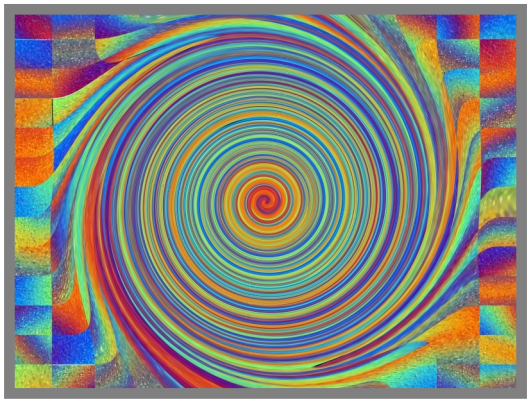
structWSF: A Framework for Data Mixing

by Mike Bergman - Tuesday, June 30, 2009

http://www.mkbergman.com/496/structwsf-a-framework-for-data-mixing/



Interoperable Naïve Data Structs, Datasets and Canonical RDF

As I noted in my review of SemTech 2009, one of the key themes of the conference was data federation. Unfortunately, data federation has been a term a bit out of vogue for a while. (Though I still think it best captures the space.)

The current vernacular has been pushing forward an alternative: *data mixing*. One of the larger product pushes at the conference was by Zepheira for its new Freemix service and product. Freemix is a hosted service largely built around the Exhibit data display application, aided by some tools to make creating an exhibit easier. Exhibit is an attractive presentation system; for nearly three years **AI3**'s own Sweet Tools dataset listing of semantic Web and -related tools has been presented via Exhibit.

Freemix looks promising and is now being offered in beta. But one thing caught my ear when listening to the company's announcement: they are not yet able and ready to show the "data mixing" part of the system. Its release is apparently being delayed until later this year because of the difficulties encountered.

This post coincides with the release of the alpha version of the **structWSF** code on the OpenStructs Web site. It is available for download under Apache 2 license.

We'll be blogging a few more times in the coming days regarding other possible uses and applications for this platform-independent Web services framework.

What is Data Mixing and Why is it So Hard?

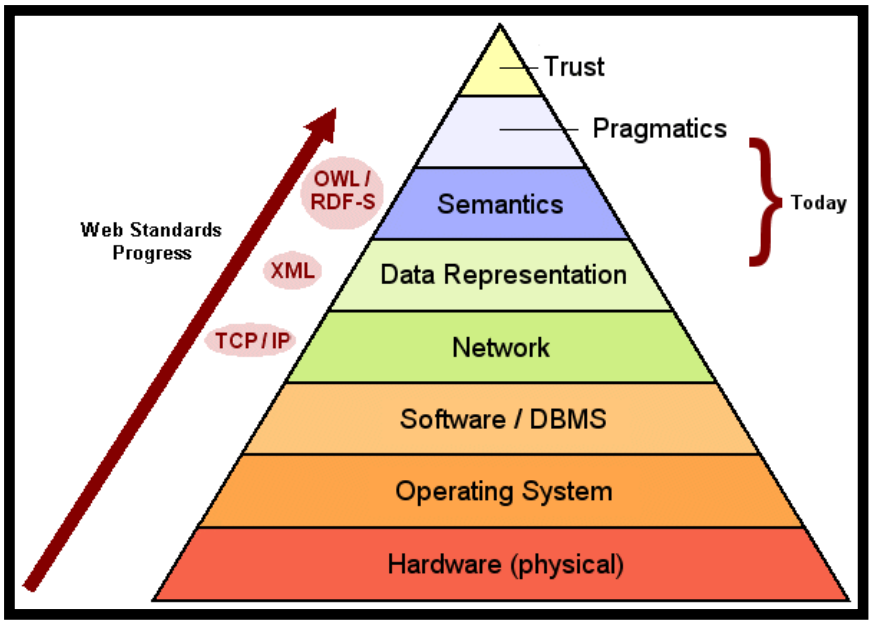
As a new term there is no "official" definition of *data mixing*. However, I think we can consider it as generally equivalent to the older data federation concept.

Data federation is the bringing together of data from heterogeneous and often physically distributed data sources into a single, coherent view. Sometimes this is the result of searching across multiple sources, in which case it is called federated search. But it is not limited to search. Data federation is a key concept in business intelligence and data warehousing and a driver behind master data management (MDM).

As I first wrote about data federation about five years ago [1]:

Data federation first became a research emphasis within the biology and computer science communities in the 1980s. At that time, extreme diversity in physical hardware, operating systems, databases, software and immature networking protocols hampered the sharing of data.

