multi
SDO

IISOMI
(Informal Inter-SDO Open Model Initiative)

mSDO IM Workshop; December 2016

Content

- IISOMI Setup
- IISOMI Guidelines
  - Publication Structure
  - UML Modeling Guidelines
  - Papyrus Guidelines
  - UML To YANG Mapping Guidelines
- Tools
  - UML To YANG Mapping Tool
  - Pruning&Refactoring Tool
- Summary

IISOMI SETUP

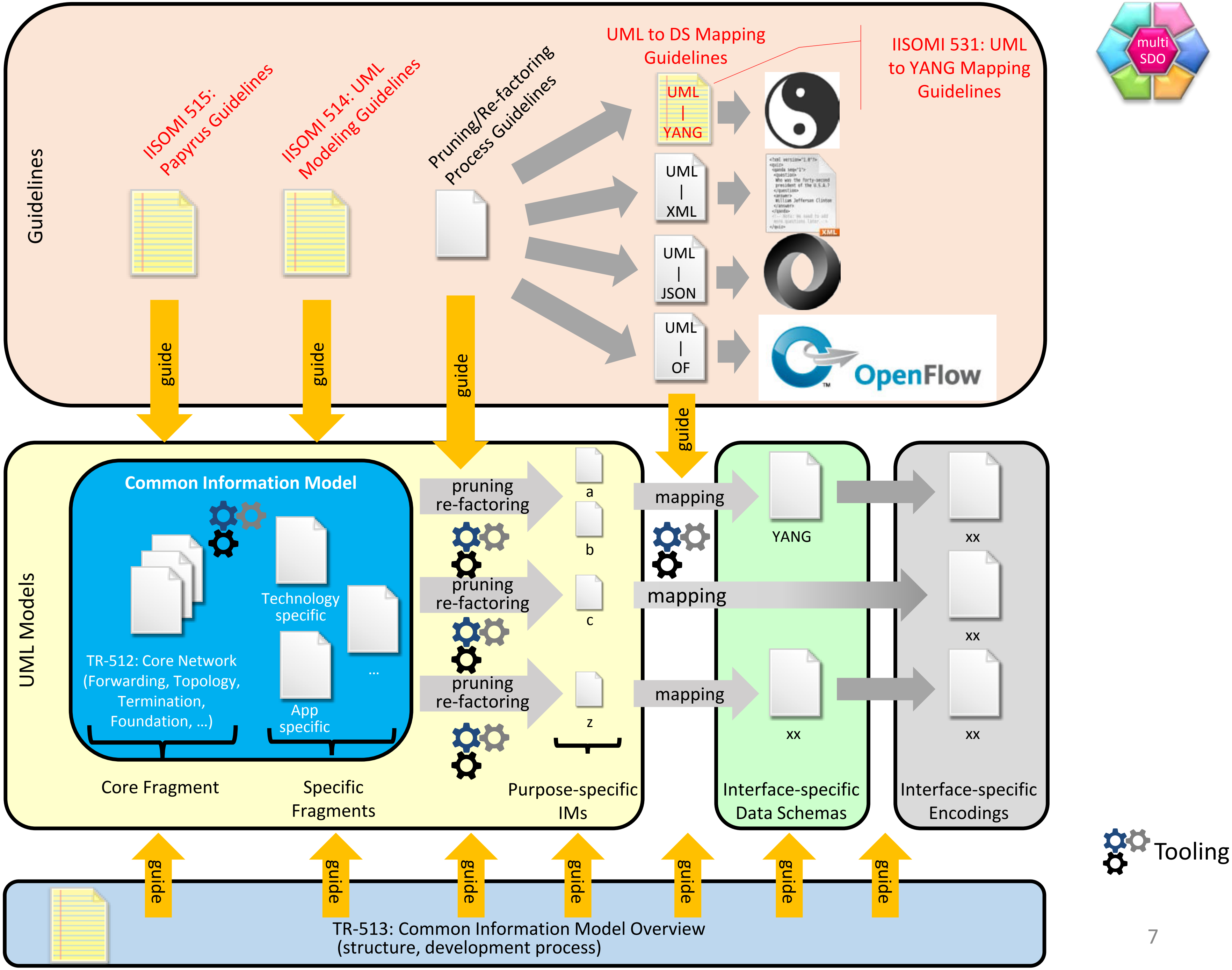
IISOMI Setup

- Informal
  - Not associated to any SDO
  - Modelers of various SDOs “sitting together”
  - Results are not binding to any SDO
- Inter-SDO
  - Previous participants who are also actively participating in various SDOs:
    - Marc Flauw, Mehmet Ersue, Klaus Martiny → ETSI ISG NFV
    - Kam Lam, Malcolm Betts, Scott Mansfield, Xiang Yun, Yuji Tochio → ITU-T
    - Andrea Mazzini, Andy Mayer, Augie Jagau, Jessie Jewitt, Scott Mansfield → MEF
    - Germano Gasparini, Hui Ding, Kam Lam, Karthik Sethuraman, Nigel Davis, Thorsten Heinze, Xiang Zhao, Yuji Tochio, Bernd Zeuner → ONF
    - Augie Jagau, Nigel Davis → TM Forum
- Open Model
  - Open Source
  - Using Open Source SDN facilities
- Initiative
  - Because of abbreviation IISOMI ☺
  - Team → IISOMT ☹; Group → IISOMG ☹

IISOMI Setup

- Organization
  - Start with Modeling Infrastructure Team
    (Core Model is covered by a joined SDO activity; refer to ONF presentation)
  - Kick-Off March 30, 2016
  - Conference calls every week (Tuesdays 17:00 – 18:00 CET)
    Moderation rotate between participants
  - Recently (Nov. 2016) ONF decided to open the UML to YANG Mapping calls – held every week (Wednesdays 14:00 – 15:00 CET) – to IISOMI
- Collaboration Platform
  - Open Source SDN EAGLE **Community**
    (https://community.opensourcesdn.org/wg/EAGLE/dashboard)
  - **GitHub** EAGLE-Open-Model-Profile-and-Tools
    (https://github.com/OpenNetworkingFoundation/EAGLE-Open-Model-Profile-and-Tools)
- Outcomes
  - IISOMI 514: UML Modeling Guidelines
    Published Version 1.2 in September 2016
  - IISOMI 515: Papyrus Guideline
    Published Version 1.2 in September 2016
  - IISOMI 531: UML to YANG Mapping Guidelines
    Published Version 1.0 in September 2016

IISOMI GUIDELINES



Guidelines

IISOMI 515: Papyrus Guidelines

IISOMI 514: UML Modeling Guidelines

Pruning/Re-factoring Process Guidelines

UML to DS Mapping Guidelines

IISOMI 531: UML to YANG Mapping Guidelines

UML | YANG

UML | XML

UML | JSON

UML | OF

OpenFlow

guide

UML Models

Common Information Model

TR-512: Core Network (Forwarding, Topology, Termination, Foundation, …)

Technology specific

App specific

…

Core Fragment

Specific Fragments

pruning re-factoring

a

b

c

z

Purpose-specific IMs

mapping

YANG

xx

Interface-specific Data Schemas

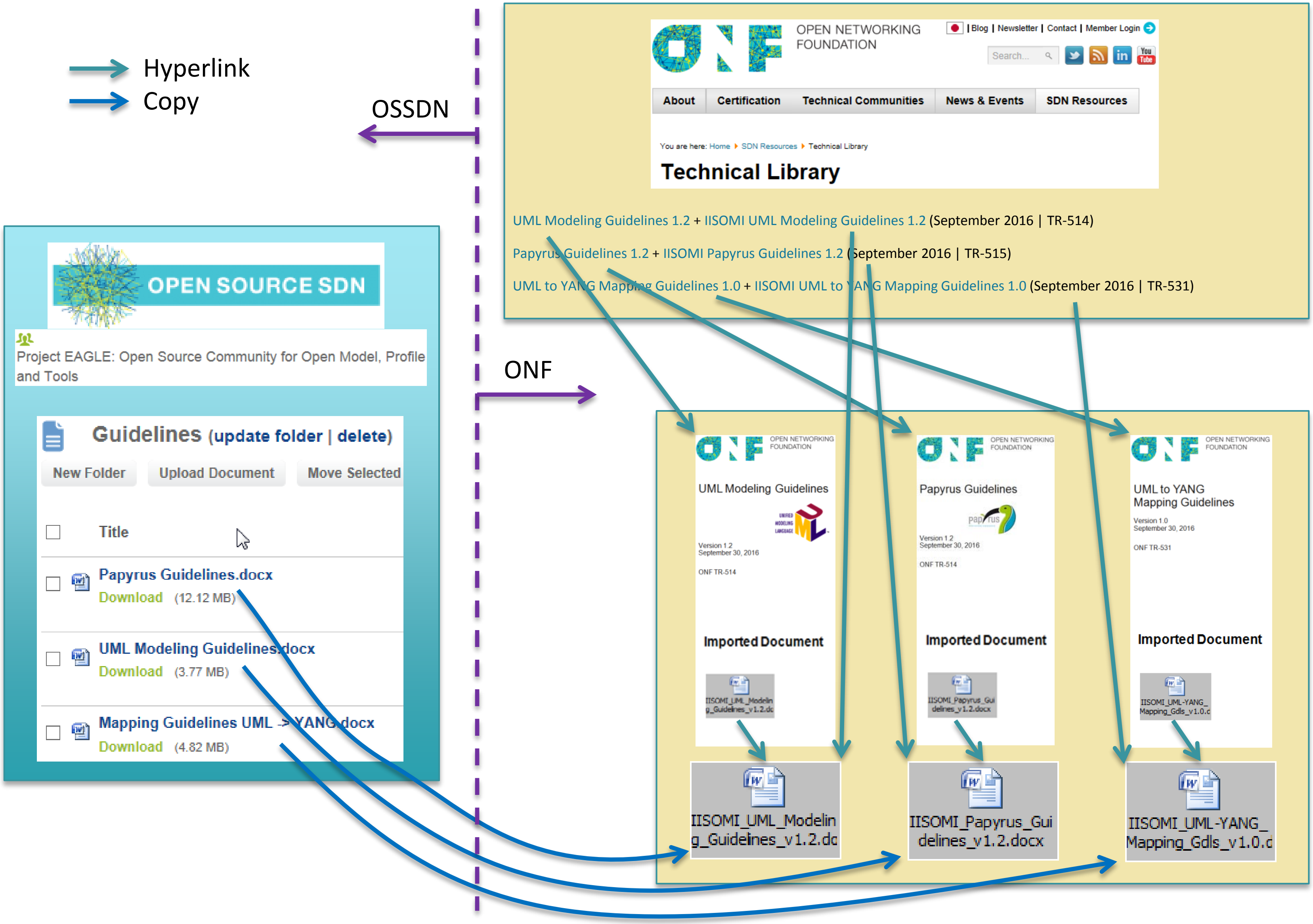
Interface-specific Encodings

TR-513: Common Information Model Overview (structure, development process)

