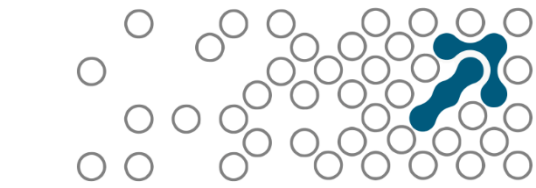
ONTOLOGIES AND APPLICATION

By: Dave McComb



semantic arts

Outlining three and a half ways that applications can have their schemas derived from an enterprise ontology…

Many people (ok, a few people) have asked us: "what is the relationship between an ontology and an application?"

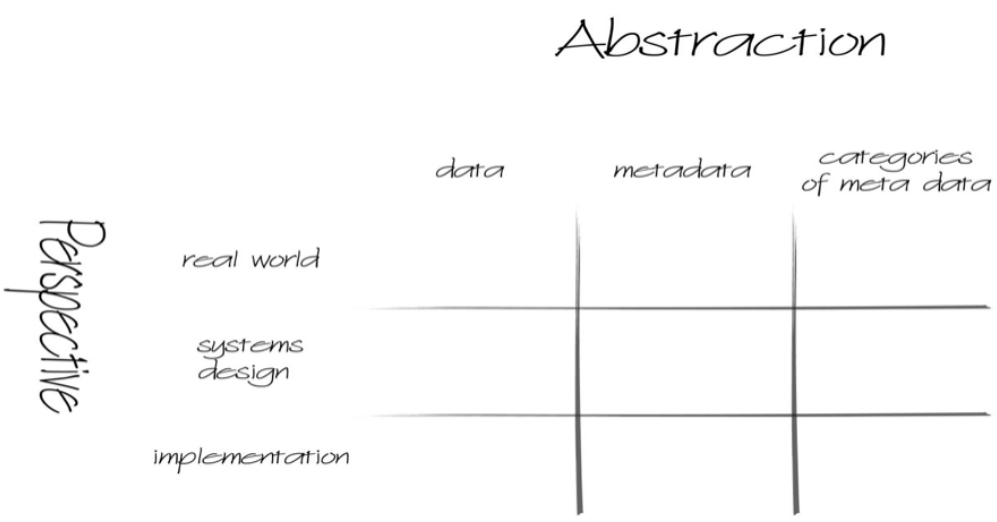
We usually say, "That's an excellent question" (this is partly because it is, and partly because these 'people' are invariably our clients).

Having avoided answering this for all this time we finally feel motivated to actually answer the question.

It seems that there are three (ok three and a half) ways that ontologies are or can be related to applications. They are:

- Inspiration
- Transformation
- Extension

But, I fail to digress… Let's go back to the 'tic tac toe' board. We call the following a 'tic tac toe' board, because it looks like one:



semantic arts

What it is attempting to convey is that there are levels of abstraction and differences in perspective that we should consider when we are modeling. An application is in the lower middle cell.

Illustration

Data models are in the middle square. Ontologies could be anywhere. An ontology is a formal way of representing a model. And so we could have an ontology that describes an application, an ontology of a logical model, even ontologies of data or meta meta data.

In our opinion the most interesting ontologies are in the middle top: these are ontologies that represent concepts independent of their implementation. This is where we find upper ontologies as well as enterprise ontologies.

Now some companies have built enterprise wide conceptual models. The IRS has one, with 30,000 attributes. But all the ones we've seen are not actually in the top center cell, they are logical models of quite wide scope. Ambitious and interesting, but not really conceptual models and typically far more complex than is useful.

What we've found (and written about in other articles (ref the Elegance article)) is that a conceptual model can cover the same ground as a logical model with a small percentage of the total number of concepts. Not only are there fewer concepts in total, there are few concepts that need to be accepted and agreed to.

In a traditional model, the reviewer of the model has to learn, and agree to, all the concepts in the model in order to use the model. With a well-designed ontology, the consumer of the ontology learns a much smaller number of concepts and the rest of the concepts are defined by recombining concepts that have already been learned.