Yet it is easy to overlook the massive strides in overcoming these obstacles in the past two decades.



The Internet and its TCP/IP and Web HTTP protocols and XML standards in particular, have been major contributors to overcoming respective physical and syntactical and data exchange heterogeneities. The current challenge is to resolve differences in meaning, or ***semantics***, between disparate data sources. Your "*glad*" may be someone else's "*happy*" and you may organize the world into countries while others organize by regions or cultures.

Resolving semantic heterogeneities is also called *semantic mediation* or *data mediation*. Though it displays as a small portion of the pyramid above, resolving semantics is a complicated task and may involve *structural conflicts* (such as naming, generalization, aggregation), *domain conflicts* (such as schemas or units), or *data conflicts* (such as synonyms or missing values). Researchers have identified nearly 40 distinct types of possible semantic heterogeneities [2].

Ontologies provide a means to define and describe these different worldviews. Referentially integral languages such as RDF (Resource Description Framework) and its schema implementation (RDF-S) or

the Web ontological description language OWL are leading standards among other emerging ones for machine-readable means to communicate the semantics of data.

Fortunately, we have climbed most of this data federation pyramid. The stumbling block now are the ***semantics.*** This is made all the harder when we place too much burden on the data transmission or "packet" itself. In other words, does ***exchange*** also carry with it the burden of ***meaning***? The rest of this post tries to explain what I mean by this and how it relates to our new **structWSF** Web services framework.

Is it Apples or Oranges?

Not to pick on any one thing or any individuals, but three recent threads on semantic Web-related mailing lists help illustrate in various ways some interesting mindsets. While there is much on each of these threads of other value, I'm only focusing on a narrow topic from each based on my thesis at hand.

And, what is that thesis? It is simply that we too often mix instance record and attribute assertions with schema representations and world views. And, when we do, we sometimes make mountains out of molehills (or mix apples and oranges to completely mix metaphors).

Example 1: Squeezing RDF into JSON

JSON (JavaScript Object Notation) is a data notation or syntax, easily created and widely used for current Web apps. It has a rather simple syntax for representing attribute-value pairs. Many useful tools and parsers for the serialization exist.

In keeping with his general and broad criticisms of how the semantic Web standards and approaches have been promulgated by the W3C to date, John Sowa most recently expressed his ideas in a posting to the ontolog-forum mailing list under the heading of 'Semantic Systems' [3]. In this thread, John proposes:

1. The recommended exchange form for RDF will become JSON. Any JSON documents that are limited to triples can use the old XML-based RDF form, but they can also use the more compact and more general full JSON.

Then, in a subsequent posting to that thread he notes:

*5. The W3C made a major blunder with a one-size-fits-all approach that tried to use a document tagging language as a knowledge representation language. The result was the *worst* notation for logic ever invented.*

Finally, he goes on to note in a further post:

*JSON could be used as an alternative to XML for the syntax, but the lack of a standard semantics for JSON means that it could *not* be used as a replacement for RDF *unless* an official standard were*

adopted for mapping RDF to and from a particular subset of JSON whose semantics was defined in Common Logic.

All of this John proposes in the spirit of:

*The goal of my proposal is nothing less than a total *integration* of the Semantic Web methodologies with the methodologies that have been used in the traditional software development community* [3].

I find common ground with a couple of the ideas in this proposal. First, accepted formats like JSON should have a prominent place in data exchange. Second, leveraging methodologies used in the traditional community is definitely a good thing.

But John, while suggesting reuse of existing traditions, is also paradoxically recommending a wholesale replacement for RDF. He is also positing a single exchange standard (JSON). And, he stops tantalizingly short of recognizing an important truth that I'm sure he knows: simple instance record assertions and representations -- the essence of data exchange -- can and should be viewed separately from schema representations.

As I have noted in my earlier naïve data 'structs' series, there are in fact scores of existing data transfer formats that have been adopted by their communities -- and are likely to remain popular within those communities for some time -- that can play a similar role to JSON. So long as the role of data exchange is kept to the assertions ("metadata") about instances, many formats can play in the sandbox.

The role of RDF may or may not reside with data exchange. To conflate and equate RDF and JSON is to reduce the power of keeping instance record representations separate from schema and world view representations. John's basic sensibilities, I think, could be more effectively promoted by not posing 'either-or' strawmen and recognizing that data exchange formats will ALWAYS be diverse and heterogeneous.

