A BPTrends Column Updated Feb. 2013

# MDA Journal

**David S. Frankel**
David Frankel Consulting

david@dFrankelConsulting.com
http://www.linkedin.com/in/davidsfrankel

Author:
Model Driven Architecture:
Applying MDA to Enterprise Computing

# Semantic Interoperability for SOA



In my last MDA Journal Column[1] I mentioned that a new organization named BIAN (www.bian.org) is defining standards for service-oriented architecture in the banking industry and is using techniques to promote semantic interoperability that I've written about in previous MDA Journal Columns.[2] In this month's Column, I explain how BIAN is using these techniques, which were designed for data modeling, and is extrapolating them so as to apply them also to service modeling.

## Semantic Interoperability Recap

In previous MDA Journal Columns I've written extensively about how integration costs in enterprise computing continue to stress resources, and in discussing how to mitigate the costs I've described techniques that enhance semantic interoperability.

In these writings I've defined semantic interoperability as the ability of multiple parties to coordinate their functioning based on a shared understanding of the meaning of the data that flows among them. This kind of interoperability is distinct from syntactic interoperability, which is based on shared understanding of the syntax of the data that flows among them. It's the

[1] "Banking Industry Architecture Network (BIAN)," BPTrends, June, 2011.

[2] Such as "Semantic Metadata: Unlocking the Potential of Semantic Ontologies," originally published as an MDA Journal Column in BPTrends in February 2010, updated version available at http://www.dfrankelconsulting.com/Articles.html with revised title "Semantic Metadata for Data Integration: Making Semantic Vocabularies and Ontologies Actionable."

difference between understanding that an element of data is a string of ten characters starting with a letter of the alphabet (such as A1343CU77Q) and understanding that an element of data is an identifier for a premier customer. Data modeling tools have long been strong in formally specifying data syntax, but less so regarding the formal specification of data's meaning.

Low levels of semantic interoperability – in other words misunderstandings as to the meaning of data – are a key factor in the persistence of high integration costs.

Service-Oriented Architecture (SOA), however, is not simply about data. Therefore, my definition of semantic interoperability needs to be extended. Before discussing the extension, it's useful to think about what is special about SOA.

## The Emergent Properties of a Service

SOA deals with the concept of services, which have contractual behavior that can be invoked by a service consumer. SOA services receive data as input and return data as output, so modeling of that data is still relevant, but a service is about more than the input of data by a consumer and the output of data to the consumer.

For example, consider a service that transfers funds from one account to another. Its input parameter is a data structure that contains the source and destination account numbers and the monetary amount of the transfer. If the specification of the service simply specified the input and output parameters, it would be inadequate, because it wouldn't be clear whether the money was supposed to be added to both accounts, subtracted from both accounts, or transferred from one account to the other. Clearly we have more to specify. We need to know the behavioral contract for the service – what is it supposed to do?

These behavioral properties are more than the sum of the properties of the inputs and outputs. Information scientists often refer to properties that are more than the sum of the properties of the parts as *emergent properties.*

## Modeling Service Behavior via Postconditions

One technique for specifying the behavioral properties of a service in a disciplined manner is to stipulate the *postconditions*, which are conditions that must be satisfied when the service is finished executing. In the case of the funds transfer we can informally state two postconditions as follows:

1. *The balance of the first input account is decreased by the input amount.*
2. *The balance of the second input account is increased by the input amount.*

These two postconditions, taken together, tell us something meaningful about the semantics of the service's behavioral contract. Some practitioners refer to postconditions as *effects* of the service.

The names of input and output parameters can suggest the intended behavior, as in this funds transfer example where the two input parameters might be named Source Account and Destination Account. However, parameter naming is not always clarifying in this regard; for example, for a more narrowly defined service the names of the two input parameters might be Savings Account and Checking Account, which would not make clear whether the funds are to be transferred from savings to checking or from checking to savings.

