

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 2:

Upotreba CIMTool alata za kreiranje CIM profila u
RDFS formatu

CIM, RDF, RDFS

- CIM (Common Information Model)
 - apstraktni model koji je razvila Radna grupa 13 Tehničkog komiteta 57 Međunarodne elekrotehničke komisije
 - postao je međunarodni standard IEC 61970-301.
 - opisan je pomoću UML-a.
- RDF (Resource Description Framework)
 - definisan familijom W3C (World Wide Web Consortium) standarda kao model za opis mašinski čitljivih meta-podataka.
 - opisuje resurse u obliku subjekat-predikat-objekat “tripleta”
- RDFS (Resource Description Framework Schema)
 - proširivi jezik koji obezbeđuje elemente za zadavanje RDF resursa.

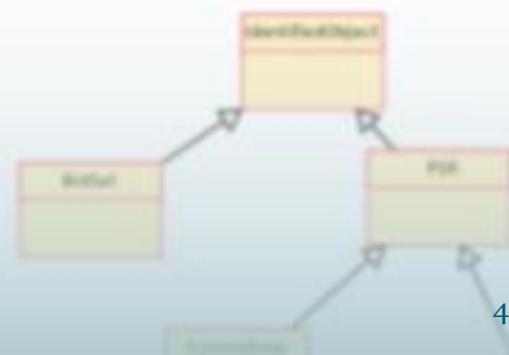
CIM Profile

- Profil predstavlja skup definicija klasa, njihovih atributa i međusobnih veza u okviru jedne šeme. Profil predstavlja opis modela podataka koji je podskup neke nadređene šeme.
- Osnovna uloga profila je definisanje domenskih i kontekstno zavisnih modela.
- Zadavanje profila je moguće u nekoliko oblika i to RDFS, XSD, kao tekstualni ili HTML dokument.
- Jednostavno rečeno svi delovi CIM-a su opcioni u zavisnosti od aplikacije i njenih zahteva. Zbog ovakvog stava, neophodan je bio poseban dokument koji bi specifikovao neophodne delove. Takav dokument je nazvan CIM Profil.

CIMTool alat

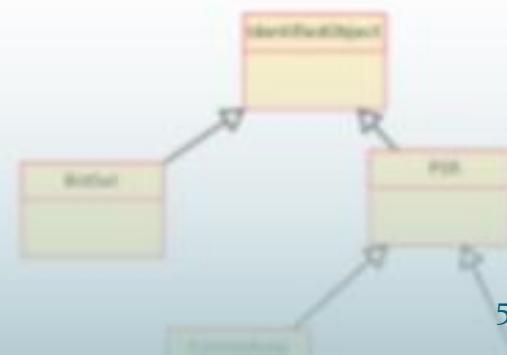
- Opensource alat
 - Site: <http://wiki.cimtool.org/index.html>
- Implementiran je kao plugin za Eclipse platformu
- Koristimo ga za dizajn CIM profila na bazi UML šeme modela date u XMI formatu

CIM  **Tool**



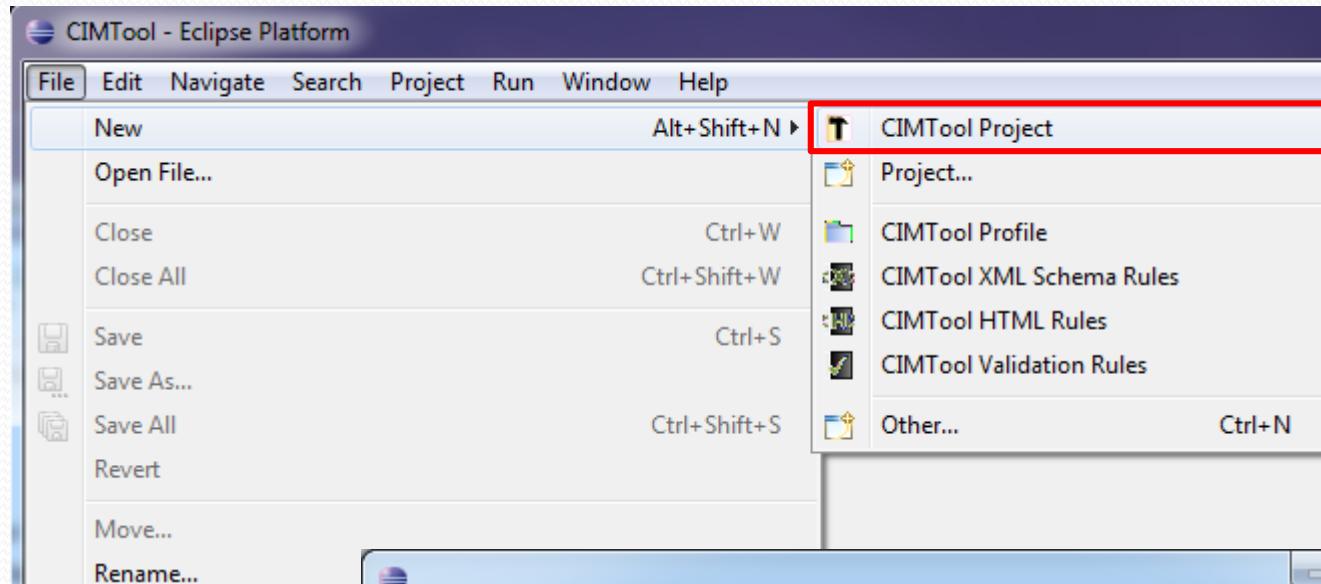
Osnovne operacije u CIMTool alatu

- Kreiranje CIMTool projekta
- Učitavanje šeme modela date u XMI formatu
- Kreiranje CIM profila
 - Biranje formata zapisa CIM profila
 - HTML
 - RDFS
 - Dodavanje klase
 - Dodavanje atributa klase
 - Konfiguracija klasa i atributa u profilu
- Import postojećeg CIM profila u projekat
- Import postojećeg CIMTool projekta

