NETMOD Working Group
Internet-Draft
Intended status: Informational
Expires: March 11, 2016

S. Mansfield, Editor
Ericsson Inc.
B. Zeuner
Deutsche Telekom AG
N. Davis
Ciena
X. Yun
Fiberhome
Y. Tochio
Fujitsu
K. Lam
E. Varma
Alcatel Lucent
September 8, 2015

Guidelines for Translation of UML Information Model to YANG Data Model

draft-mansfield-netmod-uml-to-yang-01

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 March 11, 2016.

Copyright Notice

Table of Contents

1. **Introduction**
2. **Terminology**
3. **Overview**
4. **Mapping Guidelines**
   - 4.1 Mapping Guideline Considerations
   - 4.2 Mapping of Object Classes
   - 4.3 Mapping of Attributes
   - 4.4 Mapping of Types
     - 4.4.1 Mapping of Primitive Types
     - 4.4.2 Mapping of Enumeration Types
     - 4.4.3 Mapping of Basic Data Types
     - 4.4.4 Mapping of Complex Data Types
   - 4.5 Mapping of Associations
   - 4.6 Mapping of Interfaces
   - 4.7 Mapping of Operations
   - 4.8 Mapping of Operation Parameters
   - 4.9 Mapping of Notifications
   - 4.10 Mapping of Lifecycle
   - 4.11 Other Mappings
5. **Mapping Issues**
   - 5.1 Using types defined in YANG
   - 5.2 YANG 1.0 or YANG 1.1
   - 5.3 Mapping of UML Packages
6. **Mapping Patterns**
   - 6.1 UML Recursion
   - 6.2 UML Conditional Pacs
   - 6.3 XOR Relationship
   - 6.4 Mapping of UML Support and Condition
7. **Mapping Basics**
   - 7.1 UML-YANG or XMI-YANG
8. **Acknowledgements**
9. **IANA Considerations**
10. **Security Considerations**
11. **Informative References**

A. Example

Authors' Addresses

Figures

Figure 1: Mapping of Object Classes
Figure 2: Example of Abstract Object Class Mapping (Available in PDF or HTML versions)
Figure 3: Example of Non-Abstract Object Class Mapping (Available in PDF or HTML versions)
Figure 4: Mapping of Attributes
Figure 5: Example of Attribute Mapping (Available in PDF or HTML versions)
Figure 6: Mapping of Types
Figure 7: Mapping of Primitive Types
Figure 8: Mapping of Enumeration Types
Figure 9: Example of Enumeration Type Mapping (Available in PDF or HTML versions)
Figure 10: Mapping of Basic Data Types
Figure 11: Example of Basic Data Type Mapping (Available in PDF or HTML versions)
Figure 12: Mapping of Complex Data Types
Figure 13: Example of Complex Data Type Mapping (Available in PDF or HTML versions)
Figure 14: Mapping of Associations
Figure 15: Association Mapping Example 1 (Available in PDF or HTML versions)
Figure 16: Association Mapping Example 2 (Available in PDF or HTML versions)
Figure 17: Association Mapping Example 3 (Available in PDF or HTML versions)
Figure 18: Mapping of Interfaces (grouping of operations
Figure 19: Mapping of Operations
Figure 20: Operation Mapping Example (Available in PDF or HTML versions)
Figure 21: Mapping of Operation Parameters
Figure 22: Parameter Mapping Example (Available in PDF or HTML versions)
Figure 23: Mapping of Notifications
Figure 24: Notification Mapping Example (Available in PDF or HTML versions)
Figure 25: Mapping of Lifecycle
Figure 26: Other Mappings
Figure 27: Re-engineered Example
Figure 28: IP Address Mapping Examples (Available in PDF or HTML versions)
Figure 29: Reference Based Approach (Available in PDF or HTML versions)
Figure 30: Specific Number of Recusions (Available in PDF or HTML versions)

Figure 31: Mapping of Conditional Packages (Available in PDF or HTML versions)
Figure 32: Support and Condition Mapping Example (Available in PDF or HTML versions)
Figure 33: Example UML to YANG Mapping (Available in PDF or HTML versions)
Figure 34: Example XMI (Papyrus) to YANG Mapping (Available in PDF or HTML versions)
Figure 35: Interfaces Tree (Available in PDF or HTML versions)
Figure 36: Notifications Tree (Available in PDF or HTML versions)
Figure 37: Interfaces UML Model (Available in PDF or HTML versions)

1. Introduction

As discussed in draft-lam-teas-usage-info-model-net-topology [6] 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 [7] (as constrained by the ONF's UML Modeling Guidelines [8]) to YANG (defined in RFC6020 [1] and YANG Update [4]) including mapping of object classes, attributes, data types, associations, interfaces, operations and operation parameters, notifications, and lifecycle.

2. Terminology

The following terms are defined in RFC6020 [1]

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

The following terms are defined in UML 2.4 [7]

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

3. 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.

4. Mapping Guidelines

4.1 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.

4.2 Mapping of Object Classes

Object Class --> "list" statement (key property, multiple instances) or "container" statement (single instance) or "grouping" statement		
UML Artifact	YANG Artifact	Comment
documentation	"description" substatement	
superclass(es)	abstract: "grouping" |not abstract:"augment"??	
abstract	"grouping" statement	
objectCreationNotification [YES/NO/NA]	"notification" statement	Goes beyond the simple "a notification has to be sent": a tool can construct the signature of the notification by reading the created object.
objectDeletionNotification [YES/NO/NA]	"notification" statement	Goes beyond the simple "a notification has to be sent": a tool can construct the signature of the notification by reading the deleted object. (i.e. not necessary to provide the attributes of the deleted object.
support	"if-feature" substatement	Support and condition belong together. If the "support" is conditional, then the "condition" explains the conditions under which the class has to be supported.
condition		
operation	"action" substatement	YANG 1.0 supports only rpc -> add prefix to the rpc name; i.e. objectClass::rpc; "action" requires YANG 1.1
XOR	"choice" substatement	
constraints between attribute values -> error notfication?? OCL??	"must" substatement	
object identifier	list::"key" substatement	
muliplicity on association	list::"min-elements" "max-elements" substatements	min-elements default = 0 max-elements default=unbounded mandatory default=false
Conditional PACs	container::presence" substatement	
hyperlink??	"reference" substatement	Papyrus doesn't support hyperlinks

lifecycle stereotypes	"status" substatement	"current" "deprecated" "obsolete" default="current"
constraint property	list::"unique" substatement	
complex attribute	"uses" substatement	use of a complex data type as the type of the attribute; e.g., date and time, object creation data
{<constraint>}	"when" substatement	

