IoT Semantic Interoperability Workshop Terminology

Benoit Claise

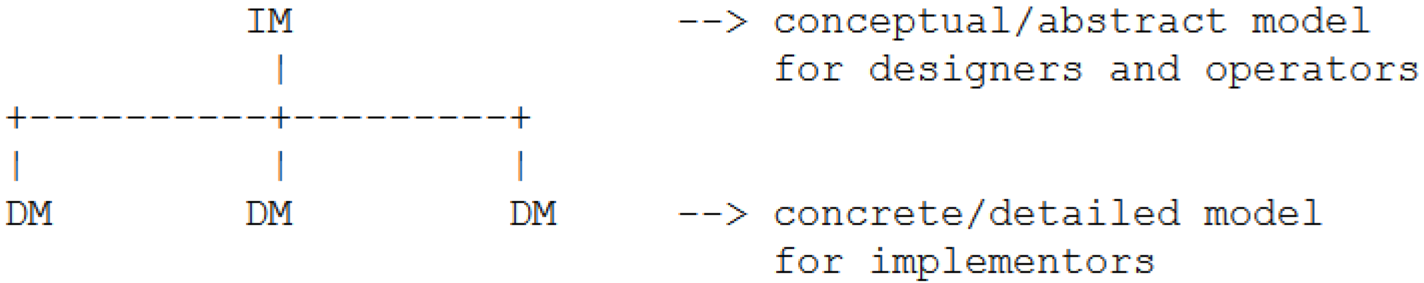
Why Focusing on Terminology?

- To avoid confusion
- To facilitate the workshop discussions
- Different background = different terminology
- To save time later

Clarification on Information Model versus Data Model

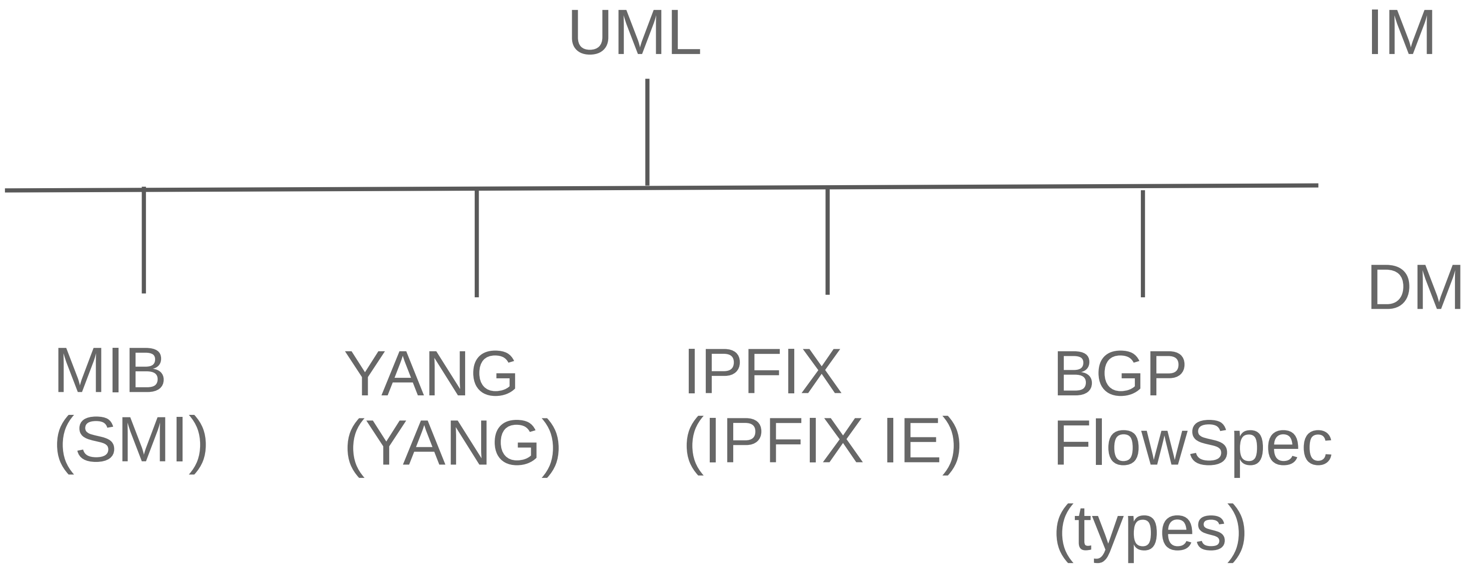
> “The main purpose of an IM is to model managed objects at a conceptual level, independent of any specific implementations or protocols used to transport the data.
>
> [...]
>
> DMs, conversely, are defined at a lower level of abstraction and include many details. They are intended for implementors and include protocol-specific constructs.”
>
> *-- RFC 3444 On the Difference between Information Models and Data Models*

