Infotech@Aerospace 2011
29 - 31 March 2011, St. Louis, Missouri

AIAA 2011-1501

Using RDF Schema and Taxonomies to Describe a Flexible, Standards Based SPA Ontology

Scott S. Pasko[1]
Solar Digital, Inc., Escondido, CA, 92026

The Space Plug-and-play Avionics (SPA) standards continue to be developed by the Air Force Research Laboratory to enable the rapid design, assembly, and integration of spacecraft. The focus of this research is on finding an ideal way to define a formal ontology that will enable the dynamic system interoperability required by the SPA program, as well as simplify the adoption of these approaches for industry participants that will build SPA-compatible components and flight software.

I. Introduction

AN ontology is a set of definitions that describe a shared vocabulary of terms and their related meanings. In current implementations of the SPA standard, the Common Data Dictionary (CDD) defines a somewhat informal ontology. The CDD consists of mostly one-word terms in a handful of categories described rather informally. When using the CDD to find specific types of resources on a satellite bus, you must combine these general terms together to try and create a qualifying expression specific enough to identify an appropriate system resource with some degree of certainty.

As an alternative, we have implemented an ontology that represents a complete logical model of the satellite bus and its related resources. We use domain names derived from formalized taxonomies to describe and resolve resources in a simpler, more reliable, and more deterministic fashion than previous approaches. The use of taxonomies introduces a greater formalism for describing collections of related terms, as each taxonomy is a hierarchical classification of related concepts. Organizing these concepts as terms within a taxonomy requires designing meaningful categories that subdivide the problem space into increasingly specialized terms. A collection of such definitions and taxonomies represents an ontology, and gives us a complete logical model of a system and its resources.

President/CEO, AIAA Member

II. Developing an Ontology

In our model a network is simply a collection of nodes, where nodes represent resources that can be utilized by one another. To dynamically configure a system, we need a reliable way for one node to locate a resource that another node provides. To locate a resource, we must be able to provide a description of the resource we are interested in. This requires us to have a reliable and consistent way to describe well known resources. The Resource Description Framework (RDF) is one way to formally describe resources. This paper focuses on how we use RDF and its related schema to describe well known resources on a satellite bus.

A. RDF Schema (RDFS)

A lot of useful work and standardization has occurred under the effort collectively known as the Semantic Web overseen by the World Wide Web Consortium (WC3). Although there is no consensus that the ambitious goals of the Semantic Web will ultimately be realized, there is much value in this body of work that can be applied to various problems in many industries, including realizing the goals of SPA. The Resource Description Framework (RDF) and the related RDF Schema (RDFS) are of particular interest to our research. We believe this work is directly relevant to SPA as well as any other domain where dynamic discovery and configuration is of value.

We use RDFS to formally describe our ontology using a collection of taxonomies. RDFS extends the RDF vocabulary to describe taxonomies of classes and properties. Using RDFS to describe our ontology results in a formal machine-processable set of descriptions. These formal descriptions not only define our ontology, but they can also be used to create tools that let us visualize and evolve the ontology, as well as help generate device descriptions that properly use and reflect it.

B. Ontology - Logical versus Physical

The satellite bus and its related components represent a physical view of the system. We can view this entire system collectively as a black box that simply provides access to resources. To locate any particular resource, we need reliable "keys" to query this system, where the keys represent standardized resource descriptions. Our ontology represents a logical view of the system, and the keys we need are derived from the related taxonomies as pathnames. Discovering and configuring a system based solely on the logical view gives us the dynamic bindings we need so that the physical system can change without effecting the logical nature of the system.

To properly describe an ontology for any domain of interest, we must abstract the problem into a set of concepts and terms that can be described systematically. Although this technique will work for any problem domain, we focus here on the satellite network bus. We will describe the system in terms of resources, and will describe these resources in one of two categories:

1. Service - A service is some utility provided by a node that is accessed through a communications interface
2. Data Point - A data point is a source of data that can be sampled over time using a communications interface

Note - A data point is ultimately accessed using a service, however describing data points separately in our ontology is useful because data-flow is a primary focus of guidance, navigation, and control systems.

C. Primary Taxonomies

We use taxonomies to collectively describe the logical view of the system. In particular, we will use three primary taxonomies to enumerate the problem domain.