So if an upper level ontology is conceptual, and really focused on what things mean, how might it relate to an implemented ontology?

As we said earlier, there are three (and a half) ways:

- Inspiration
- Transformation
- Extension

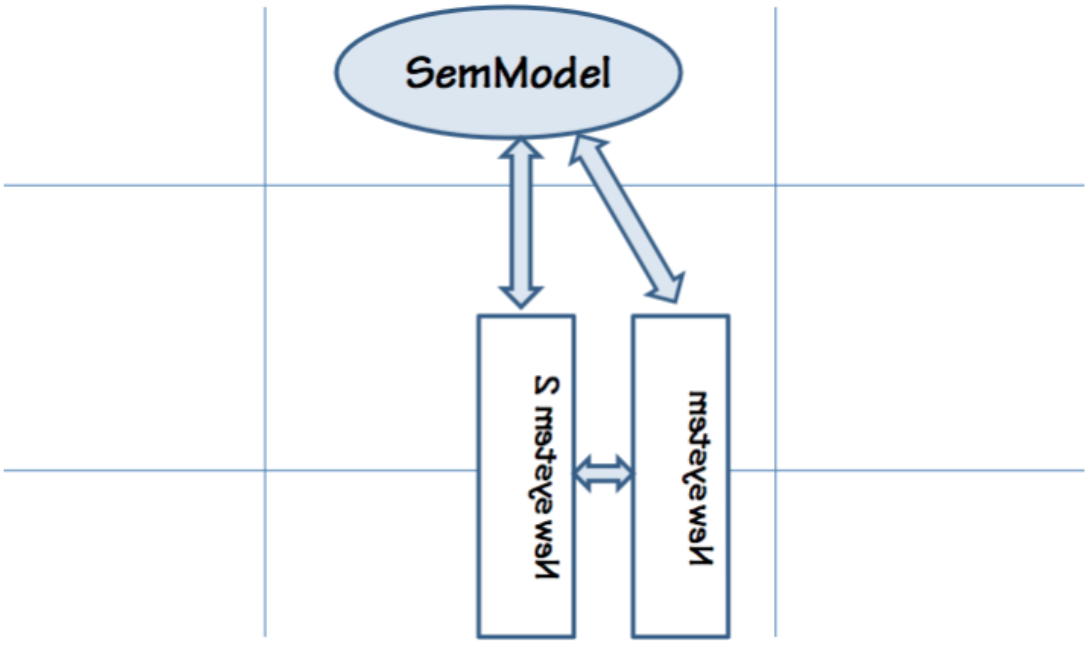
Inspiration

Normally when a designer designs an application they come up with the concepts from the requirements or from their experience. It typically doesn't occur to them that they are recreating or overlapping with concepts that have been designed and developed elsewhere in the enterprise.

Of course it does occur much later to the systems integrator who have to resolve all these similar, but not quite identical concepts.

But if you have an enterprise ontology, this can serve as the basis for the designer's model. When they have an urge to create a 'client' table they might notice that the 'customer' concept in the enterprise ontology is virtually identical to the concept they were going to implement (and if it is and needs to be slightly different there are ways to accommodate that).

If two designers create models and applications based on the same ontology, they will have implementations that should be considerably easier to integrate than the non-ontology driven approach.



The other reason we call this inspirational, is that every place we have done this, people were actually surprised that it was possible, understandable and relatively concise.

Most enterprises have given up on the idea of having an enterprise model. Building one with traditional technology, and building one from the bottom up, ends up being a huge effort that often couldn't even keep up with the changing application landscape. Having given up on the idea, they are surprised that a model can be built in less than 6 months that can be the basis for future systems development.

We believe that even at this level the enterprise ontology is typically worth many millions of dollars for most large enterprises based on effort savings in application development, elegance (future systems have smaller logical models) and integration economy.