More formal modeling of postconditions using machine-readable constraint languages is possible if we formally model the input data and output data. I've practiced and written[3] about this formal technique for quite a few years. I always make sure to state the postconditions informally in English, even if I'm specifying them via a constraint language.

[3] See, for example, "Scaling the Business Platform Up," BPTrends, December 2005, section entitled "Semantically Rich Service Contracts," pages 4-6,

## Preconditions

Preconditions, which must be satisfied when the service is invoked, also are useful for capturing the semantics of a service. If the preconditions are not satisfied, then the service provider is not obligated to execute the service and satisfy the postconditions.

An example of informally stated preconditions for the funds transfer service are as follows:

1. *The amount of the transfer must be less than or equal to the balance of the first input account.*
2. *The first and second input accounts must belong to the same customer.*

Postconditions tell us something about the scope of the service's behavior. The first postcondition declares that the service does not support overdrawing. The second declares that the service does not support transferring funds from one customer to another. A different funds transfer service might not have these restrictions; for example there could be a service, probably accompanied by a higher level of security safeguarding, that could allow inter-customer transfer, in which case the second precondition would not be declared for that service.

Preconditions imply an obligation on the part of the consumer to ensure that the conditions are satisfied before invoking the service. In practice, the consumer might not check the preconditions, being content to let the service provider reject the request for execution if the preconditions are not satisfied; however, that is an implementation detail.

## Semantic Interoperability of Services

In this light, we can define semantic interoperability of services as the ability of multiple parties to coordinate their functioning based on a shared understanding of their contractual obligations as providers and consumers of services. Semantic interoperability of the data that is input to and output by services is a necessary but not sufficient condition for semantic interoperability of services.

## Semantic Grammar

There are techniques for enhancing semantic interoperability of data, which I have highlighted in previous Columns and are in practice in industry today to some extent, that can be extended to cover semantic interoperability of services. These techniques compliment the specifying of preconditions and postconditions,

In my recent writings[4], I have discussed the semantic grammar of a data element in some detail, as defined by the ISO 11179 standard and standards derived from it such as UN/CEFACT Core Components (CCTS). For SOA, we can extrapolate this approach by defining a semantic grammar for a service.

### Recap: The Semantic Grammar of a Data Element

In this section I review the basics of semantic grammar for data elements, which are treated in more detail in my previous Columns that I've cited.

The semantic grammar of a data element can be summarized as follows:

*<object class concept><property concept><representation concept>*[5]

[4] Op. cit. BPTrends, February 2010, pages 3-4

[5] UN/CEFACT has defined 22 representation concepts, which are essentially very high-level data types. For example, saying that the representation concept for a data element is Quantity simply makes a semantic statement about the data element, without committing to the technical data type used to represent the quantity in computer systems.

where each concept can be qualified by additional concepts. For example, consider the following data elements defined in the XBRL reporting format used to file electronic financial statements with the US Securities and Exchange Commission in accordance with the US Generally Accepted Accounting Principles (US GAAP):

- CommonStockParOrStatedValuePerShare
- CommonStockSharesAuthorized
- CommonStockSharesIssued

We use the semantic grammar of a data element to represent the semantic structure of these three elements, as follows:

CommonStockParOrStatedValuePerShare

Object Class Concept: *Stock*

Object Class Qualifier Concept: *Common*

Property Concept: *Value per Share*

Property Qualifier Concept*: Par or Stated*

Representation Concept: *Ratio*

CommonStockSharesAuthorized

Object Class Concept: *Stock*

Object Class Qualifier Concept: *Common*

Property Concept: *Shares*

Property Qualifier Concept*: Authorized*

Representation Concept: *Quantity*

CommonStockSharesIssued

Object Class Concept: *Stock*

Object Class Qualifier Concept: *Common*

Property Concept: *Shares*

Property Qualifier Concept*: Issued*

Representation Concept: *Quantity*

I use the term *semantic metadata* to refer to the elements that describe the semantic structure of a data element. Semantic metadata is distinct from traditional metadata that describes the syntax of the data but says little about its meaning. Semantic metadata encodes some of the meaning of an element in a manner that allows tools to provide assistance to humans who are trying to understand data in order to use it and integrate it.