Tooling

multi SDO

IISOMI Guidelines Publication



OPEN SOURCE SDN

Project EAGLE: Open Source Community for Open Model, Profile and Tools

Guidelines (update folder | delete)

□	Title
□	Papyrus Guidelines.docx Download (12.12 MB)
□	UML Modeling Guidelines.docx Download (3.77 MB)
□	Mapping Guidelines UML -> YANG.docx Download (4.82 MB)

OPEN NETWORKING FOUNDATION

You are here: Home ▸ SDN Resources ▸ Technical Library

Technical Library

UML Modeling Guidelines 1.2 + IISOMI UML Modeling Guidelines 1.2 (September 2016 | TR-514)

Papyrus Guidelines 1.2 + IISOMI Papyrus Guidelines 1.2 (September 2016 | TR-515)

UML to YANG Mapping Guidelines 1.0 + IISOMI UML to YANG Mapping Guidelines 1.0 (September 2016 | TR-531)

OPEN NETWORKING FOUNDATION

UML Modeling Guidelines

Version 1.2
September 30, 2016

ONF TR-514

Imported Document

IISOMI_UML_Modeling_Guidelines_v1.2.dc

OPEN NETWORKING FOUNDATION

Papyrus Guidelines

Version 1.2
September 30, 2016

ONF TR-514

Imported Document

IISOMI_Papyrus_Guidelines_v1.2.docx

OPEN NETWORKING FOUNDATION

UML to YANG Mapping Guidelines

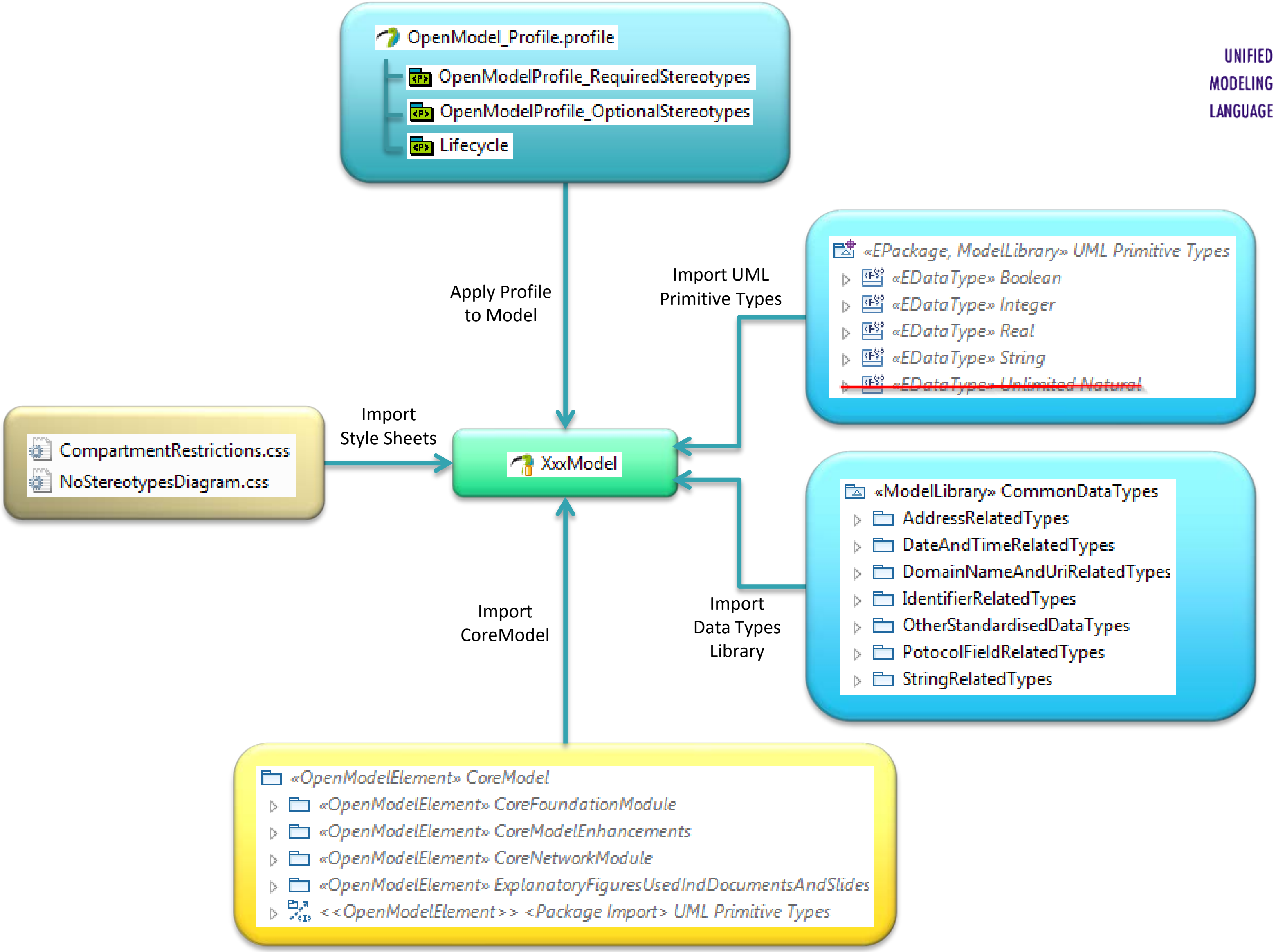
Version 1.0
September 30, 2016

ONF TR-531

Imported Document

IISOMI_UML-YANG_Mapping_Gdls_v1.0.c

UML Modeling Guidelines (IISOMI 514)



OpenModel_Profile.profile

- OpenModelProfile_RequiredStereotypes
- OpenModelProfile_OptionalStereotypes
- Lifecycle

Apply Profile to Model

Import UML Primitive Types

«EPackage, ModelLibrary» UML Primitive Types

- «EDataType» Boolean
- «EDataType» Integer
- «EDataType» Real
- «EDataType» String
- ~~«EDataType» Unlimited Natural~~

CompartmentRestrictions.css

NoStereotypesDiagram.css

Import Style Sheets

XxxModel

«ModelLibrary» CommonDataTypes

- AddressRelatedTypes
- DateAndTimeRelatedTypes
- DomainNameAndUriRelatedTypes
- IdentifierRelatedTypes
- OtherStandardisedDataTypes
- PotocolFieldRelatedTypes
- StringRelatedTypes

Import CoreModel

Import Data Types Library

«OpenModelElement» CoreModel

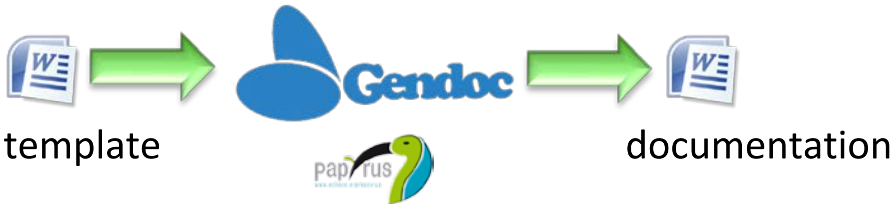
- «OpenModelElement» CoreFoundationModule
- «OpenModelElement» CoreModelEnhancements
- «OpenModelElement» CoreNetworkModule
- «OpenModelElement» ExplanatoryFiguresUsedIndDocumentsAndSlides
- <<OpenModelElement>> <Package Import> UML Primitive Types

UML Modeling Guidelines (IISOMI 514)

- Goal: Assure mSDO-wide consistent information models
- Not a UML tutorial!
- Defines guidelines for creation of protocol-neutral UML information models
  Not specific to any SDO, technology or management protocol
- Constraints the defined UML basic model elements (UML artifacts)
  Only a selected subset of these artifacts are used
- Documentation of each basic model artifact:
  - Short description
  - Graphical notation examples
  - Properties
- UML tool: Papyrus
- Model is split into a static part (object classes / signals and its attributes) and a dynamic part (operations / notifications)
- Definition of UML artifact properties → Meta Model
  - Standard properties (UML Specification)
  - Tool provided properties (Papyrus)
  - Additional specific properties (UML Profile)
- Modeling patterns
- Diagram guidelines

Papyrus Guidelines (IISOMI 515)