Figure 1: Mapping of Object Classes

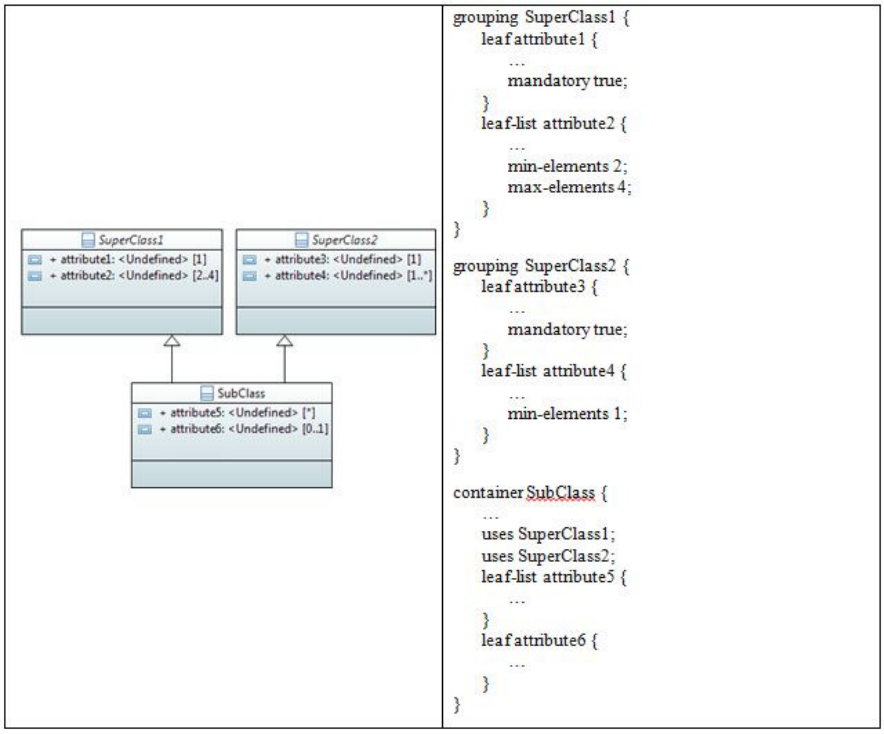
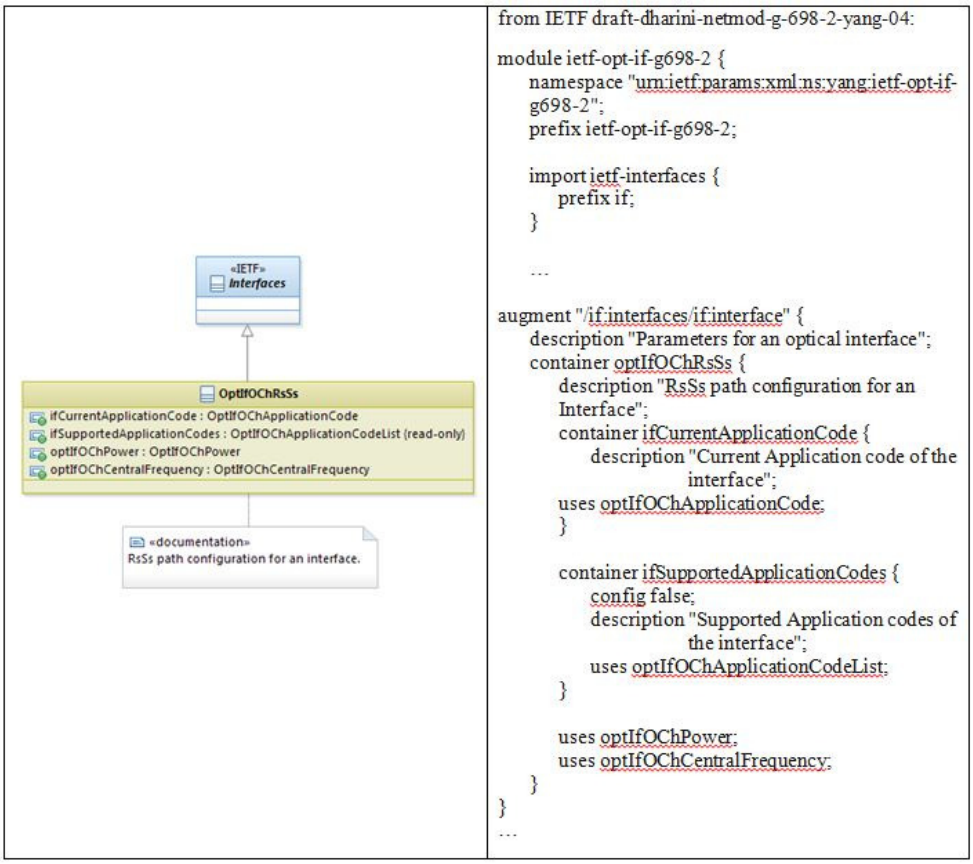


Figure 2: Example of Abstract Object Class Mapping (Available in PDF or HTML versions)




```
from IETF draft-dharini-netmod-g-698-2-yang-04:

module ietf-opt-if-g698-2 {
    namespace "urn:ietf:params:xml:ns:yang:ietf-opt-if-
    g698-2";
    prefix ietf-opt-if-g698-2;

    import ietf-interfaces {
        prefix if;
    }

    ...

augment "/if:interfaces/if:interface" {
    description "Parameters for an optical interface";
    container optIfOChRsSs {
        description "RsSs path configuration for an
        Interface";
        container ifCurrentApplicationCode {
            description "Current Application code of the
                         interface";
        uses optIfOChApplicationCode;
        }

        container ifSupportedApplicationCodes {
            config false;
            description "Supported Application codes of
                         the interface";
            uses optIfOChApplicationCodeList;
        }

        uses optIfOChPower;
        uses optIfOChCentralFrequency;
    }
}
...
```

Figure 3: Example of Non-Abstract Object Class Mapping (Available in PDF or HTML versions)

4.3 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)	config default = true
isOrdered	"ordered-by" substatement ("system" or "user")	ordered-by default = system
multiplicity	"mandatory" or "min-elements" and "max-elements" substatements [0..1]=>no mapping needed; is default substatement=false [1]=>mandatory substatement=true [0..x]=> no mapping needed; is default [1..x]=> min-elements substatement = 1 [0..3]=> max-elements substatement = 3	min-elements default = 0 max-elements default=unbounded mandatory default=false
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.
isInvariant	"extension" substatement -> ompExt:isInvariant	
valueRange	"range" or "length" substatement of "type" substatement	
passedByReference	if passedByReference = true -> type leafref { path "/<object>/<objectidentifier>"} if passedByReference = false -> either "list" statement (key property, multiple instances) or "container" statement(single instance)	Relevant only to attributes that have an object class defined as their type.
support	"if-feature" substatement	Support and condition belong together. If the "support" is conditional, then the "condition" explains the conditions under which the class has to be supported.
condition		
error notfication??	"must" substatement	
hyperlink??	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current" "deprecated"

		"obsolete" default="current"
unit??	"units" substatement	
{<constraint>}	"when" substatement	

