A Pragmatic Approach to Interoperability in the Internet of Things

Kai Kreuzer, Deutsche Telekom AG

January 24, 2016

Submission to the IAB IoT Semantic Interoperability Workshop

Abstract

More and more devices are connected to the Internet, but the result is currently rather an incomprehensible jungle of silo solutions and walled gardens instead of a flourishing eco-system.
Users do not want to be locked-in to a single vendor and rightly expect devices from different vendors to work seamlessly with each other. There are essentially two approaches for addressing the problem:

1. Have all devices speak the same language
2. Translate their communication by means of an intermediator

The first approach is pursued by standardization efforts like AllJoyn, OIC and UPnP. The intrinsic problem with this approach is the heterogeneity of the Internet of Things: It reaches out into all market segments and covers things of the size (and complexity) from a micro-bot to a spaceship. Naturally, their communication requirements and the technical possibilities differ heavily and thus make it impossible to define a common language. In the end, all such approaches will always only apply to a subset of the Internet of Things.
The downside of the second approach is that an intermediator is required, which prevents a peer-to-peer communication scheme. The positive aspect though is that such an integration can easily evolve over time. New device types can be added, new protocol versions can be supported transparently besides the old ones and even legacy devices can be integrated. Translating between many protocols and system automatically means some loss of detail, since we are moving from concrete data models to an abstract information model. Yet, this is often good enough for over-arching applications.

The open-source project Eclipse SmartHome pursues the second option and – while specifically targeted at the home automation market – its concepts can easily be applied to the IoT in general.

Functional vs. Device Perspective

There are two different perspectives for looking at interoperability and their assumptions and requirements differ heavily:

- Talking about the Internet of Things, the thing (or device) is in the center of the attention. For interoperability, it needs to be described in an abstract way. In order to be useful for the system/application that wants to support the device (through a "generic" driver), a lot of technical details that are specific to the device are required besides its functionality, e.g. how it is discovered, configured, authenticated and included – all of this can be considered to be a data model for a specific protocol.
  Eclipse SmartHome defines a Thing to have
  - a configuration (consisting out of a list of parameters)
  - properties (for static information like serial no., product ids etc.)
  - channels (reflecting the functionality, once it is operational)

  Meta-information can be referenced by declaring a *Thing Type*, for which Eclipse SmartHome defines a common meta-model. Within this meta-model, configuration parameters as well as channels can be optionally organized into groups, but there is no possibility to nest groups into each other.
  A special thing type is a so-called *Bridge*, which is a Thing with the additional possibility to expose other Things. Any number of Bridges can be chained, so that complex topologies can be represented.

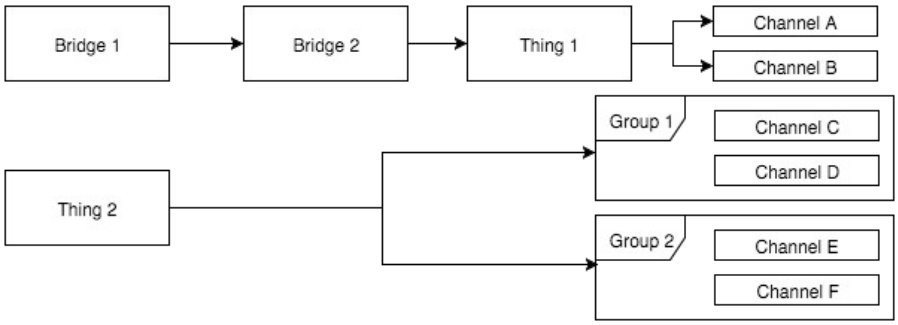

Figure 1: Topology options for Things in Eclipse SmartHome

The focus of this Thing model is to describe the physical side in order to be able to correctly discover, setup and configure devices by means of generic user interfaces that do not need to be adapted when new device types are added.

