NETMOD Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2016

S. Mansfield, Ed.
Ericsson Inc.
B. Zeuner
Deutsche Telekom AG
N. Davis
Ciena
Y. Yun
Fiberhome
Y. Tochio
Fujitsu
K. Lam
E. Varma
Alcatel Lucent
July 3, 2015

Guidelines for Translation of UML Information Model to YANG Data Model
draft-mansfield-netmod-uml-to-yang-00

Abstract

This document defines guidelines for translation of data modeled with UML to YANG including mapping of object classes, attributes, data types, associations, interfaces, operations and operation parameters, notifications, and lifecycle.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Keywords 3
3. Terminology 3
4. Overview 4
5. Mapping Guidelines 4
5.1. Mapping Guideline Considerations 4
5.2. Mapping of Object Classes 5
5.3. Mapping of Attributes 6
5.4. Mapping of Types 8
5.4.1. Mapping of Primitive Types 8
5.4.2. Mapping of Enumeration Types 8
5.4.3. Mapping of Basic Data Types 9
5.4.4. Mapping of Complex Data Types 10
5.5. Mapping of Associations 11
5.6. Mapping of Interfaces 12
5.7. Mapping of Operations 12
5.8. Mapping of Operation Parameters 13
5.9. Mapping of Notifications 14
5.10. Mapping of Lifecycle 15
5.11. Other Mappings 16
6. Mapping Issues 16
6.1. Mapping of Recursion 16
7. Mapping Patterns 16
7.1. UML Recursion 16
7.2. UML Conditional Pacs 16
7.3. XOR Relationship 17
8. Mapping Basics 17
8.1. UML-YANG or XMI-YANG 17
8.2. XMI Differences 17
9. Acknowledgements 17
10. IANA Considerations 17
11. Security Considerations 17
12. References 17
12.1. Normative References 17
12.2. Informative References 18
Appendix A. Example 18

Authors' Addresses 19

1. Introduction

As discussed in draft-lam-teas-usage-info-model-net-topology [5] a Data Model (DM) may be derived from an Information Model (IM). However, in order to assure a consistent and valid data modelling language representation that enables maximum interoperability, translation guidelines are required. A set of translation rules also assists in development of automated tooling.

This draft defines guidelines for translation of data modelled with UML [6] (as constrained by the ONF's UML Modeling Guidelines [7]) to YANG (defined in RFC6020 [2] and YANG Update [3]) including mapping of object classes, attributes, data types, associations, interfaces, operations and operation parameters, notifications, and lifecycle.

2. Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

3. Terminology

The following terms are defined in RFC6020 [2]

- o anydata
- o anyxml
- o augment
- o container
- o data node
- o identity
- o instance identifier
- o leaf
- o leaf-list
- o list
- o module

- o submodule

The following terms are defined in UML 2.4 [6]

- o association
- o attribute
- o data type
- o interface
- o object class
- o operation
- o parameter
- o signal (used to model notifications)

4. Overview

This document defines translation rules for all constructs used in a UML based IM to a data model using YANG.

While some mapping rules are straightforward, an IM in UML uses some constructs that cannot be mapped directly to a DM using YANG and conventions are described to make the translation predictable. Additionally, in some cases multiple mapping approaches are possible and selection among these is also necessary to assure interoperability.

Mapping guidelines for these constructs are provided in the following sections.

5. Mapping Guidelines

5.1. Mapping Guideline Considerations

Where "??" is inserted in the table, it means that the specific mapping is for further study as it is either as yet unclear how to map the construct or that there are multiple ways of doing the mapping and a single one needs to be selected.

A table will be included summarizing constructs in UML that do not directly map to YANG and where in this draft the associated guidelines for mapping these constructs will be provided.