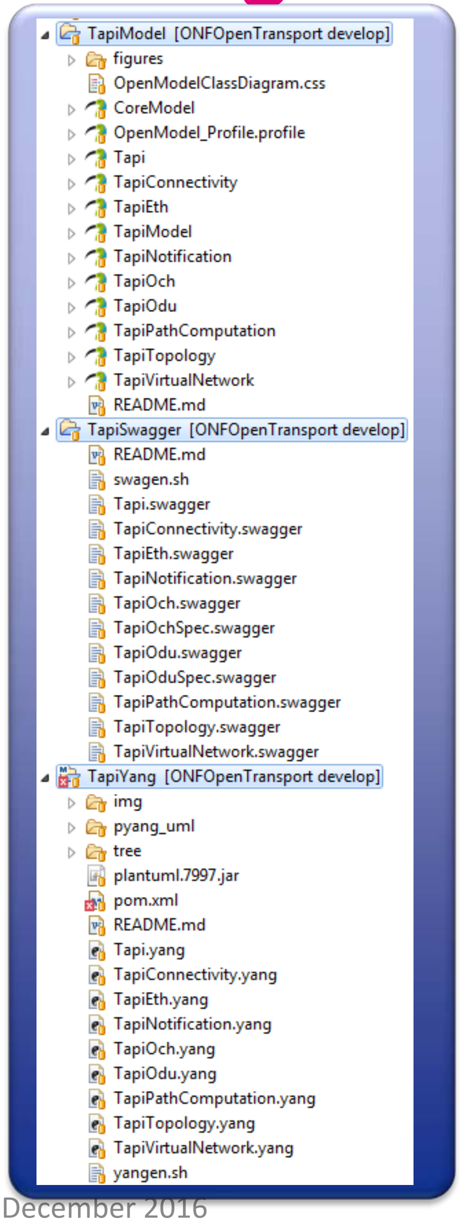
- UML Tool Papyrus
  Homepage: https://www.eclipse.org/papyrus/
- Open Source, Eclipse plug-in
  Homepage: http://www.eclipse.org/modeling/
- Running Papyrus:
  - How to download
  - How to install
  - How to import an existing model
  - How to import RSA models
  - Generating documentation
  - Models and Profiles
  - File structure
  - Team development / Model splitting
- Git
  - Document and Version Management
    Homepage: https://github.com/
  - Model structure on GitHub
  - GitHub work flow



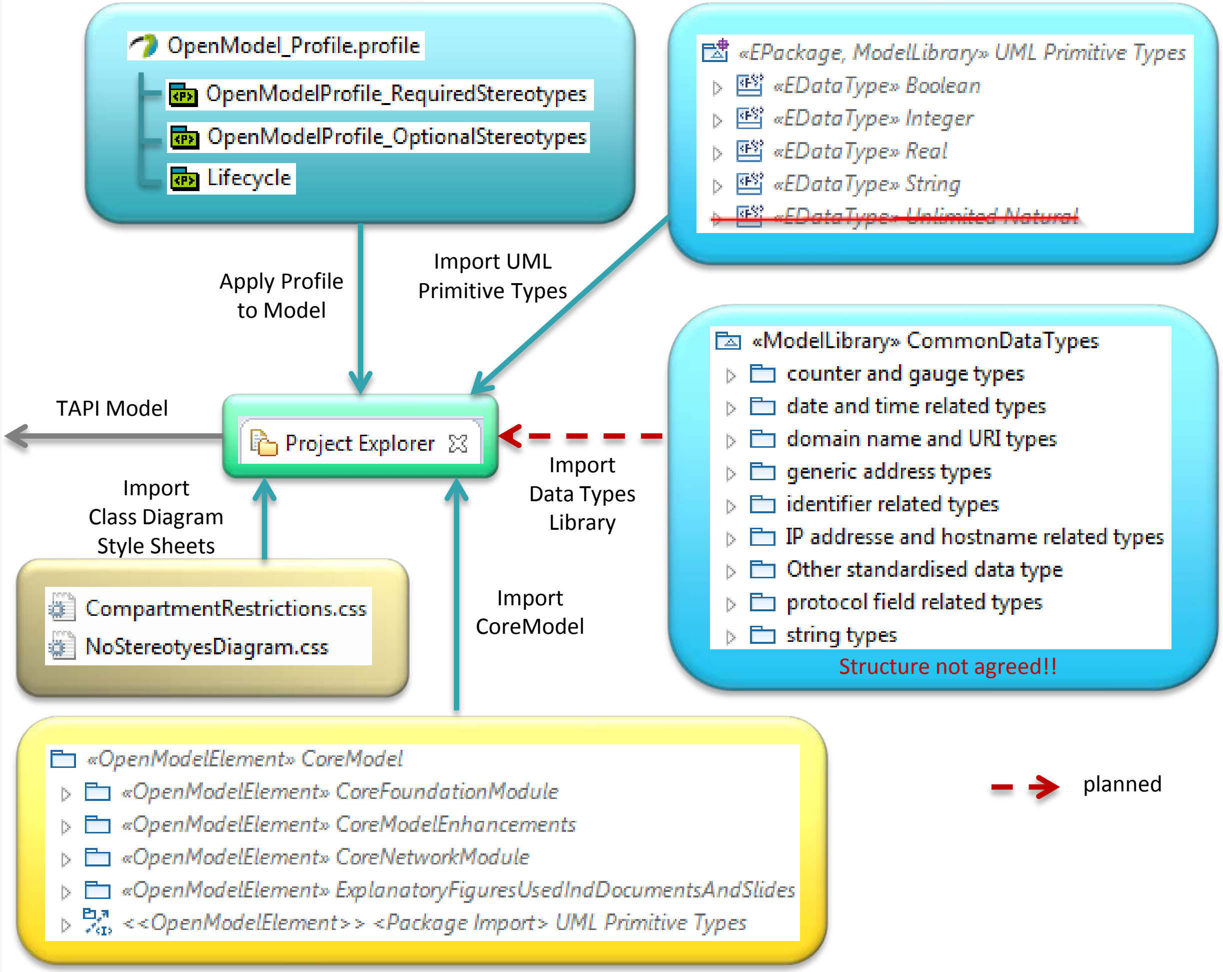
template

documentation

Tooling T-API Example



- TapiModel [ONFOpenTransport develop]
  - figures
  - OpenModelClassDiagram.css
  - CoreModel
  - OpenModel_Profile.profile
  - Tapi
  - TapiConnectivity
  - TapiEth
  - TapiModel
  - TapiNotification
  - TapiOch
  - TapiOdu
  - TapiPathComputation
  - TapiTopology
  - TapiVirtualNetwork
  - README.md
- TapiSwagger [ONFOpenTransport develop]
  - README.md
  - swagen.sh
  - Tapi.swagger
  - TapiConnectivity.swagger
  - TapiEth.swagger
  - TapiNotification.swagger
  - TapiOch.swagger
  - TapiOchSpec.swagger
  - TapiOdu.swagger
  - TapiOduSpec.swagger
  - TapiPathComputation.swagger
  - TapiTopology.swagger
  - TapiVirtualNetwork.swagger
- TapiYang [ONFOpenTransport develop]
  - img
  - pyang_uml
  - tree
  - plantuml.7997.jar
  - pom.xml
  - README.md
  - Tapi.yang
  - TapiConnectivity.yang
  - TapiEth.yang
  - TapiNotification.yang
  - TapiOch.yang
  - TapiOdu.yang
  - TapiPathComputation.yang
  - TapiTopology.yang
  - TapiVirtualNetwork.yang
  - yangen.sh



- OpenModel_Profile.profile
  - OpenModelProfile_RequiredStereotypes
  - OpenModelProfile_OptionalStereotypes
  - Lifecycle

- *«EPackage, ModelLibrary» UML Primitive Types*
  - *«EDataType» Boolean*
  - *«EDataType» Integer*
  - *«EDataType» Real*
  - *«EDataType» String*
  - ~~*«EDataType» Unlimited Natural*~~

- «ModelLibrary» CommonDataTypes
  - counter and gauge types
  - date and time related types
  - domain name and URI types
  - generic address types
  - identifier related types
  - IP addresse and hostname related types
  - Other standardised data type
  - protocol field related types
  - string types

Structure not agreed!!

- Project Explorer

- CompartmentRestrictions.css
- NoStereotyesDiagram.css

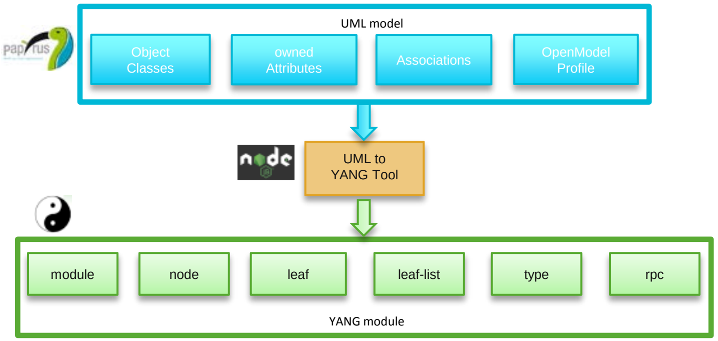
- *«OpenModelElement» CoreModel*
  - *«OpenModelElement» CoreFoundationModule*
  - *«OpenModelElement» CoreModelEnhancements*
  - *«OpenModelElement» CoreNetworkModule*
  - *«OpenModelElement» ExplanatoryFiguresUsedIndDocumentsAndSlides*
  - *<<OpenModelElement>> <Package Import> UML Primitive Types*

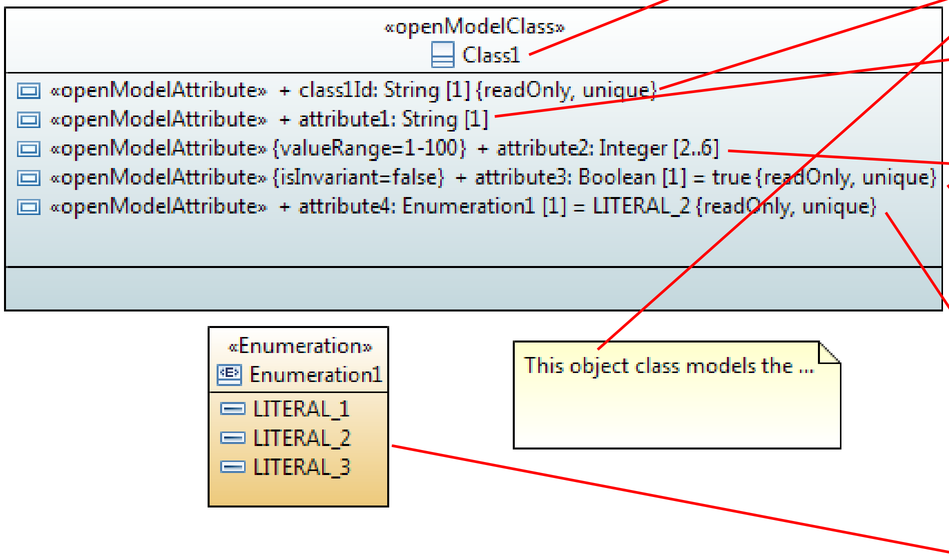
Apply Profile to Model
Import UML Primitive Types
TAPI Model
Import Data Types Library
Import Class Diagram Style Sheets
Import CoreModel
planned

