Toward the semantic web
a view of XML from outer space

Author: Stefano Mazzocchi (stefano@apache.org)

Abstract

In this paper we describe an hypotetical journey from outer space to the notion of a semantic web, describing the intents, the problems, the possible outcomes and the impact on web users.

1. Introduction

The term *semantic web* was coined by W3C Director Tim Berners-Lee to indicate a possible evolution of the current world wide web. It's a possibility, a goal; something that cannot be achieved by currently available technologies.

2. The definition of semantic web

In the English dictionary, the adjective *semantic* is defined as:

```
semantic adj.: of or relating to the study of meaning and changes of meaning
```

A semantic web is a web where the focus is placed on the meaning of words, rather then on the words themselves: information becomes knowledge after semantic analysis is performed. For this reason, a semantic web is a network of *knowledge* compared with what we have today that can be defined as a network of *information*.

3. Information Networks

Information networks are created when a number of information sources are seamlessly interconnected. This network is *distributed* when the information is stored in different geographical locations and is *decentralized* when there is no control of one node to another.

The original idea of the web was to increase scalability of information networks by allowing *information owners* (the authors or maintainer of the information) to serve it directly and become a network node with very little effort.

The lack of scalability of centralized information systems is due to the fact that management costs grow more than linearly with the amount of information stored and with the amount of people involved. This creates a *saturation point* where adding more resources to the system (time, people, money) increases costs rather than decreasing them.

The revolutionary yet extremely simple idea of what became known as the world wide web was placing management costs directly on information owners on local and very small scale.

This was based on the evidence that the management costs of single individuals (in the early days, CERN scientists) over their information grows linearly with it as long as they remain the only one to manage that information.

The idea is equivalent of the good old tune *pick up your own mess*: the incredible growth the WWW had clearly indicates it doesn't only work at home :)

4. When information becomes knowledge

Suppose you give me your favorite book: how much *information* have you passed me? Pretty easy: it's just a matter of counting the characters (to be honest, it's a hell of a lot more complex than this, but we'll ignore this issue here)

How much *knowledge* did you pass me? There is no answer.

It depends mostly on the context, which is defined as the amount of knowledge that I already have.

For example, if it's an German book, the amount of knowledge it bring me is very little since I don't know German. Or might be written in a language that I know very well, but being very badly written I don't get much out of it, or talking about situations that don't look familiar to me, etc.

Information must be semantically analyzed before it gets translated into knowledge and semantic analysis is impossible if you don't have the instruments to perform it, or simply because your *context*, your previous knowledge is not able to create connections to the new information. If it cannot *digest* it, it remains unconnected and doesn't add to your overall knowledge. It's useless information (sometimes referred to as *data pollution*)

For example, you might be totally lost from the above sentences and you might not understand my point: this implies that the information this paper conveys is not enough for you to understand the point.

But the example right above might have triggered your understanding.

Anyway, it's hard stuff because it's the deepest possible recursion of thinking: try to understand how we understand.

Don't worry if you feel lost, just keep going.

5. The web is already semantic (sort of)

When you browse a page, your browser receives a number of bits from the network and interprets them into web pages.

And what do you do with it? you read it, you try to understand it and act in consequence (insert information, click, close the browser, go back to sleep, cry, laugh, etc...).

The web is already semantic when your context is close enough for you to be able to perform a semantic analysis on it, but, if not, the information the browser gives you doesn't mean anything to you and doesn't trigger any action. It's therefore completely useless.

For example, browsing English pages if you don't understand English or browsing Chinese pages if you don't know ideograms are both examples of different degrees of knowledge conveyed by web information depending on the user semantic context.

For English, you might be able to *get* more out of it because at least the alphabet is part of your knowledge (if you live in a western country), while for Chinese this is much harder (and you probably base your assumptions on pictures and totally ignore text).