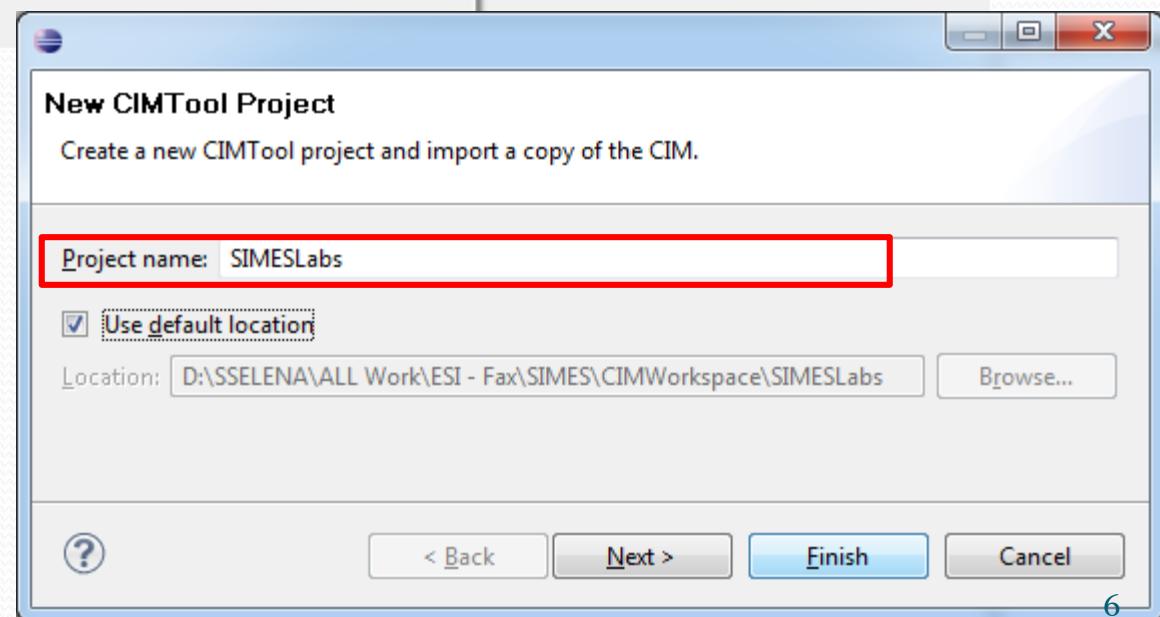


CIMTool: kreiranje projekta

- Korak 1

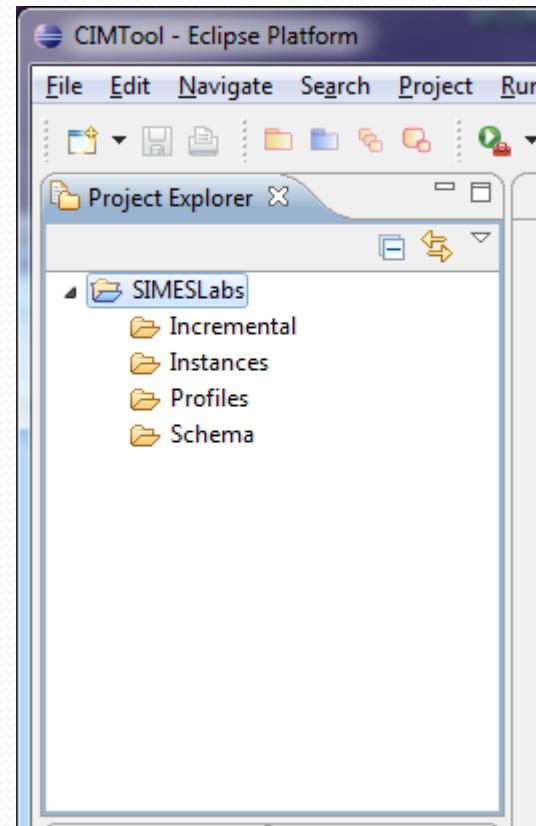


- Korak 2



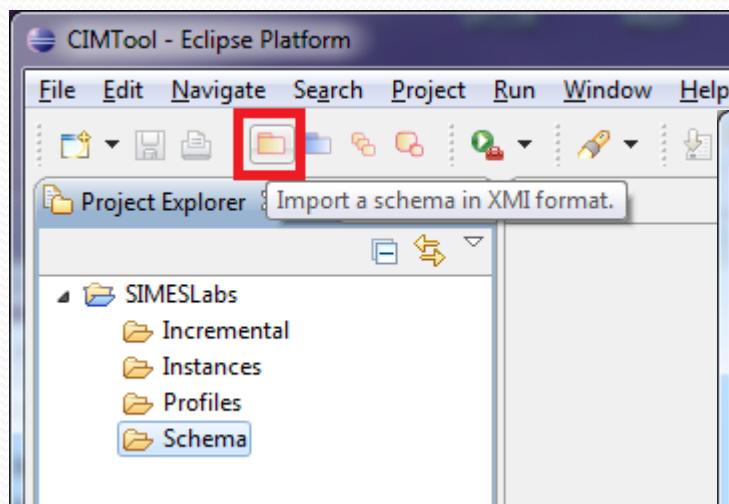
CIMTool: kreiranje projekta

- Prikaz kreiranog projekta u “*Project Explorer*” prozoru

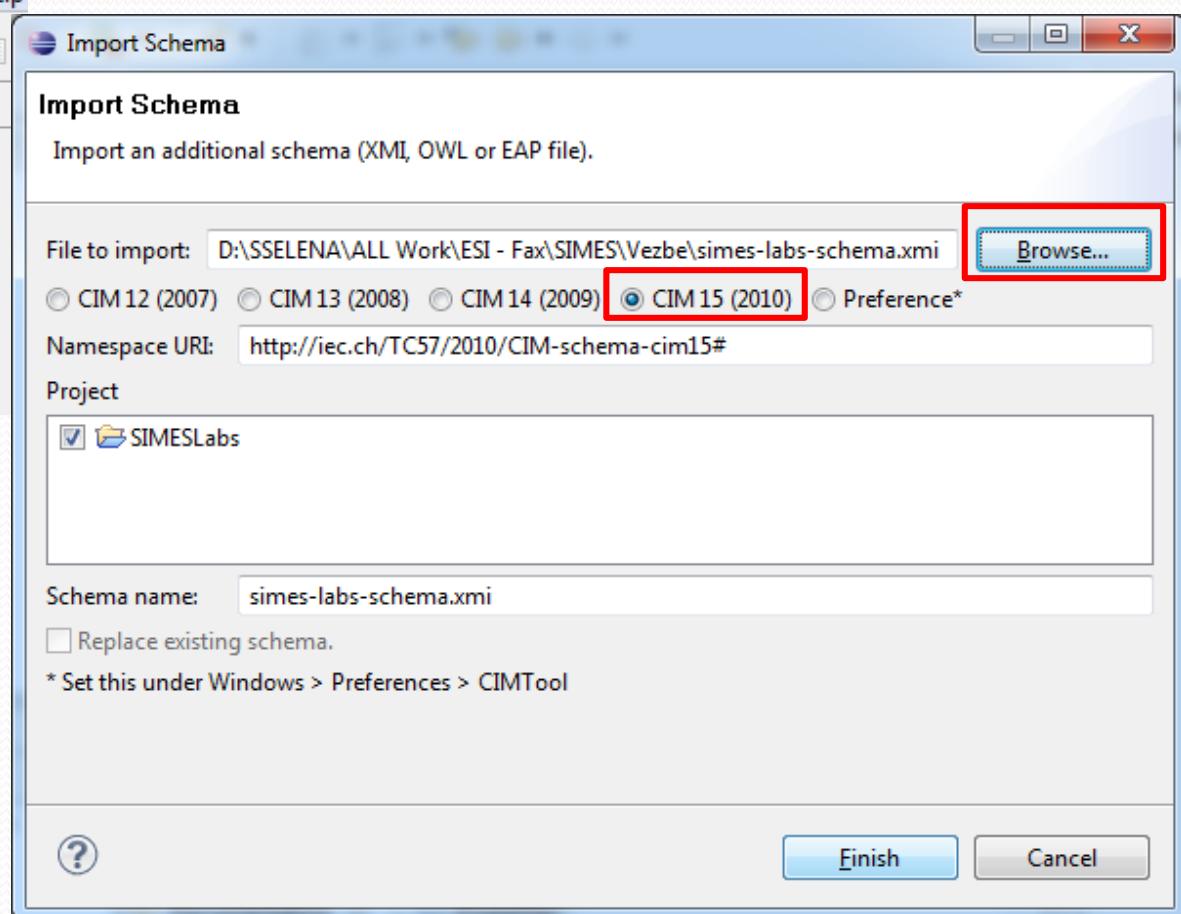


CIMTool: učitavanje šeme modela u XMI formatu

- Korak 1

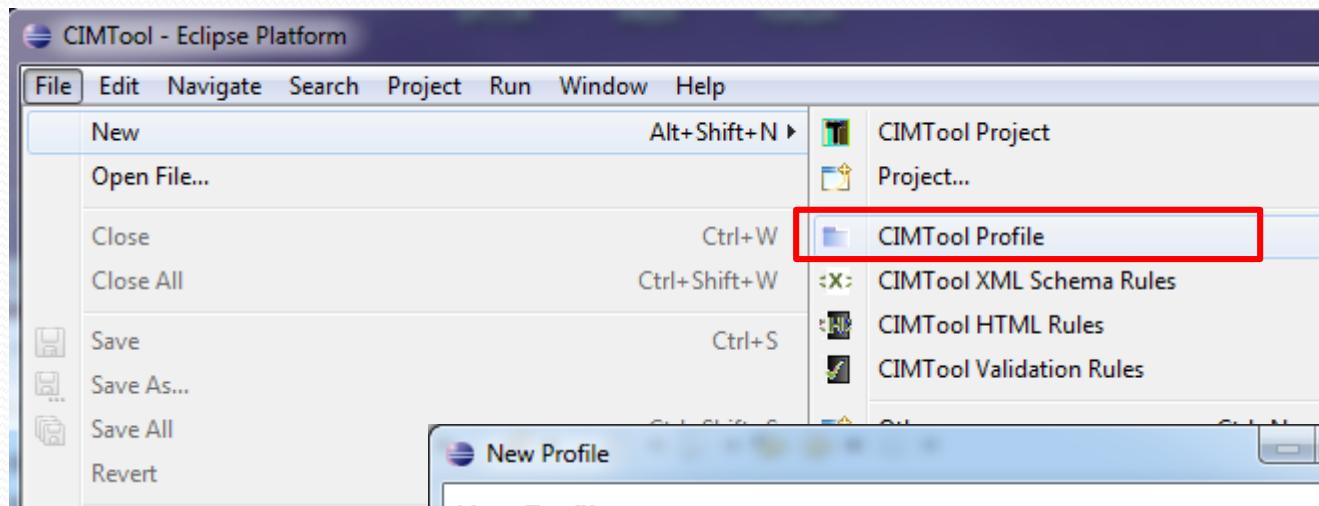


- Korak 2

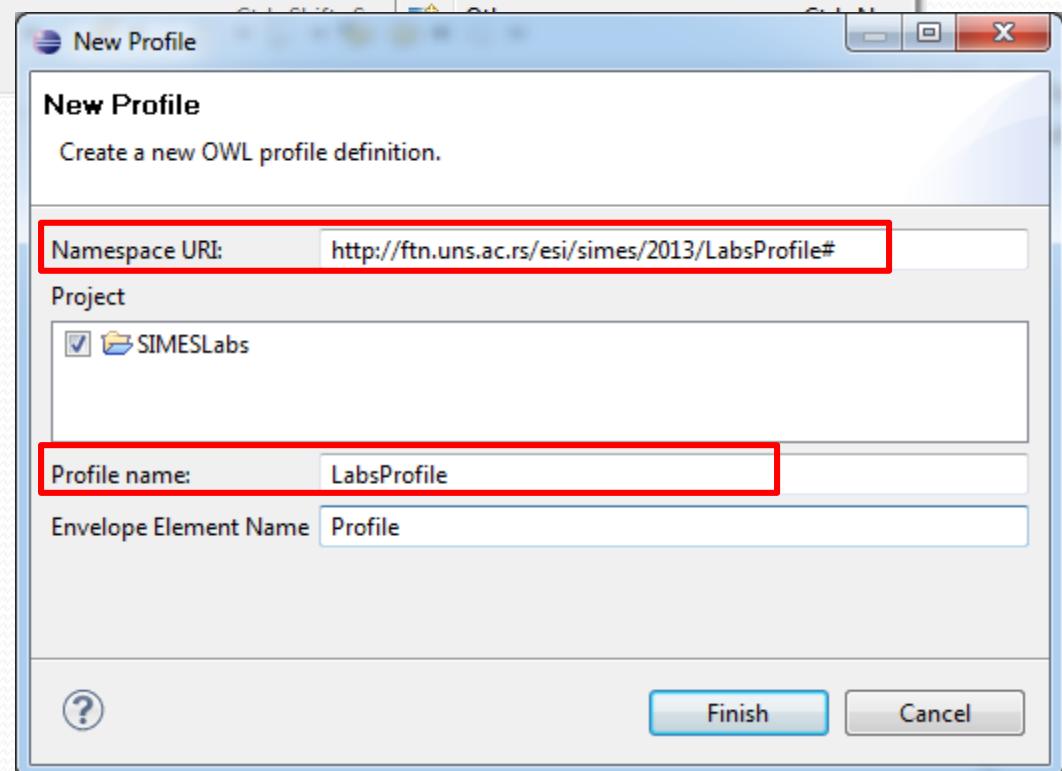


CIMTool: kreiranje CIM profila

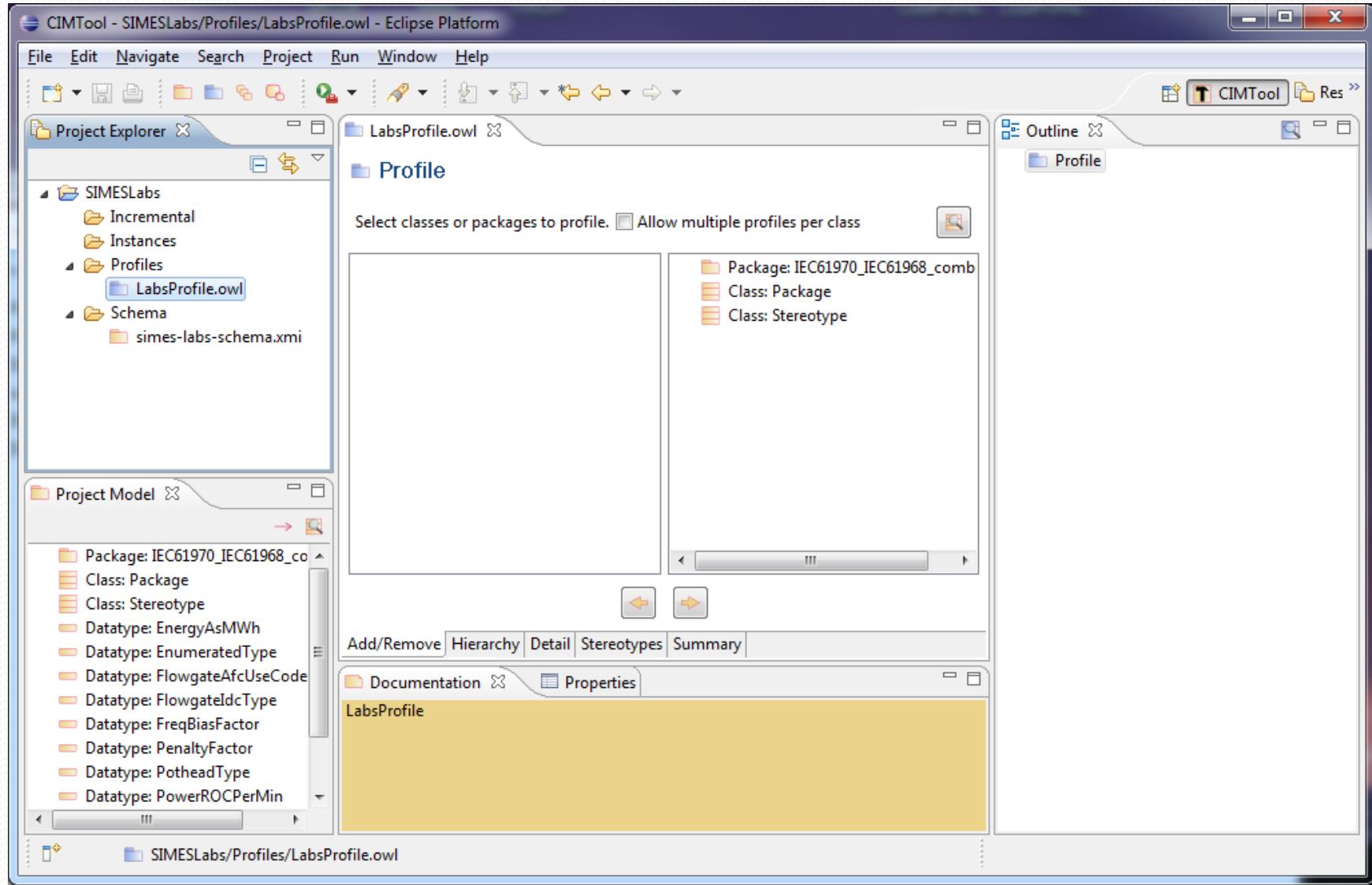
- Korak 1



- Korak 2



CIMTool: kreiranje CIM profila



CIMTool: Biranje formata zapisa CIM profila

The screenshot shows the CIMTool interface within the Eclipse Platform. The main window displays the 'Summary' tab for the profile 'LabsProfile.owl'. The 'Builder for legacy-rdfs' checkbox is selected and highlighted with a red box. The 'Project Explorer' view on the left shows the project structure, and the 'Project Model' view at the bottom shows various package and datatype definitions. A green box highlights the 'Project Explorer' view on the right, which contains generated files: 'LabsProfile.html', 'LabsProfile.legacy-rdfs', and 'LabsProfile.owl'. Numbered callouts point to specific elements: '1' points to the 'Summary' tab in the bottom bar; '2' points to the 'Builder for html' checkbox; '3' points to the 'Builder for legacy-rdfs' checkbox; and '4' points to the save icon in the toolbar.

1

2

3

4

CIMTool - SIMESLabs/Profiles/LabsProfile.owl - Eclipse Platform

File Edit Navigate Search Project Run Window Help

Project Explorer X

SIMESLabs

- Incremental
- Instances
- Profiles
 - LabsProfile.owl
- Schema
 - simes-labs-schema.xmi

Project Model X

- Package: IEC61970_IEC61968_combine
- Class: Package
- Class: Stereotype
- Datatype: EnergyAsMWh
- Datatype: EnumeratedType
- Datatype: FlowgateAfcUseCode
- Datatype: FlowgateIdcType

*LabsProfile.owl X

Summary

Location: /SIMESLabs/Profiles/LabsProfile.owl

Namespace: http://ftn.uns.ac.rs/esi/simes/2013/LabsProfile#

Edit

Reorganize and Repair

Build the following from this profile:

- Builder for html
- Builder for java
- Builder for legacy-rdfs
- Builder for legacy-rdfs-augmented
- Builder for scala
- Builder for simple-flat-owl
- Builder for simple-flat-owl-augmented
- Builder for simple-owl
- Builder for simple-owl-augmented
- Builder for sql
- Builder for ttl
- Builder for xml
- Builder for xsd

Add/Remove Hierarchy Detail Stereotypes Summary

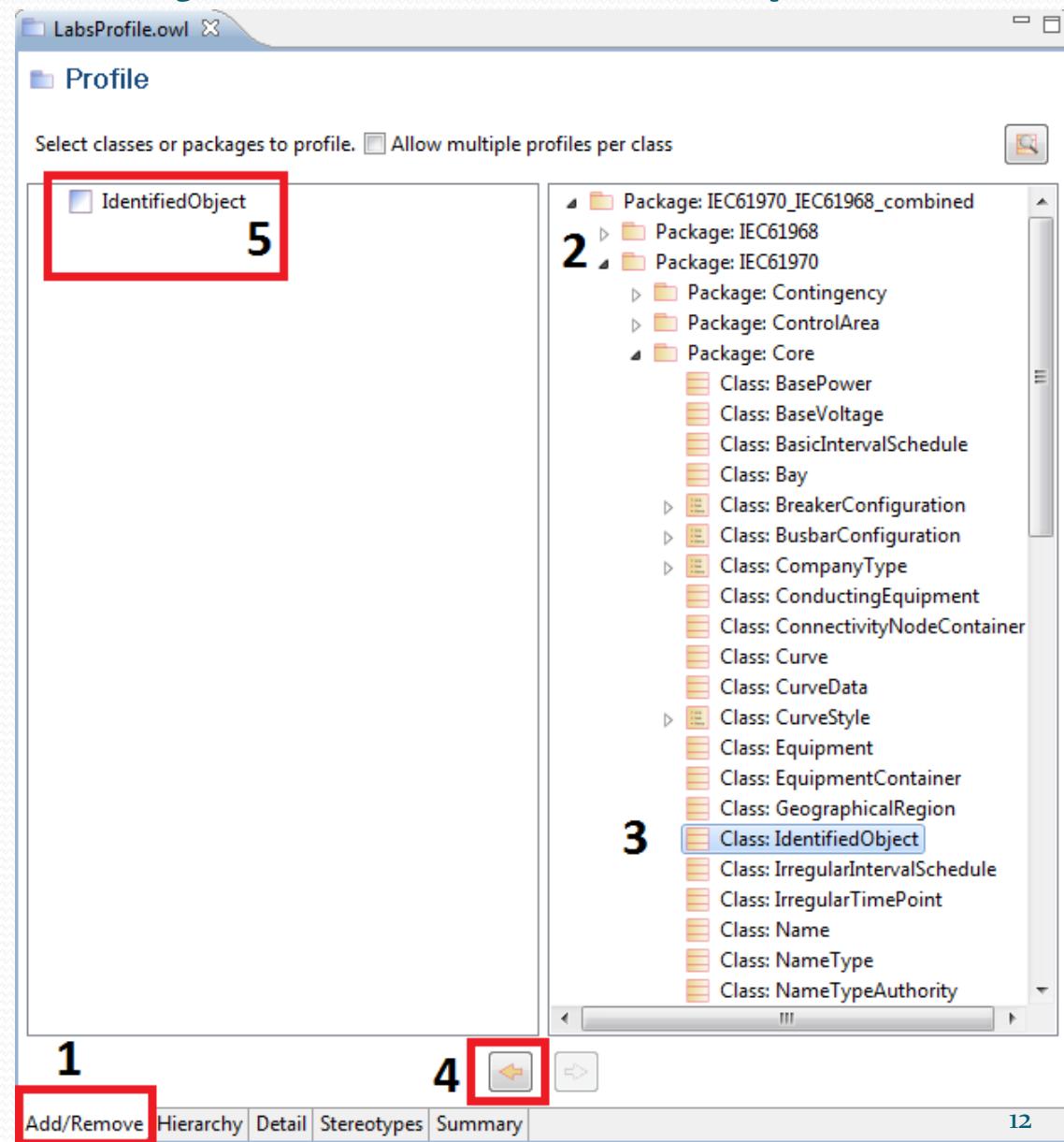
Project Explorer X

- SIMESLabs
 - Incremental
 - Instances
 - Profiles
 - LabsProfile.html
 - LabsProfile.legacy-rdfs
 - LabsProfile.owl
 - Schema
 - simes-labs-schema.xmi