Figure 4: Mapping of Attributes

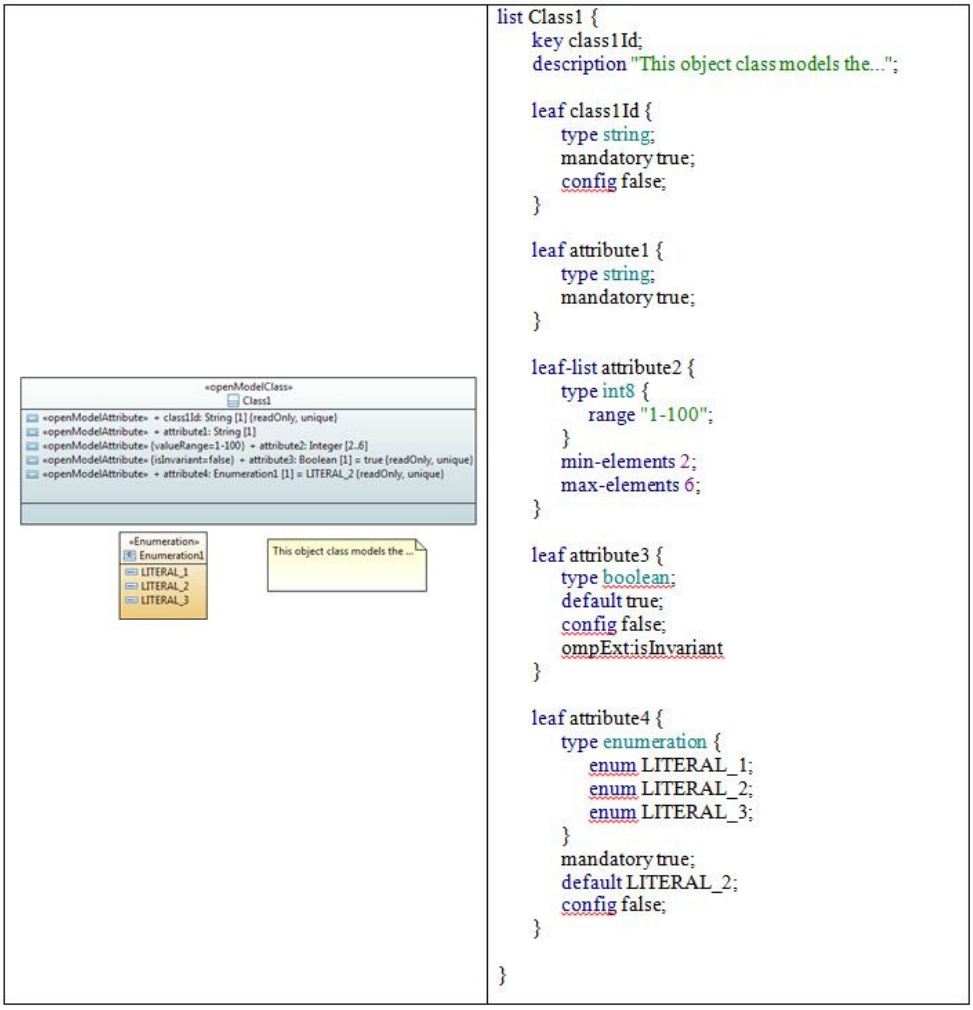


Figure 5: Example of Attribute Mapping (Available in PDF or HTML versions)

4.4 Mapping of Types

Types		
UML Artifact	YANG Artifact	Comment
Primitive Type	Built-In Type if defined; otherwise ??	e.g., Integer new built-in type??
Enumeration	"enum" statement	
Basic Data Type	"typeDef" statement	e.g., MAC address, IPv4 Address
Complex Data Type	"grouping" statement	e.g., date-time object creation data

Figure 6: Mapping of Types

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

4.4.1 Mapping of Primitive Types

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

Figure 7: Mapping of Primitive Types

4.4.2 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 8: Mapping of Enumeration Types

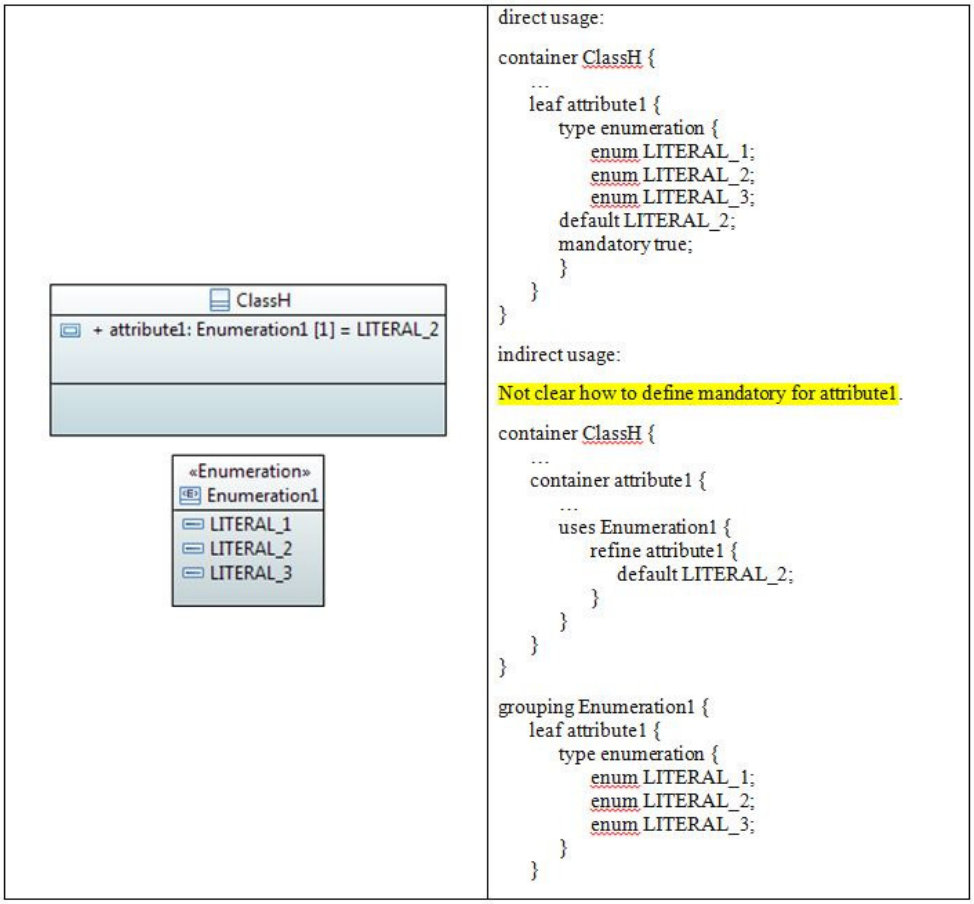


Figure 9: Example of Enumeration Type Mapping (Available in PDF or HTML versions)

4.4.3 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 10: Mapping of Basic Data Types