5.2. Mapping of Object Classes

```
+-----------------------------------------------------------------+
|       Object Class --> "list" statement (key property) or       |
|                         "container" statement                   |
+-----------------------------------------------------------------+
| UML Artifact          | YANG Artifact         | Comment         |
+-----------------------------------------------------------------+
| documentation         | "description"         |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| superclass(es)        | ??                    |                 |
+-----------------------------------------------------------------+
| abstract              | abstract: "container" |                 |
|                       | not abstract: "list"  |                 |
+-----------------------------------------------------------------+
| objectCreationNotific-| ??                    |                 |
| ation                 |                       |                 |
+-----------------------------------------------------------------+
| objectDeletionNotific-| ??                    |                 |
| ation                 |                       |                 |
+-----------------------------------------------------------------+
| support               | "if-feature"          |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| condition             | "if-feature"          |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| operation             | "action"              |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| XOR                   | "choice"              |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| ??                    | "config"              |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| error notfication?    | "must"                |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| object identifier     | list::"key"           |                 |
|                       | substatement          |                 |
+-----------------------------------------------------------------+
| ??                    | list::"min-elements"  | min-elements    |
|                       | "max-elements"        |  default = 0    |
|                       | substatements         | max-elements    |
|                       |                       |  default=unbounded|
|                       |                       | mandatory       |
```

```
   |                       |                       |  default=false    |
   +-----------------------------------------------------------------+
   | Conditional PACs      | container::presence"  |                   |
   |                       | substatement          |                   |
   +-----------------------------------------------------------------+
   | hyperlink?            | "reference"           | Papyrus doesn't   |
   |                       | substatement          | support hyperlinks|
   +-----------------------------------------------------------------+
   | lifecycle stereotypes | "status"              | "current"         |
   |                       | substatement          | "deprecated"      |
   |                       |                       | "obsolete"        |
   |                       |                       |  default="current"|
   +-----------------------------------------------------------------+
   | ??                    | list::"unique"        |                   |
   |                       | substatement          |                   |
   +-----------------------------------------------------------------+
   | complex attribute     | "uses" substatement   |                   |
   +-----------------------------------------------------------------+
   | {<constraint>}        | "when" substatement   |                   |
   +-----------------------------------------------------------------+
```

Figure 1: Mapping of Object Classes

5.3. Mapping of Attributes

```
   +-----------------------------------------------------------------+
   |       Attribute --> "leaf" (single) or "leaf list" (multiple)     |
   |                        statement                                  |
   +-----------------------------------------------------------------+
   | UML Artifact          | YANG Artifact         | Comment           |
   +-----------------------------------------------------------------+
   | documentation         | "description"         |                   |
   |                       | substatement          |                   |
   +-----------------------------------------------------------------+
   | type                  | "type" substatement   |                   |
   |                       | (built-in or derived) |                   |
   +-----------------------------------------------------------------+
   | readOnly              | "config" substatement |                   |
   |                       | (false)               |                   |
   +-----------------------------------------------------------------+
   | isOrdered             | "ordered-by"          |                   |
   |                       | substatement          |                   |
   |                       |("system" or "user")   |                   |
   +-----------------------------------------------------------------+
   | multiplicity          | "min-elements" and    | min-elements      |
   |                       | "max-elements"        |  default = 0      |
   |                       | substatements         | max-elements      |
   |                       | [0..x]=>mandatory     |  default=unbounded|
```

```
   |                       |   substatement=false  | mandatory         |
   |                       | [1..x]=>mandatory     |  default=false    |
   |                       |   substatement=true   |                   |
   |                       |                       |                   |
   +-----------------------------------------------------------------+
   | defaultValue          | "default"             | If a default value|
   |                       | substatement          | exists and it is  |
   |                       |                       | the desired value,|
   |                       |                       | the parameter does|
   |                       |                       | not have to be    |
   |                       |                       | explicitly config-|
   |                       |                       | ured by the user. |
   +-----------------------------------------------------------------+
   | isInvariant           | "config"              |                   |
   |                       | substatement (false)  |                   |
   +-----------------------------------------------------------------+
   | valueRange            | "range" or "length"   |                   |
   |                       | substatement of "type"|                   |
   |                       | substatement          |                   |
   +-----------------------------------------------------------------+
   | passedById            | ??                    |                   |
   +-----------------------------------------------------------------+
   | support               | "if-feature"          |                   |
   |                       | substatement          |                   |
   +-----------------------------------------------------------------+
   | condition             | "if-feature"          |                   |
   |                       | substatement          |                   |
   +-----------------------------------------------------------------+
   | error notfication?    | "must"                |                   |
   |                       | substatement          |                   |
   +-----------------------------------------------------------------+
   | hyperlink?            | "reference"           | Papyrus doesn't   |
   |                       | substatement          | support hyperlinks|
   +-----------------------------------------------------------------+
   | lifecycle stereotypes | "status"              | "current"         |
   |                       | substatement          | "deprecated"      |
   |                       |                       | "obsolete"        |
   |                       |                       |  default="current"|
   +-----------------------------------------------------------------+
   | unit?                 | "units" substatement  |                   |
   +-----------------------------------------------------------------+
   | {<constraint>}        | "when" substatement   |                   |
   +-----------------------------------------------------------------+
```