11

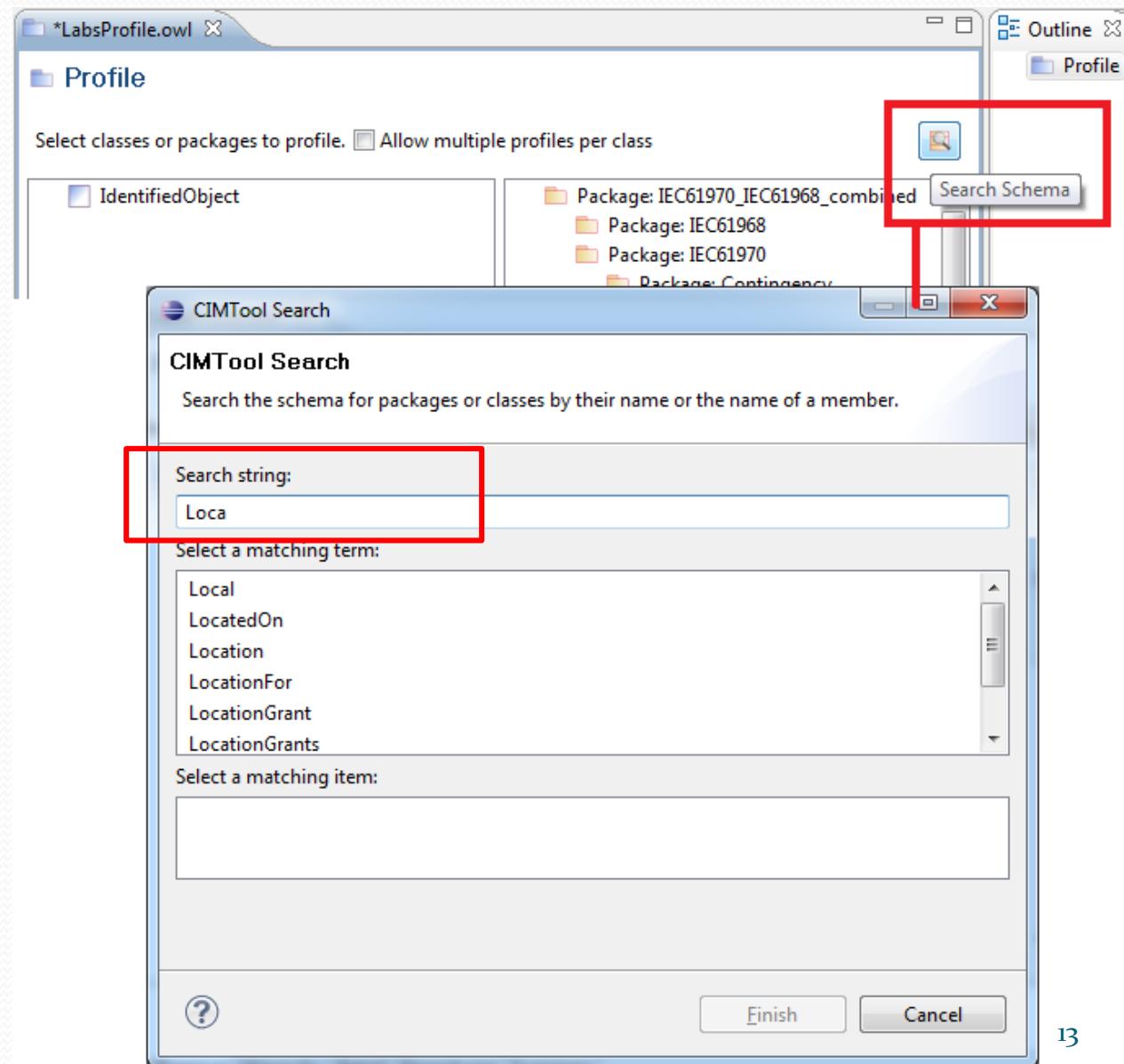
CIMTool: Dodavanje klase u CIM profil

- Prvi način:
 - odabirom iz liste



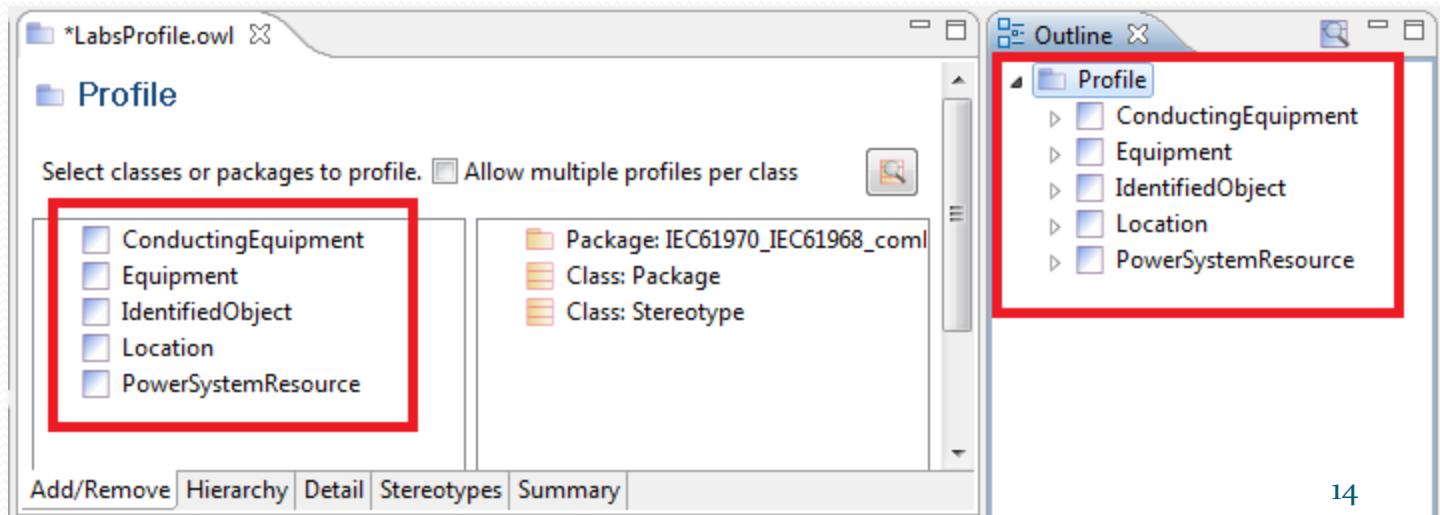
CIMTool: Dodavanje klase u CIM profil

- Drugi način:
 - *Search Schema*



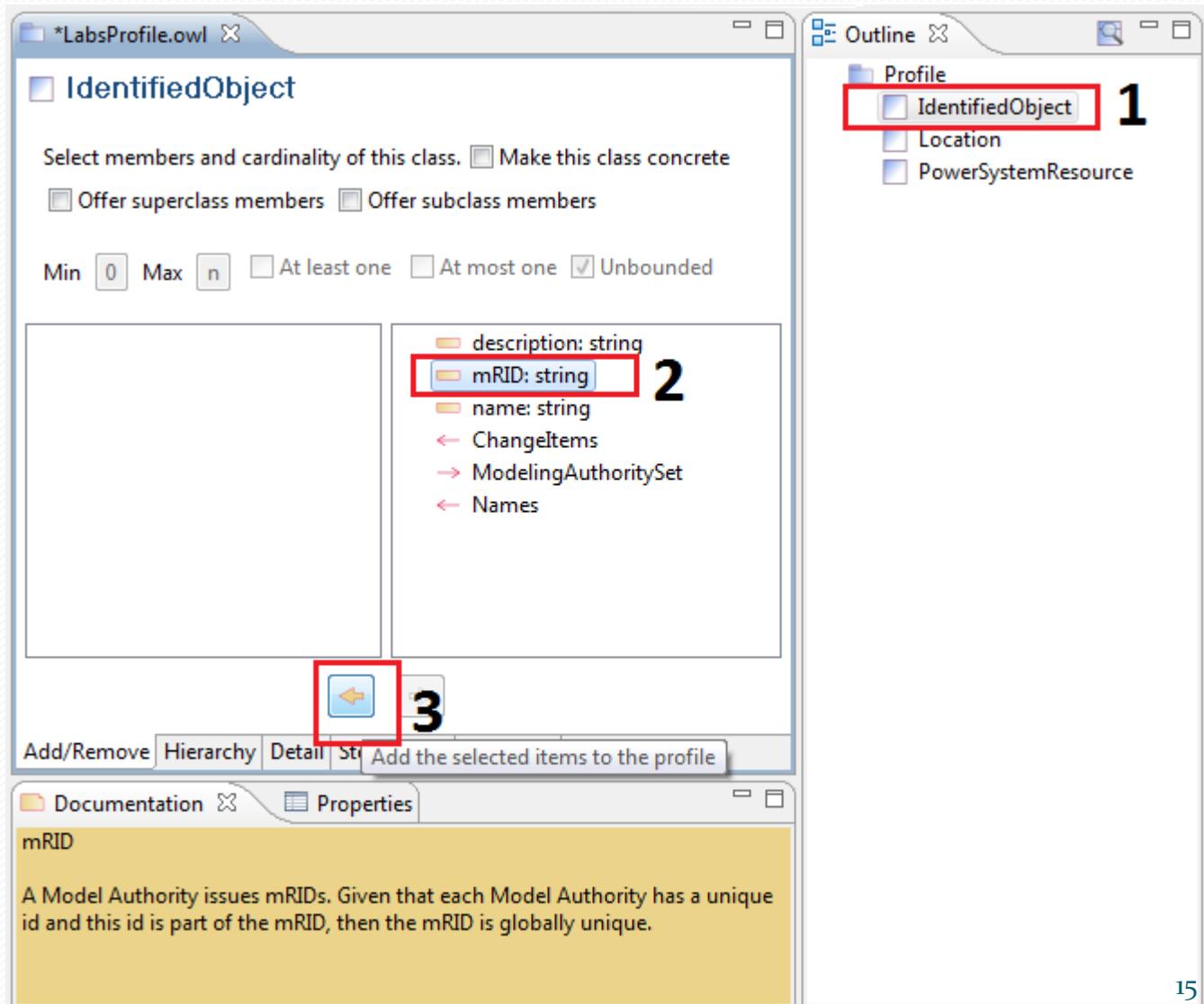
CIMTool: Dodavanje klase u CIM profil

- Vežba 2.1 :
 - Dodati u CIM profil klase:
 - IdentifiedObject
 - Location
 - PowerSystemResource
 - Equipment
 - ConductingEquipment
 - Cilj je dobiti prikaz kao na slici



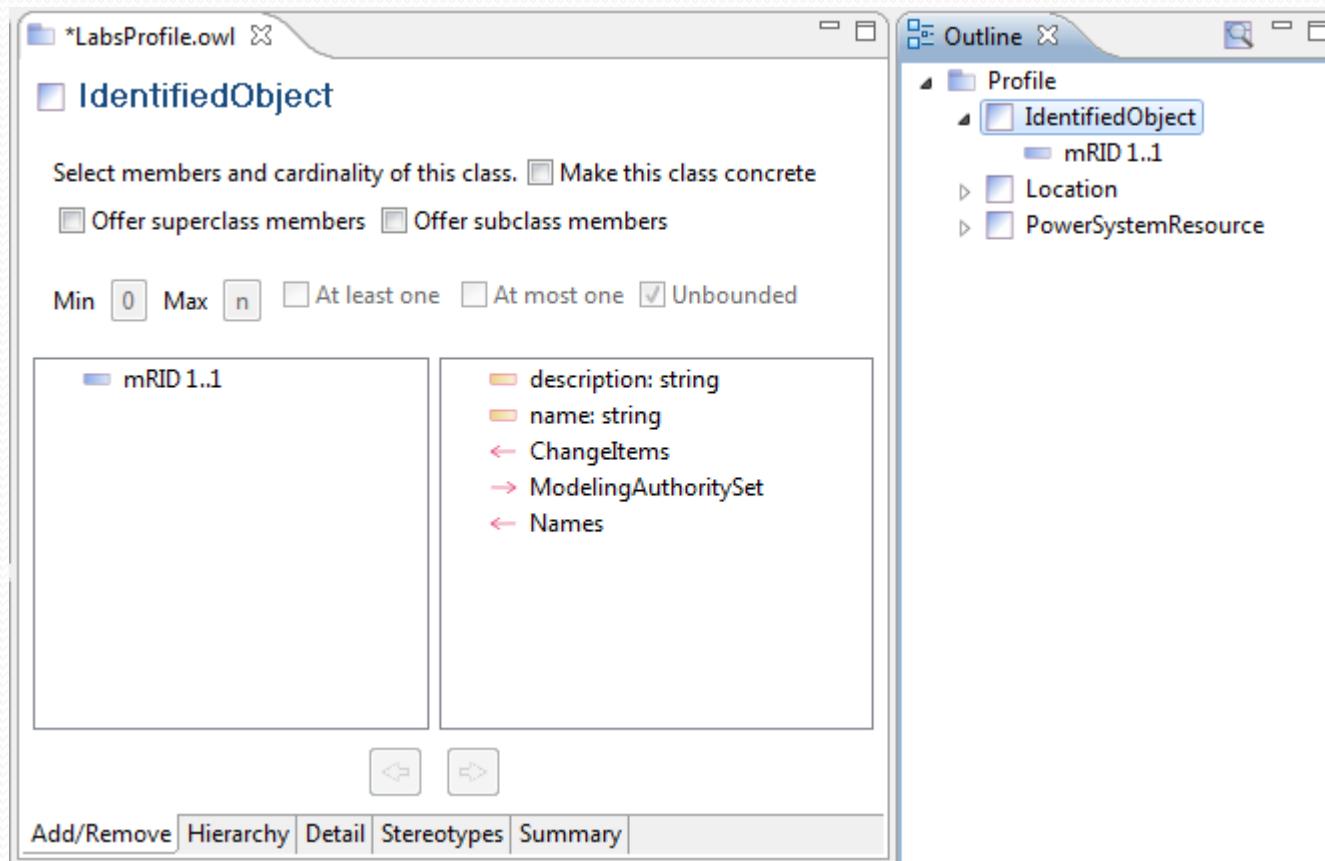
CIMTool: Dodavanje atributa u CIM profil

- Koraci:



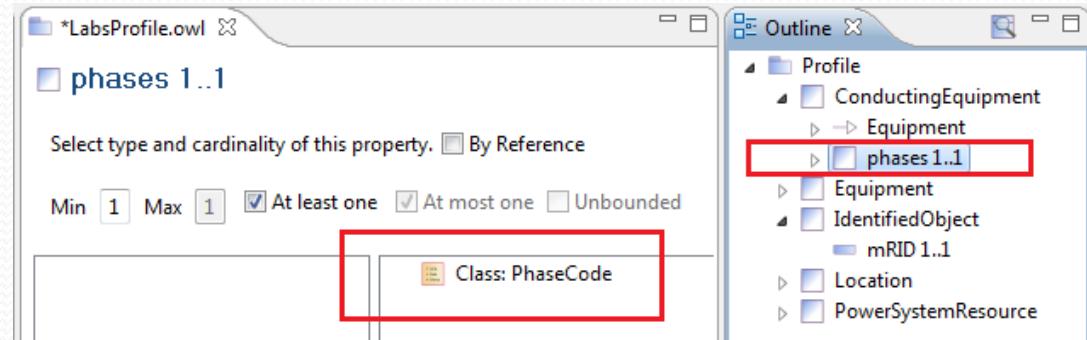
CIMTool: Dodavanje atributa u CIM profil

- Rezultat:

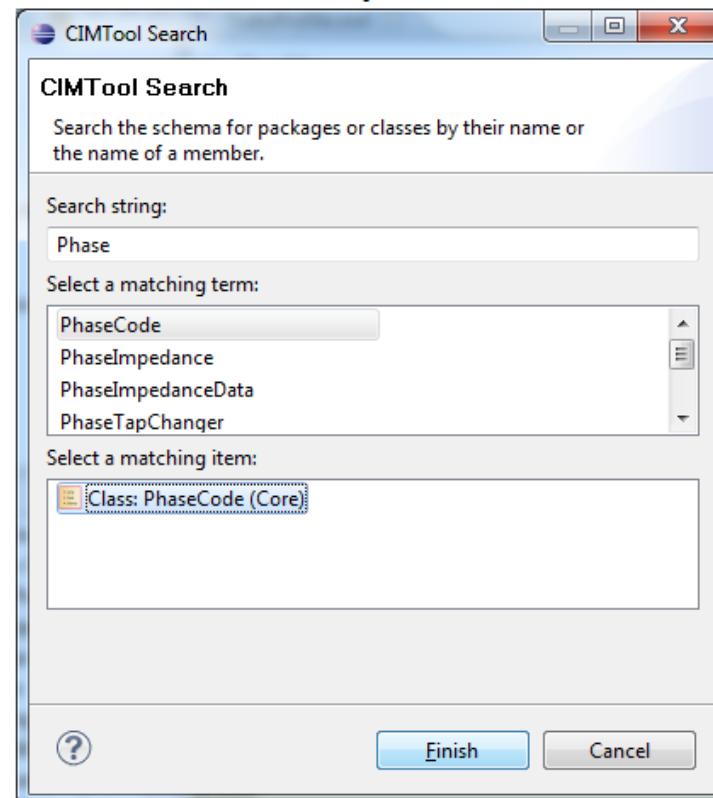


CIMTool: Dodavanje atributa u CIM profil

- Slučaj atributa čiji tip podatka je enumeracija

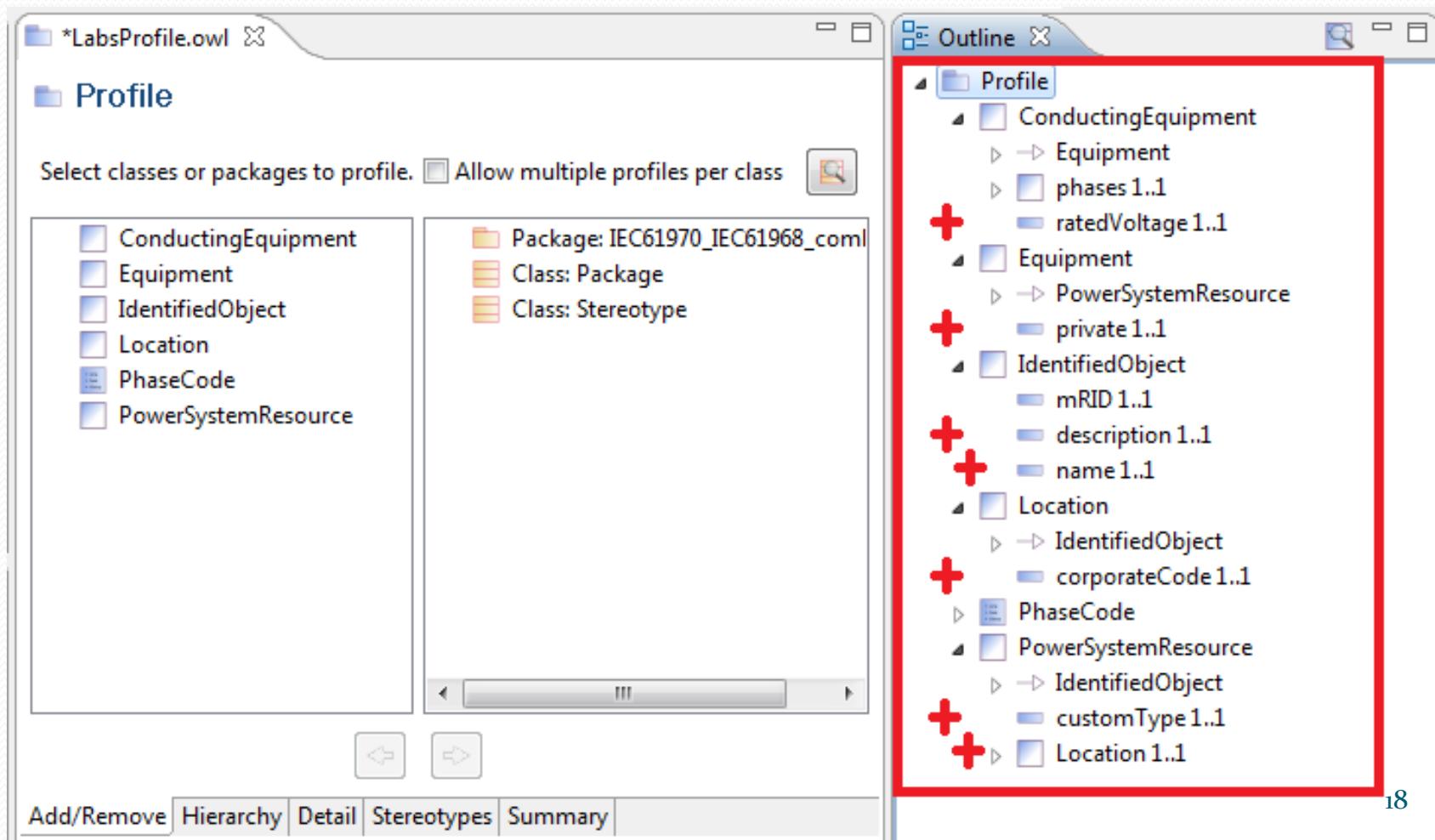


find enum class & add it to profile



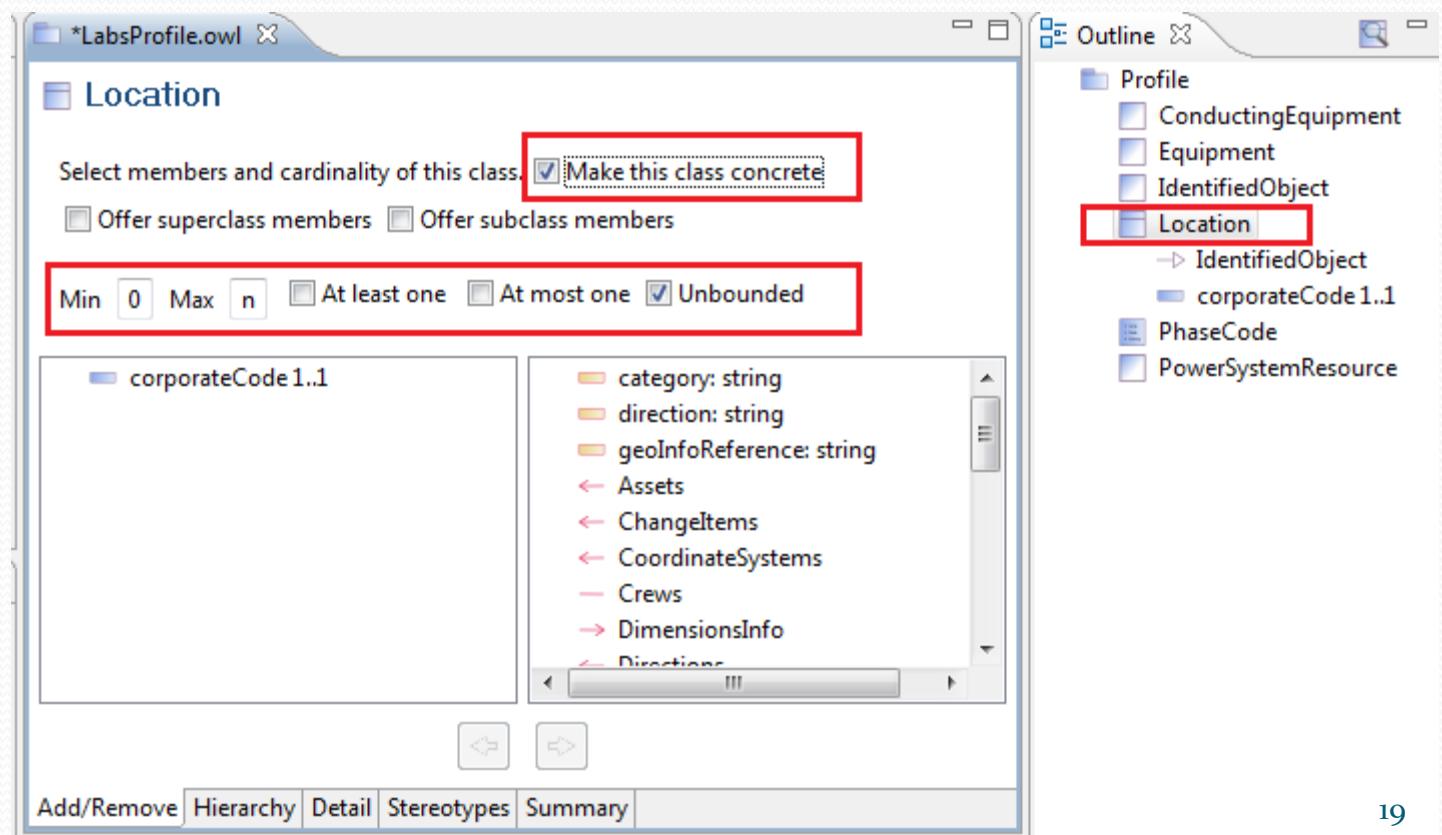
CIMTool: Dodavanje atributa u CIM profil