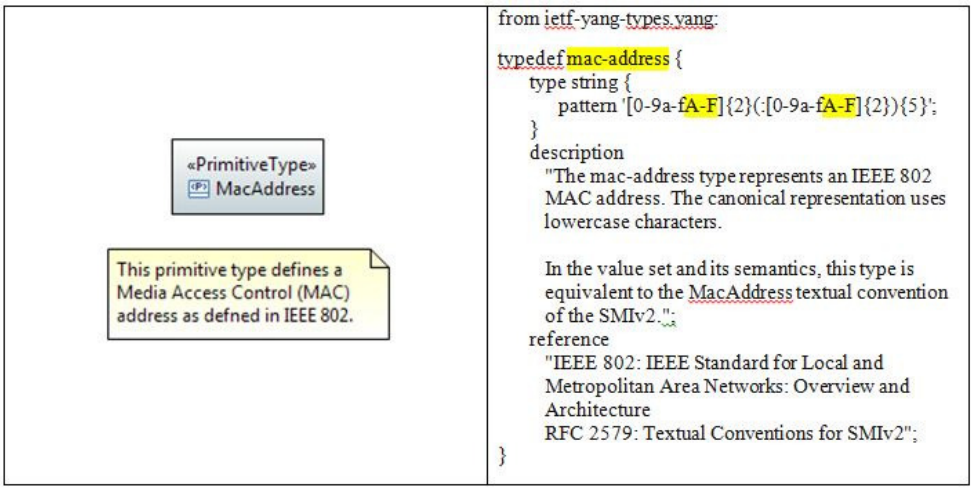


Figure 11: Example of Basic Data Type Mapping (Available in PDF or HTML versions)

4.4.4 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 12: Mapping of Complex Data Types

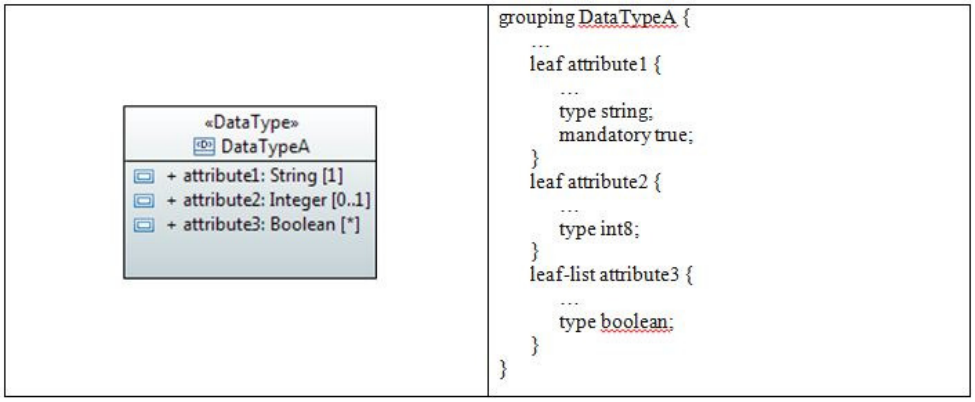


Figure 13: Example of Complex Data Type Mapping (Available in PDF or HTML versions)

4.5 Mapping of Associations

Associations		
UML Artifact	YANG Artifact	Comment
Inheritance	"grouping" or "augment" statement	Multiple inheritance can also be mapped using "groupings" Need to define when augment is used. Note: Augmentation can be conditional.
Composition with "passed by value"	"container" statement containing "list" statement(s) (multiple contained instances) or "container" state-ment(s) (single contained instances)	How to map "passed by reference"??
Aggregation with "passed by reference"	"leafref" statement	How to map "passed by value"??

Figure 14: Mapping of Associations

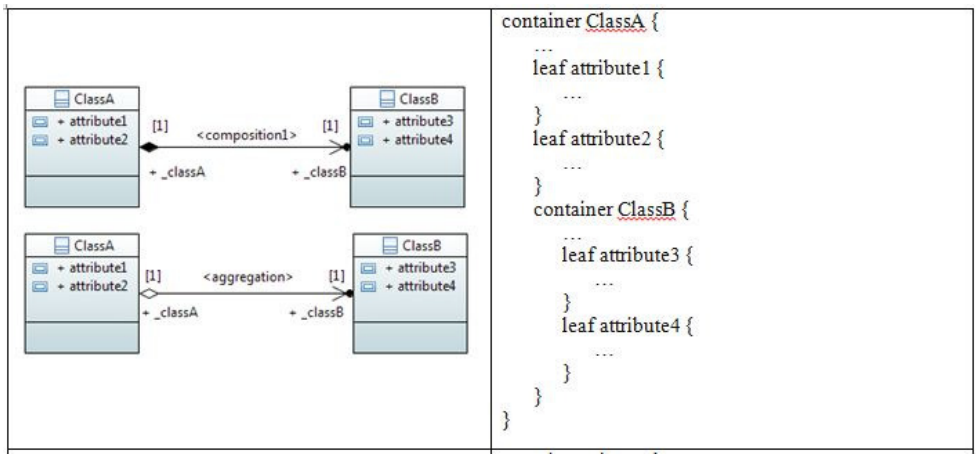


Figure 15: Association Mapping Example 1 (Available in PDF or HTML versions)

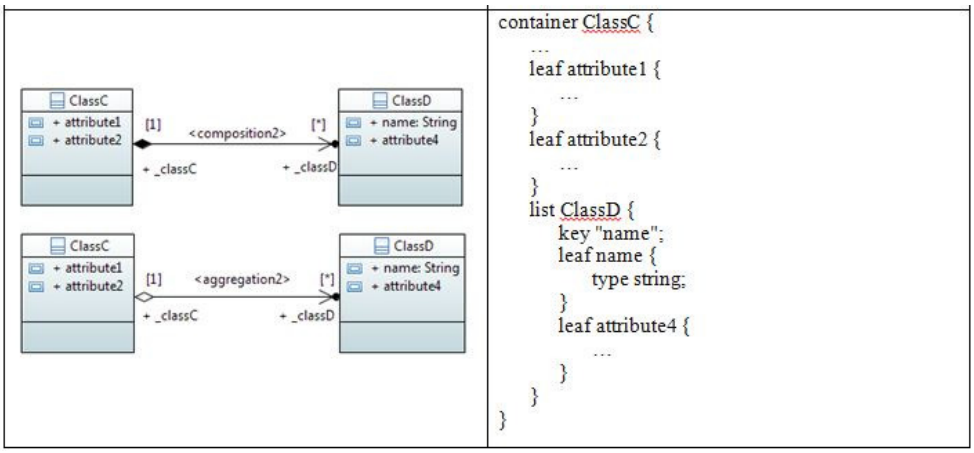


Figure 16: Association Mapping Example 2 (Available in PDF or HTML versions)

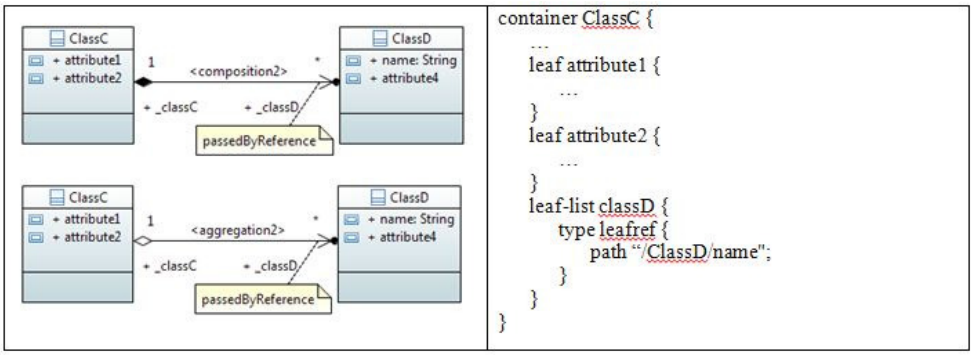


Figure 17: Association Mapping Example 3 (Available in PDF or HTML versions)