In order to get the best results, the semantic metadata should work in concert with a controlled vocabulary, which defines the concepts referred to in the semantic metadata. For example, the metadata element that says that Value per Share is the property concept for CommonStockParOrStatedValuePerShare should point directly to a definition of Value per Share sitting in a vocabulary, to reduce the possibility of misunderstandings. Referencing a vocabulary entry is also useful when the same term is used to represent somewhat different or entirely different concepts. There can be more than one vocabulary entry for “coupon,” for example,

which has different meanings in different segments of industry, and the semantic metadata should point to the vocabulary entry that has the correct meaning for the context at hand.[6]

Tools that exploit semantic metadata that is based on the semantic grammar outlined here are still in early stages of development, but, as semantic technology is now emerging from the laboratory and academia into industry, we will see more in the future.

### Semantic Grammar of a Service

BIAN, the banking standards group I mentioned, has defined the following semantic grammar for service operations:

*<action concept><object concept>[property concept]*

where each concept can be qualified by additional concepts. Note the addition of an action concept, which is not part of the semantic grammar of a data element. Note also that the square brackets indicate that the property concept is optional.

Here are some examples of how we break down service operations into their semantic structure, using this semantic grammar:

CreateSalesOrder

Action Concept: *Create*

Object Class Concept: *Sales Order*[7]

SendSalesOrderAcknowledgement

Action Concept: *Send*

Object Class Concept: *Sales Order*

Property Concept: *Acknowledgement*

These two examples are quite straightforward. However, that is not always the case. While going through the process of doing such semantic structuring for some BIAN services currently in the process of being defined, I ran across a service operation named CreateNewCurrentAccountInventory. The English description of the operation did not make clear to me whether the intent is for the service operation 1) to create inventory for a new account or 2) to create new inventory for an established account. This is the sort of subtle ambiguity that leads to integration bugs that result in costly errors and can be difficult to track down.

[6] Controlled vocabularies can be standalone, or can be part of a more complex ontology of concepts. In either case, there are advantages to defining vocabularies using tools that are based on Semantic Web standards, not the least of which is that they can leverage other vocabularies already in existence that are based on those standards.

[7] Note that the term for an atomic concept can consist of more than one word.

In the first case, the semantic structure would be as follows:

CreateNewCurrentAccountInventory

Action Concept: *Create*

Object Class Concept: *Current Account*[8]

Object Class Qualifier: *New*

*Property Concept: Inventory*

In the second case, the semantic structure would be as follows:

CreateNewCurrentAccountInventory

Action Concept: *Create*

Object Class Concept: *Current Account*

Property Concept: *Inventory*

Property Qualifier Concept: *New*

The key is whether New qualifies Current Account or Inventory. The process of modeling the structure of this service led me to spot an ambiguity and consult the service designer to get clarification. Capturing the correct semantic structure in metadata improves the ability of tools to help people avoid misinterpretation and thus improves semantic interoperability.

This benefit is not unique to service modeling. Semantically structuring data elements can catch and clarify similar ambiguities.

## Graphically Modeling Semantic Structure

BIAN is developing a technique for modeling semantic metadata with graphical modeling tools, based on a UML Profile that BIAN is defining.

I'll use the CreateNewCurrentAccountInventory example to illustrate, creating a graphical model for each of the two possible semantic interpretations of this service operation. Figure 2 models the interpretation that the meaning is to create inventory for a new current account, while Figure 1 models the interpretation that the meaning is to create new inventory for an established current account. Again, the difference lies in the fact that in the first case New qualifies the object class concept Current Account, while in the second case New qualifies the property concept Inventory, but here we capture that graphically.

[8] WordNet, the public domain project of Princeton University, has two entries for Current Account in its vocabulary: 1) *that part of the balance of payments recording a nation's exports and imports of goods and services and transfer payments* 2) *a bank account against which the depositor can draw checks that are payable on demand.* The second entry is the one to which this semantic metadata should point.

