Lost in the Semantics

Lucian **Holland**

Abstract

Recently I have been working on eXtensible Business Reporting Language (XBRL). XBRL turns on its head a domain that has traditionally been obsessed with presentation, by having accountants apply precise semantic markup to financials; what would once have been a carefully crafted document is now reduced to a set of tagged values. This provokes unease in certain quarters: financial professionals are being asked to let themselves be separated from the canonical representation of data, of enormous commercial sensitivity, by a technology abstraction. Where before they would have been able to read the official form of a company's accounts themselves, they are now faced with the prospect of relying on a piece of software to present it to them in readable form.

This set me thinking: is this not a problem for the whole XML community? Much work is currently being done on moving the web from a variety of presentational markup to a more semantic markup. In this paper, I will explore the question of whether this process risks leaving the human users of the content behind, and what we might do to address this problem.

In the first part of this paper I want to look at some examples of people publishing precisely, semantically, marked-up XML as a primary format on the web. There are still surprisingly few, but some of them are pretty influential (Amazon with Amazon Web Services, some of the big newswires using NewsML, for example). How do these content providers try and manage the relationship between a bare-bones "semantic feed" in XML, and something that communicates with a human audience? There are a few technology issues here, but they are, by and large, well understood; in a distributed, web services environment, the real issue is one of trust: if you are just publishing the raw data, what should you do to ensure that whoever/whatever shows it to a human user will do so in away that is not misleading?

The answer to this sort of question depends in part on the nature of the content and the consequences of misrepresentation; which brings me back, ultimately, to XBRL, and the issues of confidence from which I started. In the last part of the paper I look at the specific problems faced by XBRL in this area, as an example of a markup language that is entirely semantic, designed for machine-readability, and handling data whose misrepresentation can have serious consequences. I try to sketch what I see as the shape of a solution, and to draw some conclusions for publishing semantically marked-up XML more generally.

Table of Contents

1. Introduction

In their paper [SemanticWeb], Tim Berners-Lee, James Hendler and Ora Lassila give an example of how semantic web technologies might improve user-experience. "Lucy" intstructs her Semantic Web Agent to find an "excellent" (or at least "very good") doctor in a convenient location to treat her mother for a medical condition. It's an interesting and compelling example, and I think I might just be able to imagine it working; but as things stand now, I'm damned if I'm going to trust a piece of software to influence my decision on something as important as this! Reactionary? Yes. Pathetically technophobic? Guilty as charged. Unrepresentative of the average response? Not a chance. If semantic web technologies are going to take off, end users need to trust computer systems to represent and interpret a great deal of highly important information; and we're still a *long* way off achieving this.

This is not, however, a paper about the standard issues of trust that surround the semantic web, which are already comparatively well understood. There has been considerable, fruitful research around the area of distributed trust networks and their application to the semantic web, and such a model seems to me to be a highly convincing one for the task. The problem I want to consider is a much more humdrum, practical one: how do you persuade people to trust in the kind of system that tries to model trust for them?

I deal on a day to day basis with people who are concerned and challenged by the idea of attempting to codify their understanding of domains in which they are expert into structured XML data formats. On some levels, they see the long term benefits; on others they are deeply frustrated by trying to express themselves in a format that they find constraining and unfamiliar. We've got HTML and PDF as electronic formats that can be archived, categorised and searched; they look nice, everyone knows what they're getting because they can see it on the screen, and at the end of the day, when it all gets a bit much (and this is the really important bit) you can print it out and mail it to someone for analysis, happy in the knowledge that you know exactly where you stand with a bit of paper[1].

Researchers, evangelists and early adopters of semantic web technology are rightly putting a lot of thought into the thorny problems of distributed trust systems; but in reality, I think there are some much more pressing questions that need to be answered right now. The first step on the route to a semantic web is having data in formats that computers stand a fighting chance of being able to process; and I believe that one of the key barriers in the way of this goal is fear: fear of losing control of one's content to a machine. The thesis of this paper is that we should be thinking about this trust issue before we start getting excited about distributed webs of trust.