Observation: Existing and emerging data 'structs' useful to data exchange will remain manifest in format and diversity; data exchange imperatives are a different matter from schema and knowledge representation.

Example 2: RDFa is Not 'Expressive' Enough

Somewhat in contrast to this thread was a different one by Martin Hepp, editor of the excellent Good Relations ontology, on the LOD (linked open data) mailing list [4]. This thread, which sensibly questions how difficult it is for mere mortals to configure an Apache server to support publishing RDF, reached further into the realm of RDFa as a document annotation language.

As Hepp states,

The reason is that, as beautiful the idea is of using RDFa to make a) the human-readable presentation and b) the machine-readable meta-data link to the same literals, the problematic is it in reality once the structure of a) and b) are very different. For very simple property-value pairs, embedding RDFa markup is no problem. But if you have a bit more complexity at the conceptual level and in particular if there are

significant differences to the structure of the presentation (e.g. in terms of granularity, ordering of elements, etc.), it gets very, very messy and hard to maintain.

Further discussion in this thread elaborates the interest in having the documents in which the RDFa is embedded carry much more schema-level information.

Like the Sowa case, this raises the question of where to draw the line. Should embedded metadata in documents carry complex schema information as well? So, we now shift the focus from data exchange to schema representation.

I think this is really unnecessary since it is quite easy in RDFa to refer to a separately specified schema. By, in this case, conflating metadata transfer and exchange with schema, the bar has been raised unnecessarily high.

If we need to capture schema and world views, fine, let us do so directly and succinctly. Then, let our document metadata (in this case using RDFa) make attribute assertions about that "payload" simply and cleanly. The Web certainly does not need individual documents carrying with them entire schema representational views of the world.

Observation: Data exchange, even based on RDF (via RDFa), is best kept to the assertions of facts and attributes.

Example 3: Mixing Vocabularies

In a microformats context, Thomas Loertsch posed some questions on mixing vocabularies [5] and how they should be interpreted. This caused an involved discussion of intent and possible implications and best practices, with discussants including Brian Suda, Peter Mika, Ben Ward and others. It also led to the start of a useful wiki page on how objects should be represented in Web pages when multiple microformats can be invoked.

For quite some time microformats, I think, have gotten the "mix" just about right. They have created well-reasoned attributes for distinct instance types and seek to keep their embedding of that information simple in existing documents. Some advocate while others question the rigor of the microformat structure; that is not the topic here.

What is interesting about this thread is that it evolved to discuss the implications and best practices when an author posts a document with more than one microformat. How do these vocabularies relate? How should we, as "consumers" of the document, parse the vocabularies?

Yahoo!'s SearchMonkey service has recognized microformats for some time, and its questions regarding interpretation and best practices in the thread were natural. But the interesting point that seemed to come out of this thread is that users will post microformats as they wish. While care and standards in the design of the microformats can help reduce confusion and conflict, it can not guarantee it. The final responsibility for proper ingest and processing likely resides with the aggregators and publishers that consume such data.

So, here, too, we have another case of asserting metadata and embedding for data exchange in a slightly different native format than RDF. Huzzah!

Observation: Standards setters and consuming agents (often aggregators, publishers or search engines) should take lead responsibility for best practices and processing attribute data, realizing that original authors and developers may not fully comply.

Revisiting the ABox and TBox Split

These examples are a bit of a long way around the barn to reinforce what we have been arguing for some time: the need for a proper split between the ABox (assertions related to instances) and the TBox (concept relationships, schema and world views) [6]. This has been a pretty constant theme in our writing, ranging from first introductions, to its relation to description logics, relationships to existing data 'structs', and explicit discussion of ABox and TBox roles in a four-part series.

One of the key points throughout this writing is that an ABox-TBox mindset provides a context and rigor for looking at questions such as our three examples above. In all three cases, I argue, the seeming conundrums result from lacking this mindset. Once this mindset is applied, the respective roles of various data formats, RDF, schema and the like naturally fall into place.