4.6 Mapping of Interfaces

UML Interface -> Container??		
documentation	"description" substatement	
abstract	??	
support	"if-feature" substatement	Support and condition belong together. If the "support" is conditional, then the "condition" explains the conditions under which the class has to be supported.
condition		

Figure 18: Mapping of Interfaces (grouping of operations

4.7 Mapping of Operations

Operation -> "action" and "rpc" statements		
(RFC 6020: 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 associated at the module level.)		
documentation	"description" substatement	
pre-condition	"extension" substatement -> ompExt:preCondition	RFC 6020; During the NETCONF <edit-config> processing errors are already sent for: - Delete requests for non-existent data. - Create requests for existing data. - Insert requests with "before" or "after" parameters that do not exist.
post-condition	"extension" substatement ompExt:postCondition	
input parameter	"input" substatement	
output parameter	"output" substatement	
operation exceptions	"extension" substatement ompExt:operationExceptions	
isOperationIdempotent	"extension" substatement ompExt:isOperationIdempotent	
isAtomic	"extension" substatement ompExt:isAtomic	Necessary?? Not in UML Guidelines (TR-514); needs to be added??
support condition	"if-feature" substatement	Support and condition belong together. If the "support" is conditional, then the "condition" explains the conditions under which the class has to be supported.
hyperlink??	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current

Figure 19: Mapping of Operations

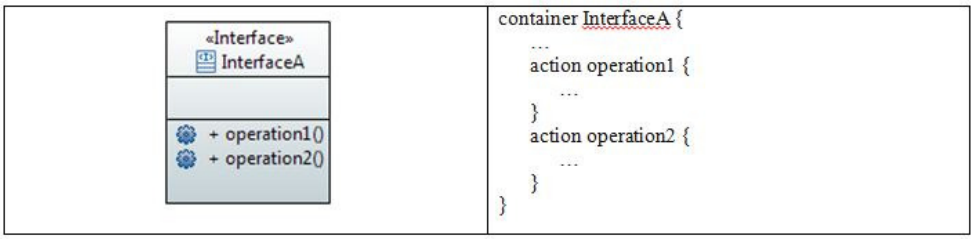


Figure 20: Operation Mapping Example (Available in PDF or HTML versions)

4.8 Mapping of Operation Parameters

Operation Parameters		
documentation	"description" substatement	
direction	"input" or "output" substatement	
type	see mapping of attribute types	
isOrdered	(grouping, leaf, leaf-list, list, typedef, uses)	
multiplicity		
defaultValue		
valueRange		
passedByReference	if passedByReference = true -> type leafref { path "/<object>/<objectidentifier>"} if passedByReference = false -> either "list" statement (key property, multiple instances) or "container" statement(single instance)	Relevant only to attributes that have an object class defined as their type.
support condition	"if-feature" substatement not defined for input and output substatements in YANG??	Support and condition belong together. If the "support" is conditional, then the "condition" explains the conditions under which the class has to be supported.
XOR	"choice" substatement	
error notification??	"must" substatement	
complex parameter	"uses" substatement	

Figure 21: Mapping of Operation Parameters

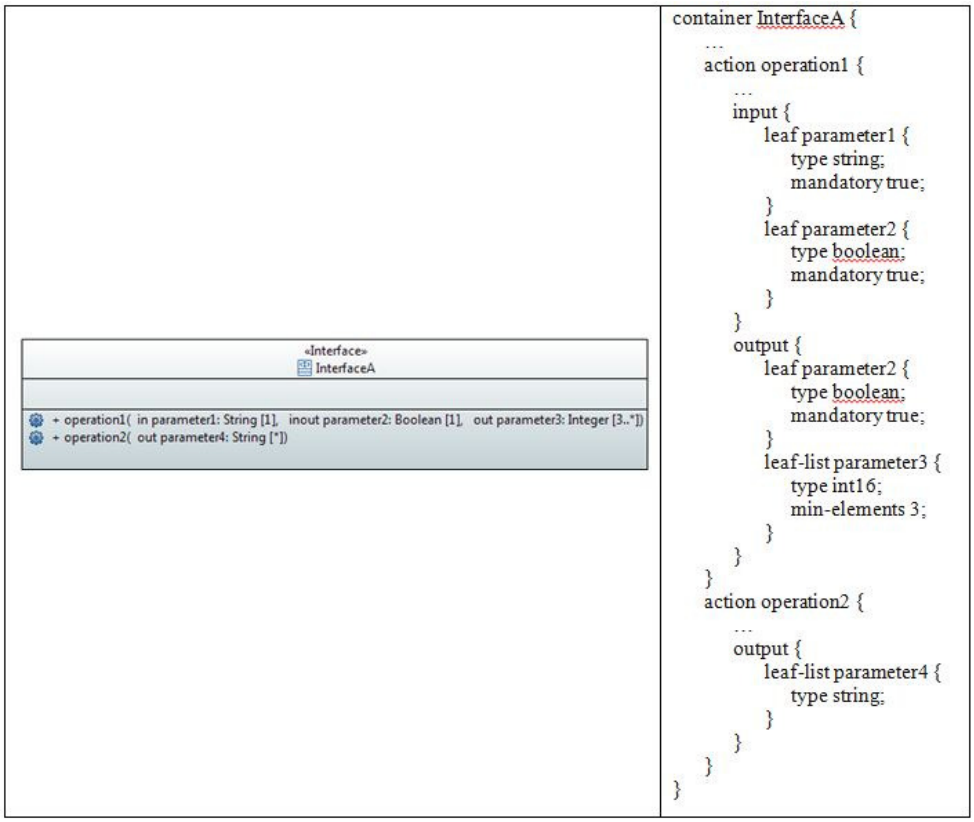


Figure 22: Parameter Mapping Example (Available in PDF or HTML versions)

4.9 Mapping of Notifications

Signal -> "notification" statement		
documentation	"description" substatement	
support	"if-feature" substatement	Support and condition belong together. If the "support" is conditional, then the "condition" explains the conditions under which the class has to be supported.
condition		
XOR	"choice" substatement	
error notification??	"must" substatement	
hyperlink??	"reference" substatement	Papyrus doesn't support hyperlinks
lifecycle stereotypes	"status" substatement	"current", "deprecated", "obsolete" default=current
attributes	see mapping of attribute types (grouping, leaf, leaf-list, container, list, typedef, uses)	
complex attribute	"uses" substatement	

Figure 23: Mapping of Notifications

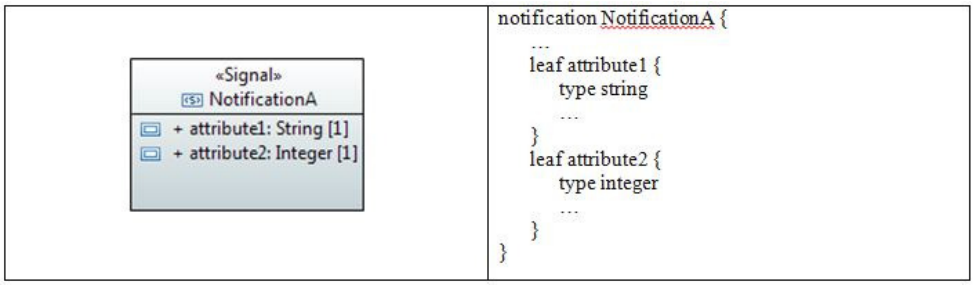


Figure 24: Notification Mapping Example (Available in PDF or HTML versions)

4.10 Mapping of Lifecycle

UML Lifecycle		
lifecycle stereotypes	"status" substatement	YANG: "current", "deprecated", "obsolete" default=current UML: <<Example>>, <<Experimental>>, <<Faulty>>, <<Preliminary>>, <<Obsolete>>, <<LikelyToChange>> How to map or enhance??