Figure 2: Mapping of Attributes

5.4. Mapping of Types

UML Artifact	YANG Artifact	Comment
Primitive Type	??	new built-in type?
Enumeration	"enum" statement	
Basic Data Type	"typeDef" statement	
Complex Data Type	"grouping" statement	

Figure 3: Mapping of Types

Note: YANG allows also in-line enumerations which are not possible in UML

5.4.1. Mapping of Primitive Types

Primitive Type -> new built-in type?		
UML Artifact	YANG Artifact	Comment
documentation	??	

Figure 4: Mapping of Primitive Types

5.4.2. Mapping of Enumeration Types

Enumeration Type -> "enum" statement		
UML Artifact	YANG Artifact	Comment
documentation	"description" substatement	
literal name	"value" substatement	
hyperlink?	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current
??	"if-feature" statement	

Figure 5: Mapping of Enumeration Types

5.4.3. Mapping of Basic Data Types

Basic Data Type -> "typeDef" statement		
UML Artifact	YANG Artifact	Comment
documentation	"description" substatement	
type	"type" substatement (built-in type)	
defaultValue	"default" substatement	If a default value exists and it is the desired value, the parameter does not have to be explicitly configured by the user.
hyperlink?	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current
unit?	"units" statement	

Figure 6: Mapping of Basic Data Types

5.4.4. Mapping of Complex Data Types

Complex Data Type -> "grouping" statement		
UML Artifact	YANG Artifact	Comment
documentation	"description" substatement	
not used	"action" substatement	
XOR	"choice" substatement	
hyperlink?	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current
complex attribute	"uses" statement	

Figure 7: Mapping of Complex Data Types

5.5. Mapping of Associations

Associations		
UML Artifact	YANG Artifact	Comment
Inheritance	"extension" or "augment" statement	
Composition	"container" statement	
Aggregation	"container" statement	

Figure 8: Mapping of Associations

Figure 9: Association Mapping Examples (Available in PDF or HTML versions)

5.6. Mapping of Interfaces

```
+-------------------------------------------------------------------+
|                     UML Interface -> Container?                   |
+-------------------------------------------------------------------+
| documentation         | "description"         |                   |
|                       | substatement          |                   |
+-------------------------------------------------------------------+
| abstract              | ??                    |                   |
+-------------------------------------------------------------------+
| support               | "if-feature"          |                   |
|                       | substatement          |                   |
+-------------------------------------------------------------------+
| condition             | "if-feature"          |                   |
|                       | substatement          |                   |
+-------------------------------------------------------------------+
```

Figure 10: Mapping of Interfaces

5.7. Mapping of Operations

Operation -> "action" and "rpc" statements		
documentation	"description" substatement	
pre-condition	??	
post-condition	??	
input parameter	"input" substatement	
output parameter	"output" substatement	
operation exceptions	??	
isOperationIdempotent	??	
isAtomic	??	
support	"if-feature" substatement	
condition	"if-feature" substatement	
hyperlink?	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current

Figure 11: Mapping of Operations

Note: The difference between an action and an rpc is that an action is tied to a node in the data tree, whereas an rpc is not.

5.8. Mapping of Operation Parameters

Operation Parameters		
documentation	"description" substatement	
direction	"input" or "output" substatement	
type	see mapping of attribute types (grouping, leaf, leaf-list, list, typedef, uses)	
isOrdered		
multiplicity		
defaultValue	??	
valueRange	??	
passedByID	??	
support	"if-feature" substatement	
condition	"if-feature" substatement	
XOR	"choice" substatement	
error notification?	"must" substatement	
complex parameter	"uses" substatement	

Figure 12: Mapping of Operation Parameters

5.9. Mapping of Notifications

Signal -> "notification" statement		
documentation	"description" substatement	
support	"if-feature" substatement	
condition	"if-feature" substatement	
XOR	"choice" substatement	
error notification?	"must" substatement	
hyperlink?	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current
complex attribute	"uses" substatement	

Figure 13: Mapping of Notifications

5.10. Mapping of Lifecycle

UML Lifecycle		
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current

Figure 14: Mapping of Lifecycle

5.11. Other Mappings

UML Lifecycle		
Conditional Package	"container" statement with "presence" substatement	
Primitive Type	Built-In Type	
Package	Submodule	

Figure 15: Other Mappings

6. Mapping Issues

When translating from UML information models to YANG data models some mapping rules are straightforward, and some are not. This section provides considerations and recommendations for the more complex translations.