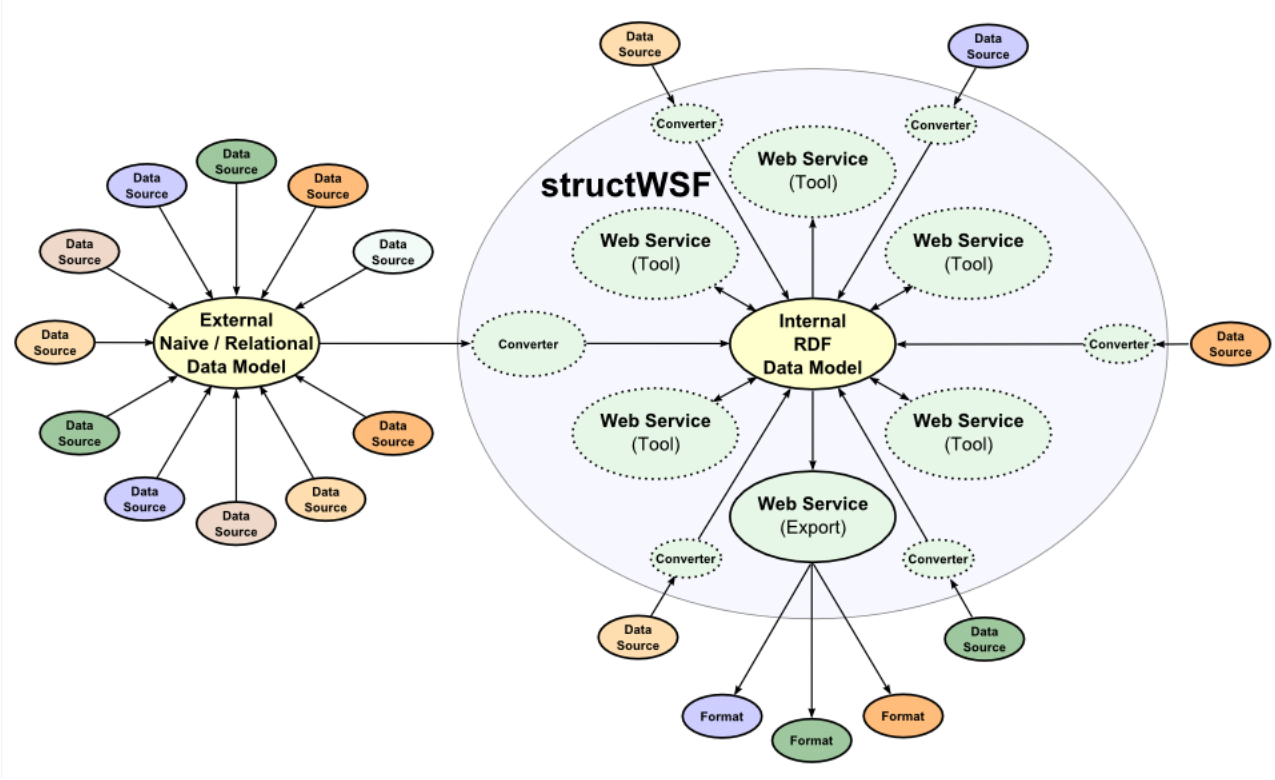
Of course, the Web is also a dirty and chaotic place where niceties of design and best practices are routinely ignored or unknown or purposefully rejected. So be it. This is reality. This reality needs to be accommodated. But good design can help overcome it and work to establish resilient, flexible architectures.

Of course, even though this might be good design, there is no ability to enforce such distinctions across the Web. However, insofar as key implementors are concerned (standards writers, major publishers, tools developers, industry experts, and the like) we can put in place better approaches. This mantra is at the heart of all that Structured Dynamics does -- including the **structWSF** Web services framework, just released as open source code.

A General Data Mixing Model

So, now we can finally turn our attention to the **structWSF** Web services framework, more broadly described here.

There are a number of perspectives and contexts to view this **structWSF** framework. In this posting, we take the boundary conditions of data formats and data exchange [7]. The key question for this perspective is: given the realities noted above, what is an adaptive framework for data mixing on the Web? Our schematic answer to this question is below:



The basic design has two key data considerations. First, all **structWSF** tools and Web services and schema work from the canonical RDF data model. It is the hub and common denominator for all **structWSF** installations. We are able to design and optimize generic tools and services (including converters) around this canonical framework.

Second, we assume most everything in the outside world to be non-compliant with this canonical model, with the data representations often naïve and incomplete. Converters (also known as translators or RDFizers) are an essential bridge to this external world, and need to be designed for re-use and extensibility.

Where the outside world is compliant, they conform to the **structWSF** APIs or are themselves **structWSF** installations. In these cases, direct data exchange and access with permission rights occurs at a dataset level (not shown).

The Naïve Part of the Spectrum

Converters are themselves *bona fide* Web services at the **structWSF** level. (Only a few are presently included in the alpha release.) While some may be one-off converters (sometimes off-the-shelf RDFizers), and often devoted to large volume external data sources, it is also helpful to emphasize one or more "standard" naïve external formats. A "standard" external format allows for a more sophisticated converter and enables specific tools to be more easily justified around the standard naïve format.