The first part of this paper is about providing some material for discussion. I want to define what I mean by semantic markup, and then look at a few examples and the trust issues they raise. In the second section of the paper I want to draw out some more abstract conclusions about the nature of the trust questions involved. In the final section, I will look in more detail at the concrete example of XBRL, and explore what can and is being done to address these sorts of problems in the immediate term for that technology.

2. Semantic Markup Examples

Belatedly, then, I should probably define my terms. When I use the buzzwords "Semantic Markup" and "Semantic Web" I'm not confining myself to RDF/OWL and the work around those standards; I'm talking about the wider movement in how content is created, stored and published online. Many (perhaps most) popular markup dialects today have a distinct similarity to programming languages; each tag that you insert constitutes some sort of instruction to a computer to process your content in some particular way. An extreme case is something like XSLT, which *is* a programming language; but in practice HTML and even DocBook also fall into this category to some degree. The `emphasis` and `ulink` tags in DocBook are pretty direct instructions to a processing application, even if others like `title` and `section` are less so. In reality, even these more abstract tags end up being instructions after a fashion: I wouldn't be terribly happy submitting my paper in DocBook without being able to preview the result, and on some level I inevitably end up seeing `title` primarily as a way of giving me some big bold text at the start of my sections with an appropriate entry in the table of contents.

[1] No, really. This actually happens in UK government. Who says eGovernment is a myth?

Nevertheless `title` is clearly a different sort of beast to `font`; the latter is a pure instruction, whereas the former says a little bit about what I mean. DocBook forces me to express more of the logical structure of my documents, rather than simply providing instructions to a renderer. Processing applications can do more sophisticated things more reliably, like generating tables of contents.

For some applications, this is enough; but people are increasingly looking to do more ambitious things with their content. They want to be able to categorise, index and search it effectively; they want to be able to reformat and repurpose it as the situation demands; sometimes they even want to be able to check it for correctness and extract data items from it. At this point, *structural* transparency ceases to be enough; it becomes necessary for the content to be much more transparent to a computer on a *propositional* level. As Mark Pilgrim says [http://diveintomark.org/archives/2002/12/29/million_dollar_markup], at this point you have a choice; you can go down the Google route and try to write a "million dollar application" that can extract this kind of information from unstructured text; or you can go for the "million dollar markup" option, and try and put structured semantic information into your content in the first place. And it is this sort of markup that I want to talk about as "semantic markup" for the purposes of this paper. With that clarified, let's move on to some concrete examples; for each one, I want to consider first what motivates people to adopt that particular form of markup, and then what problems this raises at the level of human-computer interaction.

2.1. META Tags/DocBook keywords/equivalent

The simplest form of "semantic markup" is to include a series of keywords at the head of a document. This simple step of pulling out and highlighting important words can dramatically simplify the exercise of searching and indexing documents. In crude terms this paper is about: XML, Semantic Markup, Trust, XBRL, Content Management. It may not be very sophisticated having a simple list of terms like this, but it's a big step up from having to guess based on the frequency with which words appear in the document.

In the form of HTML META tags, this is probably the most widely used form of semantic markup around today; so it is a good starting point for our investigation into trust in semantic markup.

2.1.1. Adoption

People use keywords in order to make their content easier to find. The primary driver for people putting META tags into HTML files is to get higher rankings on the search engines. I put keywords into my paper so that people will find it when they search the conference proceedings for information on the relevant topics. There's a clear and immediate benefit that drives adoption here.

2.1.2. Trust issues

Google doesn't use META tags. Why? It doesn't trust them. In part, there's a simple issue here: do you trust the publisher of the content? But underlying this is something more interesting. When people use keywords they don't do so out of a sort of "semantic altruism" in a bid to make the world a better place; they do it to achieve a specific result. How they tag their content is influenced to a large extent by what effect they expect this to have on the system that is doing the processing. Where the content author's aims are aligned with those of the people searching this doesn't cause a problem, but this tends only to be true of small closed environments. In the big wide world of the internet, these kind of keywords often become little more than a "search engine manipulation language". Rather than communicating about the content they purport to describe, they in fact document what kind of audience the author desires to reach.