6.1. Mapping of Recursion

- o Statically define a number of recursion levels

- o Reference Based Approach

In the static approach, some number of recursion levels is pre-configured. In the Reference-based approach, a flat list is maintained using hierarchical identities. The reference-based approach is generally preferred because there is no arbitrary limitation set in the solution.

7. Mapping Patterns

7.1. UML Recursion

TBD

7.2. UML Conditional Pacs

May use the "presence" property of the container statement?

Figure 16: Mapping of Conditional Packages (Available in PDF or HTML versions)

7.3. XOR Relationship

Use the "choice" property of the container statement.

8. Mapping Basics

8.1. UML-YANG or XMI-YANG

Figure 17: Example UML to YANG Mapping (Available in PDF or HTML versions)

Figure 18: Example XMI (Papyrus) to YANG Mapping (Available in PDF or HTML versions)

8.2. XMI Differences

Figure 19: Example XMI (Papyrus) / XMI (RSA) Differences (Available in PDF or HTML versions)

Figure 20: Example XMI (Papyrus) / XMI (RSA) Differences (detailed) (Available in PDF or HTML versions)

9. Acknowledgements

10. IANA Considerations

This memo includes no request to IANA.

11. Security Considerations

This document defines defines guidelines for translation of data modeled with UML to YANG. As such, it doesn't contribute any new security issues beyond those discussed in Sec. 16 of RFC6020 [2].

12. References

12.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

12.2. Informative References

[2] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[3] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", draft-ietf-netmod-rfc6020bis-05 (work in progress), May 2015.

[4] Galimberti, G., Kunze, R., Lam, H., Hiremagalur, D., Grammel, G., Fang, L., and G. Ratterree, "A YANG model to manage the optical interface parameters of "G.698.2 single channel" in DWDM applications", draft-dharini-netmod-g-698-2-yang-03 (work in progress), March 2015.

[5] Lam, H., Varma, E., Doolan, P., Davis, N., Zeuner, B., Betts, M., Busi, I., and S. Mansfield, "Usage of IM for network topology to support TE Topology YANG Module Development", draft-lam-teas-usage-info-model-net-topology-00 (work in progress), March 2015.

[6] OMG, "Unified Modeling Language (UML)", 2011, <http://www.omg.org/spec/UML/2.4/>.

[7] OMG, "ONF TR-514 v1.0 UML Modeling Guidelines", 2015, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/UML_Modeling_Guidelines_V1.0.pdf>.

Appendix A. Example

The YANG data schema (in tree format) shown below was extracted from dharini-netmod-g-698-2-yang [4] and represents the same data as UML model appearing in Figure 23 after the tree format. Note: The color code used in the tree format corresponds to the color code used in the UML class diagram.

Figure 21: Interfaces Tree (Available in PDF or HTML versions)

Figure 22: Notifications Tree (Available in PDF or HTML versions)

Figure 23: Interfaces UML Model (Available in PDF or HTML versions)

Authors' Addresses

Scott Mansfield (editor)
Ericsson Inc.
USA

Phone: +1 724 931 9316
Email: scott.mansfield@ericsson.com

Bernd Zeuner
Deutsche Telekom AG
Heinrich-Hertz-Str, 3-7
Darmstadt 64295
Germany

Phone: +49 6151 58-12086
Email: b.zeuner@telekom.de

Nigel Davis
Ciena
United Kingdom

Email: ndavis@ciena.com

Yun Xiang
Fiberhome
China

Email: yunxig@fiberhome.com.cn

Yuji Tochio
Fujitsu
Japan

Email: tochio@jp.fujitsu.com

Hing-Kam Lam
Alcatel Lucent
USA

Phone: +1 732 331 3476
Email: kam.lam@alcatel-lucent.com

Eve Varma
Alcatel Lucent
USA

Email: eve.varma@alcatel-lucent.com