As noted above, this "standard" is often JSON or a derivative of JSON. But, just as readily, the common

'naïve' format could be SQL from relational databases or another format common to the community at hand. In many ways, because the emphasis of data exchange is on the ABox and instance records and assertions (and attribute extensions), the actual format and serialization is pretty much immaterial.

Emphasizing one or a few naïve external formats allows more tools and services to be cost-effectively developed for those formats. And, even though the format(s) chosen for this external standard may lack the expressiveness of RDF (and, ultimately, OWL), because the burden is principally related to data exchange, this layer can be readily optimized for the deployment at hand.

Besides import converters it is also important to have export services for the more broadly used naïve external formats. In fact, some **structWSF** services can be devoted to data cleanup or attribute (property) or object reconciliation (including disambiguation as a possibility). In this manner, **structWSF** installations could also improve the authority and trustworthiness of standard data in the wild.

Another common service for this naïve data is to give it unique URI identifiers and to make it Web-accessible, thus turning it into linked data.

The RDF Canonical Data Model

Such generic services are possible because the "highest common denominator" for the system is the canonical RDF model. Because it is the consistent basis for tools and services, once a converter is available and the external information schema is mapped to the internal structure, all existing tools and services are available for re-use. Moreover, this system and its datasets are now ready for sharing with other **structWSF** instances, within the enterprise or beyond.

Thus, we begin to see a network of canonical "hubs" in a sea of heterogeneity, the interoperation of which is facilitated by a **structWSF** framework at every network node. This design is discussed more in the next part of this series.

Some, such as Sowa noted above, would prefer a grounding in common logic (CL) as opposed to RDF. Our choice to use RDF is based on the simplicity and understandability of the data model, plus the richness of languages and standards from the W3C that surround the framework.

Even here, however, the RDF basis of **structWSF** need not be the final word. Because of a keen intent to keep all designs and ontologies used by **structWSF** firmly grounded in description logics, it is possible for the **structWSF** basis to be converted to other languages and frameworks such as CL that can be expressed in DL.

Bringing it Back to Data Federation

Data mixing -- or more preferably, *data federation* -- has as its heart the premise of heterogeneous and distributed data sources. It implicitly acknowledges differences in syntax, semantics and serializations.

The design and architecture of **structWSF** is similarly premised. While each of us may prefer one model or one format over others, we must interoperate in the real world. And that world, for many

understandable and immutable reasons, will retain its diversity. Accepting this reality is a first step to adaptive design.

So, we control what we can control, and we adapt to what else exists. We have chosen RDF as the canonical data model that we can control and have embedded it in a Web services framework that is Web-based and scalable; in other words, a fully compliant Web-oriented architecture. These are the conceptual foundations to **structWSF**.

To be sure, **structWSF** in its current alpha release is quite raw in many areas and incomplete in others. But we will continue to work on it -- and invite your participation to do the same -- such that it can fulfill its destiny as a data federation framework for the Web.

---

[1] I first wrote about this while at BrightPlanet; a page is still up on that Web site with the text above. I have re-caste this material in various ways since.

[2] I have previously written on the "40 sources" of data heterogeneity. See here, for example.

[3] See http://ontolog.cim3.net/forum/ontolog-forum/2009-06/msg00210.html and continue to follow the noted thread.

[4] See the thread, ' .htaccess a major bottleneck to Semantic Web adoption,' at http://lists.w3.org/Archives/Public/public-lod/2009Jun/0341.html and continue to follow this thread.

[5] See http://microformats.org/discuss/mail/microformats-discuss/2009-June/012985.html and continue to follow the 'mixing vocabularies' thread.

[6] This is our working definition of the ABox and TBox in specific reference to description logics:

"Description logics and their semantics traditionally split *concepts* and their relationships from the different treatment of *instances* and their attributes and roles, expressed as fact assertions. The concept split is known as the TBox (for *terminological* knowledge, the basis for *T* in *TBox*) and represents the schema or taxonomy of the domain at hand. The TBox is the structural and intensional component of conceptual relationships. The second split of instances is known as the ABox (for *assertions*, the basis for *A* in *ABox*) and describes the attributes of instances (and individuals), the roles between instances, and other assertions about instances regarding their class membership with the TBox concepts."

[7] For functionality, download, documentation or other direct materials on **structWSF**, please see OpenStructs.org and its related resources. There is also a Drupal instantiation of the system called conStruct, also available for download.
