me spend so much time disparaging XML with our clients. Like many of you, I typically take one of the following two approaches:

1. I write a library to hide the XML and tell my customers not to worry their pretty little heads; or

2. I fail to hide the XML and treat it as a personal failure, telling my customers to prepare for pain, or, as Victorian mothers supposed told their just-married daughters, to close their eyes and think of England.

Perhaps it is time to begin caring about XML rather than just using it.

Biography

David **Megginson**
Consultant
Megginson Technologies Ltd. [http://www.megginson.com/]
Ottawa
Ontario
Canada

David Megginson, principal of Megginson Technologies, has been active within the SGML and, later, XML communities since 1991. He led the original initiative that created SAX, the Simple API for XML, which is now the most widely used streaming API for XML.

David's work includes many Open Source software packages, together with the books Structuring XML Documents and Imperfect XML, published by Prentice-Hall.

David formerly chaired the XML Information Set Working Group at the World Wide Web Consortium (W3C) and served as a member of the W3C's XML Working Group and XML Co-ordination Group.

In Spring 2000, David was proud to receive the Java Technology Achievement Award For Outstanding Individual Contribution to the Java Community from Sun Microsystems and JavaPro magazine.