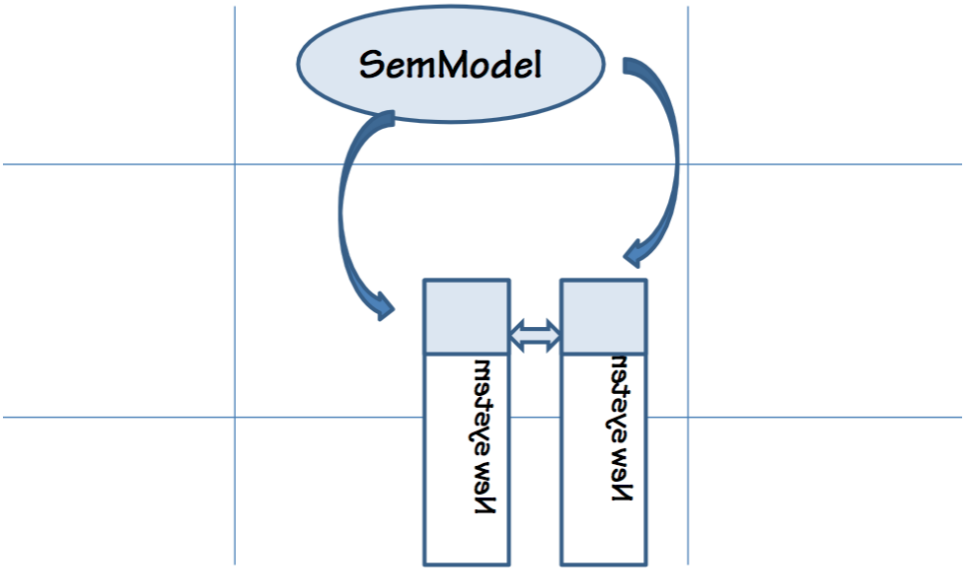
But this is just the first stage in potential benefits.

2a. Transformation

Deriving Logical and Physical Models

It is possible, as we've demonstrated for some of our clients, to automatically generate the logical models of your future systems, and with existing technology it is relatively straightforward to generate the physical models as well.

The advantage of this approach is that it cuts down on the human design time, speeding up development and reducing errors. By generating the logical models it forces the ontologists to work with the designers to make additions to the ontology if needed to accommodate the logical model. This ensures that the ontology continues to cover all, or at least most, of the concepts being implemented.



This makes future integration easier yet. Also the enterprise model can be used as a schema for federated querying, which is made far easier when the member applications have a schema that was derived from the shared schema.

But you don't always have the luxury of designing your own apps. You have legacy apps. And you have COTS apps. And you have legacy COTS apps.

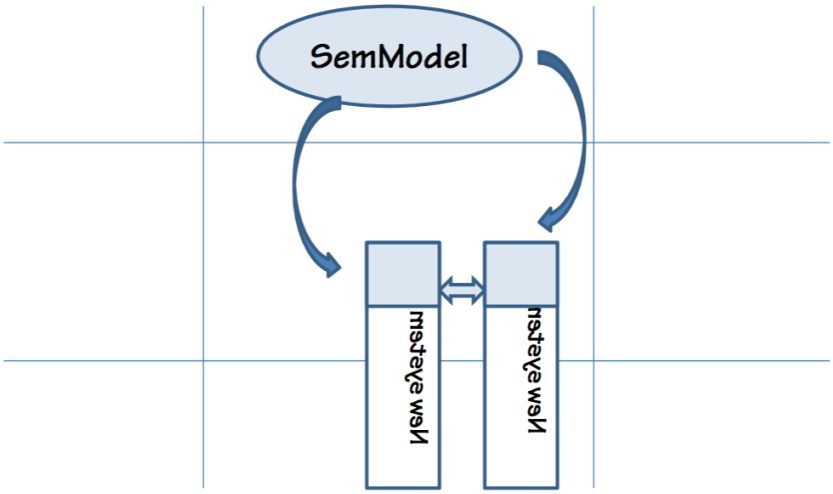
That's where the next transform approach comes in.