1. System Taxonomy

The System taxonomy describes the well known components of a satellite and its bus. This taxonomy is not as useful for dynamic discovery and configuration because it describes components instead of resources. However this taxonomy can be used to label the components of the system with standardized terminology, which is mostly valuable for diagnostic purposes. The SPA standard requires that all components have a universally unique fingerprint that uniquely identifies a system component over its lifetime, although this ID has no semantic relevance. The standardized terms in the System taxonomy can be placed within a device description to provide a semantic description of the component type. Tagging components with standardized type names can be useful when configuring, testing, or trouble shooting a system.

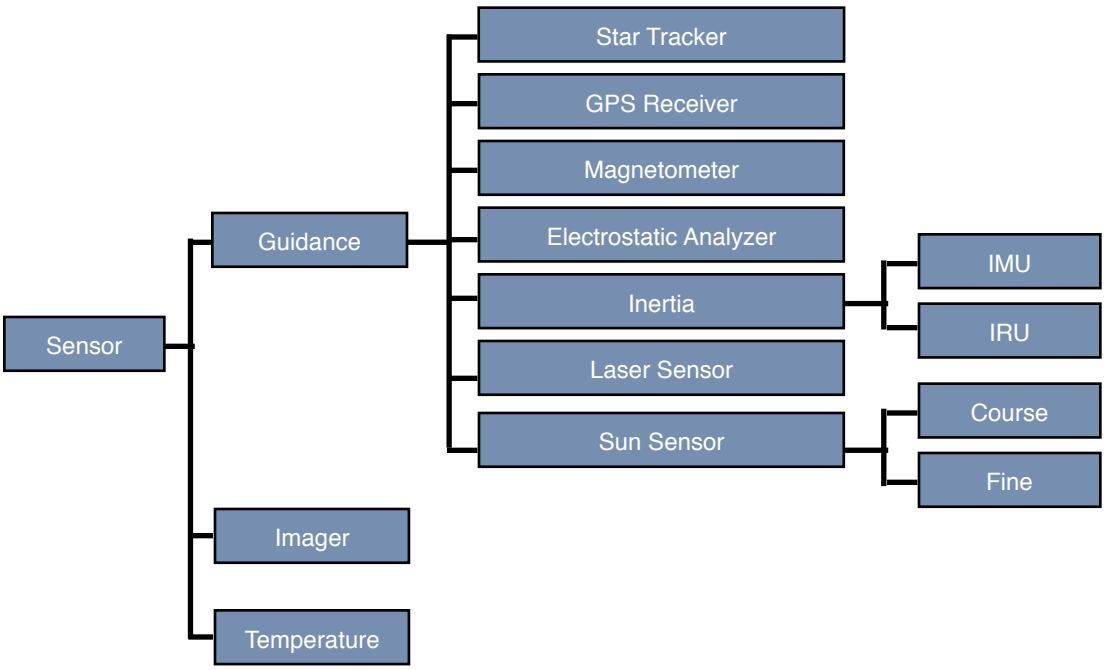

Figure 1. A subtree taken from the System taxonomy.

Assuming device descriptions are stored in XML, such as the xTEDS format used by SPA, support for this taxonomy can be added into an existing device description by simply adding one additional element that references this taxonomy.

<ComponentType>System/Spacecraft/Bus/Device/Sensor/Temperature</ComponentType>

Example 1. We use a pathname derived from the full System taxonomy to use as the component name, where the System taxonomy represents the range of allowable values for the ComponentType element in our device description.

2. Interface Taxonomy

The Interface taxonomy enumerates the well known services within the system. An interface is a collection of related messages that allow each service to be utilized over a communications path. By standardizing the well known interfaces, we can reliably locate services by name.

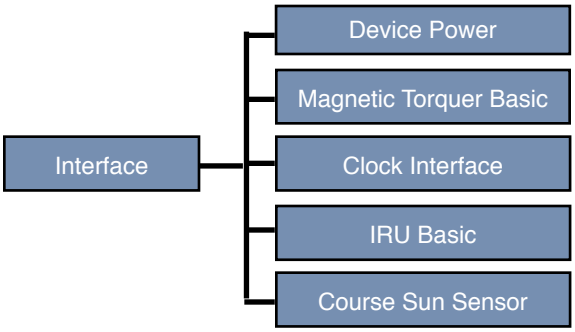

Figure 2. A subtree taken from the Interface taxonomy.