But semantic analysis on web content is not only performed by humans: computer programs crawl the web retrieving information for other programs that perform semantic analysis to index it. This processed information is then queried directly by humans or by

other computer programs at need.

It must be noted that for small amount of information, the process of semantic analysis can be performed by humans but on very large scale, the task must be performed by computer programs to keep costs reasonable and match the network rate of growth or frequency of update.

Search engines perform semantic analysis on web pages, and the algorithms that perform this semantic analysis are very complex and their heuristic is based on HTML interpretation, metainformation, link topology and natural language analysis.

Part of the problem is that no AI program is able to match the intrinsic ability of humans to perform semantic analysis, extracting meaning from information and abstracting concepts from examples. But the real problem is that most of the content found on the web is made of HTML pages and HTML was designed to contain instructions for browsers on how to display the pages to users. It contains very little information about the real semantics of the page, which where either never there (if pages were written in HTML directly), or lost when transposing from other formats.

6. The intrinsically poor semantic capabilities of HTML

Let's make an example to show how poor HTML is for expressing non-visual semantics. Let us consider the tag <b>. It stands for *bold*, which is a specific font style which is *fatter* and *darker* than the normal font style. This font style is normally used to make words *stand up* in the visual context of a particular sentence. So, when HTML writers want part of the sentence so *stand up*, they use the tag <b> because they know how this will be rendered by browsers.

Now, suppose you want to *read* this page for users with non-visual capabilities (either visual impaired or requesting from audio-only devices), what does <b> means for voice?

The mistake is subtle but evident: the most used aspect of an abstract concept was used to indicate the abstract concept itself. The tag should have been used since it represents a concept that is totally independent from the usage context and will be interpreted as *bold* for text, as *higher volume* for voice, etc.

HTML contains a number of tags that can be used to represent a visually-abstract content, but these where designed to markup pages, so they contain semantics about paragraphs, tables, images, but they cannot markup things like invoices, mathematics formulas, car spare parts, etc.

Don't get me wrong: HTML is great for describing web pages and a wise use of its markup, coupled with the use of cascading style sheets to separate the style attributes from the page structure, does a perfect job.

The problem is that HTML treats every information as pages.

Pages are great for browsing but not good for data mining; when you query a page you don't care about it's style or page structure, but only about the information it contains.

7. The XML model

Adding semantics to HTML would have fragmented its use and increased incompatibilities between client implementations, thus causing harm rather than good to the web.

So the W3C started in 1997 the creation of new technological infrastructure for web content that would have been *extensible* enough to convey more specific semantic

information, match HTML functionality for today's web needs as well as improve on them, yet remaining as simple as possible.

After three years and tens of specifications worth thousands of pages, the XML model is not yet finished, but close enough to the point where attention can be brought to the technological problems that this model poses on the web as a whole.

8. Anatomy of the XML model

The XML model can be decomposed in separate areas of concern (in parenthesis the specifications that define them)

- syntax (XML)
- validity constrains (XML, XMLSchema)
- modularity (Namespaces)
- hyperlink capabilities (XLink, XPointer, XInclude, XBase)
- metadata (RDF, RDFSchema)

plus a number of *lateral* specifications that we won't cover here.

8.1 Syntax

The XML syntax defines how information must be encoded to be considered *well-formed* by parsers. The XML syntax is text based and it's based on the same syntax used by HTML, but it improves it by specifically indicating empty elements to simplify parsing complexity and allowing incremental operation.

8.2 Validity constraints

XML documents are said *well-formed* if they follow the XML syntax, but they can be further "validated" on some validity constraints that can indicate element structure, value types, default content, allowed namespaces, etc. XML documents are said *valid* if they are well-formed and followed their schemas without errors.

This is the first improvement on HTML since validation allows both page writers and tool implementers to rely on a strong contract defined by the document schema. This allows errors in implementation or schema usage to be detected before they enter the system, thus limiting the damage they can do and improving overall security of the system.