UML To YANG Mapping Guidelines (IISOMI 531)

- Not a YANG Tutorial!
- UML is object oriented whereas YANG is tree-structured
  → need to identify a single root node
- Single UML model mapped to single YANG module
- Divide data into configuration data and state data?
- Mappings are defined for UML-
  - Object Classes
  - Class Attributes
  - Data Types (incl. Primitive Types and Enumerations)
  - Associations
  - Interfaces
  - Operations (incl. Parameters)
  - Notifications
  - Artifact Lifecycle
  - UML-Patterns
    - UML recursion
    - Conditional packages
    - XOR / Choice
    - Proxy Classes

UML To YANG Mapping Example




«openModelClass»
Class1

«openModelAttribute» + class1Id: String [1] {readOnly, unique}
«openModelAttribute» + attribute1: String [1]
«openModelAttribute» {valueRange=1-100} + attribute2: Integer [2..6]
«openModelAttribute» {isInvariant=false} + attribute3: Boolean [1] = true {readOnly, unique}
«openModelAttribute» + attribute4: Enumeration1 [1] = LITERAL_2 {readOnly, unique}

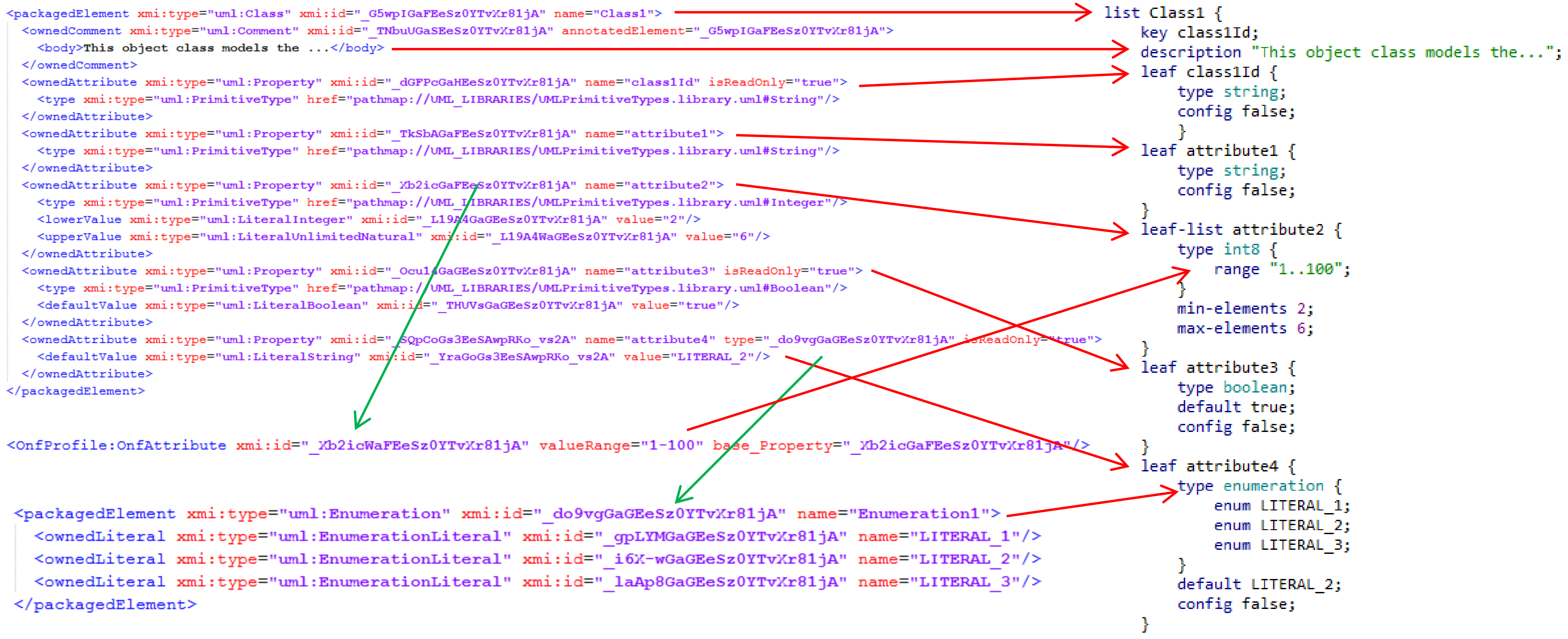
«Enumeration»
Enumeration1

LITERAL_1
LITERAL_2
LITERAL_3

This object class models the ...

```
list Class1 {
    key class1Id;
    description "This object class models the...";
    leaf class1Id {
        type string;
        config false;
        }
    leaf attribute1 {
        type string;
        mandatory true;
        config false;
    }
    leaf-list attribute2 {
        type int8 {
            range "1-100";
        }
        min-elements 2;
        max-elements 6;
    }
    leaf attribute3 {
        type boolean;
        default true;
        config false;
        ompExt:isInvariant;
    }
    leaf attribute4 {
        type enumeration {
            enum LITERAL_1;
            enum LITERAL_2;
            enum LITERAL_3;
        }
        default LITERAL_2;
        config false;
    }
}
```

(Papyrus) XMI To YANG Mapping Example



```xml
<packagedElement xmi:type="uml:Class" xmi:id="_G5wpIGaFEeSz0YTvXr81jA" name="Class1">
  <ownedComment xmi:type="uml:Comment" xmi:id="_TNbuUGaSEeSz0YTvXr81jA" annotatedElement="_G5wpIGaFEeSz0YTvXr81jA">
    <body>This object class models the ...</body>
  </ownedComment>
  <ownedAttribute xmi:type="uml:Property" xmi:id="_dGFPcGaHEeSz0YTvXr81jA" name="class1Id" isReadOnly="true">
    <type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
  </ownedAttribute>
  <ownedAttribute xmi:type="uml:Property" xmi:id="_TkSbAGaFEeSz0YTvXr81jA" name="attribute1">
    <type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
  </ownedAttribute>
  <ownedAttribute xmi:type="uml:Property" xmi:id="_Xb2icGaFEeSz0YTvXr81jA" name="attribute2">
    <type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#Integer"/>
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_L19A4GaGEeSz0YTvXr81jA" value="2"/>
    <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_L19A4WaGEeSz0YTvXr81jA" value="6"/>
  </ownedAttribute>
  <ownedAttribute xmi:type="uml:Property" xmi:id="_Ocu14GaGEeSz0YTvXr81jA" name="attribute3" isReadOnly="true">
    <type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#Boolean"/>
    <defaultValue xmi:type="uml:LiteralBoolean" xmi:id="_THUVsGaGEeSz0YTvXr81jA" value="true"/>
  </ownedAttribute>
  <ownedAttribute xmi:type="uml:Property" xmi:id="_SQpCoGs3EeSAwpRKo_vs2A" name="attribute4" type="_do9vgGaGEeSz0YTvXr81jA" isReadOnly="true">
    <defaultValue xmi:type="uml:LiteralString" xmi:id="_YraGoGs3EeSAwpRKo_vs2A" value="LITERAL_2"/>
  </ownedAttribute>
</packagedElement>
```

```xml
<OnfProfile:OnfAttribute xmi:id="_Xb2icWaFEeSz0YTvXr81jA" valueRange="1-100" base_Property="_Xb2icGaFEeSz0YTvXr81jA"/>
```

```xml
<packagedElement xmi:type="uml:Enumeration" xmi:id="_do9vgGaGEeSz0YTvXr81jA" name="Enumeration1">
  <ownedLiteral xmi:type="uml:EnumerationLiteral" xmi:id="_gpLYMGaGEeSz0YTvXr81jA" name="LITERAL_1"/>
  <ownedLiteral xmi:type="uml:EnumerationLiteral" xmi:id="_i6X-wGaGEeSz0YTvXr81jA" name="LITERAL_2"/>
  <ownedLiteral xmi:type="uml:EnumerationLiteral" xmi:id="_laAp8GaGEeSz0YTvXr81jA" name="LITERAL_3"/>
</packagedElement>
```

```
list Class1 {
  key class1Id;
  description "This object class models the...";
  leaf class1Id {
    type string;
    config false;
    }
  leaf attribute1 {
    type string;
    config false;
  }
  leaf-list attribute2 {
    type int8 {
      range "1..100";
    }
    min-elements 2;
    max-elements 6;
  }
  leaf attribute3 {
    type boolean;
    default true;
    config false;
  }
  leaf attribute4 {
    type enumeration {
      enum LITERAL_1;
      enum LITERAL_2;
      enum LITERAL_3;
    }
    default LITERAL_2;
    config false;
  }
```