2.2. Amazon Web Services

This is a little different to my other examples in that it isn't an open standard for the publication of information; furthermore, a lot of AWS (Amazon Web Services) is transactional in nature, to allow resellers to operate through the Amazon marketplace infrastructure. The core product database, however, constitutes a massive online resource that is effectively published in a highly structured XML format. In the usual scheme of things, online retailers only publish this sort of

information in the form of the HTML (probably dynamically generated) of their websites, and almost certainly don't disclose all that is available through AWS.

2.2.1. Adoption

As a proprietary format, the issue of adoption for AWS is rather different to that for the other examples we have considered: there is only ever likely to be one publisher of the information. Nevertheless it is interesting to ask the question: what persuaded Amazon of the benefits of making all this information available in structured XML? The answer is that the overwhelming majority of the people who use this XML do so in a way that pushes more business their way. It is no more than a pleasing side effect that a few people have created some pretty neat applications that don't directly benefit Amazon. It made sense for them to invest time and effort into publishing their semantic markup to the world because that markup is so closely tied to an application that makes them an awful lot of money; Amazon's XML is useful primarily to people wanting to do business with Amazon.

2.2.2. Trust issues

As a format which essentially has a single publisher, there aren't the usual trust issues with AWS data. Assuming someone hasn't spoofed the Amazon site, you know that you are getting information direct from the horse's mouth. What is interesting is that there are trust issues, but they work the other way round. We have observed that the tight linkage between the data and Amazon's application is key to the commercial success of the exercise; but it also gives them a much greater degree of responsibility for what is done with the information. When publishing in HTML, form and content are closely bound together; your information appears [2] broadly as you intend it to. When publishing in XML, however, you can be fairly confident that the end user is not going to be reading your XML directly; someone is going to process it somehow for display. In the case of AWS, there's quite a high likelihood that this will take the form of an eCommerce site built on the XML data, which ultimately leads back to the Amazon marketplace. So what happens when people process Amazon's XML in ways that are misleading? Perhaps they leave out a critical piece of information; perhaps they display old prices or old availability data. The end result is likely to be embarrassing for Amazon at best; at worst it might have legal ramifications.

Amazon's solution at the moment is very simple: they have some very carefully worded terms and conditions that specify exactly what you are and aren't allowed to do with the information. You are obliged to update pricing and availability information at certain specified intervals, for example. The fundamental problem here is that the end user needs to be confident that the information represented to him is a faithful presentation of what came from Amazon; he needs, in effect, to be able to establish trust in a particular mode of presentation. This is a requirement that we will be definitely be revisiting later in this paper.

2.3. XBRL

XBRL is in many ways the quintessential example of a semantic markup language, if not so much for technical reasons as by virtue of the nature of the domain it addresses. Business reporting, and in particular financial reporting, is still very much dominated by unstructured content. Worse still, much of the unstructured content comes in the form of glossy presentations from which it can be difficult even to extract the important text. In many cases financial statements are designed to be printed out and mailed to a regulator! XBRL aims to decompose the meat of these statements into a series of tagged data items - Revenue, Profits, FixedAssets etc. This constitutes a paradigm shift in the way such reporting is conducted, from almost completely opaque to automatic processing, to a very high degree of transparency. Furthermore, like RDF/OWL, as a meta-language XBRL is all about providing mechanisms for extending and enriching the semantic vocabularies that are used for creating such transparent reports.

[2] Ignoring for the moment the minefield of user-agent differences.

2.3.1. Adoption

Adoption is very much a live issue in the XBRL community at the moment. From the point of view of organisations who *use* business reports (especially financial statements), there is a reasonably clear case for adoption. Currently many such institutions are forced to key in the vast majority of the information they receive manually (if they even key it in at all). This is enormously expensive and error-prone. Receiving information in a rich semantic markup would be an enormous leap forward.