- Vežba 2.2:
 - Dodati u klase CIM profila atribute kao na slici



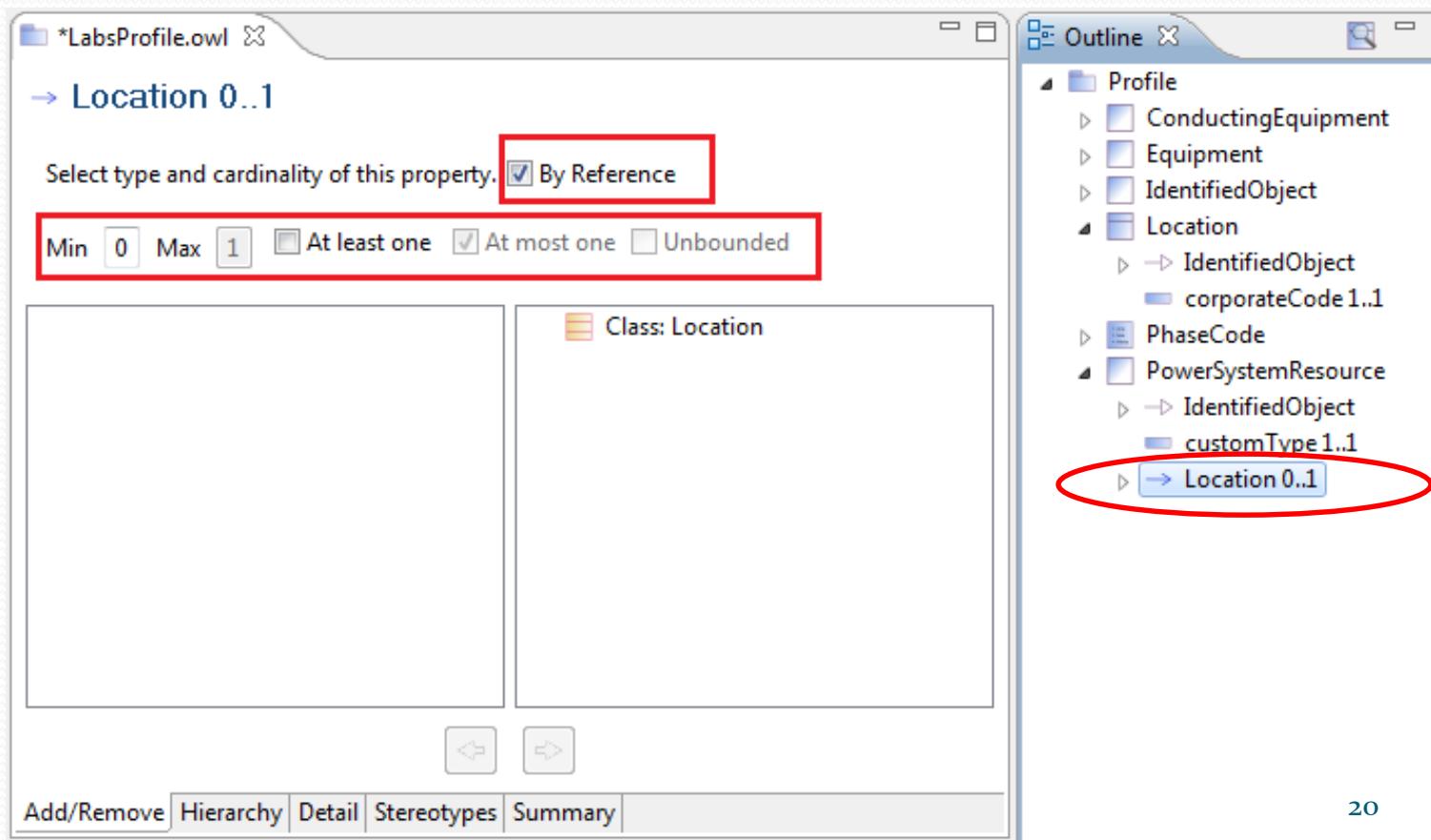
CIMTool: Konfiguracija klase u CIM profilu

- *Abstract VS Concrete class*
- Instance occurrence



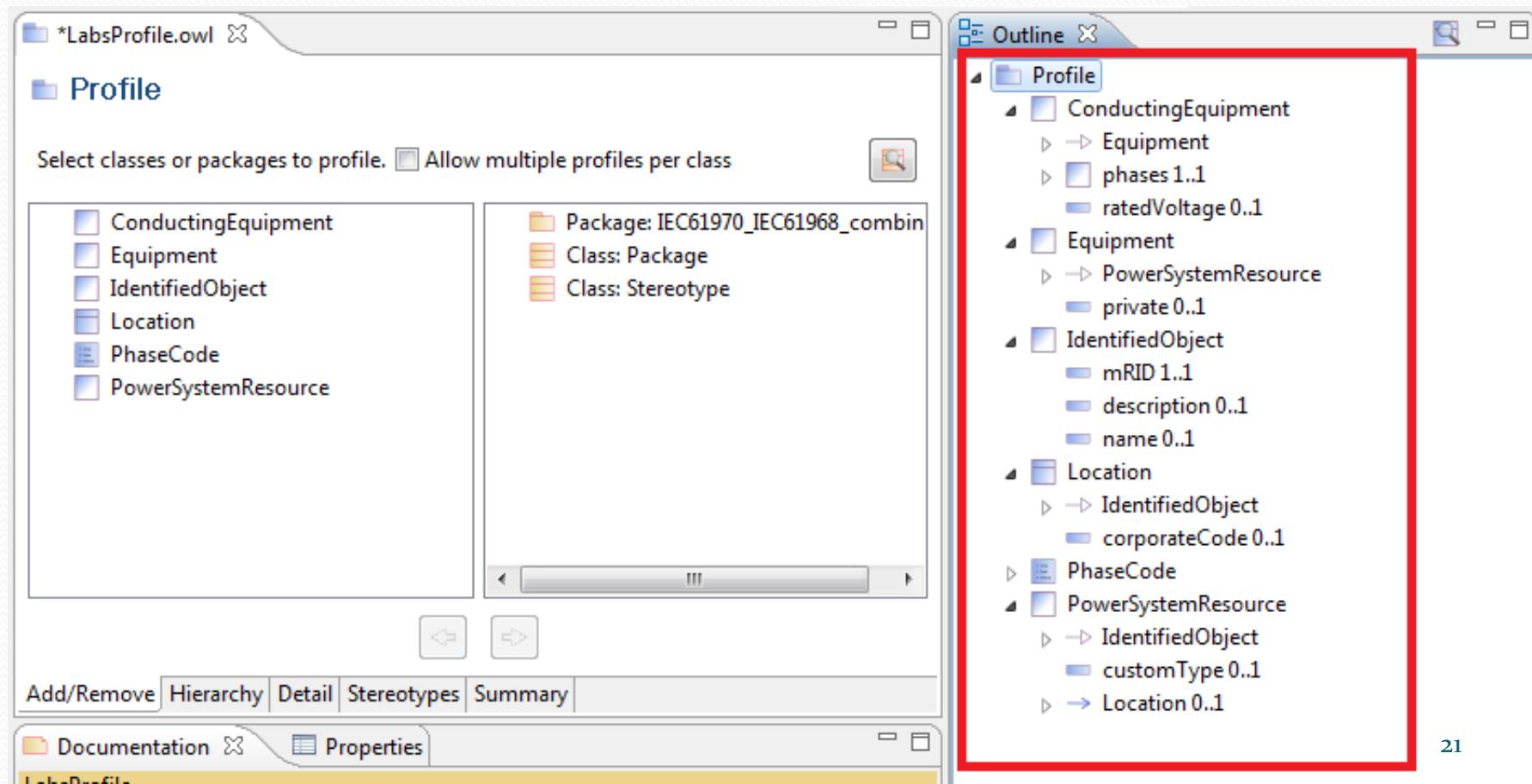
CIMTool: Konfiguracija atributa u CIM profilu

- Opcija „By reference“
- Kardinalitet



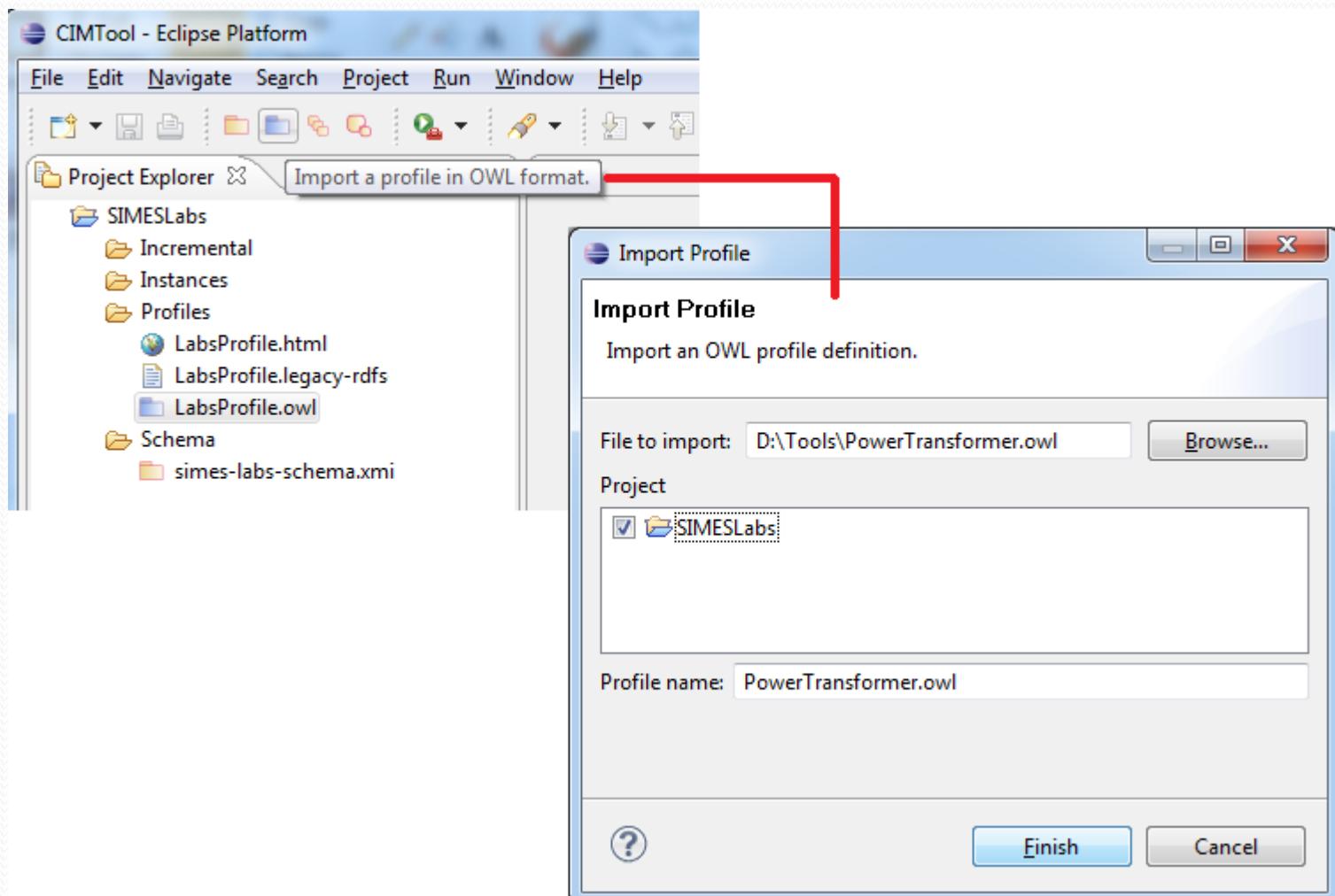
CIMTool: Konfiguracija atributa u CIM profilu

- Vežba 2.3:
 - Konfigurisati kardinalitete atributa u CIM profilu kao na slici



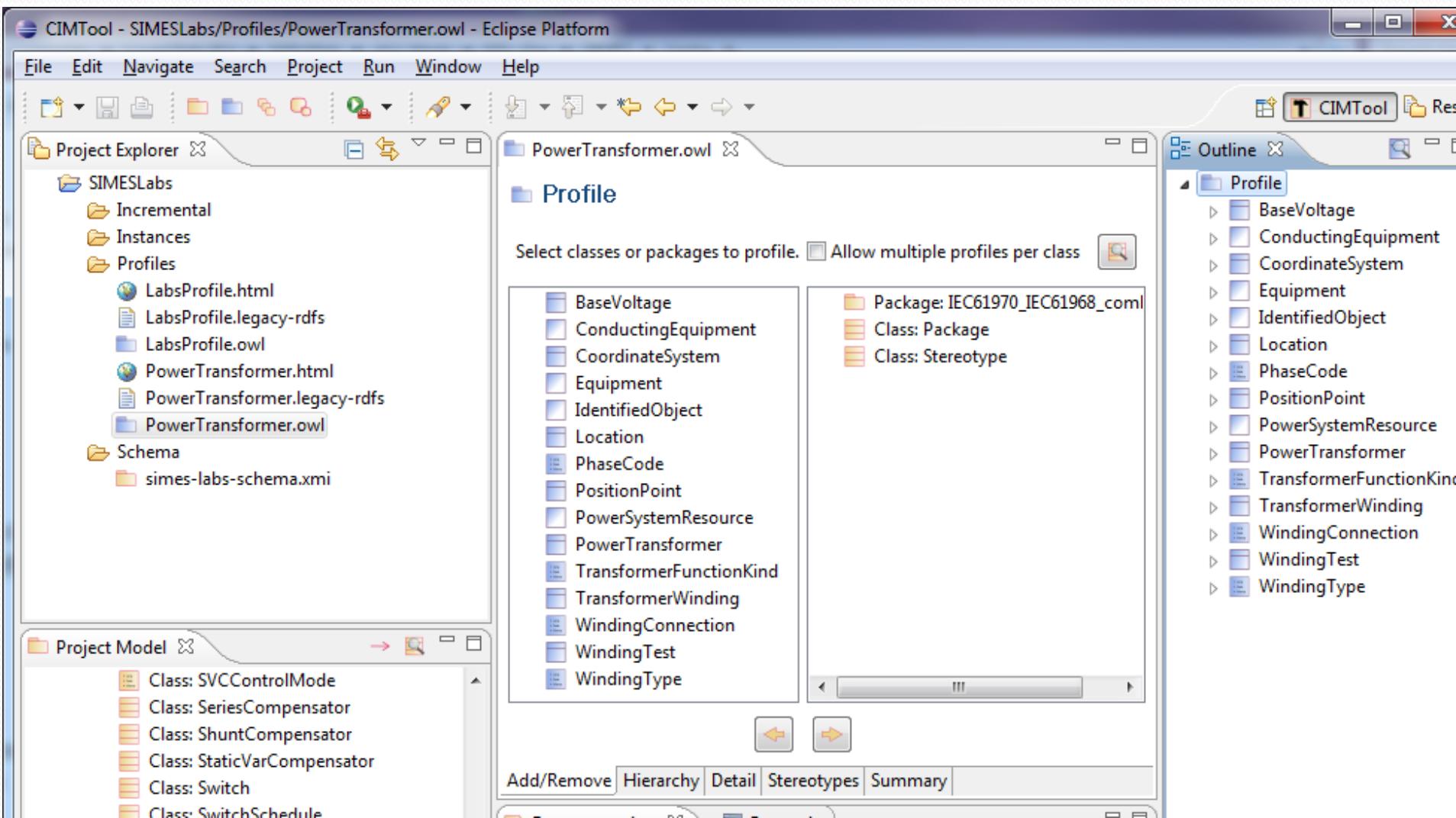
CIMTool: Import postojećeg CIM profila u projekat

- Koraci



CIMTool: Import postojećeg CIM profila u projekat

- Rezultat



CIMTool: Pregled HTML formata zapisa CIM profila

The screenshot shows the CIMTool interface running on the Eclipse Platform. The title bar indicates the file is 'PowerTransformer.html' located at 'file:///D:/SSELENA/ALL Work/ESI - Fax/SIMES/CIMWorkspace/SIMESLabs/Profiles/'. The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations.

The left side features the Project Explorer view, which lists the project structure:

- SIMESLabs
 - Incremental
 - Instances
 - Profiles
 - LabsProfile.html
 - LabsProfile.legacy-rdfs
 - LabsProfile.owl
 - PowerTransformer.html**
 - PowerTransformer.legacy-rdfs
 - PowerTransformer.owl
 - Schema
 - simes-labs-schema.xmi

The right side displays the 'Profile Documentation' view for 'PowerTransformer.owl'. The title is 'Profile Documentation'. The profile namespace is listed as <http://ftn.uns.ac.rs/esi/simes/2013/PowerTransformerProfile#>. A section titled 'Concrete Classes' contains a detailed entry for 'BaseVoltage':

BaseVoltage
Defines a nominal base voltage which is referenced in the system.

Native Members

nominalVoltage	1..1	Voltage	The PowerSystemResource's base voltage.
-----------------------	------	-------------------------	---

Below this is a 'Documentation' tab for 'PowerTransformerProfile'.

At the bottom left is the 'Project Model' view, listing various classes such as SVCControlMode, SeriesCompensator, ShuntCompensator, StaticVarCompensator, Switch, SwitchSchedule, SynchronousMachine, SynchronousMachineOperatingM, SynchronousMachineType, TapChanger, and TapChangerKind.

At the bottom right is the page number '24'.

CIMTool: Pregled RDFS formata zapisa CIM profila

The screenshot shows the CIMTool interface running on the Eclipse Platform. The title bar reads "CIMTool - SIMESLabs/Profiles/PowerTransformer.legacy-rdfs - Eclipse Platform". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Import.