Clarification on IM versus DM



```
          IM                 --> conceptual/abstract model
           |                     for designers and operators
+----------+---------+
|          |         |
DM         DM         DM     --> concrete/detailed model
                                 for implementors
```

Clarification on IM versus DM



Data Model Mapping

- Mapping data between data models, as opposed to translating data models
- Mainly hardcoded in NMS, painful, costly
- Example: prefix
  - ipAddressPrefixTable: MIB module
  - sourceIPv4Prefix(8), sourceIPv6Prefix(170) in IPFIX
  - source prefix (2), BGP flow spec
  - Syslog: plain english text
  - ….: YANG
  - ….: RADIUS
  - ….: Diameter
  - ….: you-name-it

IETF and IM/DM

- We don’t specifiy many IM, we focus on DM
  - With YANG as THE data model language for configuration
- Why?
  - Timing: We need to move faster
  - Opensource: pressure versus standards
  - Operators: « give me something I could use », for automation
  - We can’t derive the full DMs from the IM
- However, IM
  - Is good as a starting point
  - Should lead/help to DM definition

Data Model Driven Management

Acting on resources

```
Module my-interfaces {
{
 namespace ”com.my-interfaces”;

   container interfaces {
     list interface {
       key name;
       leaf name { type string; }
       leaf admin-status { type enum;}

   rpc flap-interface {
     input {
       leaf name { type string; }
     }
     output {
       leaf result { type boolean; }
     }
}
```

GET : Gets a resource

GET /restconf/data/my-interfaces:interfaces

GET /restconf/data/my-interfaces:interfaces/interface/*<some name>*

POST : Creates a resource or invoke operation

POST /restconf/operations/my-interfaces:flap-interface
+ JSON/XML Form Data (including name)

Response will have JSON/XML result

PUT : Replaces a resource

PUT /restconf/data/my-interfaces:interfaces/interface/<some name> + JSON/XML Form Data (name, admin-status)

DELETE : Removes a resource

DELETE /restconf/data/my-interfaces:interfaces/interface/<some name>

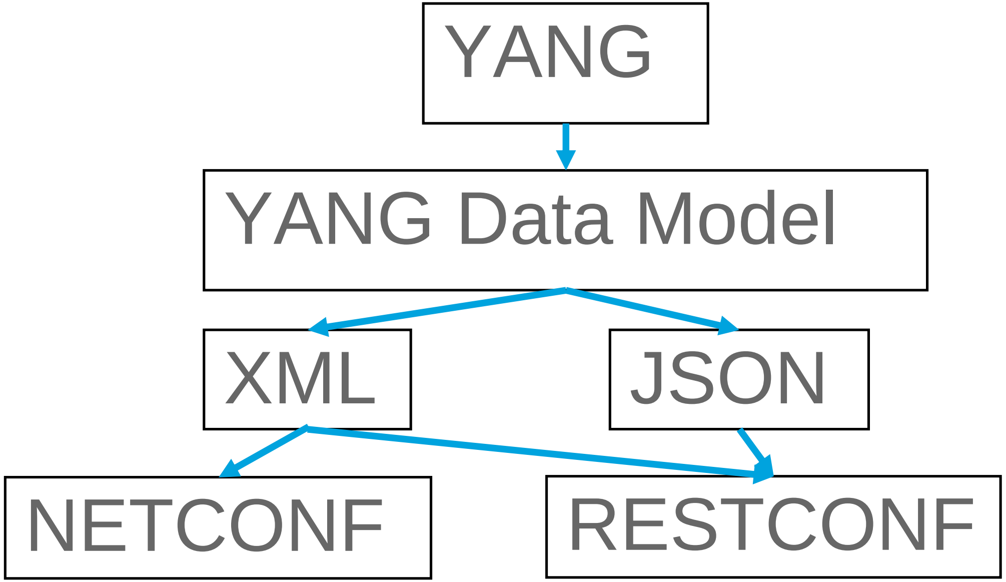
Terminology/Relationships as an example

Data Model Language (schema language)