What about those submitting the information? In theory, it would be wonderful if every company could take an automatic dump of their general ledger and have it written straight through to a set of XBRL accounts ready for publication; this would save the fabulous sums of money currently spent on pulling all this information together, copy-pasting it into a graphical layout, cross-checking it and getting it published. Furthermore, many companies, particularly smaller ones, have a vested interest in getting their financial data into analyst models. At the moment this exercise is sufficiently expensive that most analysts are only going to do it for the larger companies; but if the information is published in an electronic form that can be analysed directly, much of this cost disappears. So reduction of cost and improved market visibility are the key drivers to adoption here.

2.3.2. Trust issues

Particularly with the advent of Sarbanes-Oxley, accounting is a highly sensitive domain; accuracy is of paramount importance, since errors can have serious legal ramificiations. Unsurprisingly this gives rise to confidence issues that sometimes threaten to obscure the potential advantages that encourage organisations to consider adopting XBRL in the first place.

At the moment, the majority of the "real" XBRL that is being created is done as an afterthought. Companies take their existing, audited financial reports and tag them up manually as XBRL. This process is fraught with difficulties, not least because they have a legal responsibility to ensure that the contents of the XBRL markup are *exactly* the same as the contents of the published paper accounts. To illustrate the kind of difficulties that arise, consider the following. In an XBRL taxonomy (essentially a supercharged XML Schema) it is possible to define arithmetic calculation relationships between data items; "Revenue - Expenses = Profit(Loss)", for example. This caused some interesting problems for some of the first companies to start tagging their accounts with the US GAAP (General Accepted Accountancy Principles) taxonomy; when validating their efforts they were being told that their financial statements didn't add up properly! The semantic vocabulary they were using expected more and different information to be reported than was published in their audited accounts. In many cases, it was trivial to infer the missing information from what was present; but it wasn't legally acceptable to include the missing data explicitly!

Accountancy is sufficiently fluid that one person's definition of a concept may differ subtly from another's in a way that makes it difficult to define absolute taxonomies; all too often there isn't a single correct way to calculate something. In a paper statement, an accountant can stick a text label on a figure and rely on the experience of those reading it to interpret it correctly within the context of the rest of the document; in XBRL, he is expected to select a tag with predefined semantics to hold the information. What happens when those semantics don't align precisely with what he is expecting? He is used to a a model under which the meaning of a fact is determined by a combination of general accepted practice and the context provided by a particular financial statement; he is now expected to work with one in which the meaning of a fact is defined in an rigid fashion in an external XBRL taxonomy. The fear of being misunderstood in a mode of expression that is decidedly alien is a significant barrier to adoption for many here.

As XBRL gains ground, it will hopefully start to become with primary format for business reporting applications rather than just an adjunct. But if this alleviates the pain of managing two different versions of the same information, it will further exacerbate the concerns over semantic fidelity. In the UK, HMRC (Her Majesty's Revenue and Customs) are gearing up to accept Corporation Tax Computations in XBRL. Currently they have thousands of highly paid tax inspectors who pore over computations looking for irregularities. The goal is for the new system to triage the inbound submissions so that the inspectors don't waste most of their time looking at trivial cases that give no cause for comment. The difficulty is that the inspectors still need to look at *some* of the computations, and they're not going to do it in XBRL directly!

The preparers no longer have to submit a rendered version of the computation, which saves them having to keep two versions in sync, but the tax inspector needs something to look at. This creates a peculiar situation: people filing their Corporation Tax Computations will no doubt want to look at what they are about to send in a format they can understand; the inspector at the other end will want to do the same; and if there are problems, the inspector will want to discuss them with the submitter with reference to a human readable format; but no information about the layout of such a format is ever transmitted between the two! The preparer is thus left in the invidious situation of making a very important financial submission without knowing what it will actually look like to any tax inspector who may ultimately examine it! As you can imagine, this has caused considerable concern in the relevant quarters.

This is symptomatic of a deeper problem: the people who understand the data think in terms of text labels and visual layouts; they always have done, and, most likely, they always will do. XBRL, by contrast, relies on thinking in terms of discrete *concepts*. The challenge is persuading the content owners to trust that the mess of angle brackets accurately represents the rendered layout that they understand; and furthermore, to convince them that it does so in a way that will allow others to use it to recreate a rendered layout that will also be faithful to the original intention.