The left side features the "Project Explorer" view, which displays the project structure under "SIMESLabs". It includes "Incremental", "Instances", "Profiles" (containing "LabsProfile.html", "LabsProfile.legacy-rdfs", "LabsProfile.owl", "PowerTransformer.html", "PowerTransformer.legacy-rdfs", "PowerTransformer.owl"), and "Schema" (containing "simes-labs-schema.xmi").

The central area contains two tabs: "PowerTransformer.owl" and "PowerTransformer.legacy-rdfs". The "PowerTransformer.legacy-rdfs" tab is active, showing the XML code for the profile:

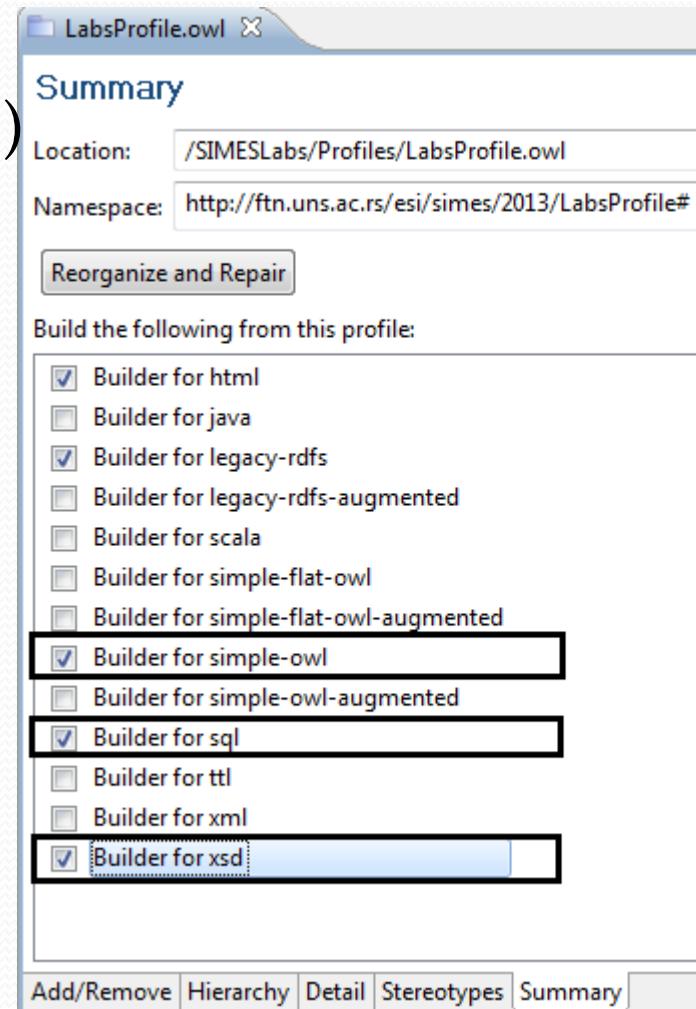
```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cims="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:msg="http://langdale.com.au/2005/Message#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:uml="http://langdale.com.au/2005/UML#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:j_0="http://iec.ch/TC57/2010/CIM-schema-cim15#"
    xml:base="http://iec.ch/TC57/2010/CIM-schema-cim15" >
<rdf:Description rdf:about="#TransformerWinding.windingType">
    <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#attribute"/>
    <rdfs:comment>The type of winding.</rdfs:comment>
    <rdfs:label>windingType</rdfs:label>
    <rdfs:range rdf:resource="#WindingType"/>
    <rdfs:domain rdf:resource="#TransformerWinding"/>
    <cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#"
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>
<rdf:Description rdf:about="#Package_Wires">
    <cims:belongsToCategory rdf:resource="#Package_IEC61970"/>
    <rdfs:comment>An extension to the Core and Topology package that models information on t</rdfs:comment>
    <rdfs:label>Wires</rdfs:label>
    <rdf:type rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#ClassCate</rdf:type>
</rdf:Description>

```

The bottom right corner shows a yellow status bar with the text "PowerTransformerProfile".

CIMTool: Biranje formata zapisa CIM profila

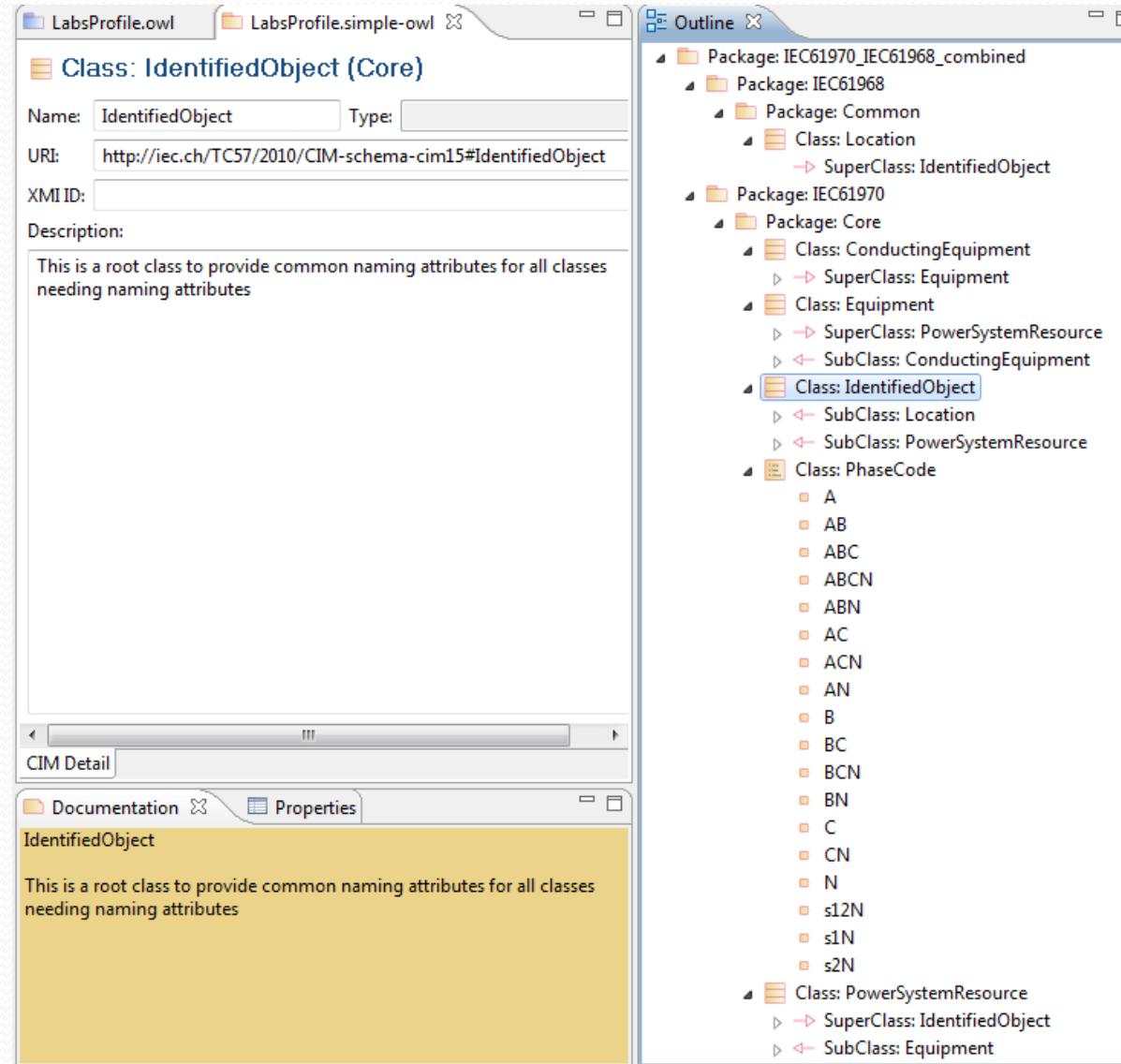
- Na raspolaganju su i drugi formati od kojih izdvajamo:
 - XSD
 - OWL (Web Ontology Language)
 - SQL



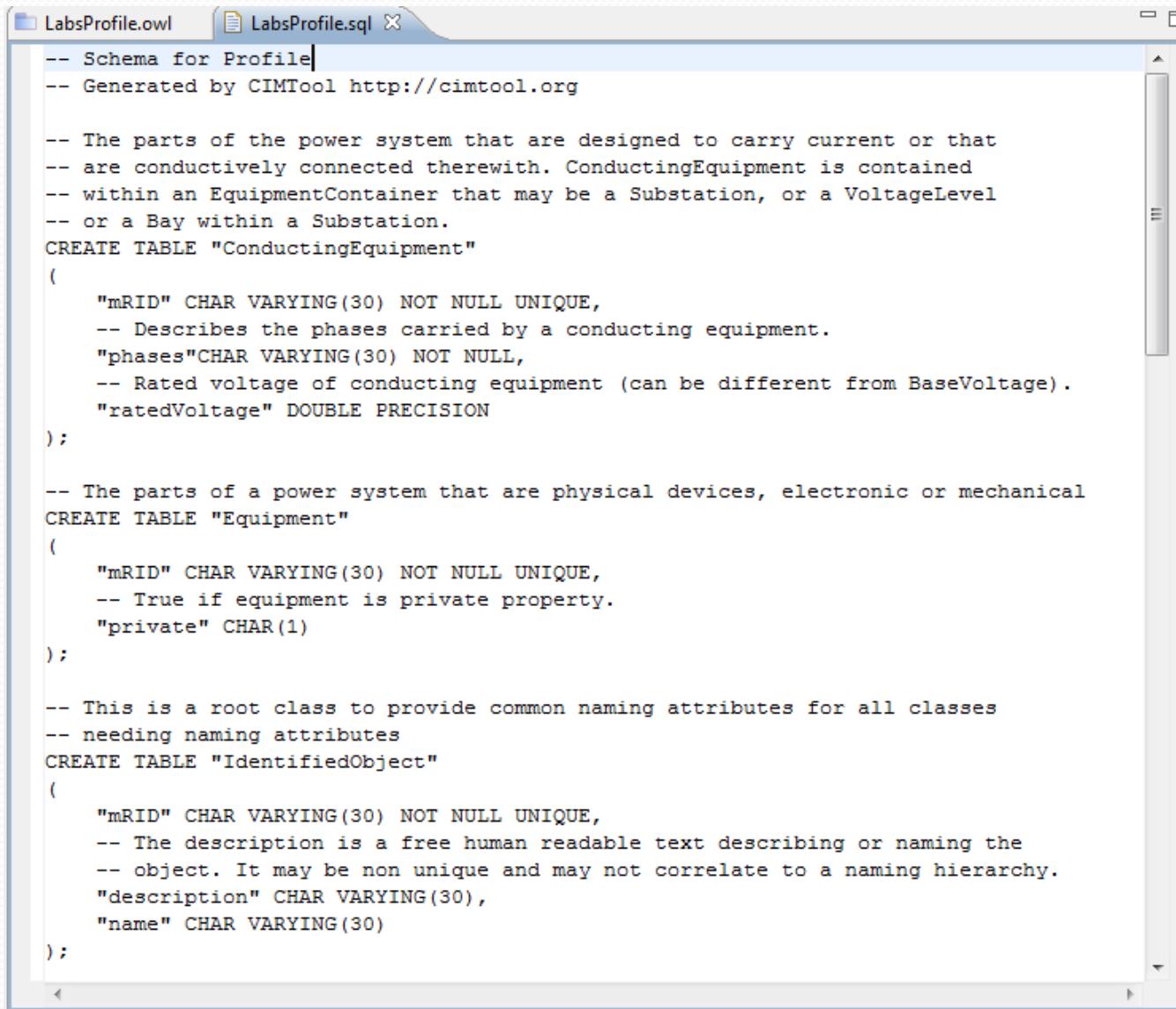
CIMTool: Pregled XSD formata zapisa CIM profila

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:a="http://langdale.com.au/2005/Message#" xmlns:sawsdl="http://www.w3.org/ns/sawsdl" targetNamespace="http://ftn.uns.ac.rs/esi/simes/2013/LabsProfile#" elementFormDefault="qualified" attributeFormDefault="unqualified" xmlns="http://langdale.com.au/2005/Message#" xmlns:m="http://ftn.uns.ac.rs/esi/simes/2013/LabsProfile#">
    <xs:annotation/>
    <xs:element name="Profile" type="m:Profile"/>
    <xs:complexType name="Profile">
        <xs:sequence>
            <xs:element name="Location" type="m:Location" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ConductingEquipment" sawsdl:modelReference="http://iec.ch/TC57/2010/CIM-schema-cim15#ConductingEquipment">
        <xs:annotation>
            <xs:documentation>The parts of the power system that are designed to carry current or that are conductively connected therewith. ConductingEquipment is contained within an EquipmentContainer that may be a Substation, or a VoltageLevel or a Bay within a Substation.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="m:Equipment">
                <xs:sequence>
                    <xs:element name="phases" minOccurs="1" maxOccurs="1" sawsdl:modelReference="http://iec.ch/TC57/2010/CIM-schema-cim15#ConductingEquipment.phases">
                        <xs:annotation>
                            <xs:documentation>Describes the phases carried by a conducting equipment.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType sawsdl:modelReference="">
                            <xs:attribute name="ref" type="xs:string"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="ratedVoltage" minOccurs="0" maxOccurs="1" type="xs:float" sawsdl:modelReference="http://iec.ch/TC57/2010/CIM-schema-cim15#ConductingEquipment.ratedVoltage">
                        <xs:annotation>
                            <xs:documentation>Rated voltage of conducting equipment (can be different from BaseVoltage).</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

CIMTool: Pregled simple-OWL formata zapisa CIM profila



CIMTool: Pregled SQL formata zapisa CIM profila



```
-- Schema for Profile
-- Generated by CIMTool http://cimtool.org

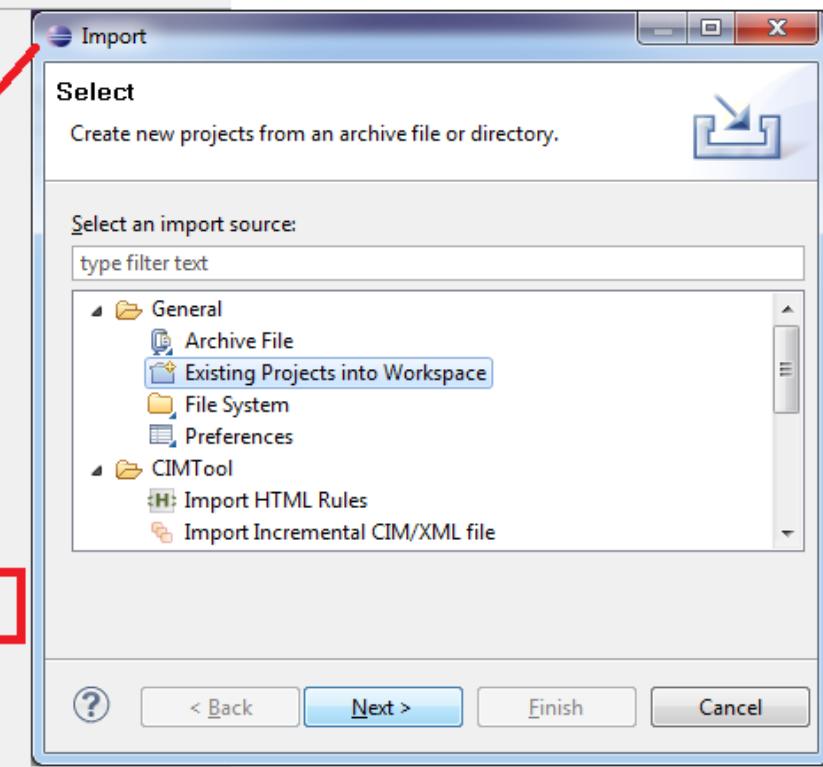
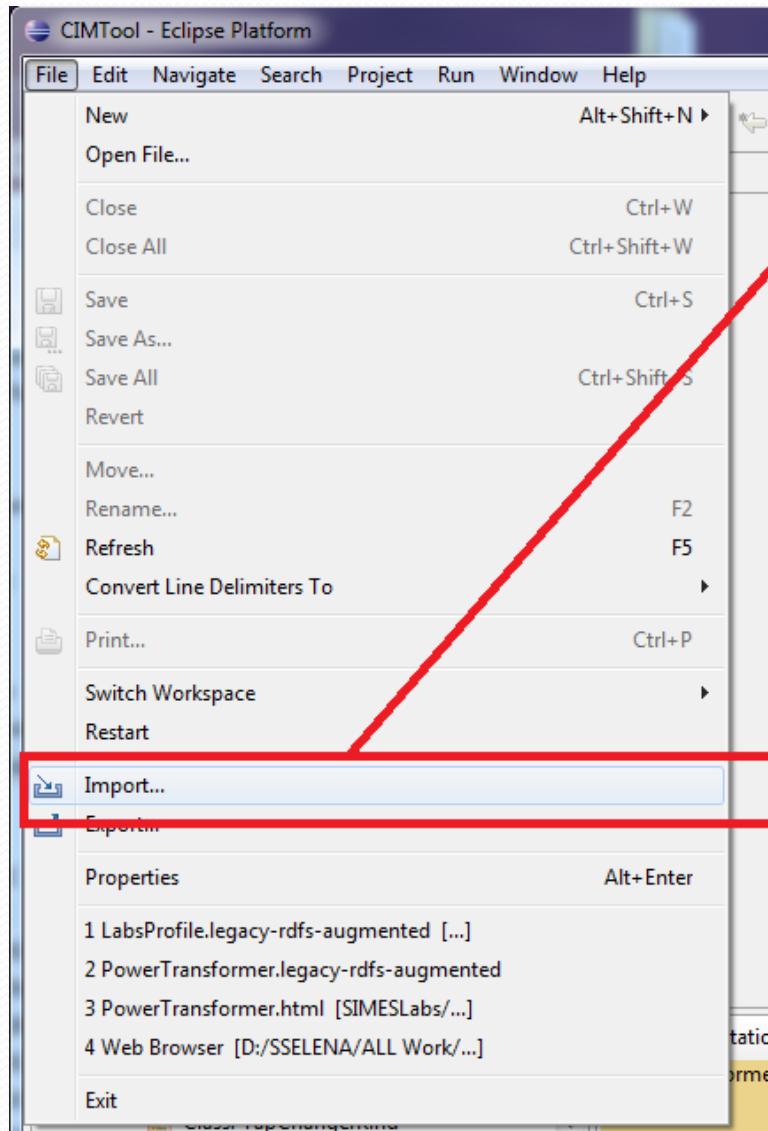
-- The parts of the power system that are designed to carry current or that
-- are conductively connected therewith. ConductingEquipment is contained
-- within an EquipmentContainer that may be a Substation, or a VoltageLevel
-- or a Bay within a Substation.
CREATE TABLE "ConductingEquipment"
(
    "mRID" CHAR VARYING(30) NOT NULL UNIQUE,
    -- Describes the phases carried by a conducting equipment.
    "phases"CHAR VARYING(30) NOT NULL,
    -- Rated voltage of conducting equipment (can be different from BaseVoltage).
    "ratedVoltage" DOUBLE PRECISION
);

-- The parts of a power system that are physical devices, electronic or mechanical
CREATE TABLE "Equipment"
(
    "mRID" CHAR VARYING(30) NOT NULL UNIQUE,
    -- True if equipment is private property.
    "private" CHAR(1)
);

-- This is a root class to provide common naming attributes for all classes
-- needing naming attributes
CREATE TABLE "IdentifiedObject"
(
    "mRID" CHAR VARYING(30) NOT NULL UNIQUE,
    -- The description is a free human readable text describing or naming the
    -- object. It may be non unique and may not correlate to a naming hierarchy.
    "description" CHAR VARYING(30),
    "name" CHAR VARYING(30)
);
```