Figure 25: Mapping of Lifecycle

4.11 Other Mappings

UML Lifecycle		
Conditional Package	"container" statement with "presence" substatement	
Package??	Submodule	

Figure 26: Other Mappings

5. Mapping Issues

5.1 Using types defined in YANG

Many common types (primitive and complex) are already defined in YANG. E.g., ietf-inet-types, ietf-yang-types (others to be investigated):

```
ietf-inet-types
 - domain name and URI types
   - DomainName
   - Uri
   - <<Union>> Host
 - types related to IP addresses and hostnames
   - Ip4Address
   - Ipv4AddressNoZone
   - Ipv4Prefix
   - Ipv6Address
   - Ipv6AddressNoZone
   - Ipv6Prefix
   - <<Union>> IpAddress
   - <<Union>> IpAddressNoZone
   - <<Union>> IpPrefix
 - types related to protocol fields
   - IpVersion
   - DSCP
   - IpV6FlowLabel
   - PortNumber
```

```
- ietf-yang-types
 - Counter32
 - Counter64
 - DateAndTime
 - DottedQuad
 - Gauge32
 - Gauge64
 - HexString
 - MacAddress
 - ObjectIdentifier
 - ObjectIdentifier128
 - PhysAddress
 - Timestamp
 - Timeticks
 - Uuid
 - Xpath1.0
 - YangIdentifier
 - ZeroBasedCounter32
 - ZeroBasedCounter64
```

Figure 27: Re-engineered Example

It is proposed to define for the commonly used YANG types corresponding UML primitive or complex data types respectively. These types will be available (by default) for use in all UML information models. This "re-engineering" needs to be done without making the UML models YANG-depended.

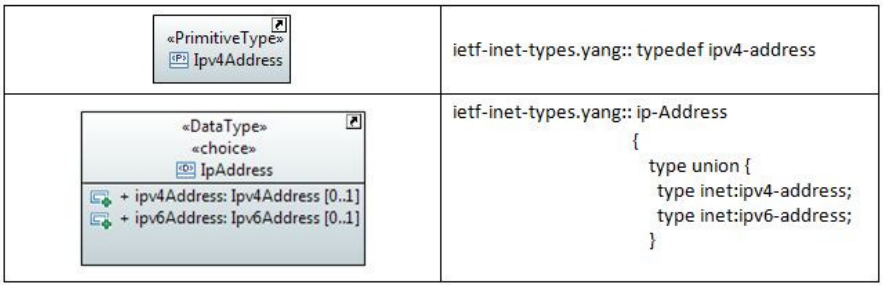


Figure 28: IP Address Mapping Examples (Available in PDF or HTML versions)

5.2 YANG 1.0 or YANG 1.1

YANG 1.0 is approved and defined in RFC6020 [1].

YANG 1.1 is not currently approved and its definition is ongoing in draft-ietf-netmod-rfc6020bis [4]. Main enhancements are the action and anydata statements.

5.3 Mapping of UML Packages

Need to define mapping rules for UML package into YANG modules or the new draft YANG package statement (draft-bierman-netmod-yang-package [3])?

6. Mapping Patterns

6.1 UML Recursion

As YANG defines hierarchical data store, any instances that need to store recursive containment will require translation. A mapping between object-oriented store and a hierarchical store is possible; however, there is more than one option:

- Reference Based Approach - have a flat list of objects, where the objects are linked into a hierarchy using references. An example of a two-way navigable approach is in RFC7223 [2]. See example in Figure 29 below.
- Assume some specific number of "recursions"; i.e., specify some default number of recursion levels, and define a configurable parameter to allow changing the number of levels. See example in Figure 30 below, in which there is one level of recursion.

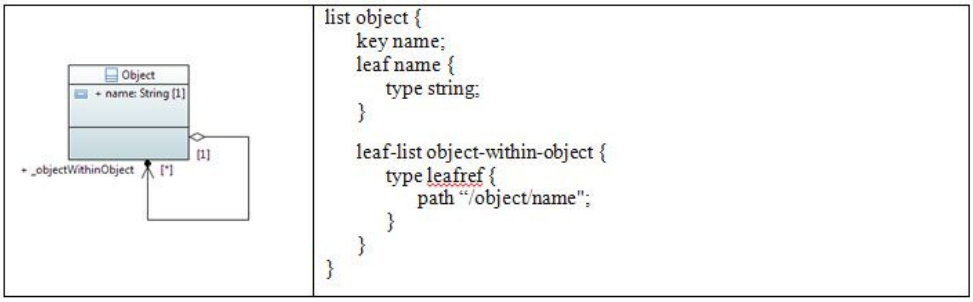


Figure 29: Reference Based Approach (Available in PDF or HTML versions)

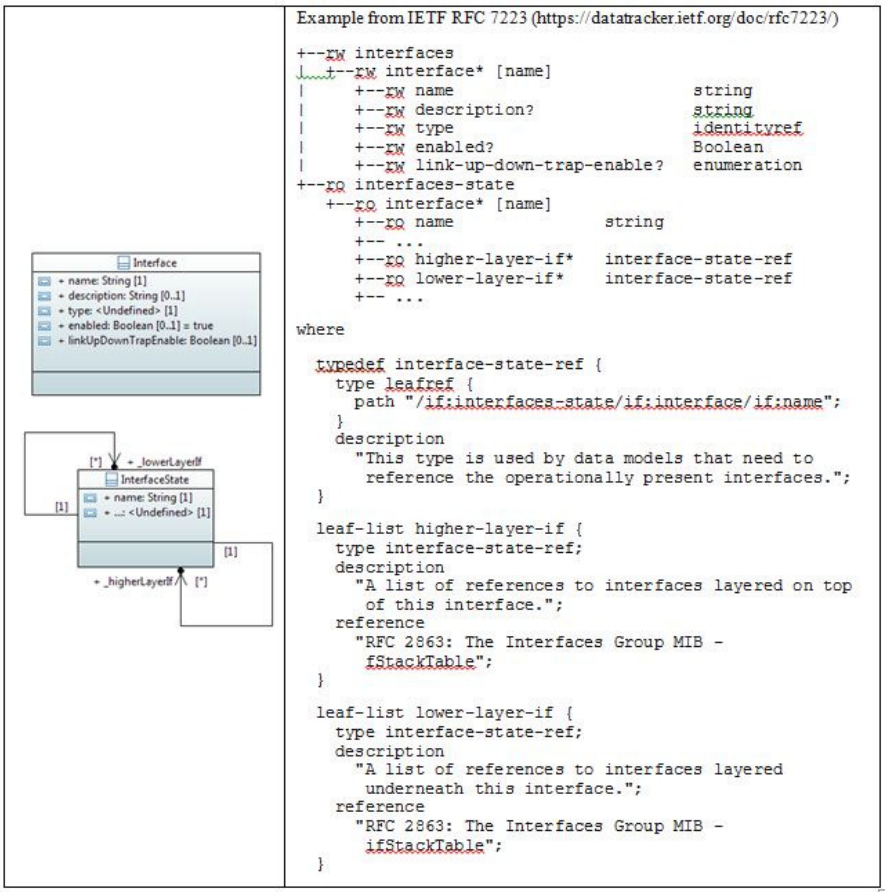
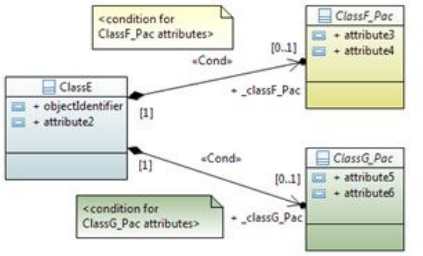