8.3 Modularity

Modularity in the XML context is the ability to merge content belonging to different semantic contexts into the same document. For example, mixing page layout with mathematics formulae and vector graphics for scientific papers.

This capacity is achieved by mapping each schema to a specific and unique URI,thus avoiding name collisions and preserving semantic independence.

This is another innovation from the HTML model which didn't have such capabilities.

8.4 Hyperlink capabilities

While HTML defined hyperlink semantics using a particular tag, the XML model uses the notion of namespaced attributes to add particular semantics to existing elements.

This is a big improvement since it allows to add hyperlink capabilities to any document schema without requiring the creation of specific linking elements.

This is a much cleaner and powerful approach than HTML.

8.5 Metadata

Metadata is data about data.

HTML introduced the *meta* tag to allow page authors to help indexing of the page by placing keywords that specified the page content. The best search engine make extensive use of these tags, when available, to categorize the information

The XML model extends this ability by defining a framework for resource definition (RDF) which can be considered the *syntax* for metadata, along with the metadata schemas (RDFSchema) that define the metadata structure and its relation with other schemas.

We'll discuss metadata in greater detail in following sections of this paper.

9. A machine-friendly architecture

The ultimate goal of a semantic web is to allow computer programs to help humans by doing semantic analysis for them on very large information sets and improve the user experience on data mining.

This is the single most important difference between the HTML model where the user experience was greatly improved on browsing, but totally ignored on searching (in fact, the first search engines were *hacks* rather than engineered methodologies in the original web design).

The goal is not to substitute humans with computer programs that automatically search the web for you, simulating your semantic analysis capabilities very poorly, but to create a technology infrastructure that allows web content owners to semantically markup their information and create algorithmically certain ways for computer programs to perform specific semantic queries on very large datasets.

Let's have an example. Take a look at this HTML page:

```
<html>
  <body>
   <h1>My trip to the Java Island</h1>
   <h3>by Stefano Mazzocchi (stefano@apache.org)</h3>
   <p>The trip on the island was great:
    <ul>
     <li>we tasted great Java coffee</li>
     <li>we finished my latest program written in Java</li>
     <li>we did a great trip on the islands around Java</li>
    </ul>
   </p>
   <p>The trip was arranged by Travels@Java.com Inc. (info@javatravels.com).</p>
   <p>Saluto tutti gli amici italiani che erano con me.</p>
  </body>
 </html>
```

The semantic analysis of this page is very hard for humans and close to impossible for computer programs (at least nowadays). For example, the second sentence could mean that I finished my latest program that I wrote when visiting the island of Java, or, more

reasonably, that I finished my latest programming written using the Java programming language.

But the page in general requires your knowledge to contain notions about Java as an island, as a programming language and as a coffee brand.

We could continue our semantic analysis by saying that the page is written using the English language, written by Stefano Mazzocchi who can be reached at the email address stefano@apache.org and the page is mainly about a trip to the island of Java.

But computer programs that perform on syntax analysis rather than semantic analysis might not be able to understand that Travels@Java is not the author of the page, nor it's a valid email address.

But I'm the author of the page and I know the meaning of what I write and I would like to transmit it as such to humans as well as computer programs and not letting them *guess* about what it contains. (of course, literature and art in general do not have such mechanical vision of semantic interpretation, but we ignore this for now)