CIMTool: Import postojećeg CIM projekta

- Korak 1



- Korak 2

CIMTool: Import postojećeg CIM projekta

- Korak 3

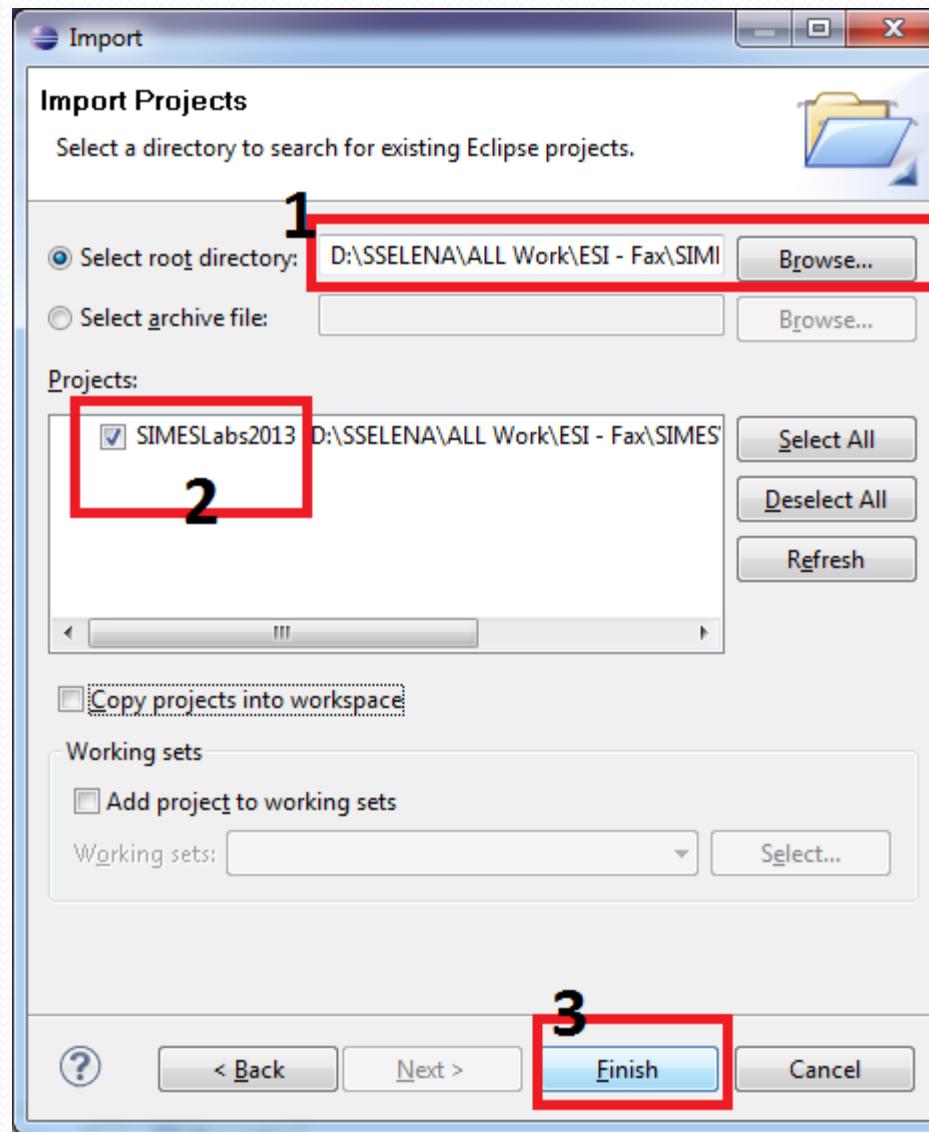
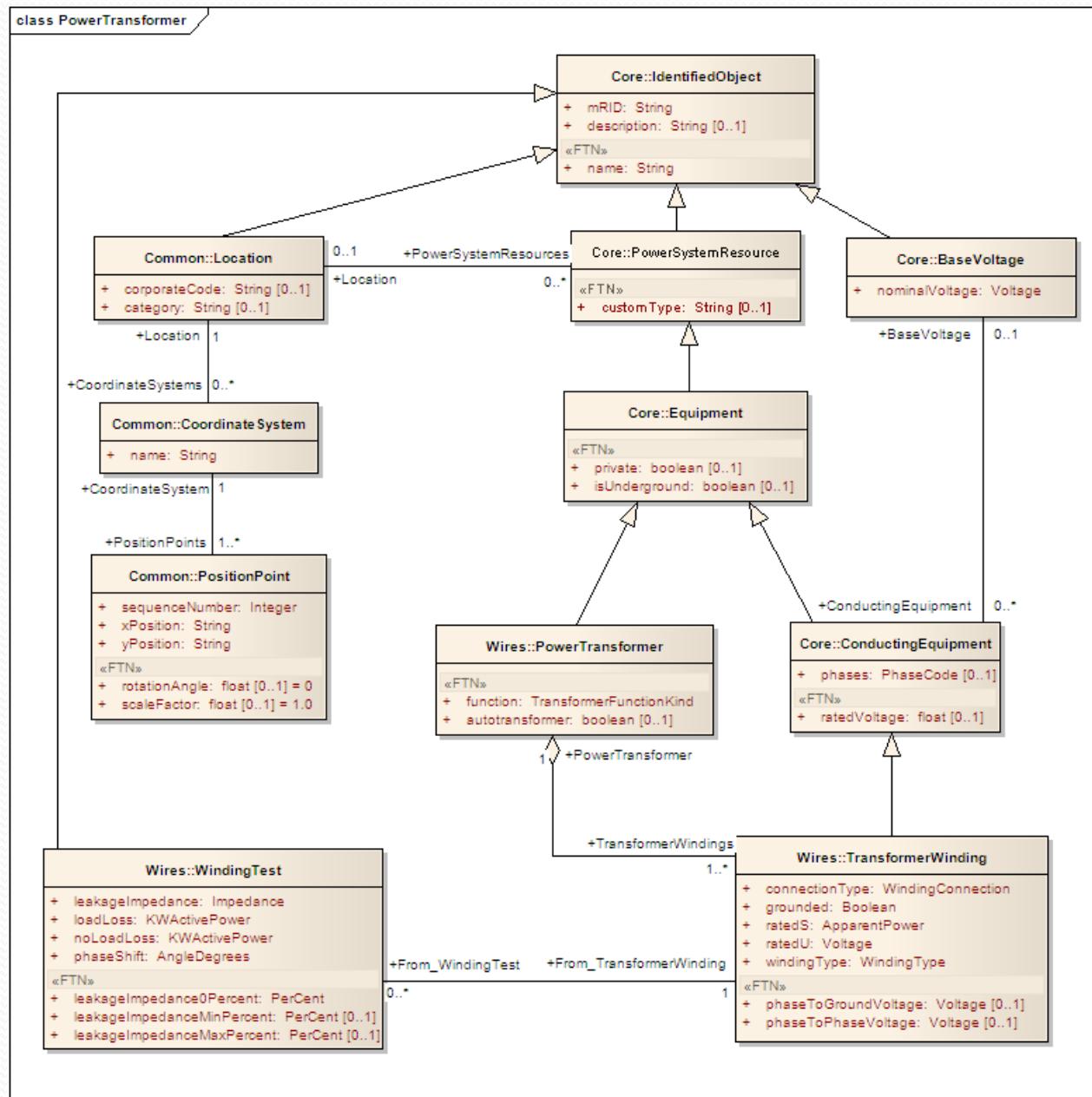


Diagram klasa CIM profila



Zadatak: Izmena CIM profila

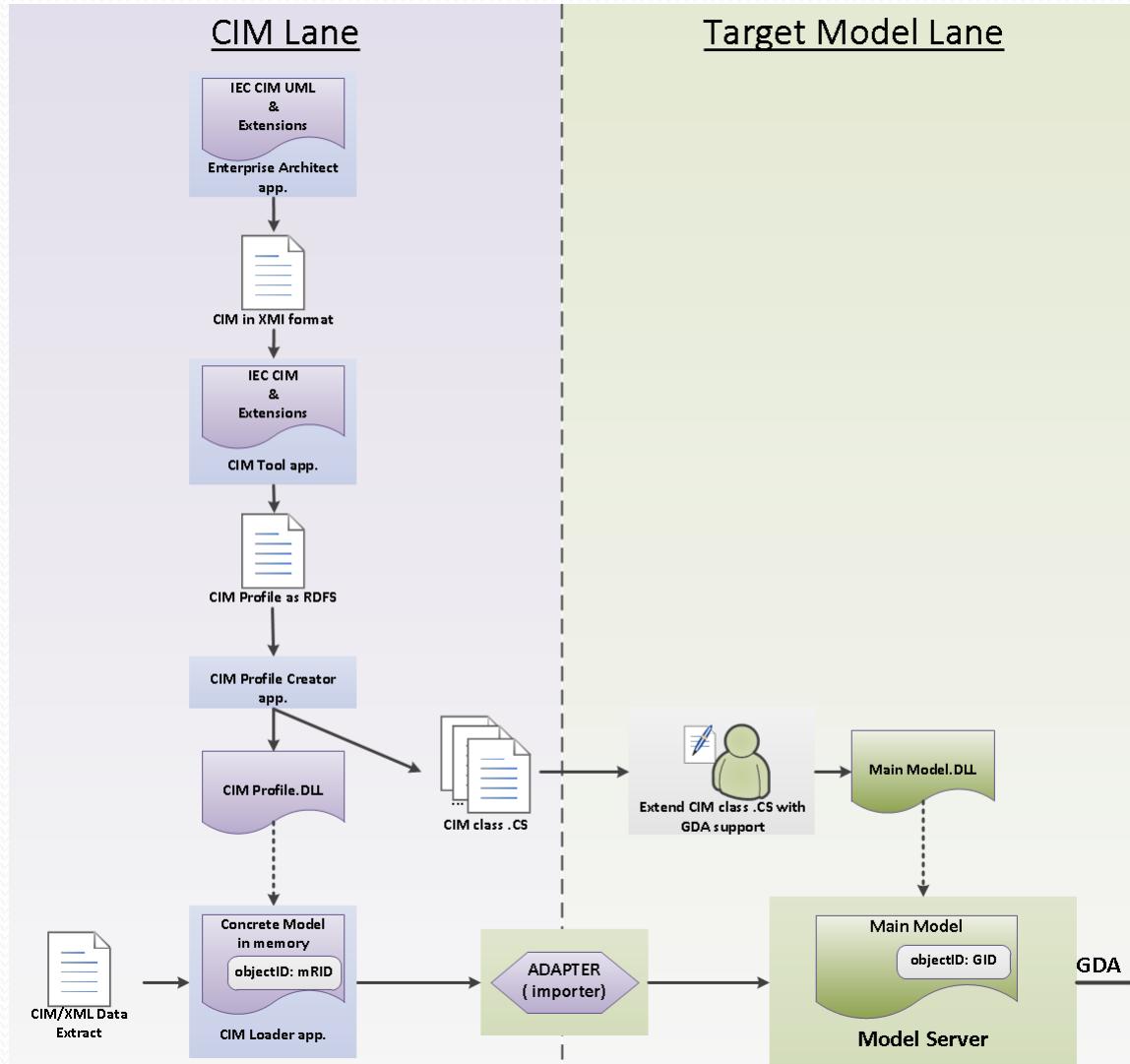
- 1) U EA alatu napraviti izmene *PowerTransformer* diagrama:
 - 1) dodati novi atribut u postojeću klasu
 - 2) dodati novu klasu (koristiti generalizaciju)
 - 3) definisati asocijaciju izmedju 2 klase (razlikovati tipove asocijacija).
- 2) Izgenerisati XMI file.
- 3) Učitati izgenerisani XMI file u dati CIMTool projekat.
- 4) U *PowerTransformer* profilu ispratiti izmene prethodno uradjene na diagramu klasa.
- 5) Snimiti izmenjen profil u RDFS formatu i pronaći dodate elemente (klase, attribute).

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 3:

Uvod u Parser CIM profila datog u RDFS formatu

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Utvrđivanje

- Enterprise Architect alat
- CIMTool alat
- Zadatak

CIMTool: Pregled RDFS formata zapisa CIM profila

The screenshot shows the CIMTool interface running on the Eclipse Platform. The title bar reads "CIMTool - SIMESLabs/Profiles/PowerTransformer.legacy-rdfs - Eclipse Platform". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Import.

The main window is divided into several panes:

- Project Explorer**: Shows the project structure. It contains a folder "SIMESLabs" which includes "Incremental", "Instances", "Profiles" (containing "LabsProfile.html", "LabsProfile.legacy-rdfs", "LabsProfile.owl", "PowerTransformer.html", "PowerTransformer.legacy-rdfs", "PowerTransformer.owl"), and "Schema" (containing "simes-labs-schema.xmi").
- Editor**: Displays two tabs: "PowerTransformer.owl" and "PowerTransformer.legacy-rdfs". The "PowerTransformer.legacy-rdfs" tab shows the XML code for the RDFS profile, which includes namespaces for RDF, CIMS, OWL, and other schema definitions. The code defines descriptions for "windingType" and "Package_Wires" properties.
- Project Model**: Shows a list of packages and datatypes defined in the model. Packages include "IEC61970_IEC61968_combine" and "Package". Datatypes listed are: EnergyAsMWh, EnumeratedType, FlowgateAfcUseCode, FlowgateIdcType, FreqBiasFactor, PenaltyFactor, PotheadType, and PowerROCPerMin.
- Documentation**: A tab labeled "PowerTransformerProfile".
- Properties**: A tab labeled "PowerTransformerProfile".

Namespaces u RDFS dokumentu

```
<rdf:RDF  
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
    xmlns:cims="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#"  
    xmlns:owl="http://www.w3.org/2002/07/owl#"  
    xmlns:msg="http://langdale.com.au/2005/Message#"  
    xmlns:dc="http://purl.org/dc/elements/1.1/"  
    xmlns:uml="http://langdale.com.au/2005/UML#"  
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"  
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  
    xmlns:j.0="http://iec.ch/TC57/2010/CIM-schema-cim15#"  
    xml:base="http://iec.ch/TC57/2010/CIM-schema-cim15" >
```

Deklaracija paketa u RDFS formatu

- Primer paketa

```
<rdf:Description rdf:about="#Package_Wires">
  <cims:belongsToCategory rdf:resource="#Package_IEC61970"/>
  <rdfs:comment>An extension to the Core and Topology package that models information
on the electrical characteristics of Transmission and Distribution networks.
This package is used by network applications such as State Estimation,
Load Flow and Optimal Power Flow.</rdfs:comment>
  <rdfs:label>Wires</rdfs:label>
  <rdf:type rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#ClassCategory"/>
</rdf:Description>
```

- Bitni elementi

- rdf:about – identifikator resursa
- belongsToCategory – pripadnost drugom paketu
- type – tip resursa:
 - <http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#ClassCategory>

Deklaracija klase u RDFS formatu 1/3

- Primer apstraktne klase

```
<rdf:Description rdf:about="#IdentifiedObject">
  <cims:belongsToCategory rdf:resource="#Package_Core"/>
  <rdfs:comment>This is a root class to provide common naming attributes for
  all classes needing naming attributes</rdfs:comment>
  <rdfs:label>IdentifiedObject</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

- Bitni elementi

- rdf:about – identifikator resursa
- belongsToCategory – pripadnost paketu
- type – tip resursa:
 - <http://www.w3.org/2000/01/rdf-schema#Class>

Deklaracija klase u RDFS formatu 2/3

- Primer *concrete* klase

```
<rdf:Description rdf:about="#PowerTransformer">
  <rdfs:subClassOf rdf:resource="#Equipment"/>
  <cims:belongsToCategory rdf:resource="#Package_Wires"/>
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#byreference"/>
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#concrete"/>
  <rdfs:comment>An electrical device consisting of two or more coupled windings,
with or without a magnetic core, for introducing mutual coupling between
electric circuits...</rdfs:comment>
  <rdfs:label>PowerTransformer</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

- Bitni elementi

- ...
- subClassOf – roditeljska-klasa
- stereotype – iskorišćen za dodatni opis resursa:
 - http://langdale.com.au/2005/UML#concrete
 - http://langdale.com.au/2005/UML#byreference