Figure 30: Specific Number of Recusions (Available in PDF or HTML versions)

The reference-based approach is generally preferred because there is no arbitrary limitation set in the solution.

6.2 UML Conditional Pacs

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




```
list ClassE {
    key "objectIdentifier";
    leaf objectIdentifier {
        type string;
    }
    ...
    leaf attribute2 {
        ...
    }
    container ClassF_Pac {
        presence " <condition for ClassF_Pac attributes>";
        ...
        leaf attribute3 {
            ...
        }
        leaf attribute4 {
            ...
        }
    }
    container ClassG_Pac {
        presence " <condition for ClassG_Pac attributes>";
        ...
        leaf attribute5 {
            ...
        }
        leaf attribute6 {
            ...
        }
    }
}
```

Figure 31: Mapping of Conditional Packages (Available in PDF or HTML versions)

6.3 XOR Relationship

Use the "choice" property of the container statement.

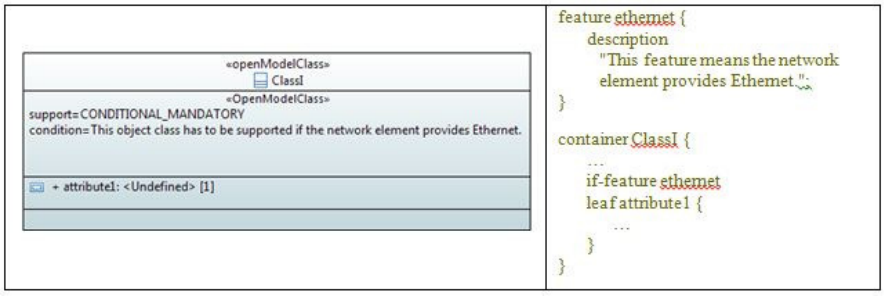
6.4 Mapping of UML Support and Condition

The UML Modeling Guidelines [8] define support and condition for all UML artifacts (M - Mandatory, O - Optional, C - Conditional, CM - Conditional-Mandatory, CO - Conditional-Optional). Support qualifies the support of the artifact at the management interface. Condition contains the condition for the condition-related support qualifiers.

M - Mandatory maps to the "mandatory" substatement in choice and leaf or to the "min-elements" substatement in leaf-list and list.

O - Optional need not be mapped since the per default the "mandatory" and "min-elements" substaments define optional.

All conditional UML support qualifiers are mapped to the "if-feature" substatement.




```
feature ethernet {
    description
      "This feature means the network
      element provides Ethernet.";
}

container ClassI {
    ...
    if-feature ethernet
    leaf attribute1 {
        ...
    }
}
```

Figure 32: Support and Condition Mapping Example (Available in PDF or HTML versions)

7. Mapping Basics

7.1 UML-YANG or XMI-YANG

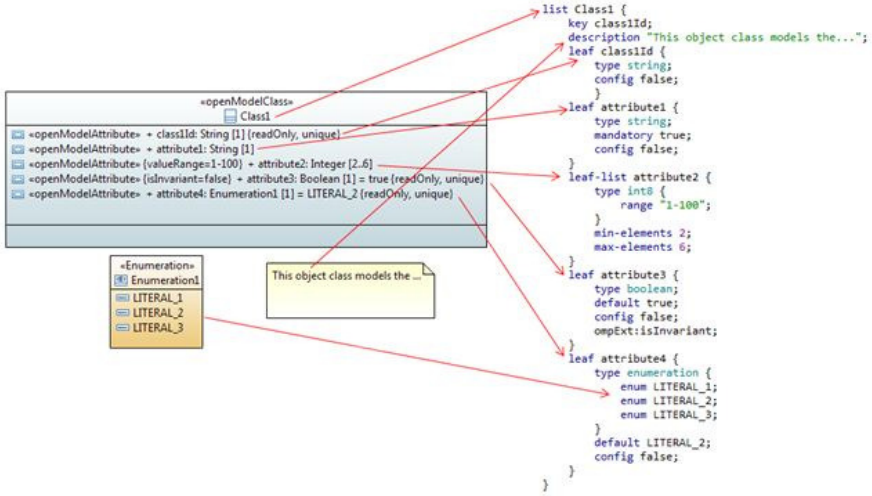


Figure 33: Example UML to YANG Mapping (Available in PDF or HTML versions)

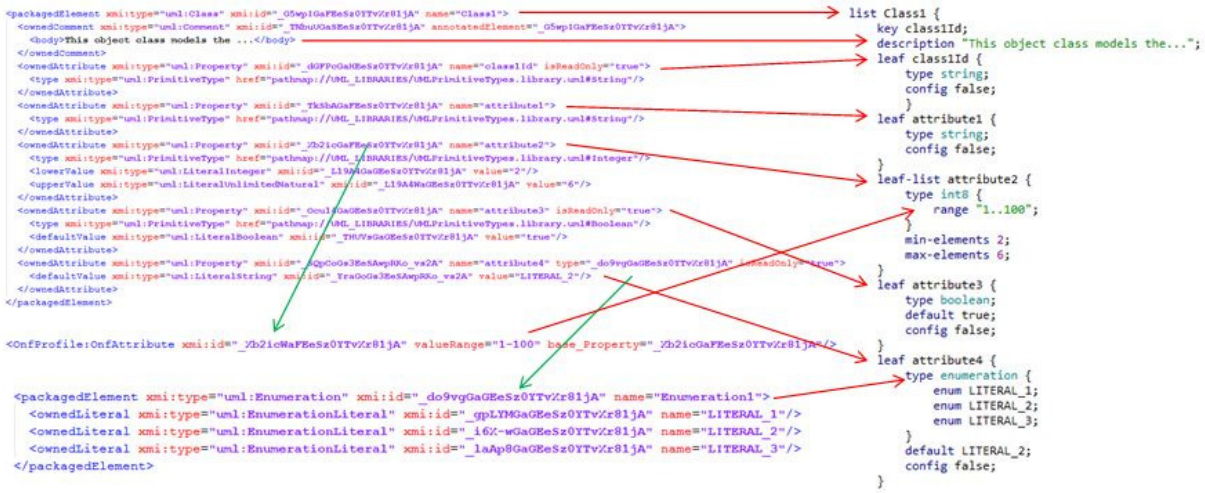


Figure 34: Example XMI (Papyrus) to YANG Mapping (Available in PDF or HTML versions)

8. Acknowledgements

9. IANA Considerations

This memo includes no request to IANA.