So I write:

```
<?xml version="1.0"?>
 <page xmlns="http://mysite.org/page"
       xmlns:geo="http://geography.gov/terms"
       xmlns:food="http://fao.org/metadata/food/en"
       xmlns:man="http://onu.gov/metadata/mandkind/en"
       xmlns:man="http://acm.gov/metadata/computing"
       xmlns:email:http://ietf.org/schemas/email"
       xmlns:com:http://nafta.org/terminology/en"
       xml:lang="en">
  <title>My trip on the <geo:island>Java Island</geo:island></title>
  <author>
   <man:man man:age="25" email:address="stefano@apache.org">
    Stefano Mazzocchi
   </man:man>
  </author>
  <content>
   <p>My trip on the island was great:
    <ul>
     <li>we tasted great <food:coffee>Java</food:coffee> coffee</li>
     <li>we finished my latest program written in <comp:lang>Java</comp:lang></li>
     <li>we did a great trip on the islands around <geo:island>Java</geo:island></li>
    </ul>
   </p>
   <p>The trip was arranged by
    <com:company com:type="incorporated" email:address"info@javatravels.com">
     Travels@Java.com Inc.
    </com:company>
   </p>
   <p xml:lang="it">Saluto tutti gli amici italiani che erano con me.</p>
  </content>
 </page>
```

This page uses XML and Namespaces to add specific semantic information about the included content while removing all style information. While it's harder for humans to understand this page as it is, it's much easier for computer programs since all the necessary semantic information is placed by the page author and namespaces identify the

semantic connections univoquely.

No heuristic or guess takes place when a computer program parses this page because it doesn't need to understand the page to be able to associate content to its semantic areas, which are here identified by namespaces.

So, the word *Java* is connected to three different meanings and will be up to the user requesting the page to determine which meaning she/he is interested it, the page parser does not have to understand any of this but simply index the page with this information and use it a query time to rate the page.

You must understand that the above is nothing new since all centralized information systems have been able to perform context sensitive searches for decades and, in fact, this is a particular flavor of queries that relational databases have been performing for decades.

Instead of specifying rows and tables, you search for some text included in specific contexts. For example, search for *Java* inside the *island* tag of the *http://geography.gov/terms* namespace and *trip* in English *en* language.

But compared to what we can do today, this would be indeed huge step forward.

10. The Babel problem

Searching in database is fairly easy once you have the database schema, but suppose to have a database schema with millions of tables with millions or columns each: how do you know what query to perform?

This is the *Babel problem* that the semantic web faces: if everybody publishes its own schema for their data, how am I going to search for it?

Let's make another example:

```
<ordine xmlns="http://miosito.it/schema/ordine" xml:lang="en">
  <data giorno="10" mese="09" anno="2000/>
  <per>Stefano Mazzocchi</per>
  <articolo codice="484974984">3</articolo>
  <articolo codice="38479847">38</articolo>
  <nota>This is a very important order</nota>
 </ordine>
```

This is perfectly legal: the language for content is English, but the language used for creating the schema is Italian since it was created by Italians. Suppose your favorite semantic search engine comes here and indexes this document, how are you going to search for it?

You might search for pages where *order/for* contains *Stefano Mazzocchi* but the search engine doesn't know (nor will have the ability to guess) that *ordine* is the Italian translation of *order* and *per* means *for*.

How can we solve this problem?

11. Semantic networks as metadata schemas

Humans learn by extending their semantic networks.

Semantic networks are networks of terms that our mind connect based on our life

experience. For example, *father* is a *man* but not a *stone*, etc.

In order for a computer program to understand that *ordine* means *order*, it must have information that connects the two words together, sort of mechanical semantic network that the program is able to use with simple inference rules to come up with the right meaning.

So, if the search engine is aware of the fact that *ordine* means *order*, the metadata is abstracted from the language used since the element *ordine* inherits all meanings of the element *order*, which we suppose known by the querying user.

The RDFSchema specification defines a way to indicate such mechanical semantic connections between metadata schemas and allow the search engines to be more flexible during searching.

For example, suppose you have an RDFSchema that connects an hypothetical *Mankind Metadata Set* with another hypothetical *Zoology Glossary*, indicating that *men* are *mammals* as well as *omnivores*.

Then a search for *mammals that swim* will return you a page where a *father teaches* his *kids* how to *swim*, since *father* is part of *men* and *men* are *mammals*.

This is a stupid example, but it gives an idea on how powerful such system might be even with very simple inference rules.