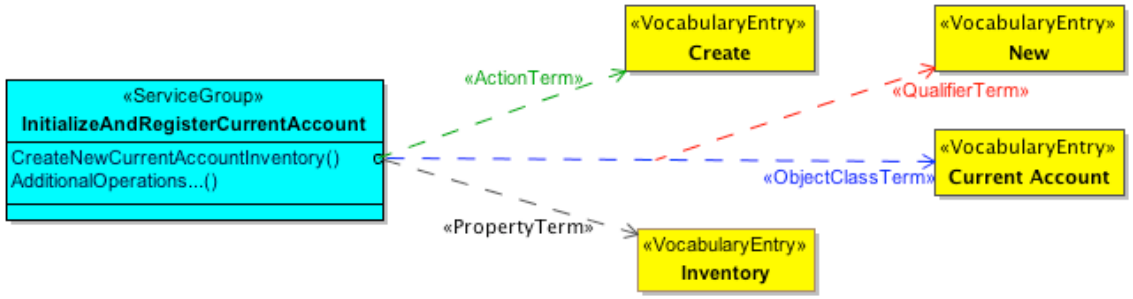


**Figure 1: Create Inventory for a New Current Account**

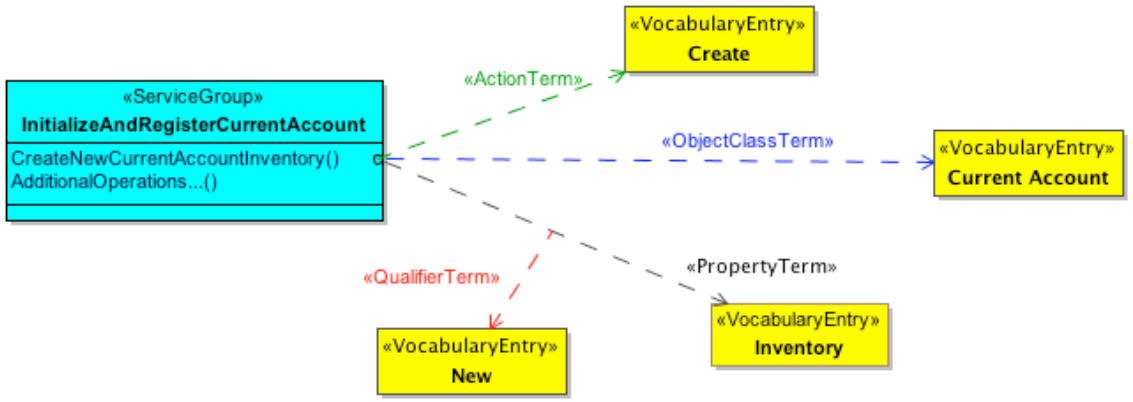


**Figure 2: Create New Inventory for a Current Account**

Readers who attended primary school prior to the 1970s may find this technique of visual diagramming according to a semantic grammar to be strongly reminiscent of sentence diagramming, a tool for training students in English grammar that has regrettably fallen into disuse. Just as the visual technique demonstrated above is driven by the semantic grammar we've established for service operations, sentence diagramming is driven by the structure of English grammar.

The Reed-Kellogg sentence diagramming technique has a basic structure as shown in Figure 3. Figure 4 uses this structure to diagram the sentence "Parents give children food." The technique has additional nuances for more complex sentences.