The xTEDS approach in SPA currently relies on completely describing the syntax of every message in each interface within a device description. We feel strongly that well known interfaces should be referenced by name and

version, not through the syntactical expression of the entire interface. Arguments have been made suggesting possible benefits to this approach, however it opens the door for many new problems to occur.

We have optionally explored using a taxonomy to formally describe an entire messaging interface. There is no known prior practice of doing this. However, having machine-processable descriptions of the messaging interfaces can be useful for building tools, managing standards, and designing new interfaces.

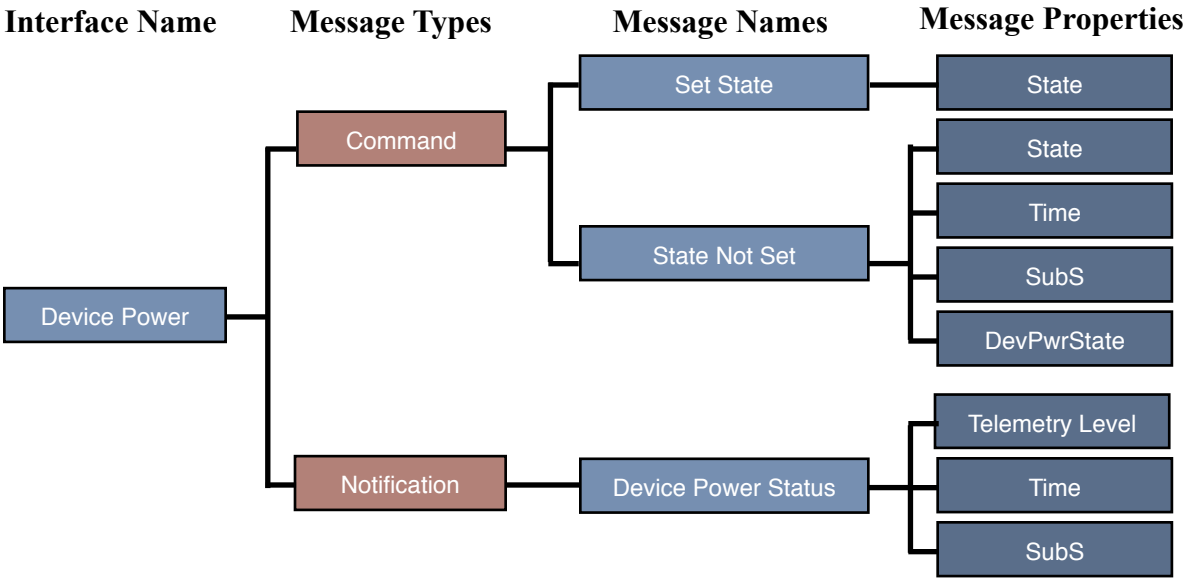

Figure 3. An example of an interface taxonomy description, where the leaves have additional RDFS properties such as Units, Type, Model, etc.

3. Data Taxonomy

The Data taxonomy describes the well known data points within the system. This taxonomy is used to enumerate these data points in a standard manner. The data points include sensor data, as well as data synthesized from sensor data. This taxonomy allows guidance, navigation, and control software to dynamically locate and interface hardware or other processes that provide data points using only a logical model.

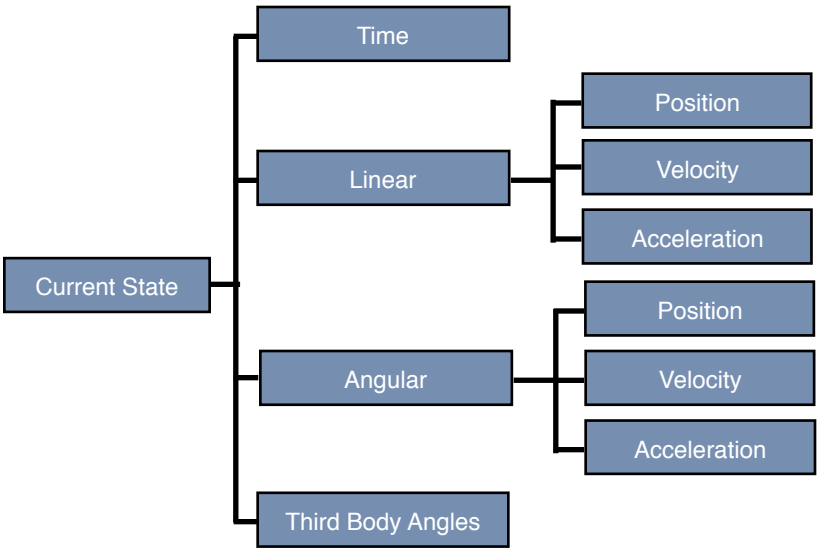

Figure 4. A subtree of the Data taxonomy.