- Looking at the needs of (use-case driven) applications, it becomes clear that this device perspective is not ideal:
  - The general assumption of applications is that all setup and configuration is already done and that the functionality can "simply" be accessed.
  - Not all functions of a device might be of interest, e.g. if an application might only want to have temperature values, but a multi-sensor also includes motion, humidity and luminance information.
  - The grouping of functionality does not necessarily need to follow the physical setup. A multi-channel actuator could switch lights in 5 different rooms – an application might prefer grouping by room instead.
  - A device might only be a technical means. An application is interested in the door, not in the magnetic contact attached to it. It wants to switch a light and not merely an outlet.
  - Low-level functionality might already be wrapped in hardware or software and only exposed on a higher-level (e.g. switching on a power plant) – in the end, it does not matter to the application, if it is talking to a device or some intermediator.

  Applications therefore have a functional perspective. It is not devices that need to be described, but whole systems, which might aggregate, group, modify or even simulate device functionality.

Eclipse SmartHome strictly separates the functional perspective from the device perspective. Functionality is expressed as *Items*, which are comparable to data-points that can be read and written and which are evented. They are usually based on primitive types such as booleans, integers or string enumerations. Other abstractions refer to similar concepts as capabilities. Items can be freely grouped, and an item can be a member of several groups. Groups can be nested and can themselves serve as a functionality (e.g. "all lights" can be switched). Items can be linked to a channel of a device, but likewise could also be implemented in software.

As a result of these thoughts, one has to state that the problems (and therefore also the solutions) for applications in the IoT are similar to any other application that uses services on the Internet, be it a large scale SOA application or a loosely-coupled REST-based mashup.

The Semantics of Things

In the typical situation, the semantics of a used device/service are known to the application programmer, so that he can wire the application in a meaningful way. Meta-models such as the Thing meta-model of Eclipse SmartHome or the Smart Device Template (SDT) help on defining the structure, the functionality and the data types of devices. The meaning behind it is described as prose for the application programmer – but it is not formalized and thus not deducible by algorithms.

Semantics can be achieved through reuse by abstraction, which basically means establishing a common vocabulary (an ontology) for similar things. This needs to be done on several levels, though:

1. The “Thing” itself, e.g. the washer dryer combo
2. The sub-component, e.g. the washing machine
3. The feature, e.g. the “washing program selection”
4. The specific values, e.g. the “Wool”

Even for small domains, such a vocabulary can be huge and it requires immense effort to come to agreements on the definitions. Just as in natural language, words can have different meanings in different domains, therefore such ontologies are likely to be domain specific.
What is more, the description of the Thing is not sufficient for applications; it must also be defined what something is used for. This might be obvious for a washing machine, but this is not the case for e.g. a temperature sensor. For an application, it makes a huge difference, whether it is measuring a room temperature or the temperature inside a fridge – this information can often only be supplied through the concrete system setup and hence cannot be provided through an information model of a device.

Eclipse SmartHome therefore defines the semantics on the items (the functional layer) and does this through *tags*. Similar to JSON-LD, tags can generally reference classes from any ontology. It is up to the solution that is built with the Eclipse SmartHome framework to decide, which ontologies are supported, so that applications can make sense of this information. Thing descriptions can include suggestions for tags to be put on linked items, but ultimately the user or system administrator decides on the correct tags on the items.
Alone in the Smart Home domain, there are many competing and overlapping ontologies and their number increases. As there are often commercial interests involved, it is unlikely that these get merged, hence there is definitely the necessity to have mappings between different ontologies. Naturally, some level of detail gets lost by this and often mappings are simply not possibly, because there is no correspondent.

Summary

There are good meta-models available for describing the structure and functionalities of Things.
For semantic interoperability on application level though, a functional perspective must be taken and thus device descriptions and taxonomies are of limited value. The requirements are rather very similar to those of the Semantic Web, and so are the challenges, like e.g. vastness, vagueness and inconsistency.
There is a large choice of ontologies, but none of them will ever be complete. Systems and their applications have to chose a best fit. Eclipse SmartHome provides the technical means but leaves the decision to the solution. For automatically reasoning, applications will always require hints from the user on the context – this can be supplied through simple configuration or interactive on-demand feedback methods.