semantic arts

Telecom Framework Model: Simplified with gist

Phil Blackwood, Semantic Arts (2021)

We recently recast large portions of the telecom Frameworkx Information Model into an Enterprise Ontology using patterns and reusable parts of the gist upper ontology. We found that extending gist with the information content of the Frameworx model yields a simple telecom model that is easy to manage, federate, and extend, as described below. Realizing accelerating time to market along with simplifying for cognitive consumption being typical barriers for success within the telecom industry, we're certain this will help overcome a few hurdles to expediting adoption.

The telecommunications industry has made a substantial investment to define the Frameworx Information Model (TMF SID), an Enterprise-wide information model commonly implemented in a relational data base, as described in the GB922 User's Guide.

Almost half of the GB922 User's Guide is dedicated to discussing how to translate the Information Model to a Logical Model and then translate the Logical Model to a Physical Model. With gist and our semantic knowledge graph approach, these transformations were no longer required. The simple semantic model and the data itself are linked together and co-exist in a triple-store data base without requiring transformations.

The gist upper ontology is the result of many years of implementing knowledge graphs for Enterprises; using gist as our starting point gave us concepts with clear non-overlapping semantics and also provided a compact set of standard Object Properties (types of relationships) that was especially helpful. As an upper ontology, gist is well suited to be the starting point for creating an Enterprise knowledge graph.

Our telecom knowledge graph is based on the W3C Semantic Web standards, which are designed to operate at web scale and are a good fit for large networking and communications enterprises. In fact, many of the world's largest companies are already using knowledge graphs.

We found that the inherent simplicity of the semantic approach is directly applicable to the Frameworx Information Model in the following ways:

- we don't need to maintain 3 types of models (conceptual, logical, and physical)
- the data and the model are integrated, and they can be interpreted programmatically
- the data in the knowledge graph is self-describing and discoverable
- types of relationships are highly re-usable

Furthermore, the semantic approach has the major benefit of interoperability.

Some key aspects of the W3C Semantic Web standards that we applied include:

- every Class and every Property in our knowledge graph has a globally unique identifier
- the Web Ontology Language (OWL) was used to define the model and the data
- constraints are defined using the SHACL constraint language
- queries are written in the SPARQL query language
- federation with existing relational data is done using the R2RML data mapping language
- an open-source reasoner automatically makes inferences and helps detect errors

In OWL, the basic unit of capturing information is a triple of the form:

subject – predicate – object

This one simple structure for expressing all data is significantly simpler than the UML representation of the Frameworx Information Model. For example:

- the name of a predicate is typically just a verb, e.g. “conformsTo”
- a given predicate may be used multiple times with the same subject
- a Class can be thought of as a set, with subclasses as subsets
- a Class does not have pre-defined attributes or properties

Let’s explore how the patterns of the Frameworx Information Model became simpler as a result of using gist (refer to the GB922 User’s Guide for the UML representations of these patterns). The Frameworx Information Model includes the following patterns:

1. Atomic/Composite
2. Classification Groups
3. Entity/Entity Role
4. Entity/Entity Specification
5. Characteristic Specification/Characteristic Value
6. Resource/Service/Product

We will examine each of these …

1- The Atomic/Composite pattern:

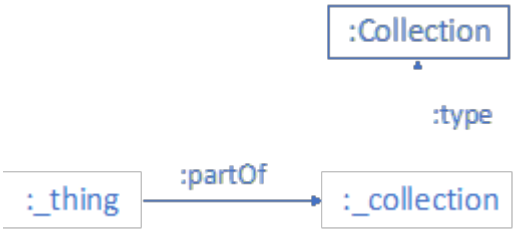
In this Frameworx pattern, an individual of a Class is composed of other individuals of the same Class. Below is an example of how we implemented this pattern:

In the diagram:

- :Agreement is a Class
- :_agreement1 and :_agreement2 are individual instances of Agreement
- a colon indicates the item belongs to some namespace
- an underscore is a convention to show the item is an instance of a Class

2- The Classification Group pattern:

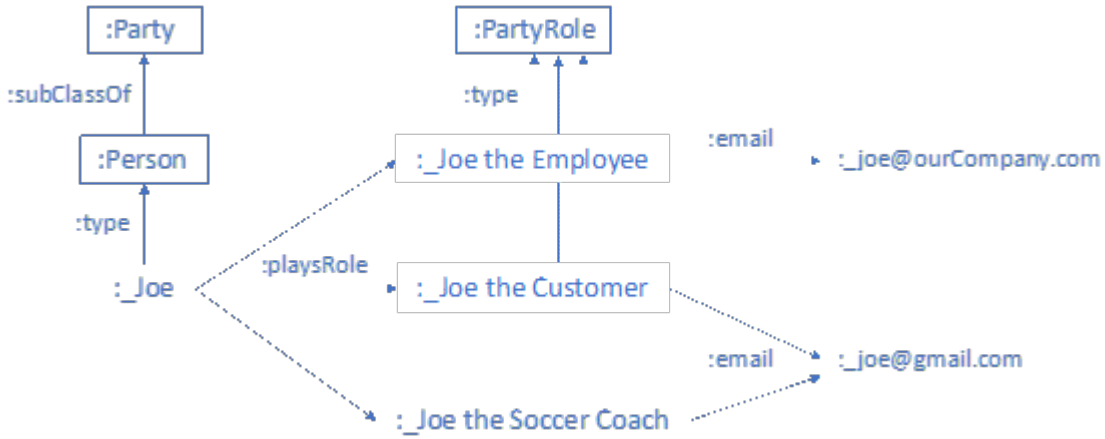
A Frameworx classification group can be used for statistics or to apply some action to all members of the group. We used the term “collection” instead of “group” (which might be misinterpreted as a group of people).



