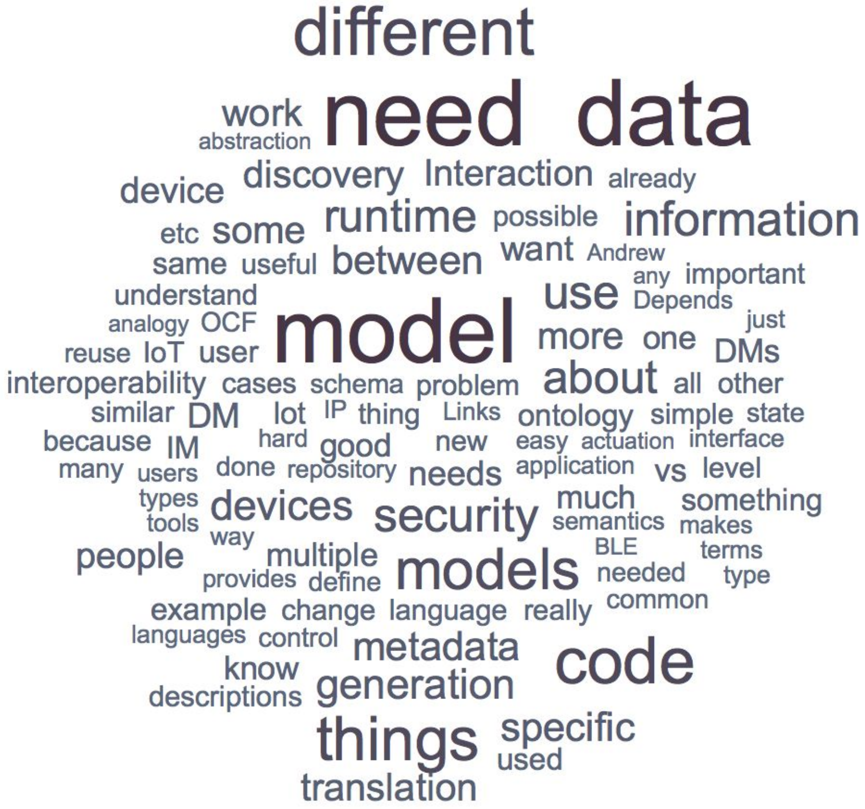
Agenda Day #2

9:30 - 10:00	Atoms
10:00 - 10:30	Summary of day 1 (Koster & Jimenez)
10:30 - 11:30	Security
11:30 - 12:00	Demos
12:00 - 12:30	*Coffee break*
12:30 - 14:00	**What should we do next?**
	Future communication channels
	Common repository
14:00-	Lunch on own

Summary of Day #1

Michael Koster, Jaime Jiménez


different
work need data
abstraction
device discovery Interaction already
runtime possible information
etc some
same useful between want Andrew
any important
understand use Depends
analogy OCF model just
reuse IoT user more one DMs
interoperability cases schema problem about all other
similar DM lot IP thing Links ontology simple state
because IM hard good new easy actuation interface
many users done repository needs application vs level
types devices security much something
tools semantics makes
way BLE terms
people multiple models needed type
provides define common
example change language really
languages control metadata code
know
descriptions generation
things specific
used
translation

Terminology

- Need to align on basic terms (RFC3444), Information Model and Data Model.
- Do we need to distinguish between the two? Some don’t...
- We need to talk about the interaction model and express it too.
- Need to define more explicitly the - many - missing terms.
- A common model at the higher level helps companies
  - No need to drop work in which they have invested.

Architecture

- Current definitions are focused on devices. Need to represent the state of abstract concepts according to user perspective (e.g. parking lot).
- Need to focus on interactions between “things” not just towards human users.
- The device should be the one telling what it uses, what it is.
- New domains might come up in the future, not in a centralized registry, the solution should be open-ended.
- Missing insights of what users actually expect (UX, ...).
- We shouldn’t build things for a single purpose.
    - smart phone led → heartrate monitor

Common Problems

- Collaborative approach to create models.
- Tools used for creation and validation of models.
- Simple vs Verbose data models.
    - Balance between over-definition and too little.
- Interaction Model: minimal shared vocabulary interaction through hypermedia.
- Metamodel as a higher layer of abstraction DMs are derived from.

Mapping IMs and DMs

- Runtime translation of data vs Translating a Data model once.
- "Translation Hub/s" (★) and how to implement it/them.
- Translation is easier if there is REST.
- Design patterns (REST, PubSub, RPC) and discovery in them.
- “Loss” in translation? Maybe not for sensing?
- No multiprotocol option for constrained devices.

Runtime discovery

- Discovery of devices and abstract entities
- How much must be shared beforehand?
- Incremental discovery + Bookmarking.
- Predefined interfaces problem
  - What if you don’t have the logic already implemented.
- Real time vs Pushing code
  - Need to avoid locked-in situation and provide scalability.
- Intelligence on device vs intelligence on GW and elsewhere.
- Automatic mapping models and discovery requires code.

Code Generation

- Several examples of code generation used in some ecosystems today
- Discussed whether and where it makes sense
- Code Generation is useful for validation and generating stubs/execution plan.
- Using strong types can help app developers and reduce bugs
- Code generation removes friction between the developer and the API.

Reuse of Models

- Linked-Data makes it possible
- Use of a common repository (schema.org) for "atoms" and procedures for interaction with the repository.
- Reusing often does not work, groups implement without sharing.
- Schemas are also made with specific use cases in mind (schema.org/Car).

Common Learning

Information Model vs. Data Model

> 2 levels of interoperability, 1.how to know whether the parking lot is full or not, regardless of the method used to determine it, 2. how that state is represented in the communication between entities, This could be at many levels e.g. space occupancy, image analysis, pre processed data vs summary %full, etc)

Semantic Interoperability is not exclusively provided by one or the other

Data models are in part handled by media types, content formats, encodings

Metadata handling "in-band" should be considered also cached, offline, design time, compile time, run time binding options

Discussion points

Abstraction layer, semantic overlay, extensible and inclusive properties

Ontology based information models

Simple/general vs. specific/expressive tradeoff, enable innovation and differentiation

Modularity and reuse, molecules composed of atoms

Interoperabilty doesn't require reuse

Code generation is a developer optimization - (is it a path to scale?)

Model translation vs metadata translation, ALGs

Schema.org type approach, repository vs. distributed, inclusive vs. prescriptive

Discussion points

Abstraction layer, semantic overlay, extensible and inclusive properties

Ontology based information models

Simple/general vs. specific/expressive tradeoff, enable innovation and differentiation

Modularity and reuse, molecules composed of atoms

Interoperabilty doesn't require reuse

Code generation is a developer optimization - (is it a path to scale?)

Model translation vs metadata translation, ALGs

Schema.org type approach, repository vs. distributed, inclusive vs. prescriptive