Architecture Features for a Semantic Web of Smart Objects

Michael J. Koster
August 21st, 2012

In this post I'd like to take a brief architectural look at the Semantic Web of Smart Objects I described in the first 2 posts. This provides a framework from which to understand the choices and direction.

Architecture as a set of constraints

Here I am looking at the notion of architecture as broadly consisting of a set of constraints imposed in order to achieve a particular goal.

*The goal is to enable the creation of a web ecosystem of **sensing**, **reasoning**, and **action** around the internet of things.*

The mechanism is broad interoperability across sensors, services, and devices.

I propose to do this by creating a semantic web template (set of constraints) for the internet of things, and a set of common practices that can also be semantically defined.

In particular, the architecture of a semantic web of smart objects is a set of constraints imposed on the use of http/REST and semantic web standards in order to achieve broad interoperability at a granular scale of things and collections of things. (These constraints can themselves be semantically defined, making this a sort of "declarative architecture").

Seven Layer Model

The first organizing principle I'd like to invoke is the well-known seven layer model for networked applications. Given the predefined set of constraints (http/REST and semantic web) the session state management layer is well established by the RESTful interface.

Thus we need to define the constraints necessary to make interoperable the layers for data presentation and the interface to the application layer. *This architecture therefore constrains the **data presentation layer** and **API**.*

Constraints will be applied to the RESTful interface to enable an interoperable service laver which provides data presentation services and a harmonized application API to enable data sources and applications to interact seamlessly and semantically with each other. In other words, data sources may be plugged into applications and applications may provide pluggable data sources using a uniform set of mechanisms.

Smart Objects

The next organizing principe I'd like to invoke is object encapsulation. The definition of an object also becomes the de facto definition for the representation of a thing or set of things on the Semantic Web and on the Internet.

The basic object encapsulation is an important architectural choice, as this constraint becomes a constraint on the way applications interact with data and metadata.

I propose to encapsulate observable property data and metadata together under a RESTful interface model that can be semantically shared between the web interface to an object and the API to application software executing within the context of the object.

Object state, object metadata, software agents, and software agent metadata share a RESTful API, with a single instance of each object pointed to by a URL. The entity this URL points to is referred to as a ***Smart Object*** *for the purpose of this discussion.*

Semantic discovery and linkage

The constraints needed to define the architecture here are related to semantic web operations. Some well defined method for getting the semantic metadata from an object to provide for indexing and discovery is needed. Fortunately the metadata are unstructured RDF triples and a simple list-all mechanism is sufficient. All that is needed is the URL to point to the object and a well known type to retrieve the triples. The triples can then be indexed for semantic linkage.

Semantic linkage can be accommodated by adding methods to return the URLs of resources in the object whose metadata properties satisfy semantic queries. A type for semantic queries and a semantically defined subset of SPARQL is indicated. An additional mtype for embedded semantic linkage is also possible and may prove useful to improve efficiency.

Additional constraints and use cases

It's important for several use cases to provide a mechanism to conditionally push linked data from one Smart Object to another. This should not impact architectural constraints, since the push operation can be semantically defined, as can the application agent call back handler linkage. There may be a need to push metadata describing the push operation itself. Also of note is that metadata triples may also be conditionally pushed.

The use case for Smart Objects includes bridging to constrained network endpoints. In this case, the bridge may create a Smart Object proxy for resources on the constrained network, and record associations of resource metadata with resource addresses on the constrained network, functioning as a semantic reverse proxy router. The bridge can also function as metadata repository and agent execution environment.

A simple bridge endpoint can also function as a standalone smart sensor and interact directly with other Smart Objects on the Internet. The sensing/actuating process is performed by an agent with well defined properties.

The other endpoint is at a user device. Rather than generate HTML for a web page or XML for AJAX controls, a Smart Object proxy can run directly on the user interface device and provide the Smart Object agent API directly to a user information/control app. A Smart Object proxy also allows the easy connection of a broad range of user interface devices, including the use of simple devices on constrained networks.

Standards

The architectural constraints apply to a set of existing web and internet standards. No new methods or protocols need to be invented. The standards drawn from include:

http(s)
REST
HTML
XML
JSON
Xpointer/Xlink
RDF/Turtle
SPARQL
SSN ontology
DUL ontology
OAuth

In addition it is expected to use and interoperate with:

6LoWPAN
CoRE
CoAP
Arduino
Processing
Python/RESTlite

Summary

The goal is to enable the creation of a web ecosystem of ***sensing****,* ***reasoning****, and* ***action*** *around the internet of things.*

The above describes what may be a minimum set of constraints and models needed to enable broad interoperability. It also defines an architectural approach to achieving a ***Semantic Web of Things.***

The architecture described above defines a data presentation layer and API which use a common RESTful interface model.

A Smart Object abstraction encapsulates an instance of observable data properties, semantic metadata, application agent processing, and agent process metadata under a URL.

Smart Objects can be semantically linked together through a metadata query interface or dynamic semantic linkage language.

Endpoints of constrained sensor nets, Smart Sensors, and diverse user interface devices are enabled through Smart Object proxies.

No new methods or protocols need be invented. The architecture consists mostly of declarative constraints.

Additional constraints may be found as the investigation proceeds toward a prototype Smart Object framework.