<DataPoint>System/CurrentState/Linear/Acceleration</DataPoint>

Example 2. An XML example of a data point description derived from the Data taxonomy.

The data points described in this taxonomy also have properties associated with them. These properties can be expanded to provide any utility necessary, and are described formally in our ontology using RDFS.

Some examples of properties for a given data point are the following:

Units - The unit of measurement that this data is provided in (centimeters, meters, etc.)
Type - The physical type of data that values are provided in (32 bit integer, 32 bit float, etc.)
Precision - how precise the value is from the point of origin
Model - The mathematical model used to represent the data (Cartesian, Spherical, RGB , HSV, etc)
Description - An informal human-readable textual description of what this value contains
Max Frequency - The maximum rate that this value changes
Writeable - Determines whether this data point can have values written to it

D. Secondary Taxonomies

Several of the Data properties require standardization as well to build reliable systems where interoperability between software and hardware from different manufacturers works as intended. For this reason, we have created three more taxonomies that enumerate the domains for these properties. We refer to these as secondary taxonomies since they are used solely to further constrain the property values of elements in a primary taxonomy.

1. Unit Taxonomy

For a consumer to properly utilize a data point from another source, we must constrain the range of allowable unit types to a well defined set. For this purpose, we created the Unit taxonomy which is used to enumerate the only allowable values for the Unit property of any given data point.

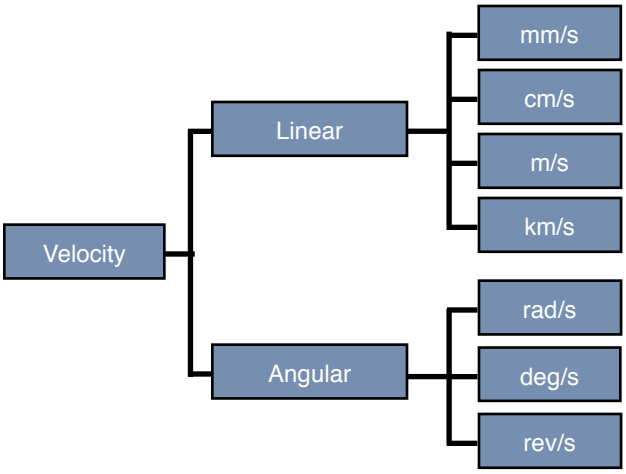

Figure 5. A subtree of the Unit taxonomy.

The taxonomy in Figure 4 shows that a data point providing a velocity value is limited to the set of unit types shown.

<Unit>Unit/Velocity/Linear/mm_s</Unit>

Example 3. An XML example of a Unity property derived from the Unit taxonomy.

2. Type Taxonomy

For a consumer to properly utilize a data point from another source, we must constrain the range of allowable types to a well defined set. For this purpose, we created the Type taxonomy which is used to enumerate the only allowable types for the Type property of a given data point.