The solution chosen by the Corporation Tax project was to publish an official stylesheet along with the taxonomy; this allows all parties to see exactly what any given piece of Corporation Tax XBRL will look like to a tax inspector. As a first stab, it's not a bad solution. But for a taxonomy with 6000 elements modeling a domain in which preparers have previously been free to submit information in any layout they like, it's pretty limiting. Is there a better way to gain people's trust here?

2.4. UK eGovernment XML Schemas

The final example I want to consider briefly is drawn from what I do on a daily basis: helping the UK government create XML Schemas. For the most part, this is about modeling existing business processes, often based around paper forms as web services available for electronic submissions.

2.4.1. Adoption

The driver to adoption here is a fairly standard one: the government wants people to stop sending it bits of paper and start submitting information in electronic formats that can be processed and validated automatically. Done well, this has the potential to save them a lot of money, and should also massively increase the quality of the data that they hold.

2.4.2. Trust issues

Much of the information about the definitions of the data to be collected and the rules governing what is allowed for a particular type of submission is currently contained in the heads of a handful of experts within the relevant government departments. The challenge is to find a way to allow them to convert this knowledge into the specification of a semantic markup vocabulary with associated validation rules. Because the people who can speak authoritatively about the business requirements generally don't understand the formats that are being created it is extremely difficult to validate that those formats are correct. An example of the sort of problems that arise: many of the paper forms that are modeled contain questions like "Is this an Occupational Pension Scheme? (yes/no)"; if the answer to such a question is "yes", then one set of further questions should be answered, if it is "no" then another set of further questions should be answered instead. The logical way to model this is to have two container elements `OccupationalPensionScheme` and `NonOccupationalPensionScheme` which then contain elements for the subsequent questions for each case. All too often, however, such an approach is rejected because there is no single element that contains the "yes/no" value that was specified in the original rules. From a logical point of view this is madness, because it is clear that no information has really been lost here; but this is not necessarily clear to the experts who are asked to sign off on the new data format, because they don't understand the relationship between the tree-based markup structure and the text-based specification that they originally provided.

When business experts do finally codify their understanding of a particular piece of information into a specific set of rules and definitions, another problem arises: they have a tendency to become very "parochial" in their understanding.

"A National Insurance Number is a structure like *this* because that's how *we* have always collected them." "The information required to uniquely identify a UK bank account doesn't include a building society reference because we've never asked anyone for one of those." This kind of approach results in an enormous number of subtly different definitions of the same basic units of information, and is frequently not obvious how to convert from one to another. At last count there were something like 19 different XML formats for representing an address within HMRC alone; very few of them have defined relationships to any of the others! The problem arises because no one believes that anyone else's definitions will meet their requirements. In part this is self-perpetuating: if everyone creates independent, incompatible data formats, there is little incentive for the next person to reuse what is already there. But it also stems from a reluctance to think in more general terms about how information should be modeled, because there is a concern that this might jeopardise satisfying the needs of the immediate application. What is needed is a vivid and comprehensible way of demonstrating how the definitions at the level of the markup map onto one another and the current requirements so that these issues can be explained and addressed rather than falling prey to the "this works for me" mentality.

3. The Human Consequences

So what do we learn from all this? I think there are two key lessons here. The first is obvious and already well understood: semantic markup must stop being an optional extra. The costs and risks associated with having to keep semantic markup synchronised with other, less structured forms of content lead to rapidly diminishing returns,[3] and as long as unstructured content remains the primary format, there will be insufficient incentive to come up with really good solutions to some of the challenges of repurposing more complex types of structured content.

The obvious solution is to do away with the parallel streams. Rather than trying to keep structured in sync with unstructured content, we should be trying to migrate the core data to structured markup and then generating whatever other formats we need from that. Anyone who attended David Haslam's "Automate Your Publishing" keynote [http://www.idealliance.org/proceedings/xml04/papers/267/267.html] at XML 2004 will remember the impressive ROI figures quoted for undertaking an exercise of this nature; if you can build all your content automatically from underlying data, semantic markup can be an *extremely* attractive proposition.