12. XML is only the first step

When I talk to people about this next web, they stare at me excited and ask me to estimate how long it will take for it to happen. My usual reply is: *might not happen at all!*

The XML model is clearly a much better model than the one created by HTML and improves on all the previous designs for information systems. It can be considered state of the art, the sum of what happened before, but technology alone cannot make it without the people supporting it and without a wise and progressive evolutionary path.

XML faces many problems, the first of which is its own complexity and somewhat design incoherence between specs. While I'm sure that incoherence between specs will be removed over the years by progressive refinement, its complexity cannot be decreased without breaking the whole model.

Compared to the relative simplicity of the HTML model, this will make it harder for people to start using the technology.

Another one is the *Babel problem*: even if many people estimate that de-facto standards for both document schemas and metadata schemas will soon emerge, especially in the areas of e-commerce or economical data interchange, more reasonable analysis forecast that many different schemas will emerge and schema transformation will provide the glue between them.

It's too early to estimate the outcome of such *schema wars* and even if it's very likely that they will be settled by standard bodies later on, it's not easy to predict how long this would take and what the results would be.

13. The chicken/egg problem

But the biggest problem the semantic web faces today is the lack of XML-capable and compliant browsers. The leading browser Microsoft IE performs very poorly in terms of standard compliance (even if it's very latest version) and it's main rival Mozilla is not yet

ready for prime time. This forces those pioneer sites using XML for their web publishing infrastructures to hide it behind transformations that send adapted HTML content.

It's the chicken/egg problem: semantically capable search engines don't emerge until there is enough semantic content to index and people don't write semantic content until there will be enough semantic search engines to make the effort economically worth.

Moreover, there is no way for search engines to identify themselves to web sites, so there is no way for web sites to feed back XML content instead HTML created from their hidden XML content.

Many sites are starting to use XML technologies on the server side because this helps them to reduce their management costs, but it's only a selfish reason: if everybody hides their XML content to clients and pass only graphical semantics to both user agents and search engine crawlers, the web will still be globally non-semantic, even if every site makes extensive use of the XML model internally.

14. Adding dimensions to the web: resource views

A possible solution to the problem was proposed by myself on the Apache Cocoon project and it's currently implemented in Cocoon2.

The solution is based on the evidence that all web resources are inherently monodimensional in the HTML model: when you ask for a particular URL, you are requesting the URL for browsing. Period, nothing else.

In publishing frameworks, even static URLs are not mapped directly to files stored on the file system but a different output is created depending on the user agent that requests the resource. For example, WAP agents are fed with WML content, HTML browsers with HTML, Voice clients with VoxML and so on.

But even if the outcome is different depending on request parameters, all clients are able to access one and only one *view* of the resource: the view that allows them to browse it.

Our solution is to extend the URI framework to allow the requestor to specify which view it's interested in. Browsers will mostly ask for browsable views (that will return content with associated visual and linking semantics), but there are many possible views of a resource:

- **default view** - returns the main view of the resource (and mostly depends on the user agent requesting the page)
- **link view** - returns the list of links included into the resource. This is very useful for crawlers that do not have to perform analysis on the very different MIME-types that could be returned by the URI. (for HTML is easy, but think about finding links inside PDF documents)
- **semantic view** - returns the semantic view of the resource.
- **schema view** - returns the schema used to validate the resource.
- **metadata-schema view** - returns information on how to map the given metadata schema to the one contained into the page.

while other views are specifically used in Cocoon2:

- **pipeline view** - returns information on what actions performed to create the particular resource along with logs, timings and debug information (security sensible: should not be accessible by the general public)
- **link-translating view** - returns the default view but all links have been translated according to the rules given in POST by the client. This is mostly used by Cocoon

itself when saves snapshots of web sites for offline browsing (security sensible: will not be accessible by the general public)

At the moment of writing (but this might change in the near future) a view is requested by sending a special query parameter attached to the URI, but the ultimate goal is to propose an extension to either the HTTP protocol or the URI definition to the IETF to formalize its notion and push its adoption more globally.