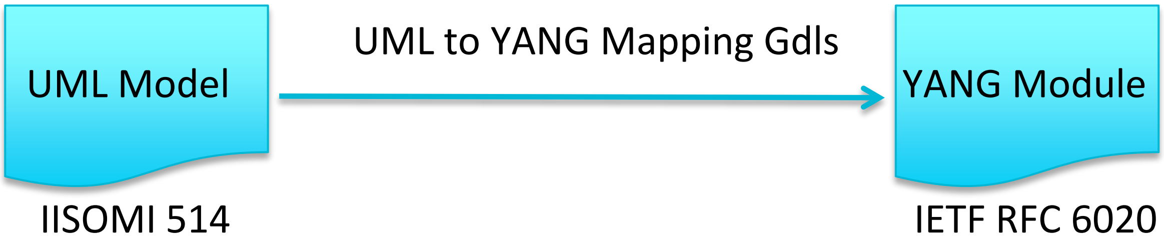
→ Mapping XMI → YANG

→ Mapping within XMI

TOOLS

Mapping Tools

- Mapping tools
  - Initially UML → YANG; already working

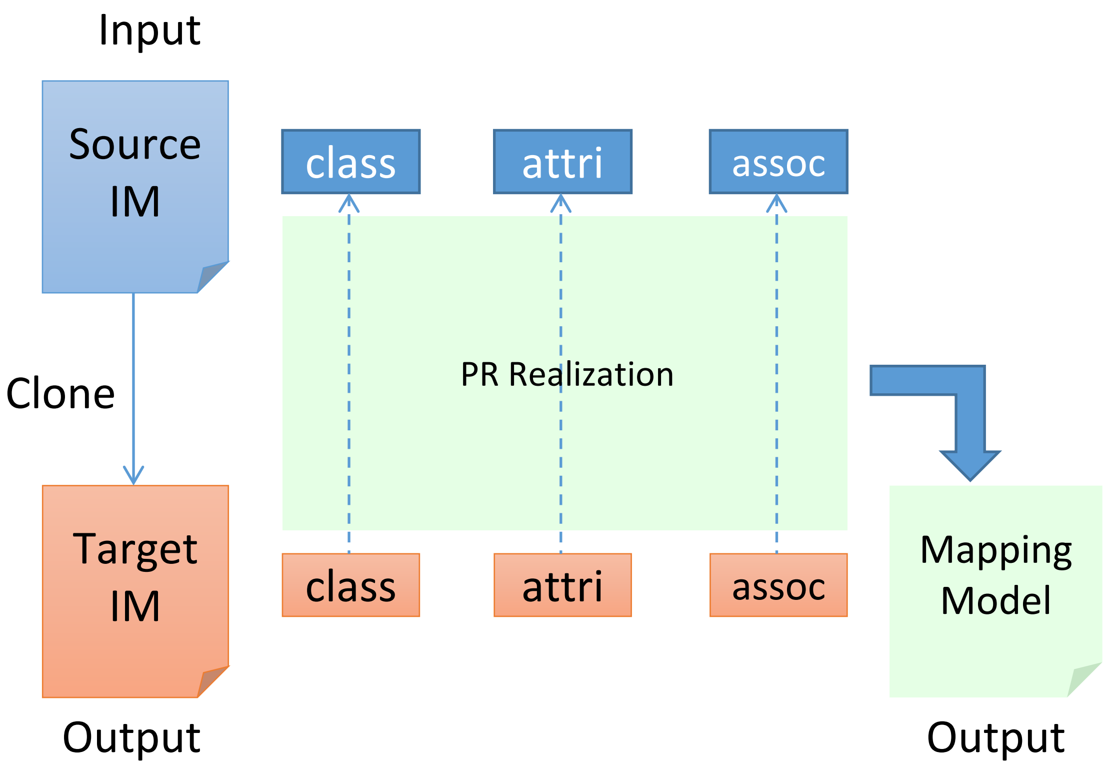


  - Others like UML → JSON will follow
  - Yang → Swagger JSON/RestConf Generation Tool is already available
  - https://github.com/OpenNetworkingFoundation/EAGLE-Open-Model-Profile-and-Tools/tree/UmlYangTools

Pruning&Refactoring Tool

- Supporting tool under development
- Some work is carried out on tooling to prepare a cloned model that is dealing with the key pruning&refactoring capability that gave rise to TAPI IM from the Core Model and also highlight subsequent changes compared to the Core Model
- Work is continuing
- https://github.com/HuiDingCATR/Pruning-and-Refactoring-Tool

Pruning&Refactoring Tool Basic Functions



- Clone Function
  - Clone every class, attribute and association to form the target information model
- Comparison Function
  - Link peer component in source and target with PR Realization
  - Compare features of the peer components and record results in the comment of the realization

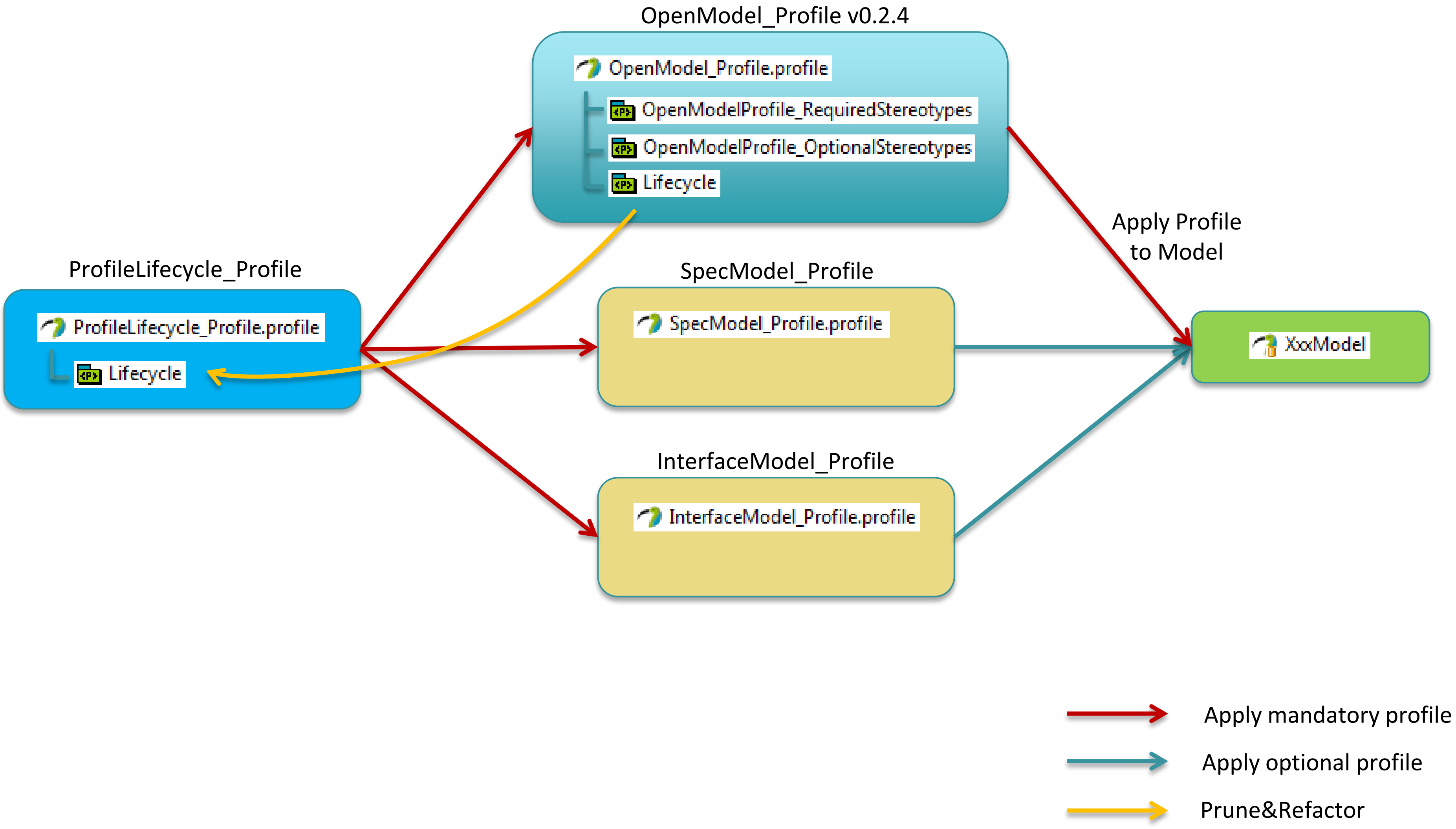
IISOMI Summary

- Progress made in 2016
  - Kick-Off March 2016
  - ONF submitted initial Guidelines to IISOMI for further development
  - Latest versions of Guidelines published in September 2016
- Issues (Louisville workshop) addressed by IISOMI
  - #6: Mixing of information model and management protocols
    Developing a modeling infrastructure for protocol neutral information models
  - #9 Different design patterns for service constructs
    #12b Interfaces are generated by hand inconsistently
    #13 Common approach to the development of information models. Common guidelines, methodology, tooling, etc.
    #20 + #32 Lack of translation from IM to DM
    Developing a modeling and tooling infrastructure; common UML Modeling, Papyrus and Mapping Guidelines; Mapping and Pruning&Refactoring Tools
  - #26 + #49 Difficulty coordinating between SDOs and Open Source Communities
    IISOMI is an Open Source community participated by members from many SDOs
- NEW issues identified by IISOMI
  - YANG Best Practices are missing (RFC6087(bis)?)
  - UML Profile Architecture
- Top priority topics of cross-industry interest, and possibly a way forward
  - **Broadening the engagement of participation**
  - Make SDO executive level be aware of IISOMI
  - Defining a list of common data types

Thank you!