This is the classic case for XML as the basis for repurposing. It works well where there is detailed technical data within a well-defined domain. Crucially, it works well when the owner of the content has a pretty clear understanding of the type of application that will be processing it. A recurring theme within real-world XML repurposing success stories is the presence of clear specification of how the content is going to be used. In the case of Daimler-Chrysler, the whole process exists within a single (albeit large) organisation; it was comparatively easy to determine what data needed to be present in the XML, and in what format, in order to produce the various different outputs effectively. It might be argued that the success of Amazon Web Services constitutes a significant counter-example, in that Amazon have little idea of what third party developers will do with the data that is published. The crucial feature of AWS, however, is that Amazon have almost total control over the meaning of the markup they publish, because its significance is irrevocably tied to the Amazon store itself. For example, many people on the internet have started using ASINs (Amazon Standard Identification Number) almost as generic product identifiers; but the only place you can resolve an ASIN is the Amazon website. For users of AWS, the meaning of the tags in the markup is quite simple: they mean what Amazon say they mean!

The second of the key lessons concerns those cases where the domain being modeled is not so well-defined, or there is no single organisation that "owns" the definitions to be used. Even for comparatively simple concepts, like an Address, or the Author of a book, there are likely to be subtle differences in definition between different content authors.[4] The problems that this creates discourage many from embarking on the exercise of putting their content into a structured format, because they don't trust the machine to understand it correctly. Computers have a nasty habit of being entirely literal-minded about what you tell them, which people find disconcerting; the fluidity of natural language and unstructured

[3] In some cases this can be a very serious issue indeed. Bill Zoellick gives a nice summary [http://gilbane.com/blog/archives/2005/04/the_matter_of_a.html] of the assurance problem that such duplication creates for XBRL on the Gilbane report blog.

[4] Even the standard semweb examples of the Library of Congress' "Author" vs. the British Library's "Creator" is, I suspect, not immune to this; there are surely cases where the British Library would classify someone as "Creator" but the LoC would not classify the same person as "Author".

content is comforting because one can, by and large, rely on other people to interpret correctly what one is trying to communicate. One of the deepest fears here is of the failure of human communication over electronic channels; if I only put my content into a structured format, and this ultimately gets rendered for another human being, I want to be sure that my information is communicated correctly and in full to that person at the far end of the process, whatever formats it may have gone through in between.

If we accept that people will continue to feel more comfortable with visual layouts than careful conceptual distinctions, then we must ask how we can translate reliably between the two. Such a mechanism needs to inspire confidence in a non-technical userbase; in the most extreme case, sufficient confidence that an auditor can audit the visual representation of a set of accounts, and be satisfied that he has meaningfully audited the underlying XBRL data. Given that this probably provides the toughest requirement to meet, I would like to dicuss potential solutions in more concrete terms by looking at what can and is being done to solve this problem for XBRL.

4. XBRL - A Case Study for a Solution

XBRL models an extremely complex domain. An XBRL taxonomy, which defines a particular reporting vocabulary, is a complex, multidimensional datastructure in its own right. The reports themselves that are expressed in these vocabularies are mapped along further, different dimensions. Presented as PDF, such reports usually consist of a series of complex tabular layouts pivoting the same data across a variety of different dimensions. Matters are then further complicated by the fact that the reporting concepts are themselves quite fluid. So how do we go about builidng the confidence of potential adopters in XBRL's ability to model such a domain in a way that accurately reflects their understanding of it?

There are two sides to this problem: what tools do the people who provide the taxonomies require, and what tools to the people creating the final reports require? Although it might seem perverse, I'll start with the latter question, because I think it will lead us to the right answer to the first.

At the moment the majority of tools designed to help people create XBRL reports work in much the same way: you start with a financial report in a traditional layout, and you annotate this with information that identifies how the resulting markup should be structured. Generally speaking, this tagging process reflects the dimensional nature of the data in question: you tag a column (or part thereof) as corresponding to one dimension (for example, time - a particular financial year) and you tag a row as corresponding to another (such as the "concept dimension" - e.g. "Revenue"). At the end of the process XBRL the application collates all the information and outputs some XBRL-tagged data.

