Information Models, Data Models, and YANG

Jürgen Schönwälder

IETF 86, Orlando, 2013-03-14

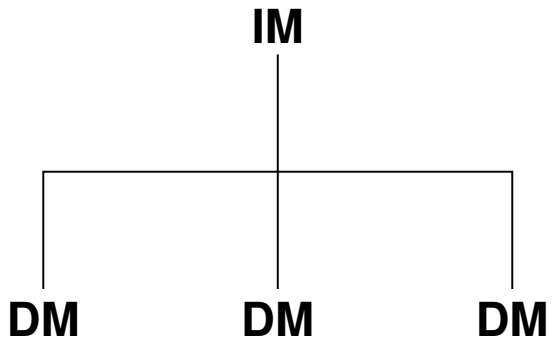
Information Models (RFC 3444)

- Information Models are used to model managed objects at a *conceptual level*, independent of any specific protocols used to transport the data (*protocol agnostic*).
- The *degree of specificity* (or detail) of the abstractions defined in the information model *depends on the modeling needs* of its designers.
- In order to make the overall design as clear as possible, information models should *hide protocol and implementation details*.
- Information models focus on *relationships between managed objects*.
- Information models are often represented in *Unified Modeling Language* (class) diagrams, but there are also informal information models written in *plain English language*.

Data Models (RFC 3444)

- Data Models are defined at a *lower level of abstraction* and include *many details* (compared to information models).
- They are *intended for implementors* and include implementation- and protocol-specific constructs.
- Data models are often represented in *formal data definition languages* that are *specific to the management protocol* being used.

Information Models vs. Data Models

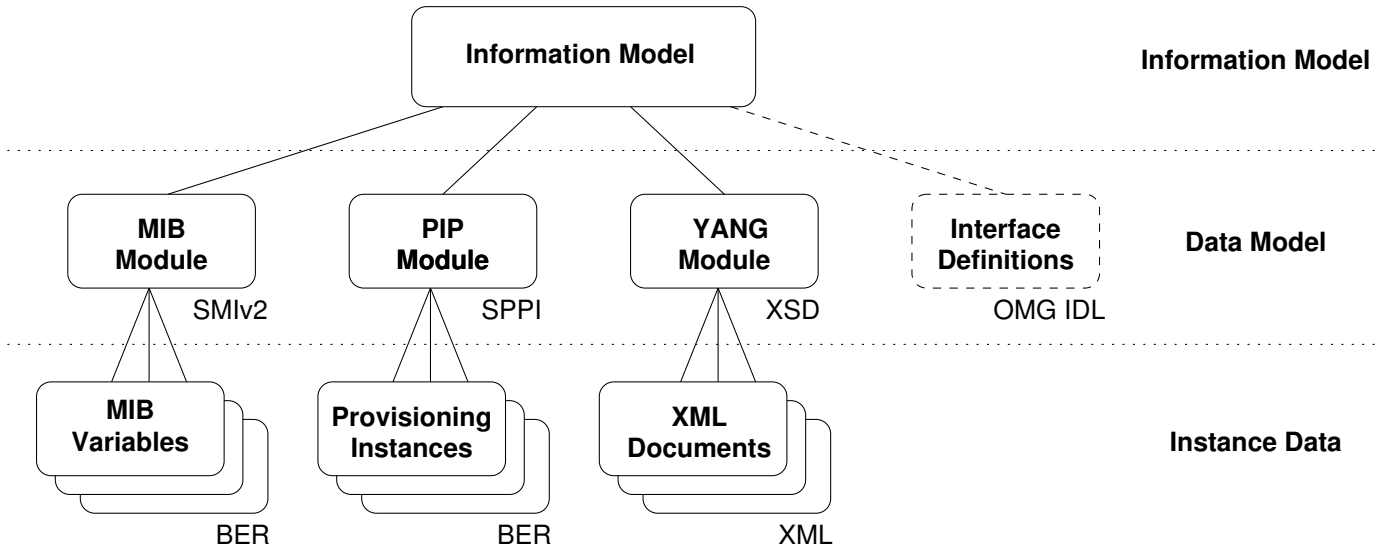


conceptual/abstract model for designers and operators

concrete/detailed model for implementors

- Since conceptual models can be implemented in different ways, multiple data models can be derived from a single Information Model.
- Although information models and data models serve different purposes, it is not always easy to decide which detail belongs to an information model and which belongs to a data model.
- Similarily, it is sometimes difficult to determine whether an abstraction belongs to an information model or a data model.