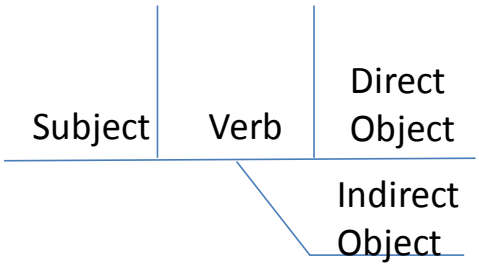


**Figure 3: Basic Structure of a Sentence Diagram**

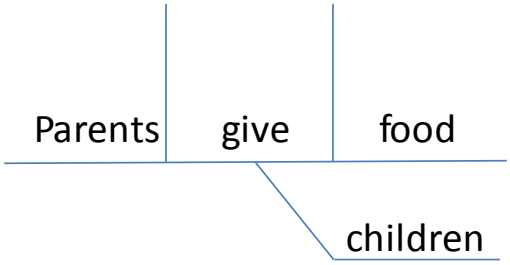


**Figure 4: Diagram of "Parents give children food."**

The Reed-Kellogg sentence diagramming system has been criticized by some linguists because it does not convey the order in which the words appear in a sentence. A defense of the Reed-Kellogg system, excerpted from a book they wrote, is illuminating:

> *The Objections to the Diagram.*--The fact that the pictorial diagram groups the parts of a sentence according to their offices and relations, and not in the order of speech, has been spoken of as a fault. It is, on the contrary, a merit, for it teaches the pupil to look through the literary order and discover the logical order. He thus learns what the literary order really is, and sees that this may be varied indefinitely, so long as the logical relations are kept clear.
>
> The assertion that correct diagrams can be made mechanically is not borne out by the facts. It is easier to avoid precision in oral analysis than in written. The diagram drives the pupil to a most searching examination of the sentence, brings him face to face with every difficulty, and compels a decision on every point.[9]

These words, written in 1877 when there was no notion of digital computing, stand the test of time, and apply very well to the intent behind modeling semantic structure according to a well-defined semantic grammar. Although we can use the semantic structure of service operations and data elements to drive how we name them, the name is not the key – the meaning is the key. Furthermore, we need to retrospectively model the semantic structure of many service operations and data elements that have already been named, in which case any such naming convention is moot.

[9] Alonzo Reed and Brainerd Kellogg, *Higher Lessons in English*, first published in 1877, as quoted in Wikipedia at http://en.wikipedia.org/wiki/Sentence_diagram

Capturing the semantic structure allows software to examine systems and, for example, reveal all the instances where the notion of New Account appears, thereby enhancing the ability of humans and machines to manage the systems. It reduces the likelihood of mistakenly including instances where, for example, the intended meaning is New Inventory, not New Account.

## Will it Scale?

Accurately capturing semantic structure can be laborious, whether using the graphical or non-graphical techniques. Industrial systems have large numbers of data and service elements. From a pragmatic standpoint, a first pass at applying these techniques may require us to focus on capturing the semantic structure only of elements where the likelihood of misinterpretation is high.

When designing new data and services, these techniques can pinpoint ambiguities and trigger a refactoring of designs in order to remove ambiguities. For example, after I pointed out the ambiguity in CreateNewCurrentAccountInventory's name and textual description, the service designer updated the name and description to further minimize the possibility of misinterpretation.

Rather large-scale uses of ISO 11179 / CCTS semantic structuring of data elements are in progress in several industrial sectors including manufacturing and retail / consumer products. These initiatives use spreadsheets for capturing the semantic structure. The spreadsheet's tabular form is certainly more compact than the visual technique, although it may not always be as straightforward for a human to grasp, particularly in complex cases where there are multiple qualifiers.

We're still learning, honing our approach as we practice it, and will be doing so well into the future, but the techniques have progressed to the point where we are ready to derive real value from them.

*David Frankel has over 30 years of experience in the software industry as a technical strategist, architect, and programmer. He is recognized as a pioneer and international authority on the subject of model-driven systems and semantic data modeling. He has published two books and dozens of trade press articles, and has co-authored a number of industry standards including XBRL, ISO 20022, BIAN, and UML®.*

## WordNet® Copyright

Footnote number 8 quotes definitions from WordNet, the online dictionary project of Princeton University. WordNet is freely available for this kind of reuse, as long as it is accompanied by a required copyright notice (see http://wordnet.princeton.edu/wordnet/license/). Here is that copyright notice:

RIGHTS.

WordNet is a registered trademark of Princeton University.