Broadly speaking, this seems to be an approach that preparers are happy with; it allows them to continue to work with the data in formats that they understand, and to see and control the relationship with the underlying XBRL tags. Two things cause problems, however. First, they still need to understand a lot about the way the markup works in order to be able to select the correct tags to apply to their existing data. Second, the output from the system is just the XBRL: most of the information about how the XBRL elements relate to a particular presentation is "lost in translation".

One way round this "lossiness" is to publish the tagged original rather than just the XBRL; this lets people see the intended layout whilst also being able to see how this relates to a structured content version of the same data. What this requires from a technical point of view is a standardised format for such tagged, rendered layouts. There are a few technologies already available that potentially fit the bill. [XForms] is one possibility; it is built around exactly this idea of binding data tightly to a presentation without losing the independence of form and semantics. Probably more appropriate for the XBRL arena, however, is a conceptual very similar technology: the XML Forms Architecture ([XFA]) that drives the latest generation of Adobe data-driven PDF formats. This too offers the right kind of data-binding architecture, but in addition provides more in the way of detailed rendering control, which is important in a domain used to glossy publications!

Knowing which tag to use for a particular line of a financial statement is a more difficult issue to tackle. At the moment users are usually presented with a tree of concepts available in the current taxonomy. Each concept will have a label, some documentation, and possibly a calculation relationship with other concepts in the taxonomy. In some cases, the label and the position of the concept in the taxonomy structure will correspond quite closely with the text and structure

of the financial statement being tagged; but frequently it will not. In such cases, the user has a difficult choice: does he try and search for a similar concept that has a reasonable correspondence with what they are trying to tag; or does he define a new extension to the taxonomy he is using? In either case there's a very real risk that the resulting markup will not be as semantically transparent to a receiving system as it ought to be.

So what's the alternative? Well, we saw earlier one solution to communicating how elements in the taxonomy were intended to be used: publish a rendering mechanism (in the form of a stylesheet) along with taxonomy. Interestingly, when we created that stylesheet, we found that by far the easiest way of creating meaningful sample data was to modify the stylesheet to create an HTML form that corresponded to the rendered layout, into which values could be input. This reinforces the idea that the mechanisms for working with semantic markup have to be very closely tied to the rendering layouts that people are used to.

The observation I made earlier, however, was that a single stylesheet producing a fixed layout was not really adequate to capture the many different ways of setting out the same logical data; what is actually needed is something finer-grained and more flexible. Addressing this requirement is going to be challenging and is probably one for the longer term, but what I would ideally like to see is small-scale formatting "patterns" being encoded into the taxonomy that describe standard ways of displaying an element or group of elements. This information could be used to construct a library of miniature pre-rendered taxonomy sections from which a user could select. Since these layouts would be closer to what the user was accustomed to, and since there could potentially be a number of alternative layouts for the same elements to cover common variations, there would be a reduced likelihood of misunderstanding/miscommunication in the tagging process. In addition, where there was no existing data to tag, the layout components could be composed to create a form into which new data could be entered.

Such a system could be used in reverse to help those faced with the task of designing a new taxonomy. Rather than trying to sit down and analyse the domain into a set of orthogonal concepts, a taxonomy author might draw out a series of these miniature formatted sections. By then identifying which figures were equivalent across the different sections, he could provide sufficient information for an automated tool to start building up a taxonomy of distinct concepts; and there would already be a wealth of information available about how to display these concepts.

This is all largely speculative, if thoroughly grounded in the experience of what has worked for our clients and what has not. I have no doubt that there would be significant challenges involved in getting such a system to work effectively. I do think, however, that it is the right *sort* of solution to the problems we have been discussing, because it recognises the crucial importance of allowing users to specify semantics in terms of the presentations which which they are most familiar.