10. 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 [1].

11. Informative References

[1] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[2] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[3] Bierman, A., "The YANG Package Statement", Internet-Draft draft-bierman-netmod-yang-package-00 (work in progress), July 2015.

[4] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", Internet-Draft draft-ietf-netmod-rfc6020bis-06 (work in progress), July 2015.

[5] 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", Internet-Draft draft-dharini-netmod-g-698-2-yang-04 (work in progress), July 2015.

[6] 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", Internet-Draft draft-lam-teas-usage-info-model-net-topology-01 (work in progress), July 2015.

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

[8] 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>.

A. Example

The YANG data schema (in tree format) shown below was extracted from dharini-netmod-g-698-2-yang [5] and represents the same data as UML model appearing in Figure 37 after the tree format. Note: The color code used in the tree format corresponds to the color code used in the UML class diagram.

```
augment /if:interfaces/if:interface:
    +--rw optIfOChRsSs
       +--rw ifCurrentApplicationCode
       |  +--rw applicationCodeId?   uint8
       |  +--rw applicationCode?     string
       +--rw ifCurrentVendorTransceiverClass
       |  +--rw vendorTransceiverClassId?   uint8
       |  +--rw vendorTransceiverClass?     string
       +--ro ifSupportedApplicationCodes
       |  +--ro numberApplicationCodesSupported?   uint32
       |  +--ro applicationCodesList* [applicationCodeId]
       |     +--ro applicationCodeId   uint8
       |     +--ro applicationCode?    string
       +--ro ifSupportedVendorTransceiverClass
       |  +--ro numberVendorTransceiverClassSupported?   uint32
       |  +--ro vendorTransceiverClassList* [vendorTransceiverClassId]
       |     +--ro vendorTransceiverClassId   uint8
       |     +--ro vendorTransceiverClass?    string
       +--rw outputPower?                     int32
       +--ro inputPower?                      int32
       +--rw wavelengthn?                     uint32
```

Figure 35: Interfaces Tree (Available in PDF or HTML versions)

```
notifications:
    +---n optIfOChWavelengthChange
    |  +--ro if-name?      leafref
    |  +--ro wavelength
    |     +--ro wavelength?   uint32
    +---n optIfOChApplicationCodeChange
    |  +--ro if-name?              leafref
    |  +--ro newApplicationCode
    |     +--ro applicationCodeId?   uint8
    |     +--ro applicationCode?     string
    +---n optIfOChVendorTransceiverCodeChange
       +--ro if-name?                    leafref
       +--ro newVendorTransceiverClass
          +--ro vendorTransceiverClassId?   uint8
          +--ro vendorTransceiverClass?     string
```

Figure 36: Notifications Tree (Available in PDF or HTML versions)

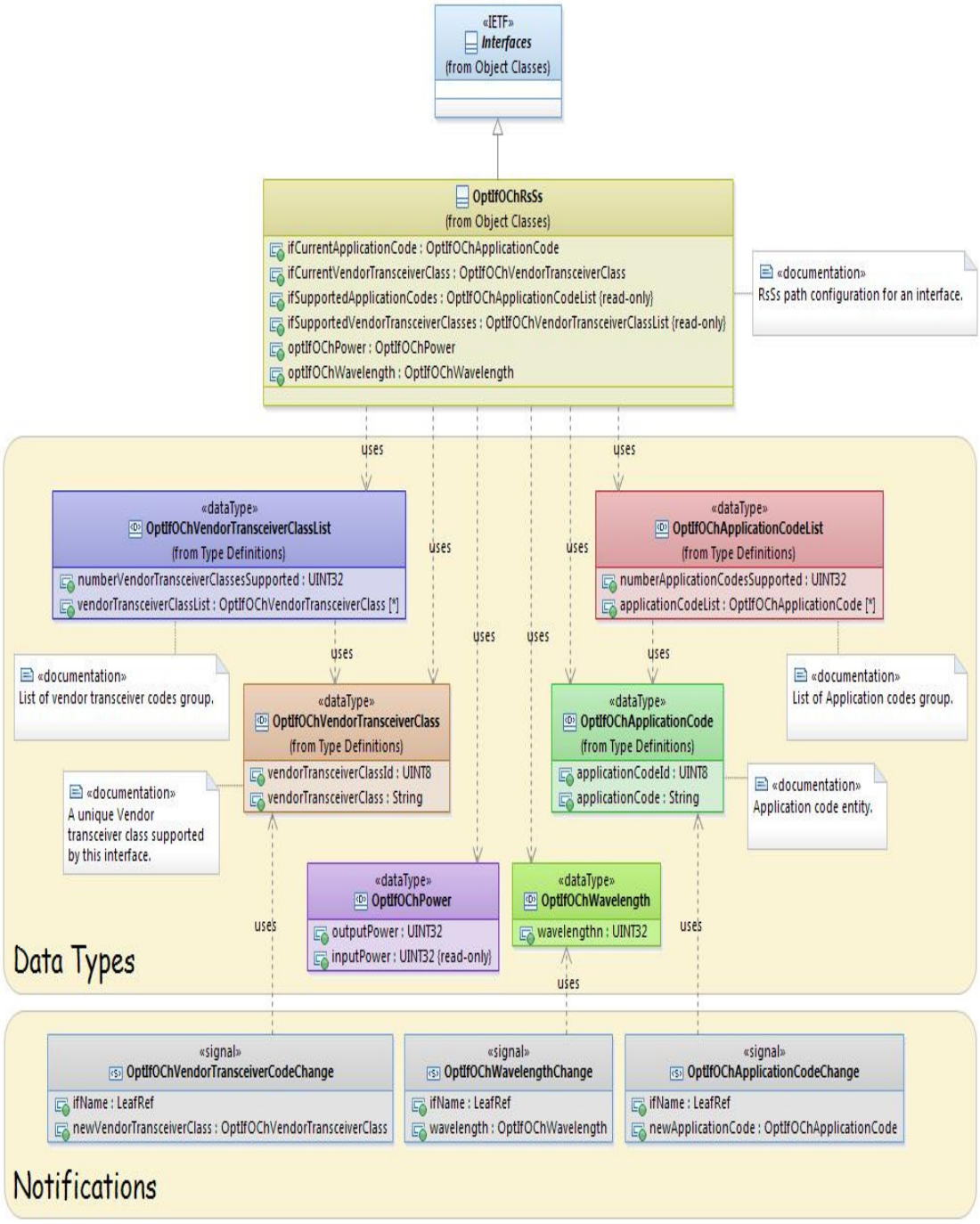


Figure 37: Interfaces UML Model (Available in PDF or HTML versions)

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

Xiang Yun
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