3- The Entity/Entity Role pattern:

In this Frameworx pattern an instance can play multiple roles, and the way the instance is related to other instances may depend on which role it is playing. A network element may play a role in the network (e.g. as an edge router, core router, or LAN router), while a person may play a role in an event or an agreement.

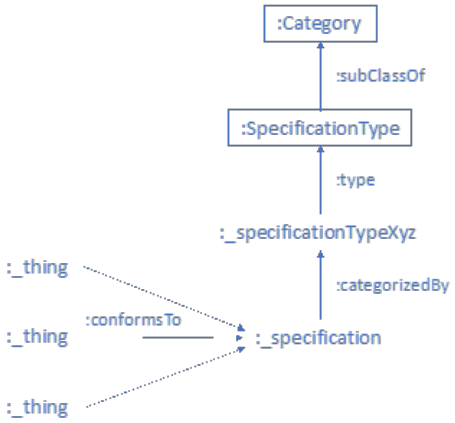
In our example below, Joe has several email addresses and the one we use to contact Joe depends on his role.



The diagram is a direct translation of the Frameworx pattern to a knowledge graph. In some cases, we found it was simpler to capture the role as part of a relationship, which is a pattern commonly used in knowledge graphs.

4- The Entity/Entity Specification pattern:

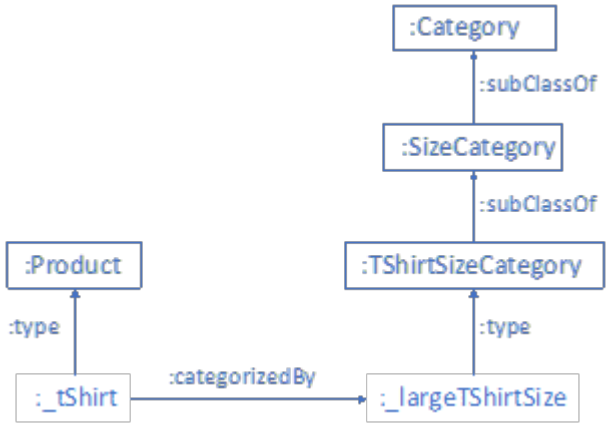
With this Frameworx pattern, attribute values that are common to all instances of the entity are captured in a specification.



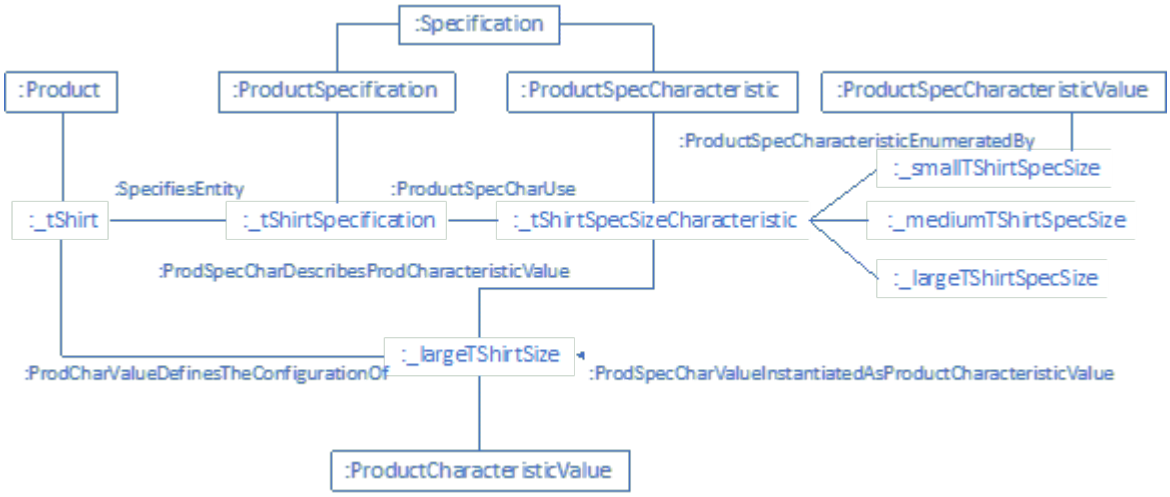
For example, identifying a fiber in a cable is always done the same way for a given type of cable. We found that when an attribute can take on multiple values, however, it was simpler to use a category than a specification, as seen in the next example.

5- The Characteristic Specification/Characteristic Value pattern:

This Frameworx pattern is used to extend the model by adding attributes. We found it useful for modeling any attribute that can take on a fixed set of values. Below is a simple example of sizes (small, medium, or large) to illustrate how we implemented the pattern in a knowledge graph using categories:



By contrast, the Frameworx pattern is more complex; the same information in Frameworx looks like this:



Our knowledge graph implementation is simpler to understand and manage, and it satisfies the original intent of the Frameworx pattern, which is to support extensions of the model as new attributes are discovered.

6- The Resource/Service/Product pattern:

This Frameworx pattern models the way products are constructed from the raw materials represented by Resources, with Services as an intermediary. The diagram below shows how we implemented the pattern in a knowledge graph:

The six patterns above illustrate some of the simplifying power of semantics we observed while implementing the Frameworx Information Model in a knowledge graph.

Refactoring the Frameworx Information Model using semantics drops the complexity by nearly an order of magnitude. If you are already using the Frameworx Information Model (TMF SID), this will make it easier for consumers to understand and map into the many distinctions of the telecom domain. If you have not yet adopted the Frameworx Information Model, you might consider this an easier on-ramp.

Semantic Arts has been helping clients bring semantic technology and knowledge graphs to their enterprise applications and enterprise architecture for over twenty years. Five years ago, we crystallized an approach we call the “data-centric approach,” and we continue to refine our methodology and tools to make this approach predictable and repeatable.