5. Conclusions

I hope this paper does not seem excessively reactionary or technophobic: it is not intended to be. If the promises of semantic markup are empty, then I'm likely to be out of a job fairly shortly! I also recognise that by focusing primarily on the issue of creating structured content from unstructured, I am looking at only one area of the quest for a semantic web; as Tim Berners-Lee et al. are keen to point out, there are already many sources of highly structured information out there on the web that just need to be exposed in the appropriate way to allow the project of semantic interconnection to progress.

Nevertheless, it's worth bearing in mind that there's an *awful* lot of unstructured content out there on the web. Think of [Wikipedia], for example. The project [http://meta.wikimedia.org/wiki/Semantic_MediaWiki] to make MediaWiki[5] support semantic assertions is, for my money, one of the most exciting things going on in this field[6]. If such projects are to succeed, they need to take careful account of the practical and psychological issues that surround the shift they are attempting to bring about. I am not entirely convinced by the Semantic MediaWiki syntax, but I think it may well be on the right lines: one is encouraged to embed semantic assertions with the fabric of the normal unstructured text content. People aren't forced to give up on their normal ways of laying things out and expressing themselves; but at

[5] MediaWiki is the Wiki software on which Wikipedia runs
[6] Ironically, this project is using neither XML nor RDF!

the same time the structured markup they do use is an integral part of what they create, rather than being second class content. Where semantic markup wholly replaces existing methods of authoring, and rendered views are generated automatically, the markup often ends up being abused: users get wise to which tags produce which effects in the output rather than thinking about the underlying meaning of what they are expressing[7]. Another interesting, if less ambitious, example along the same lines is [NITF], designed for marking up news story text. Indvidual words and phrases within the body of the news story are surrounded by XML tags like `person`, `org` or `event` which add a bit of extra semantic richness to the content, but are entirely orthogonal to the remaining presentational markup of the story: there is no temptation (as there is in DocBook, for example) to misuse the semantic elements for a particular rendering effect.

So in sum, what I'm suggesting is this: let's persuade more people to publish their content in formats that can be automatically processable; but let's do this in a way that recognises that human communication is paramount here. For the time being, people want to stay in control; they want to have the final say in the auditing of financial statements; they want to understand exactly what they're sending to the government; they want to explore in detail the reasons why they should pick one doctor rather than another. They want to continue to create, view and understand content in the same presentation structured that they are used to; but let's create some tools that start to make such renderings into flexible templates which give a window onto some underlying, tightly integrated semantic data. People can remain in direct control of their content, how it is structured, and how it appears, whilst at the same time having a transparent view of how it will be communicated to the machine. That way, people will easily be able to see when the machine has got it wrong, and they will trust that they are not losing control; and ultimately, as they see the quality and quantity of the semantic data increasing, they will perhaps come to trust that semantic web systems can be relied on to get it right without quite so much monitoring.

Bibliography

[XBRL] *eXtensible Business Reporting Language [http://www.xbrl.org].*

[AWS] *Amazon Web Services [http://www.amazon.com/aws]*

[Wikipedia] *Wikipedia [http://www.wikipedia.org]*

[SemanticWeb] *The Semantic Web*, Tim Berners-Lee, James Hendler and Ora Lassila, Scientific American, May 2001. Available on the Scientific American website. [http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2]

[NITF] *News Industry Text Format [http://www.nitf.org]*

[XForms] *The W3C XForms homepage [http://www.w3.org/MarkUp/Forms/]*

[XFA] *Adobe XFA 2.1 Specifications [http://partners.adobe.com/public/developer/xml/index_arch.html]*

[7] Matthew Thomas makes this point very well in his blog. [http://mpt.net.nz/archive/2004/05/02/b-and-i]

Biography

Lucian **Holland**
Client Projects Manager
DecisionSoft [http://www.decisionsoft.com]
Oxford
United Kingdom

Lucian works for DecisionSoft, well known for their cutting-edge XML expertise. In his roles first as technical architect and now client projects manager he has been responsible for the design of a number of XML applications including their industry leading True North XBRL validator. He is a regular contributor to XML and XBRL conferences. Currently he spends most of his time talking to UK government departments about processes to help them develop XML Schemas in an efficient, consistent and robust manner; sometimes they even listen to him...