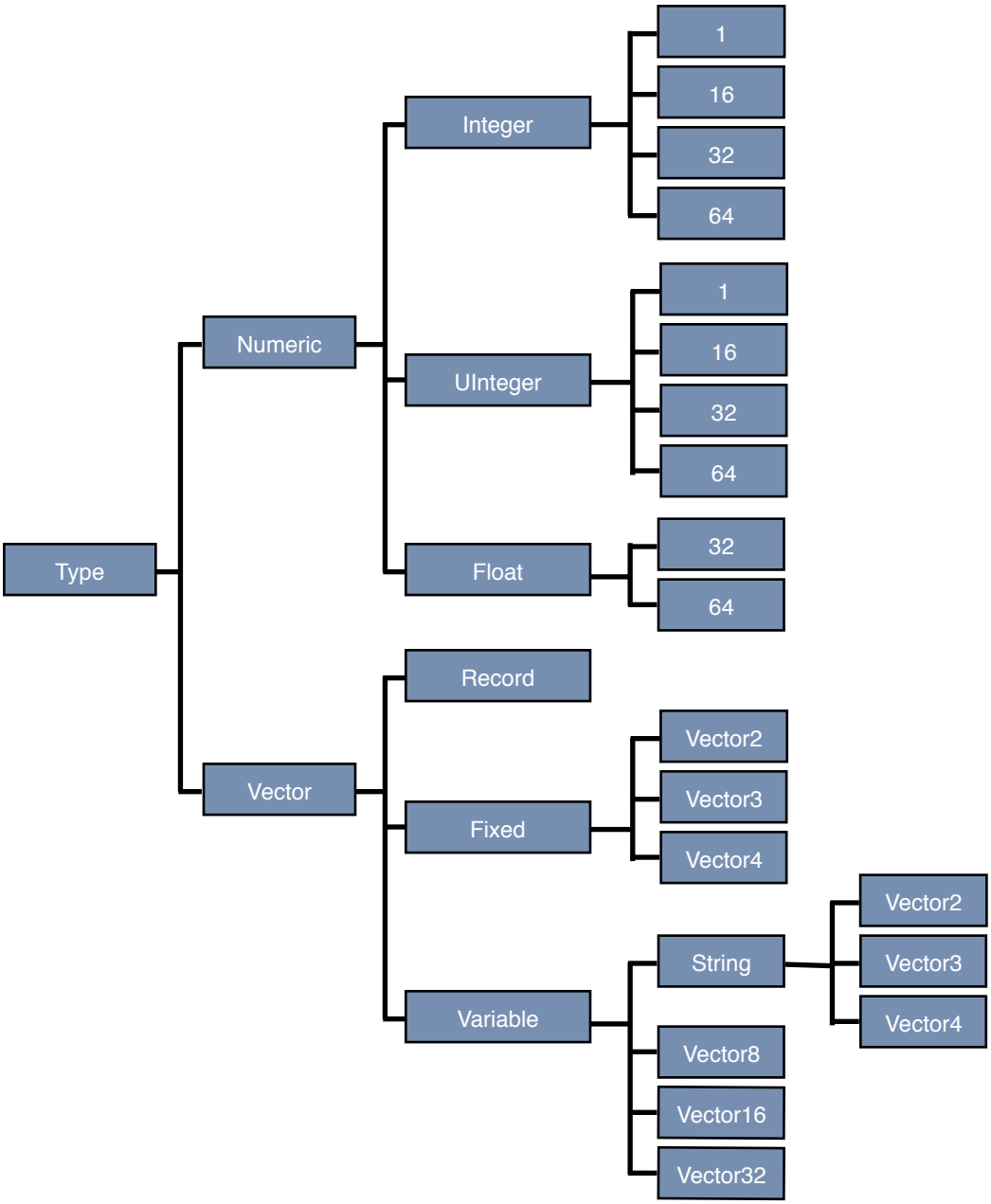

Figure 6. A subtree of the Type taxonomy.

The taxonomy in Figure 5 shows some of the basic types allowable for numeric and vector data.

<Type>Type/Numeric/Integer/32</Type>

Example 4. An XML example of a Type property derived from the Type taxonomy.

3. Model Taxonomy

For some data types, Units and Type are not sufficient to determine how to properly use the data. For example, if a position value provides an array of two Float32 values, we would still need to know what specific coordinate system the data is described in. This position might represent a 3D point in spherical coordinates or a 2D point in cartesian coordinates.