Data Modeling (schema)

Encoding (serialization)

Protocol



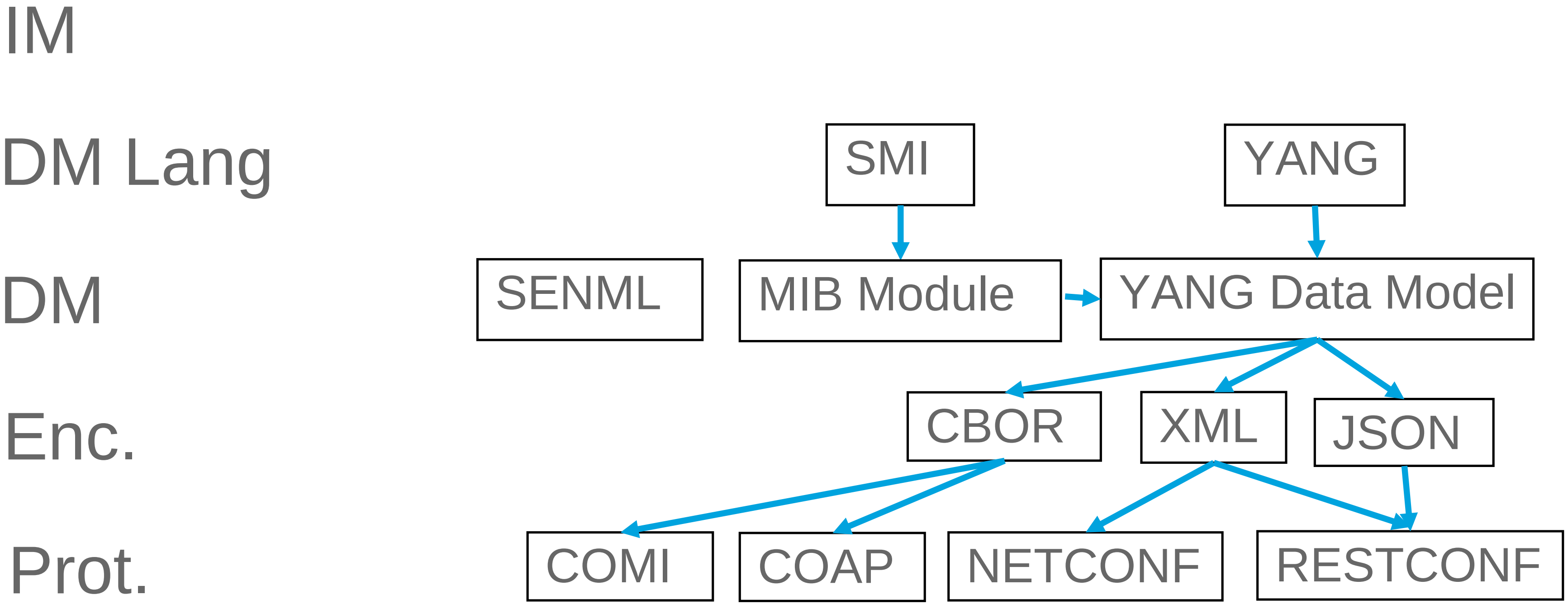
Data Models Driven Management

- APIs derived from the data models:
  - Data Model Language: YANG
  - The protocol: NETCONF or RESTCONF
  - The encoding: JSON or XML
  - The programming language: Python, Ruby, Java, C, Erlang, ...
- Industry focusing on YANG as *the* data modeling language for services and devices
- Scripting: easy to create, hard to maintain/clean-up

  => Data model-driven set of APIs

Data Models = APIs

Inserting some (IoT) keywords in here…



IM

DM Lang

DM

Enc.

Prot.

SMI

YANG

SENML

MIB Module

YANG Data Model

CBOR

XML

JSON

COMI

COAP

NETCONF

RESTCONF

Which Terms are we Missing?

- Metadata: additional information that complements an object instance
- Instance: an instantiation of a managed object
- Ontology: …

Conclusions

- Let’s be precise about terminology
- Automation is required. Hence data model driven management
- Think carefully about your (common) data model language(s)

IoT Semantic Interoperability Workshop Terminology

Benoit Claise