BACKUP

Draft Profile Architecture



OpenModel_Profile v0.2.4

OpenModel_Profile.profile

OpenModelProfile_RequiredStereotypes

OpenModelProfile_OptionalStereotypes

Lifecycle

ProfileLifecycle_Profile

ProfileLifecycle_Profile.profile

Lifecycle

SpecModel_Profile

SpecModel_Profile.profile

InterfaceModel_Profile

InterfaceModel_Profile.profile

Apply Profile
to Model

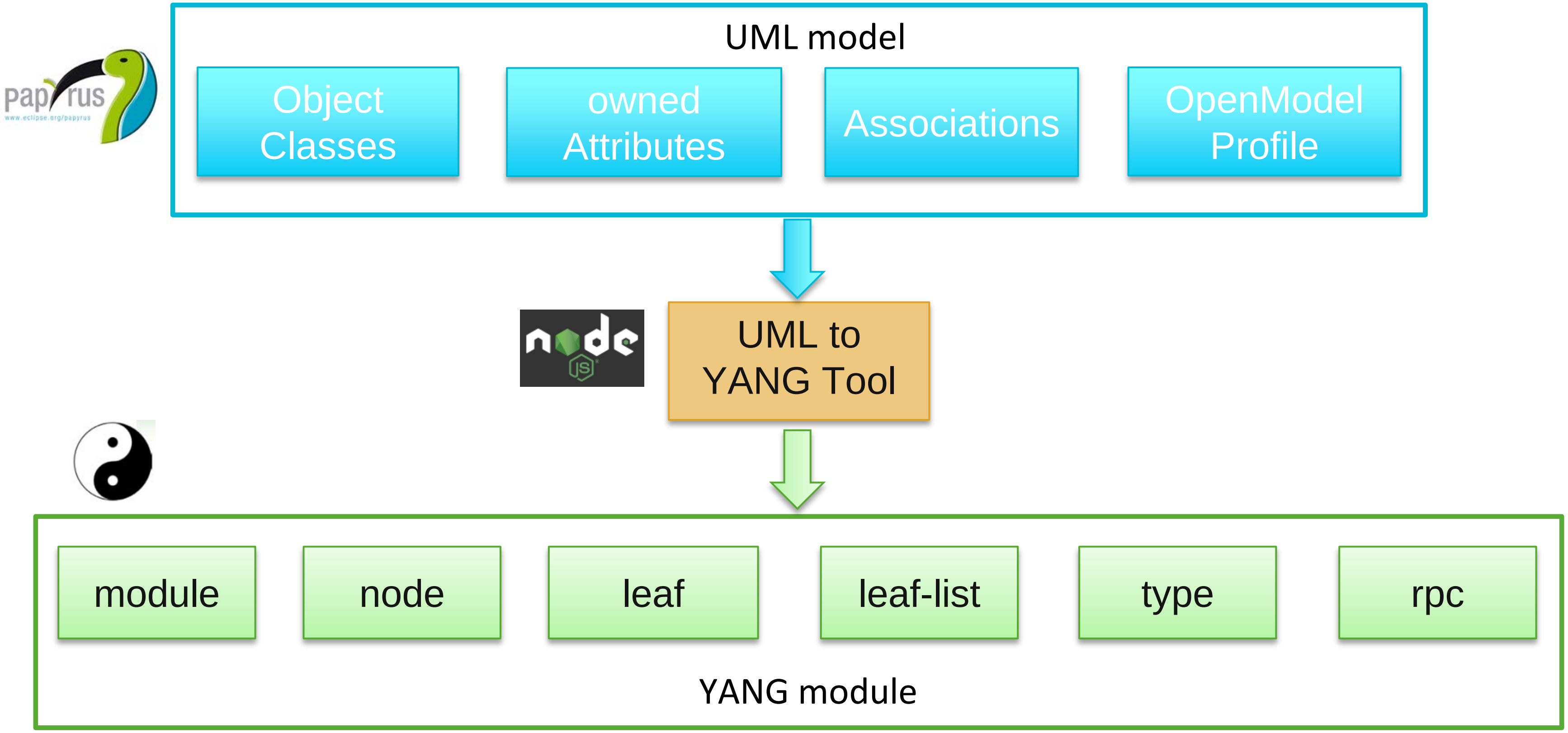
XxxModel

Apply mandatory profile

Apply optional profile

Prune&Refactor

UML to YANG Tool



UML model

Object Classes

owned Attributes

Associations

OpenModel Profile

UML to YANG Tool

module

node

leaf

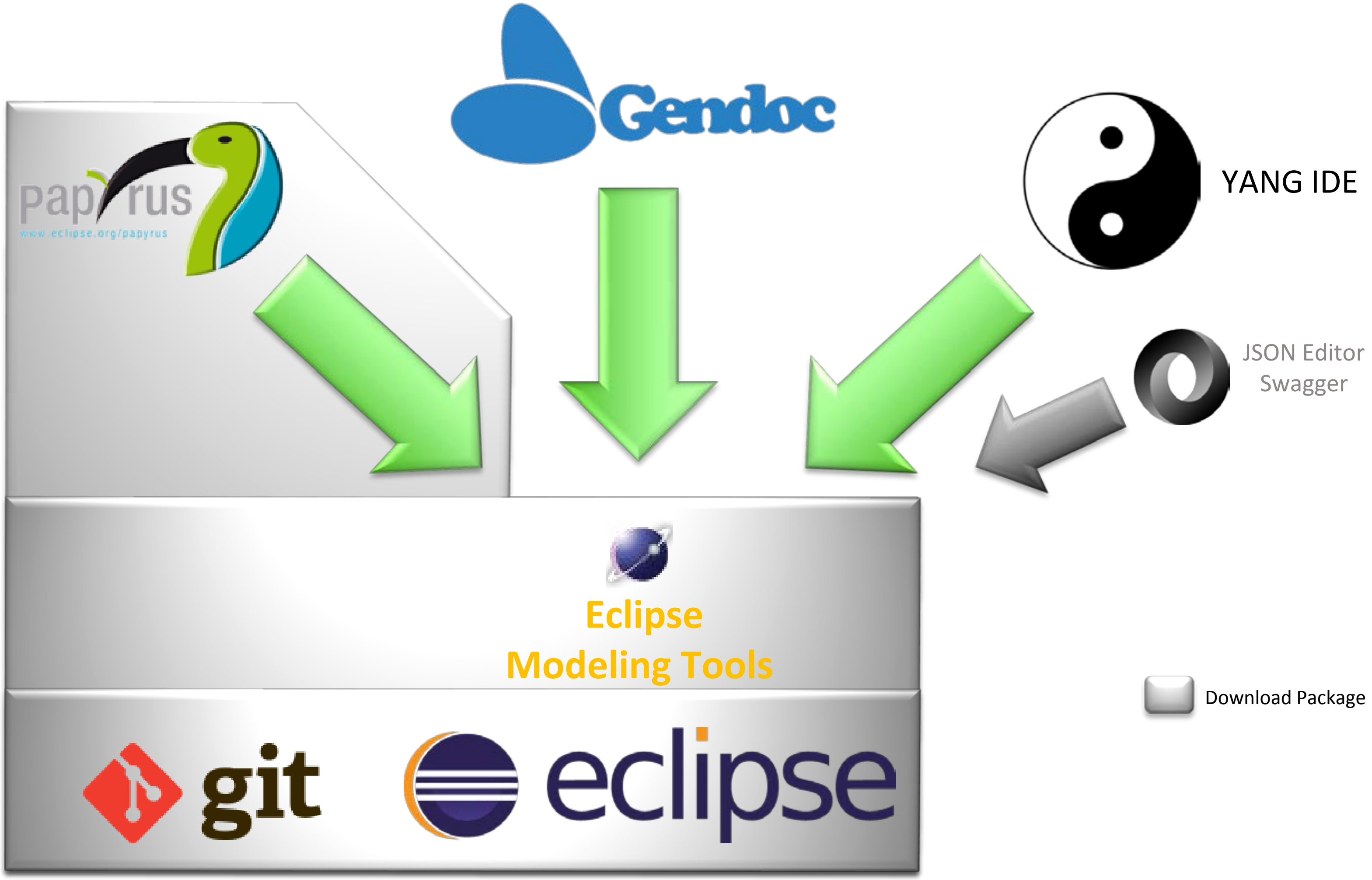
leaf-list

type

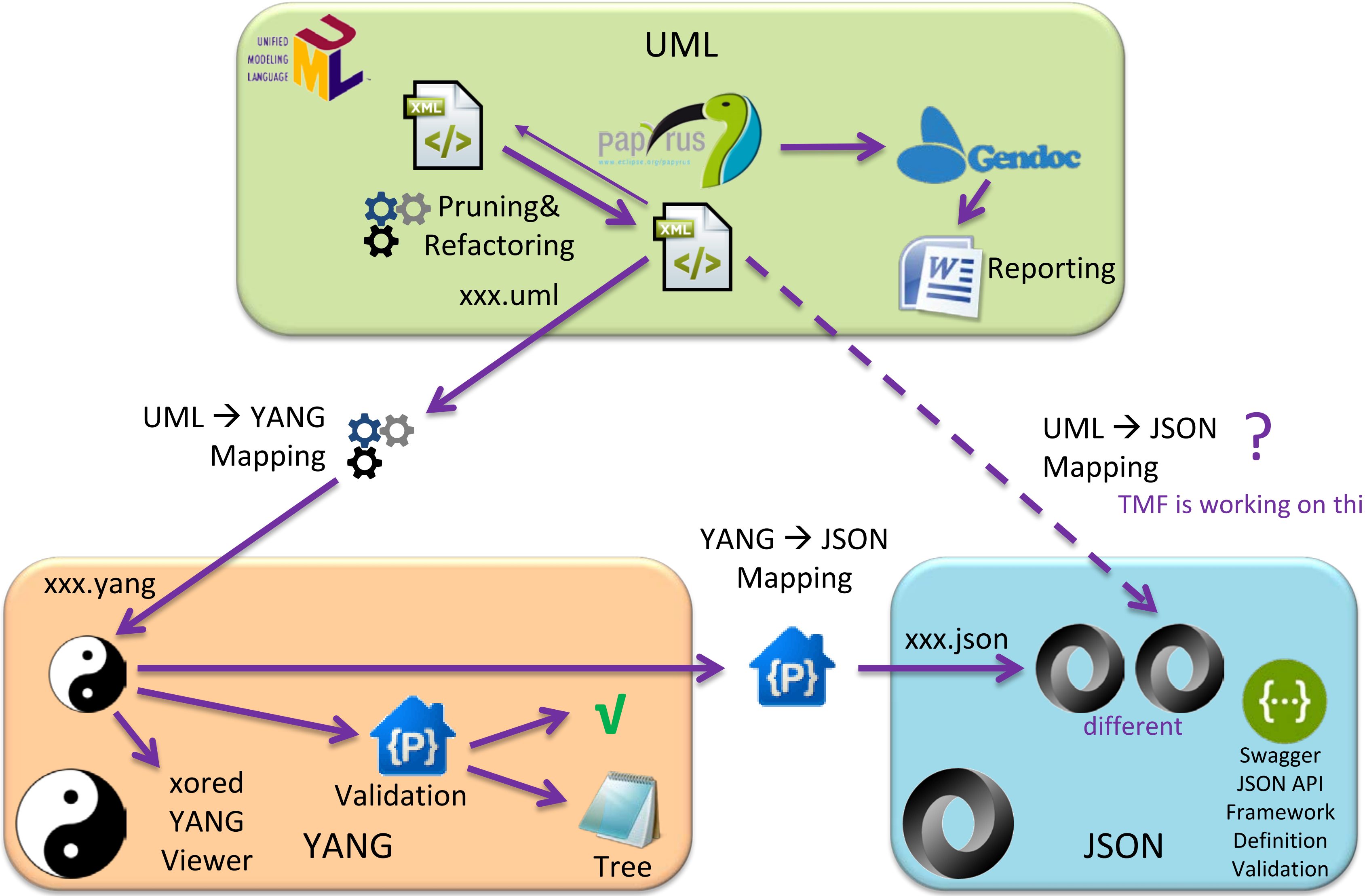
rpc

YANG module

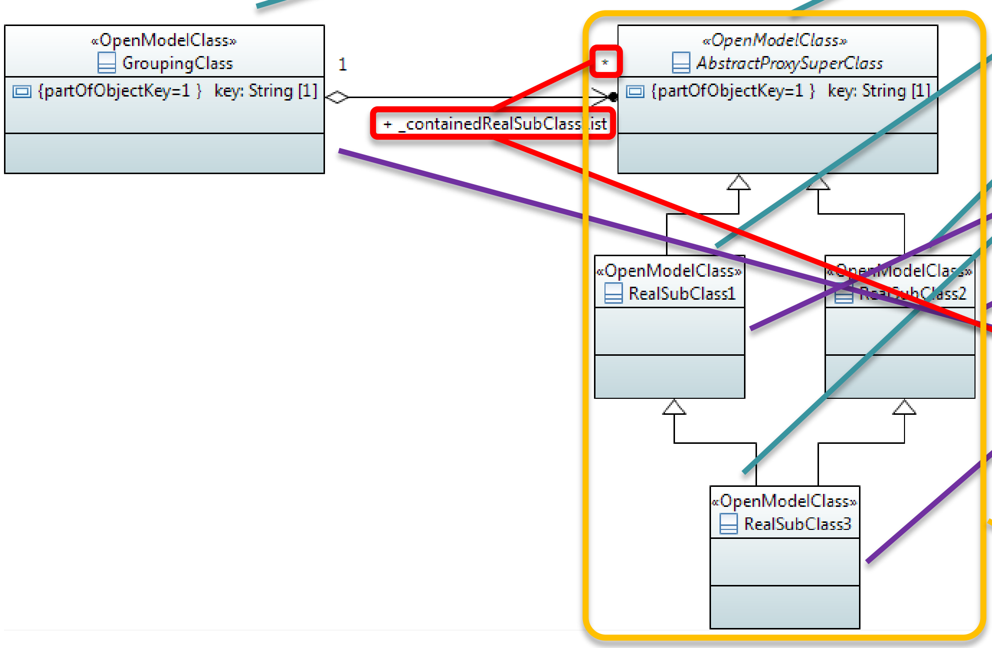
Tooling Overview



Tool Chain


UML
UNIFIED MODELING LANGUAGE
XML
papyrus
www.eclipse.org/papyrus
Gendoc
Pruning&
Refactoring
xxx.uml
Reporting
UML → YANG
Mapping
UML → JSON
Mapping
?
TMF is working on this
YANG → JSON
Mapping
xxx.yang
xored
YANG
Viewer
{P}
Validation
YANG
√
Tree
xxx.json
different
{…}
Swagger
JSON API
Framework
Definition
Validation
JSON

UML To YANG Mapping Example



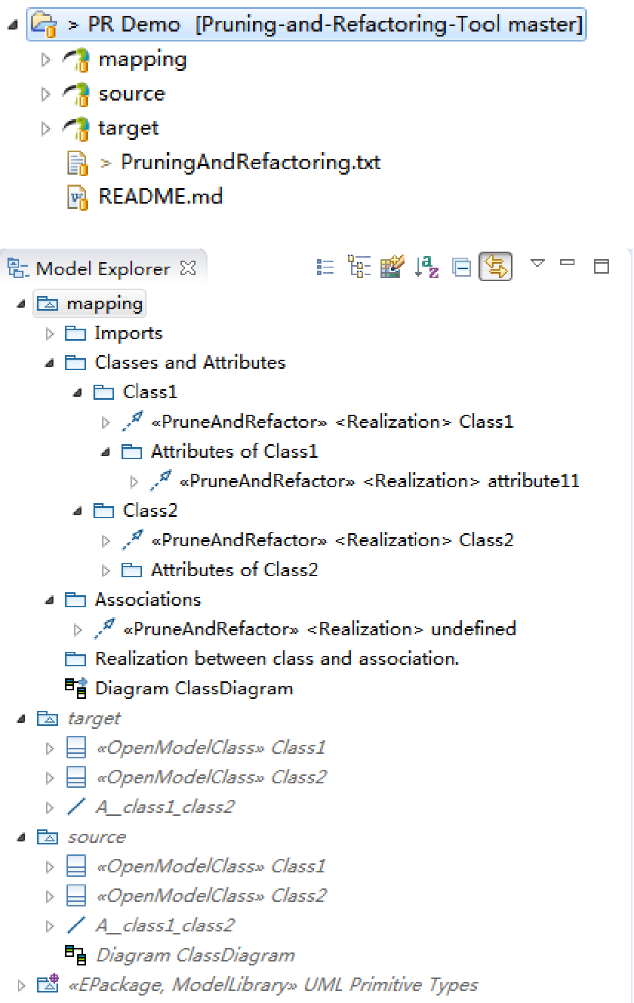
```
grouping GroupingClass {
   leaf key {
      type string;
   }
}
grouping AbstractProxySuperClass {
   leaf key {
      type string;
   }
}
grouping RealSubClass1 {
    uses AbstractProxySuperClass;
}
grouping RealSubClass2 {
    uses AbstractProxySuperClass;
}
grouping RealSubClass3 {
    uses AbstractProxySuperClass;
}
list RealSubClass1 {
   key key;
   uses RealSubClass1;
}
list RealSubClass2 {
   key key;
   uses RealSubClass2;
}
list RealSubClass3 {
   key key;
   uses RealSubClass3;
}
list GroupingClass {