15. Why views matter

User agents are very likely to be very differentiated in the future and both costs and computational capabilities will push to simplify them to the bare minimum.

This clearly goes against of current technological guidelines in the XML model where heavy semantic information must be passed *as is* to the client and let them come out with a graphical view helped with stylesheets sent by the server.

Instead, it seems very unlikely that XSLT will have a big win on the client market and will mostly remain as a server side feature to *adapt* the same content to different devices and usage contexts.

If this happens, the semantic web is born dead since all the semantic information will not have a way to be presented in a uniform way across web sites and will remain hidden inside web sites which no general way to access it.

Another solution might be to replicate the entire web site in its semantic form using virtual hosts (something like semantic.www.yourside.com compared to www.yourside.com) but the chicken/egg problem comes back: people will never duplicate their efforts if no benefit is obtained.

Our solution, instead, minimizes effort duplication by making the publishing engine take care of view generation automatically (in our specific case Cocoon, but this could be applied to any other web publishing system) with virtually no added costs beside that of using XML technology.

16. Possible future scenarios

There are several possible future scenarios that might happen, I'll try to outline a few possible ones

16.1 XML on the client side

If all clients support XML, the semantic web is just a matter or standardizing document and metadata schemas and associate them with proper namespaces. The problem is, as already described, that will not be easy for all web clients to support XML. This is very likely to create an inertial drag that will slow the adoption of new web technologies.

16.2 XML on the server side

XML will very likely be adopted on server side soon because it already allows cost reductions in both creation and maintenance of site information. Such web publishing engines are able to support both XML-capable and XML-agnostic user agents and the inertia of the system is only local and doesn't depend on client technology to happen.

The problem is due to the current incapacity of information indexers to obtain the original

semantic information of a resource in a standardized way. Such a web will allow better searches locally on sites, but won't improve data mining on global scale.

16.3 Multidimensional web resources

This improves on the previous scenario by introducing standardized way for sites to share semantic-rich information. This will remove the chicken/egg problem if local indexing capability is added on the semantic information. This will push for adoption of more semantic-rich technologies and it will allow its incremenal adoption.

16.4 ???

Future innovation in the XML model or in the web infrastructure might change the picture again in the future. It's still very early to tell.

17. Conclusions

We came from outer space and we approached the *semantic web* planet, looking at it from low orbit, but even if the technologies being proposed and developed are very promising, we showed an overview of the problems that *landing on the planet* might pose.

Today's web barely scratched the surface of what a globally distributed and decentralized information system can do in terms of social and economic impact, but there is much still to do and the inertia of such a system will be incredibly hard to influence.

Technologies alone won't make it.

In this paper I outlined both the problems that such a revolutionary transition faces and possible solutions to turn it into an *evolutionary* transition. These solutions are being currently pioneered by the Apache Cocoon project which hopes to lead the creation of a long term plan for the adoption of XML technologies on the server side with the goal of creating a semantic web.

And everybody is invited to share the fun of doing it and shape what the future of the web will be :)

Appendices

A.1 The making of this document

This document was created by out of a valid XML file and applying an XSLT stylesheet that transformed it into XSL:FO. Then these formatting objects were formatted into a PDF file.

A.2 Software used to create this paper

- [Xerces] "The Apache Xerces-J XML Parser", http://xml.apache.org/xerces-j/
- [Xalan] "The Apache Xalan XSLT Processor", http://xml.apache.org/xalan/
- [FOP] "The Apache FOP FO+SVG Formatter", http://xml.apache.org/fop/

A.3 Software cited in this paper

- [Cocoon] "The Apache Cocoon Publishing Framework", http://xml.apache.org/cocoon/

A.4 Bibliography

All the specifications cited in this paper can be found at the URL:

```
http://www.w3.org/TR/
```