IMs and DMs in the Real World



- The architecture for differentiated services (RFC 2475) is an example for an informal definition of the DiffServ information model. The DiffServ MIB module (RFC 3289) and the DiffServ PIP module (RFC 3317) are examples of data models conforming to the DiffServ information model.
- The IPFIX configuration specification (RFC 6728) contains both an information model and a data model.

So what is YANG?

YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol, NETCONF remote procedure calls, and NETCONF notifications.

- hierarchical configuration/state data models
- reusable types and groupings
- data model extensibility through augmentations
- supports the definition of operations (RPCs)
- formal constraints for configuration validation
- data model modularity through features / sub-modules
- versioning rules and development support
- well defined ways to extend the language
- easy to read and process textual representation

Even more reasons to use YANG...

- produced and maintained by the IETF
- tool support (written by people active in the IETF)
- growing set of (reusable) definitions (typedefs, groupings)
- support via YANG doctors
- JSON encodings possible (currently not used by NETCONF)
- flexibility and extensibility

For a more detailed technical introduction to NETCONF and YANG, see the IETF 84 tutorial available on the IETF EDU pages:

`http://www.ietf.org/edu/technical-tutorials.html`

Defining YANG Data Models – I

- Start by identifying the basic concepts that need to be modeled, give those concepts good names, and identify relationships between them.
- Sketch the structure of the data model; if necessary break things into meaningful pieces (different modules or submodules, consider defining features for optional things).
- Use tools to generate summaries like YANG tree diagrams since they help to understand the structure of a larger data model (existing tools usually work fine with somewhat incomplete data model definitions).
- Include the generated diagrams as supporting documentation into the specification.
- Write example data instances in XML and validate them against the model; make sure you like the instance data and consider including some of the examples in the documentation.

Defining YANG Data Models – II

- Sometimes it is useful to factor out reusable components (e.g., type definitions or groupings).
- Sometimes a modeled function or data type is rather generic and not WG specific (talk with YANG doctors – they might help getting the definition into the "right" place).
- For type definitions, think about canonical representations if certain values may have multiple representations.
- Do not over constrain data models (be careful with `when` and `must` statements); design for exensibility by both future standards and vendor-specific extensions.
- Manage your namespaces and be consistent with your naming conventions.
- Use tools to validate your data model (and sample data instances), pick tool options that ensure compliance to IETF rules (which are a bit more strict than YANG itself).

YANG Tree Diagrams

Tree diagrams summarize the hierarchical structure of YANG data models:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node and "*" denotes a "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

YANG Tree Diagram Example

```
module: foo
 +--rw mycontainer
    +--rw mylist [mykey]
       +--rw mykey          string
       +--rw (alternative)?
       |  +--:(simple)
       |  |  +--rw simple?        uint8
       |  +--:(multi)
       |     +--rw multivalued*   string
       +--ro state          enumeration
```

```
module foo {
  container mycontainer {
    list mylist {
      key mykey;

      leaf mykey { type string; }
      choice alternative {
        case simple {
          leaf "simple" { type uint8; }
        }
        case multi {
          leaf-list "multivalued" { type string; }
        }
      }
      leaf "state" { config false; mandatory true; type enumeration; }
    }
  }
}
```

References

A. Pras and J. Schönwälder.

On the Difference between Information Models and Data Models.
RFC 3444, University of Twente, University of Osnabrueck, January 2003.

M. Bjorklund.

YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF).
RFC 6020, Tail-f Systems, October 2010.

J. Schönwälder.

Common YANG Data Types.
RFC 6021, Jacobs University, October 2010.

A. Bierman.

Guidelines for Authors and Reviewers of YANG Data Model Documents.
RFC 6087, Brocade, January 2011.