   key key;
   uses GroupingClass;
   list _containedRealSubClassList
      key key;
      choice RealSubClasses {
        case RealSubClass1 {
           leaf RealSubClass1 {
              type leafref {
                 path '/RealSubClass1/key';
              }
           }
        }
        case RealSubClass2 {
           leaf RealSubClass2 {
              type leafref {
                 path '/RealSubClass2/key';
              }
           }
        }
        case RealSubClass3 {
           leaf RealSubClass3 {
              type leafref {
                 path '/RealSubClass3/key';
              }
           }
        }
      }
   }
}
```

First step: Map all object classes to Groupings

Second step: ”Instantiate“ all non-abstract object classes

Third step: Proxy class mapping

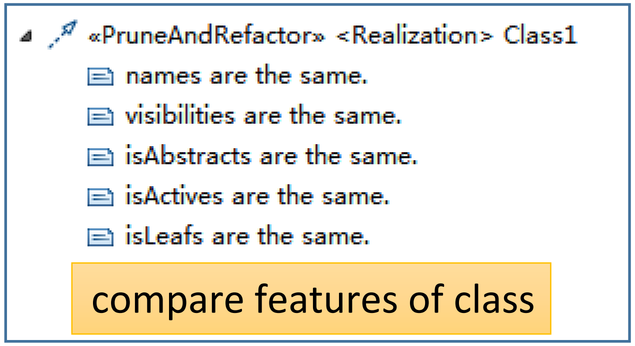
Pruning&Refactoring Tool In Papyrus



> PR Demo [Pruning-and-Refactoring-Tool master]
- mapping
- source
- target
- > PruningAndRefactoring.txt
- README.md

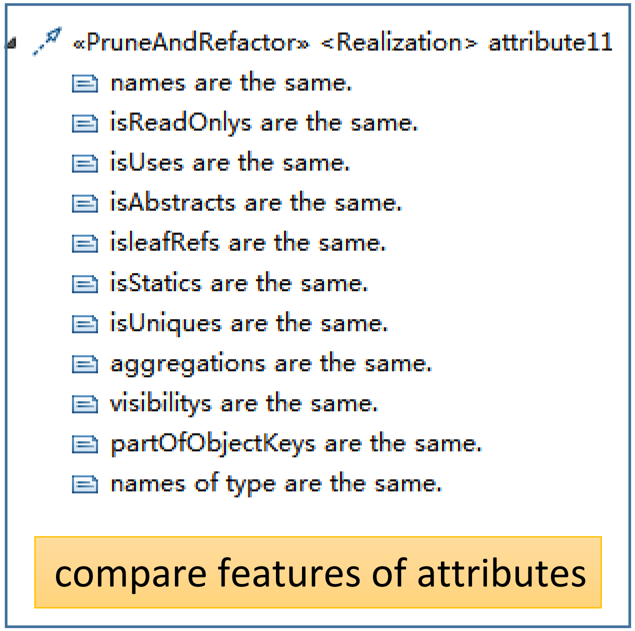
Model Explorer

- mapping
  - Imports
  - Classes and Attributes
    - Class1
      - «PruneAndRefactor» <Realization> Class1
      - Attributes of Class1
        - «PruneAndRefactor» <Realization> attribute11
    - Class2
      - «PruneAndRefactor» <Realization> Class2
      - Attributes of Class2
  - Associations
    - «PruneAndRefactor» <Realization> undefined
  - Realization between class and association.
  - Diagram ClassDiagram
- *target*
  - *«OpenModelClass» Class1*
  - *«OpenModelClass» Class2*
  - *A_class1_class2*
- *source*
  - *«OpenModelClass» Class1*
  - *«OpenModelClass» Class2*
  - *A_class1_class2*
  - *Diagram ClassDiagram*
- *«EPackage, ModelLibrary» UML Primitive Types*