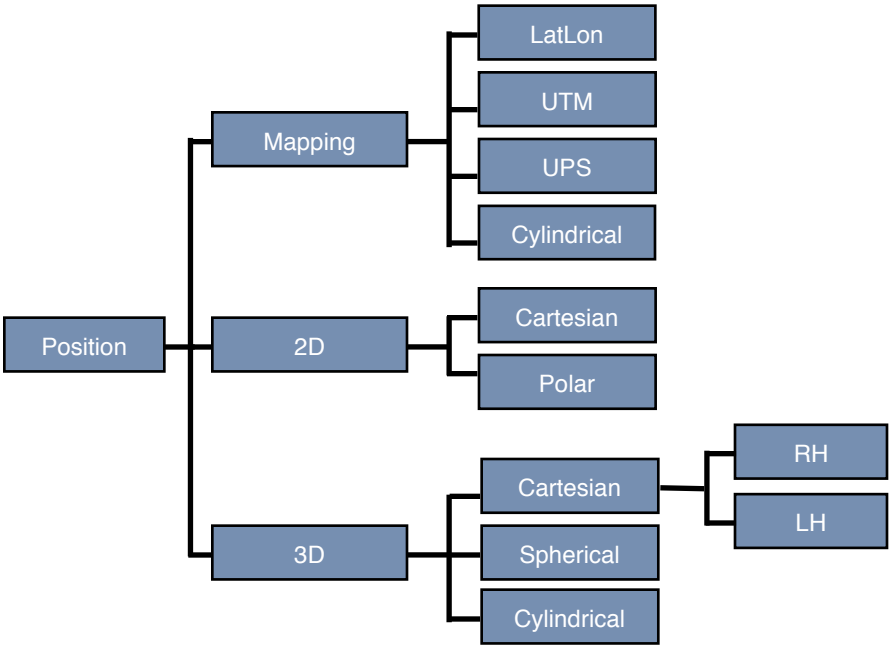

Figure 7. A subtree within the Model taxonomy.

<Model>Model/Position/3D/Cartesian/LH</Model>

Example 5. An XML example of a Model property derived from the Model taxonomy.

E. Supporting Multiple Ontologies

The taxonomies we use ultimately result in namespaces within our device descriptions. If we want to build an extensible system that can take into account more than one ontology, all that is necessary is to add a root level namespace to our taxonomies to differentiate the ontologies they belong to. In our device descriptions, we can include multiple terms that correspond to different ontologies.

<ComponentType>SPA/System/Spacecraft/Bus/Device/Sensor/Temperature,NASA/Satellite/TemperatureSensor</ComponentType>

Example 6. This Component Type describes more than one Ontology, where multiple descriptions have been comma separated.

As this example demonstrates, any field in a device description can include multiple entries. This allows a device description to correspond to more than one standard. Ultimately these two namespaces correspond to two different ways to reference the same resource, ie. two different keys to query against that will result in the same set of resource providers.

F. Version Management

As mentioned earlier, the taxonomies we use ultimately result in namespaces. We must assume that our logical view of the system will evolve over time, as will our respective ontology. To account for this, we can add a version number to the root of our namespace.

<ComponentType>SPA 1.0/System/Device/Sensor/Temperature,SPA 2.0/System/Spacecraft/Bus/Device/Sensor/Temperature,NASA 1.5/Satellite/Sensor/TemperatureSensor</ComponentType>

Example 7. This example demonstrates a component type that corresponds to three different ontologies, two of which are simply modified versions of the same ontology. These three descriptions represent three unique keys that will result in the same set of resource providers.

To further improve backwards compatibility, a newer system (one which is connected to an older component) can also have a mapping table that allows older devices to be mapped to newer ontologies without an older device having to be modified to express this directly. This would allow the system and its ontology to evolve without breaking existing hardware that has fixed device descriptions.

III. Conclusion

We have demonstrated that a set of primary and secondary taxonomies can define a complete logical model of the resources represented on a satellite bus. This logical view of the system provides a common set of well known terms that can be used to resolve the resource relationships between nodes on a system in a completely dynamic manner, agnostic to the underlying physical topology.

We chose RDFS as the syntax for formally describing this ontology because it is a well known standard in other industries and is well suited to the task. We have shown how these techniques can account for multiple ontology standards as well as deal with ontologies that evolve over time. We have also implemented a system that successfully demonstrates and uses these techniques in simulation, including tools that allow us to develop, modify and evolve ontologies. These tools can also produce device descriptions, such as xTEDS, in a highly constrained fashion that corresponds to our ontology.

We believe that using a formal ontology as our logical model gives us a strong framework for capturing and leveraging domain expertise in standards based plug-and-play solutions. We believe this framework improves upon previous approaches by providing increased formalism, specificity and determinism, while also being simpler to implement and utilize for all parties involved. We also believe that the greatest value in this research is in the approach itself, and less so in the taxonomies that we have used to fabricate and demonstrate within our system. We can imagine a collaborative approach where multiple communities work together to develop and establish more elaborate taxonomies, where the best and most practical ideas are captured and combined into an improved set of standards and an improved logical model.