Deklaracija klase u RDFS formatu 3/3

- Primeri prostih klasa tj. *Datatype* klasa

```
<rdf:Description rdf:about="#Voltage">
    <rdfs:comment>Electrical voltage.</rdfs:comment>
    <rdfs:label>Voltage</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:about="#KWActivePower">
    <rdfs:comment>Active power in kilowatts.</rdfs:comment>
    <rdfs:label>KWActivePower</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:about="#Impedance">
    <rdfs:comment>Ratio of voltage to current.</rdfs:comment>
    <rdfs:label>Impedance</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

- Razlika u odnosu na klase je da ne sadrže sledeće elemete
 - belongsToCategory – podrazumeva se paket *Domain*
 - subClassOf – nema generalizacije među njima

Deklaracija enumeracije u RDFS formatu

- Primer enumeracije

```
<rdf:Description rdf:about="#WindingType">
  <cims:belongsToCategory rdf:resource="#Package_Wires"/>
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#enumeration"/>
  <rdfs:comment>Winding type.</rdfs:comment>
  <rdfs:label>WindingType</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

- Bitni elementi

- rdf:about – identifikator resursa
- belongsToCategory – pripadnost paketu
- type – tip resursa:
 - <http://www.w3.org/2000/01/rdf-schema#Class>
- stereotype – iskorišćen za dodatni opis resursa:
 - <http://langdale.com.au/2005/UML#enumeration>

Deklaracija vrednosti enumeracije u RDFS formatu

- Primer vrednosti enumeracije

```
<rdf:Description rdf:about="#WindingType.primary">
  <rdfs:label>primary</rdfs:label>
  <rdf:type rdf:resource="#WindingType"/>
</rdf:Description>
```

- Bitni elementi
 - rdf:about – identifikator resursa
 - type – referencira Enumeraciju putem identifikatora

Deklaracija atributa u RDFS formatu 1/5

- Primer atributa prostog (*primitive*) tipa

```
<rdf:Description rdf:about="#IdentifiedObject.mRID">
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#attribute"/>
  <rdflib:comment>A Model Authority issues mRIDs. Given that each Model Authority has a unique id and this id is part of the mRID, then the mRID is globally unique.</rdflib:comment>
  <rdflib:label>mRID</rdflib:label>
  <cims:datatype rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdflib:domain rdf:resource="#IdentifiedObject"/>
  <cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:1"/>
  <rdflib:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>
```

- Bitni elementi
 - rdf:about – identifikator resursa
 - type – tip resursa:
 - <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>
 - stereotype – iskorišćen za dodatni opis resursa:
 - <http://langdale.com.au/2005/UML#attribute>
 - dataType – tip podatka koji je prost tip (string, integer, boolean, ...¹²)

Deklaracija atributa u RDFS formatu 2/5

- Primer atributa prostog (*primitive*) tipa

```
<rdf:Description rdf:about="#IdentifiedObject.mRID">
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#attribute"/>
  <rdfs:comment>A Model Authority issues mRIDs. Given that each Model Authority has a unique id and this id is part of the mRID, then the mRID is globally unique.</rdfs:comment>
  <rdfs:label>mRID</rdfs:label>
  <cims:dataType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#IdentifiedObject"/>
  <cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:1"/>
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>
```

- Bitni elementi
 - ...
 - domain – referencira klasu kojoj atribut pripada
 - multiplicity - kardinalitet
 - http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:o..1
 - http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:1

Deklaracija atributa u RDFS formatu 3/5

- Primer atributa izvedenog (*datatype*) tipa

```
<rdf:Description rdf:about="#TransformerWinding.ratedU">
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#attribute"/>
  <rdfs:comment>The rated voltage (phase-to-phase) of the winding,
  usually the same as the neutral voltage.</rdfs:comment>
  <rdfs:label>ratedU</rdfs:label>
  <cims:dataType rdf:resource="#Voltage"/>
  <rdfs:domain rdf:resource="#TransformerWinding"/>
  <cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:0..1"/>
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>
```

- Razlika u elementima
 - dataType – referencira tip podatka koji je izведен tip tj. klasa

Deklaracija atributa u RDFS formatu 4/5

- Primer atributa tipa enumeracije

```
<rdf:Description rdf:about="#PowerTransformer.function">
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#attribute"/>
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#ftn"/>
  <rdfs:comment>Purpose of the transformer within the network. (Copied from 61970:TransformerAsset)</rdfs:comment>
  <rdfs:label>function</rdfs:label>
  <rdfs:range rdf:resource="#TransformerFunctionKind"/>
  <rdfs:domain rdf:resource="#PowerTransformer"/>
  <cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:1"/>
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>
```

- Razlika u elementima
 - range – referencira tip podatka koji je **enumeracija** (za razliku od prostih/izvedenih tipova ne koristi se dataType element)
- Obratiti pažnju na
 - stereotype – <http://langdale.com.au/2005/UML#ftn>

Deklaracija atributa u RDFS formatu 5/5

- Primer atributa tipa asocijacije (referenca)

```
<rdf:Description rdf:about="#TransformerWinding.PowerTransformer">
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#byreference"/>
  <cims:stereotype rdf:resource="http://langdale.com.au/2005/UML#ofAggregate"/>
  <rdfs:comment>A transformer has windings</rdfs:comment>
  <rdfs:label>PowerTransformer</rdfs:label>
  <rdfs:range rdf:resource="#PowerTransformer"/>
  <rdfs:domain rdf:resource="#TransformerWinding"/>
  <cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:1"/>
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>
```

- Razlika u elementima

- range – referencira tip podatka koji je **klasa** (za razliku od prostih/izvedenih tipova ne koristi se dataType element)
- stereotype – iskorišćen za dodatni opis resursa:
 - <http://langdale.com.au/2005/UML#byreference>
 - <http://langdale.com.au/2005/UML#ofAggregate>

Parser CIM profila datog u RDFS formatu

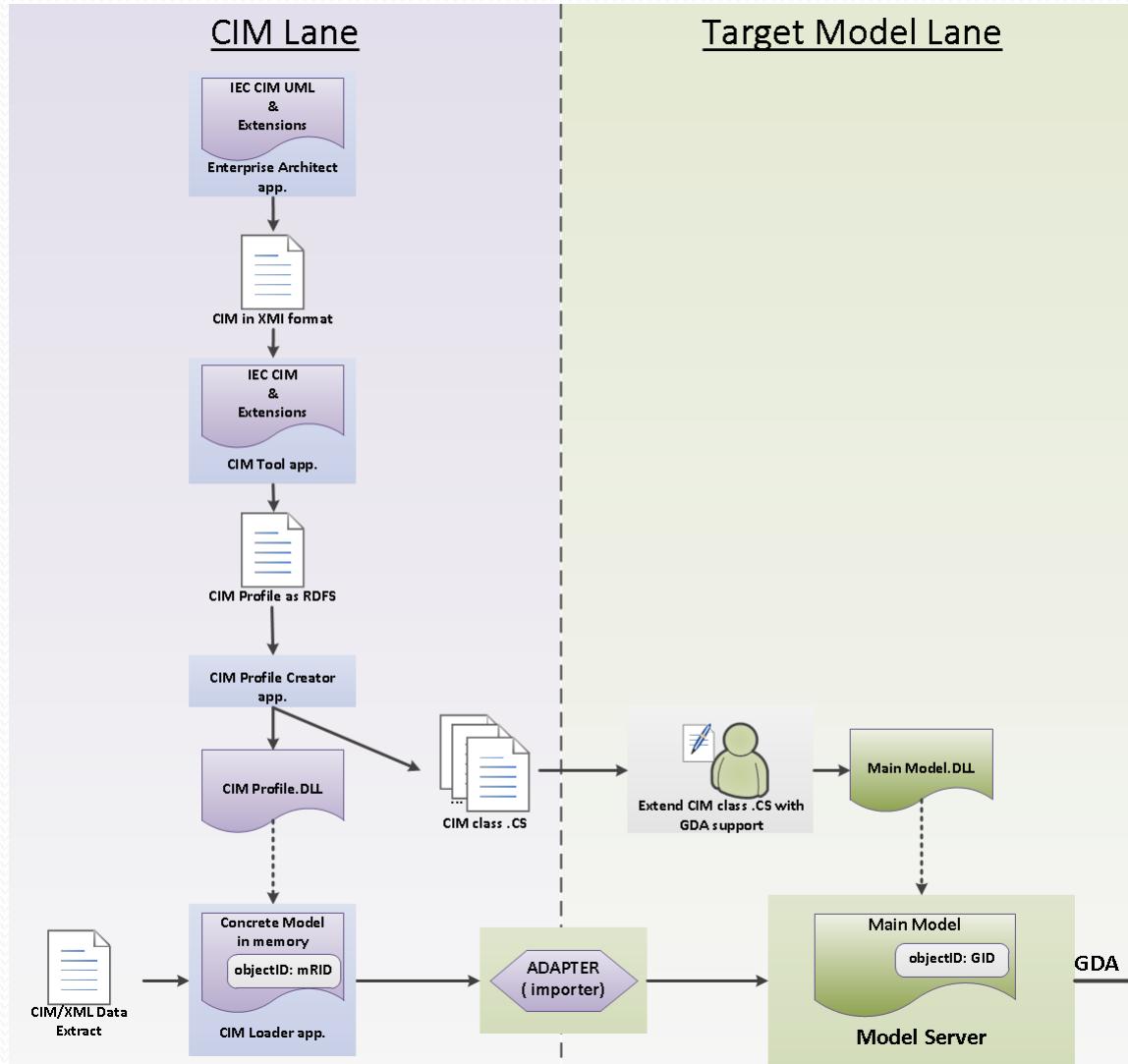
- Postavka zadatka
 - učitati RDFS dokument (koji definiše CIM profil) u odgovarajući objektni model u memoriji;
 - odštampati sadržaj objektnog modela u tekstualni izveštaj (paketi, klase u paketu, atributi u klasi).
- Implementacija
 - C#
 - Bazirana na klasi: *System.Xml.XmlReader* ([doc link](#))

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 4:

Parser CIM profila datog u RDFS formatu

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Pregled RDFS formata zapisa CIM profila

The screenshot shows the CIMTool interface running on the Eclipse Platform. The window title is "CIMTool - SIMESLabs/Profiles/PowerTransformer.legacy-rdfs - Eclipse Platform".

- Project Explorer:** Shows the project structure under "SIMESLabs". The "Profiles" folder contains "LabsProfile.html", "LabsProfile.legacy-rdfs", "LabsProfile.owl", "PowerTransformer.html", "PowerTransformer.legacy-rdfs", and "PowerTransformer.owl".
- PowerTransformer.owl Editor:** Displays the XML code for the PowerTransformer profile. The code defines the "TransformerWinding.windingType" property, which is a range of "WindingType" and has a domain of "TransformerWinding". It also defines the "Package_Wires" class, which belongs to the "IEC61970" category and is an extension of the "Core and Topology" package.
- Project Model View:** Shows the UML model elements. Under "Package: IEC61970_IEC61968_combine", there are stereotypes for Class, Package, and Stereotype. Under "Datatype", there are entries for EnergyAsMWh, EnumeratedType, FlowgateAfcUseCode, FlowgateIdcType, FreqBiasFactor, PenaltyFactor, PotheadType, and PowerROCPERMin.
- Bottom Bar:** Contains tabs for "Documentation" and "Properties", with "PowerTransformerProfile" selected.

Parsiranje XML-a

- DOM (Document Object Model) parser
 - Učitava se ceo XML dokument od jednom
 - .NET - XmlDocument
 - Može nastati problem ukoliko su u pitanju veliki XML dokumenti
- SAX (Simple API for XML) parser
 - Stream parser - Čita se element po element
 - Event driven – Kada se pročita predodređeni element XML-a, pozivaju se odgovarajuće metode za obradu
 - .NET - XmlReader
 - Obrada velikih XML dokumenata ne predstavlja problem

SAX parser 1/3

- Obrada XML dokumenta
 - Čita se element po element XML dokumenta
 - Od interesa su najčešće sledeći elementi: početak tag-a, atributi definisani u tagu, vrednost(tekst) navedena u tag-u, kraj tag-a
 - U zavisnosti od tipa elementa pozivaju se određene metode za obradu.
- Primer 1: *<item type=“typeValue”> text</item>*
 - *<item>* - početak tag-a
 - *type* – atribut definisan za tag *item*
 - “*typeValue*” – vrednost atributa *type*
 - *characters* – vrednost (tekst) navedena u tag-u
 - *</item>* - kraj tag-a

SAX parser 2/3

- Primer 2: <*item type="typeValue"*>
 - Empty tag
 - Sadrži samo atribut, nema vrednosti (tekst)
- Metode za obradu XML dokumenta
 - StartDocument(...) - Poziva se pre početka parsiranja. U ovoj metodi se uglavnom rade inicijalizacije potrebnih prmenljivih.
 - StartElement(...) – Poziva se kada se pročita početak tag-a.
 - EndElement(...) – Poziva se kada se pročita kraj tag-a.
 - Characters(...) – Poziva se kada se pročita vrednost (tekst) navedena unutar tag-a
 - EndDocument() - Poziva se na kraju parsiranja. Uglavnom se oslobođaju zauzeti resursi.

SAX parser 3/3

- Primer jednostavnog SAX paresera: pročitati zaposlene opisane u XML dokumetu i dodati ih u kolekciju svih zaposlenih

```
<?xml version="1.0"?>
<Personnel>
    <Employee typeOfEmployment="permanent">
        <Name>Marko</Name>
    </Employee>

    <Employee typeOfEmployment="contract">
        <Name>Nikola</Name>
    </Employee>
</Personnel>
```

- Algoritam
 - Čitati element po element
 - Za svaki otvoren tag *Employee* programski kreirati objekat tipa *Employee*
 - Pročitati sve atribute dodeljene tagu *Employee* i programski ih dodeliti odgovarajućim property-ima objekta *Employee*
 - Pročitati sve podtagove taga *Employee* i njihove vrednosti (tekst) programski dodeliti odgovarajućim property-ima objekta *Employee*
 - Za svaki zatvoren tag *Employee* programski dodati *Employee* objekat u kolekciju zaposlenih
 - Implementacija u projektu *SaxParserEmployee*

RDFS SAX parser 1/2

- Koristimo ga radi programske manipulacije definisanog CIM profila
 - Rezultat parsiranja predstavlja objekat tipa *Profile* koji u sebi sadrži sve informacije potrebne za opis CIM profila definisanog RDFS-om
- Implementacija
 - StartDocument() – Inicijalizacija svih potrebnih promenljivih za kreiranje
 - StartElement()
 - Pre poziva metode prikupljaju se svi atributi odgovarajućeg resursa
 - Poziva se metoda i prikupljaju podaci potrebni za kreiranje objekata koji opisuju resurse
 - Ukoliko je u pitanju prazan element `<item/>` ne očekujemo da cemo pročitati zatvarajući tag za ovaj element, pa se nakon *StartElement()* metode automatski poziva *EndElement()* metoda.

RDFS SAX parser 2/3

- Implementacija...

- EndElement() – Na osnovu prkupljenih informacija u metodi *StartElement()* kreiraju se objekti koji opisuju RDFS resurse i dodeljuju profilu.
- Obrada je podeljena na nekoliko celina:
 - Prvi deo obrađuje resurse definisane RDFS-om. Određen je delom koda ograničenog uslovom: if (*qName.Equals(rdfProfileElement)*) || ...
 - Ova celina je dalje izdeljena na obradu resursa različitog tipa:
 - Paket: if (*string.Compare(ExtractSimpleNameFromResourceURI(type), "ClassCategory") == o*)
 - Klasa: else if (*string.Compare(ExtractSimpleNameFromResourceURI(type), "Class") == o*)
 - Atribut: else if (*string.Compare(ExtractSimpleNameFromResourceURI(type), "Property") == o*)
 - Ostatak metode obrađuje podatke vezane za
 - Labele: else if (*qName.Equals(rdfsLabel)*)
 - Komntare: else if (*qName.Equals(rdfsComment)*)
 - itd.