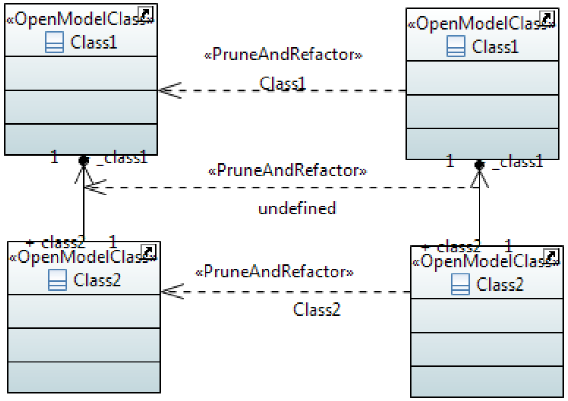
«PruneAndRefactor» <Realization> Class1
- names are the same.
- visibilities are the same.
- isAbstracts are the same.
- isActives are the same.
- isLeafs are the same.

compare features of class

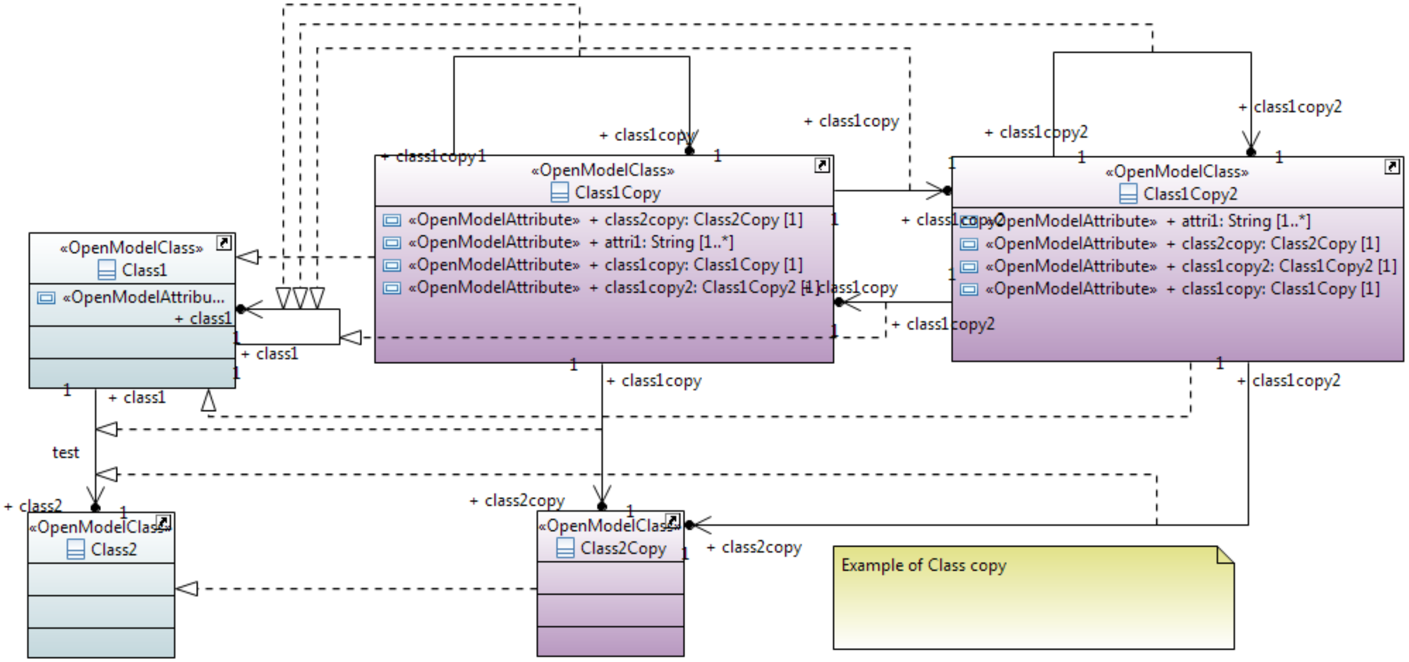


«PruneAndRefactor» <Realization> attribute11
- names are the same.
- isReadOnlys are the same.
- isUses are the same.
- isAbstracts are the same.
- isleafRefs are the same.
- isStatics are the same.
- isUniques are the same.
- aggregations are the same.
- visibilitys are the same.
- partOfObjectKeys are the same.
- names of type are the same.

compare features of attributes

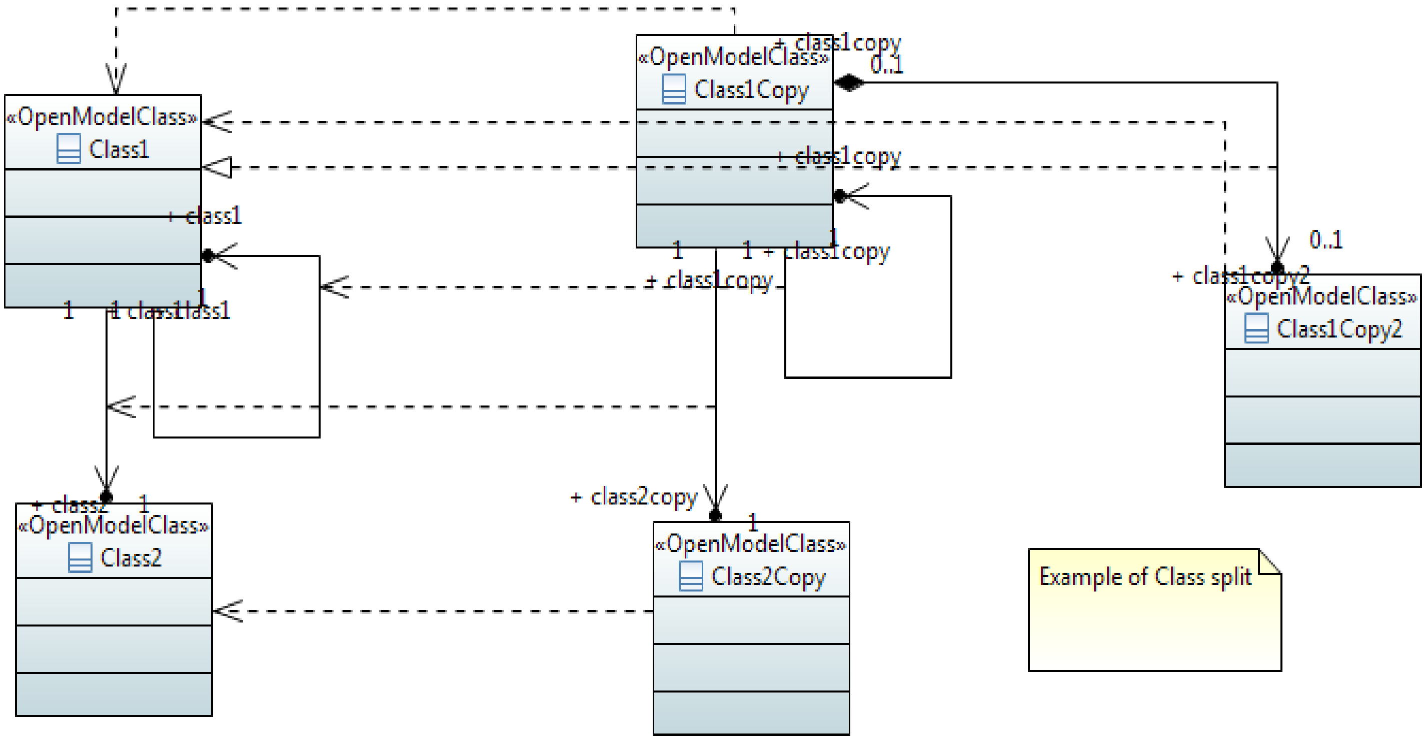


Pruning&Refactoring Tool
Class Copy


«OpenModelClass»
Class1
«OpenModelAttribu...
+ class1
«OpenModelClass»
Class1Copy
«OpenModelAttribute» + class2copy: Class2Copy [1]
«OpenModelAttribute» + attri1: String [1..*]
«OpenModelAttribute» + class1copy: Class1Copy [1]
«OpenModelAttribute» + class1copy2: Class1Copy2 [1]
«OpenModelClass»
Class1Copy2
«OpenModelAttribute» + attri1: String [1..*]
«OpenModelAttribute» + class2copy: Class2Copy [1]
«OpenModelAttribute» + class1copy2: Class1Copy2 [1]
«OpenModelAttribute» + class1copy: Class1Copy [1]
«OpenModelClass»
Class2
«OpenModelClass»
Class2Copy
+ class1copy1
+ class1copy
+ class1copy
+ class1copy2
+ class1copy2
+ class1copy2
+ class1copy
+ class1copy
+ class1copy2
+ class1copy
+ class1copy2
+ class1
+ class1
test
+ class2
+ class2copy
+ class2copy
1
Example of Class copy

- User identifies the class and the number of copies
- The tool perform class copy
  - copies all the attributes from source class
  - adds associations in target model
  - adds P&R realizations in mapping model

Pruning&Refactoring Tool
Class Split



- User identifies the class and the number of Splits
- The tool perform class split
  - copies all attributes from the source model
  - adds associations in target model for cloned class but not split class
  - adds P&R realizations in mapping model