2b. Transformation

Deriving your Canonical Message Model

Most large organizations have an SOA architecture in place. Very few are actually reusing messages or achieving much service reuse. Why is this?

The main reason is that in most organizations individual applications and services are allowed to publish their interface to the bus for consumption by others. Invariably they create messages using terms and structures from their own internal schemas and APIs. As a result they export their complexity onto the rest of the organization, and any other service or application that might have had a similar requirement will publish another different message and the opportunity will have been lost.



With the ontology driven approach, we can generate what is called a 'canonical message model' from the ontology. The canonical message model is analogous to a logical model, but rather than being the template for a database it is the basis of the shared SOA messages.

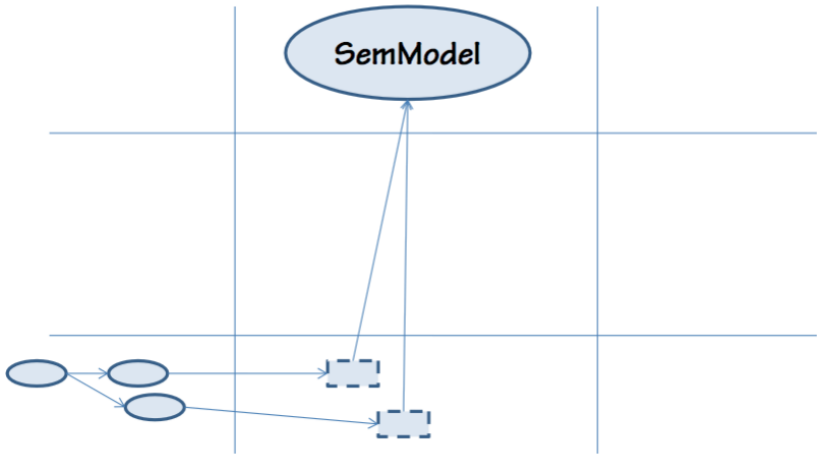
From the message model, individual SOA messages are defined which share terms, characteristics and structure with all other messages derived from the same model. And because the model was built from the concepts of the business rather than the accident of implementation, the incidence of unintentional overlap is very slight.

Extension

Finally, for some applications the ontology can be used directly. All concepts in an ontology are identified with URIs (Uniform Resource Identifiers) which can, and generally should, also be designed to double as URLs. The advantage of having the identifiers double as URLs is primarily for human understanding and consumption.

But because the concepts are URIs, and because data in a semantic system is expressed as 'triples' of URIs (the subject/predicate/object triple of semantic based systems is always and only either a uri/uri/uri or uri/uri/literal).

Because of this systems can be built where the instance data (in the lower left hand corner) is expressed as uri triples where the uris were defined in the enterprise ontology.



Now it is possible that the concepts have been cached, or that there is a subset or even that they have been augmented or translated, as suggested in the dotted boxes, but strictly speaking this isn't necessary.

The actual data in the system can be expressed as assertions based on uris from the enterprise ontology.

Of course the running system will generate new URIs for each new instance it creates, but any shared concepts, and therefore any analog to what would have been a schema in a traditional system, can be directly implemented from the enterprise ontology.

Wrap up

The models on which applications are built came from somewhere. Traditionally they came from the requirements as described to the analyst working on the application. But that sort of derivation inevitably leads to applications that have arbitrarily different schemas and therefore end up being quite expensive to integrate and result in fragile architectures.

Taking a different approach, building out application schema definition from a shared model, results in applications that are cheaper to build and much cheaper to integrate.

We've outlined three and a half ways that applications can have their schemas derived from an enterprise ontology. None of these is necessarily 'better' than the others, they just show differing degrees of commitment to the ontology. All of them result in greater productivity and greater integration.

11 Old Town Square
Suite 200
Fort Collins, CO 80524

970-490-2224
305-425-2224
info@semanticarts.com