RDFS SAX parser 3/3

- Implementacija...
 - Characters() – Prikuplja definisane vrednosti (tekst)
 - EndDocument()-Oslobađe resurse i vrši dodatno procesiranje objekta tipa *Profile* ukoliko je potrebno

Parser CIM profila datog u RDFS formatu

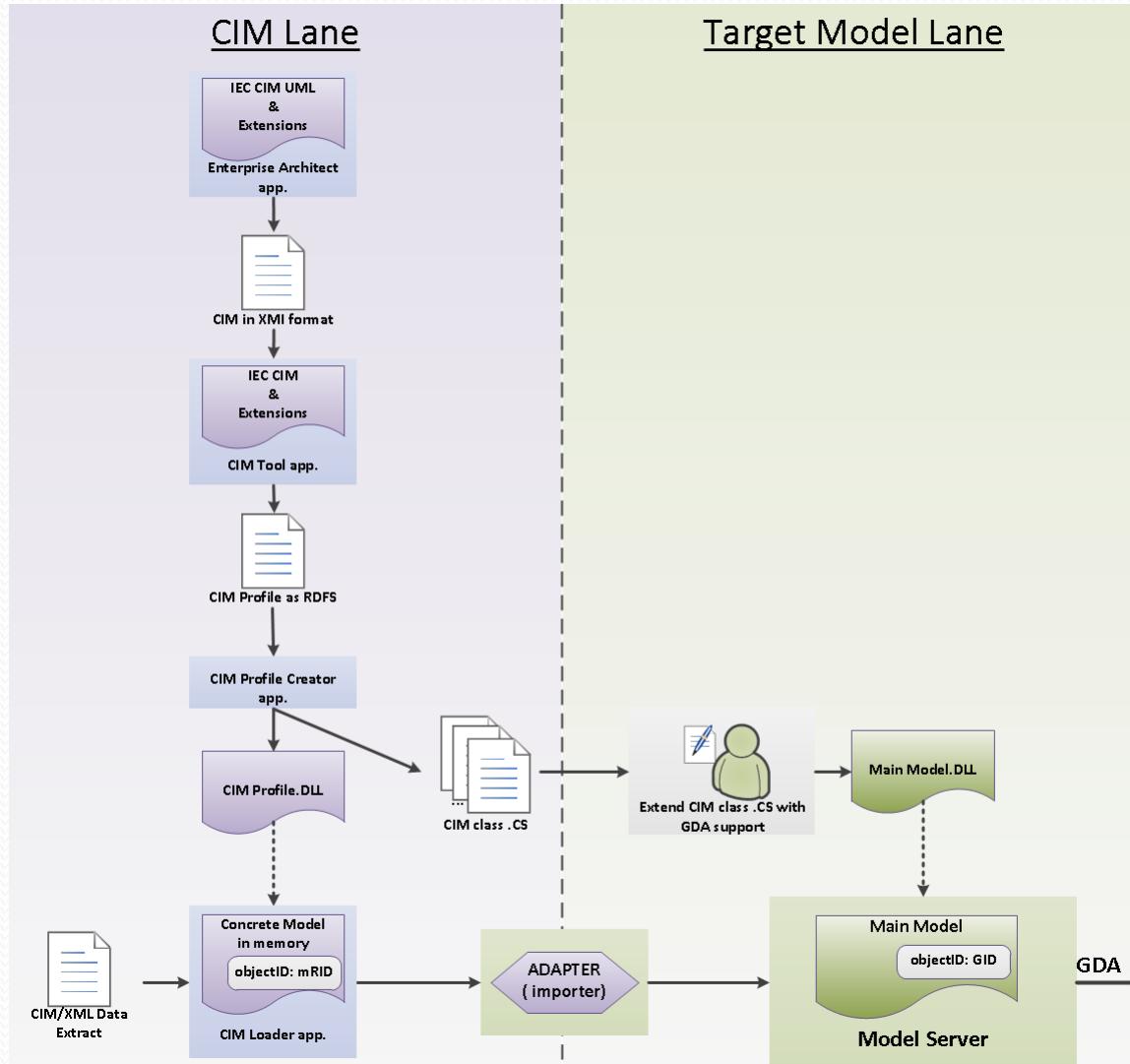
- Opis pripremljenog rešenja
 - učitati RDFS dokument (koji definiše CIM profil) u odgovarajući objektni model u memoriji;
 - odštampati sadržaj objektnog modela u tekstualni izveštaj (paketi, klase u paketu, atributi u klasi).
- Zadaci
 - U jedinoj formi dodati TextBox u kome će se ispisati
 - Broj klasa definisanih u RDFSu
 - Ukupan broj atributa definisanih u RDFSu
 - Broj komentara definisanih u RDFSu
 - Demonstrirati rad sa conditional breakpoint-ima
 - Prikazati mogućnosti optimizacije datog koda

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 5:

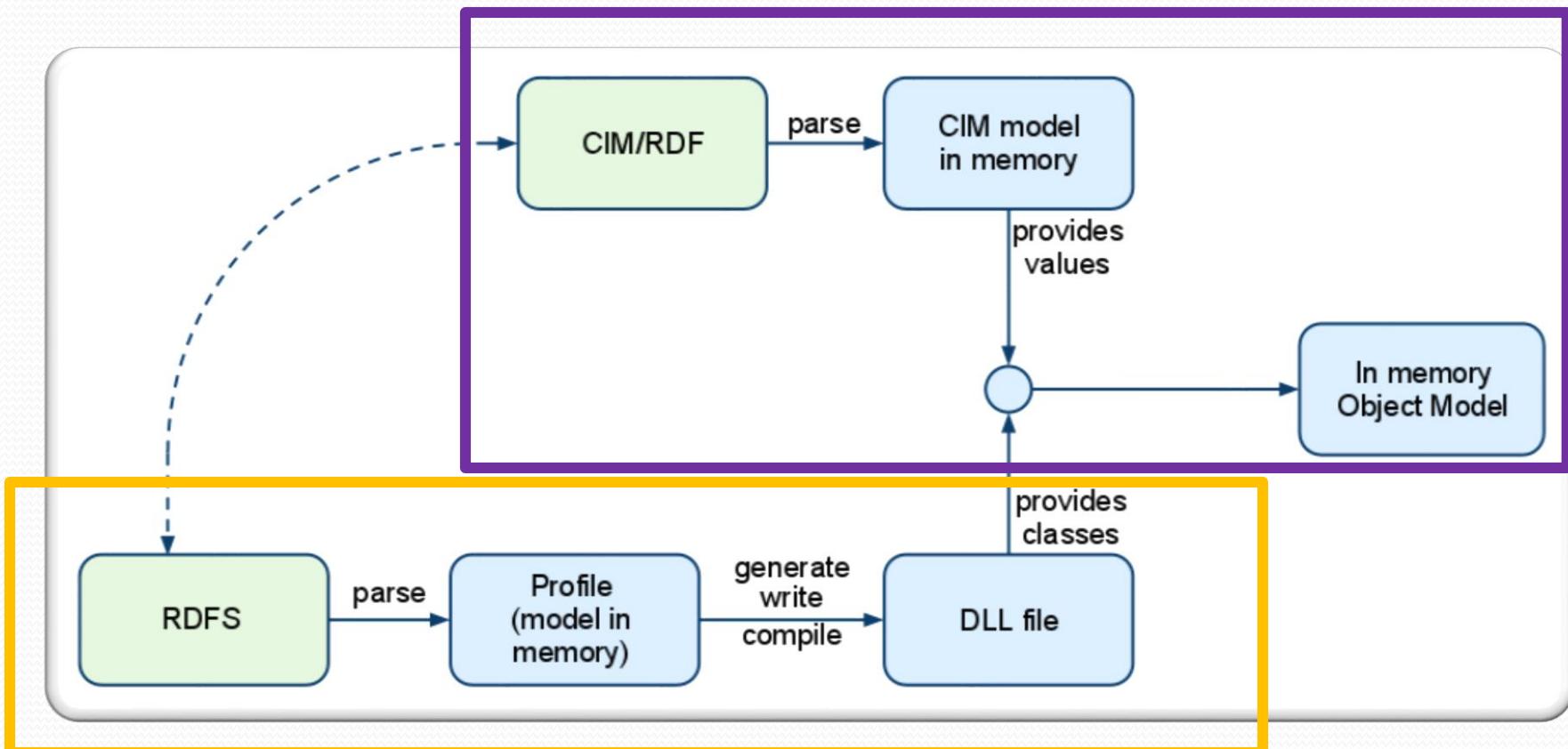
Generator klasa definisanih CIM profilom

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Pregled procesa instanciranja modela elektroenergetske mreže datog u CIM formatu

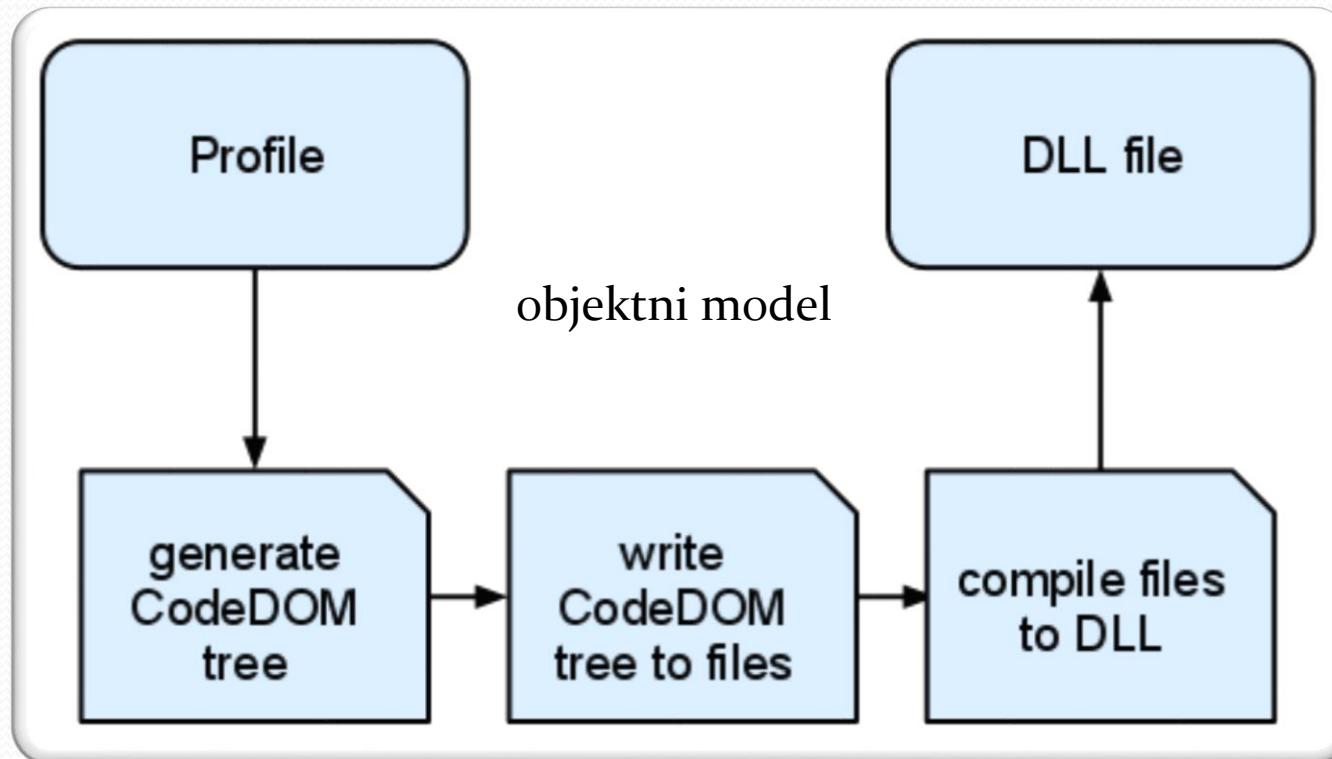
- Generator klasa
- Model loader



Generator klasa definisanih CIM profilom 1/3

- CodeDOM:
 - [http://msdn.microsoft.com/en-us/library/y2k85ax6\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/y2k85ax6(v=vs.110).aspx)
 - Omogućava implementaciju programa koji generiše *source code*.
 - Osnova je graf CodeDOM elemenata.
 - Koristi se i za *compile source code-a u binarni assembly (.DLL)*

Generator klasa definisanih CIM profilom 2/3

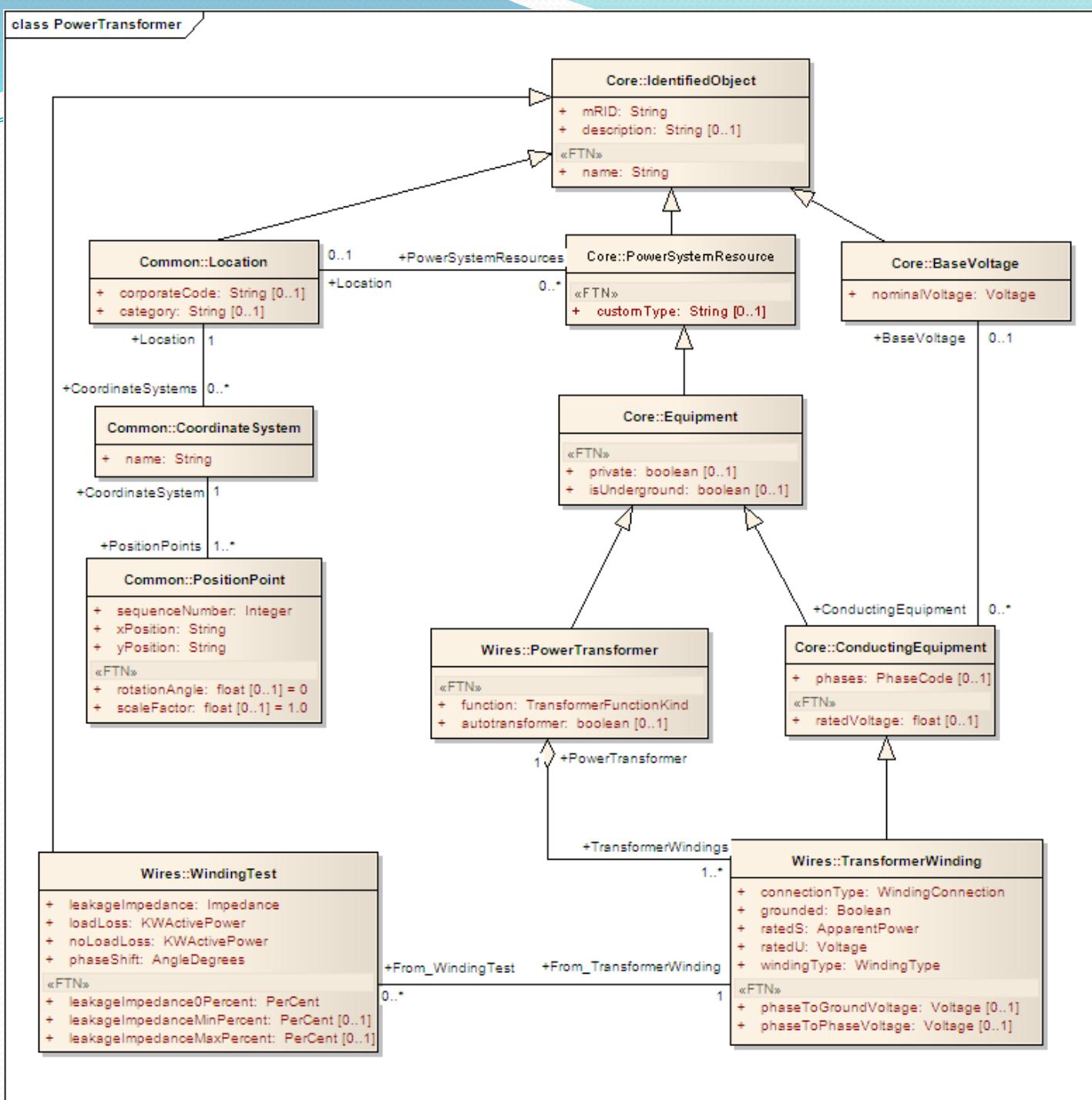


Generator klasa definisanih CIM profilom 3/3

- Opis pripremljenog rešenja
 - učitati RDFS dokument (koji definiše CIM profil) u odgovarajući objektni model u memoriji;
 - odštampati sadržaj objektnog modela u tekstualni izveštaj (paketi, klase u paketu, atributi u klasi);
 - izgenerisati .DLL na osnovu objektnog modela CIM profila.

Zadaci

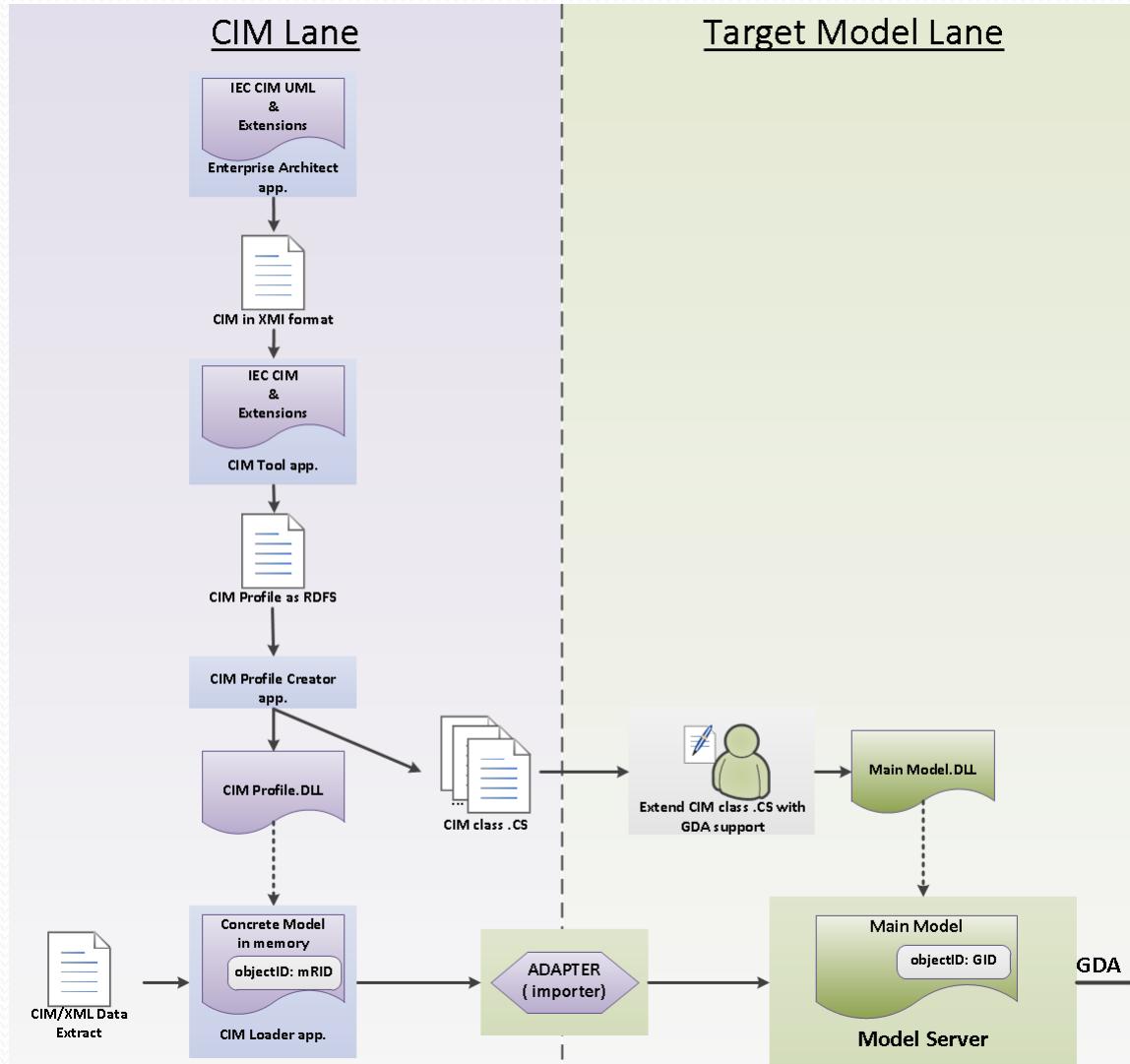
1. Kreirati .DLL na osnovu definisanog .RDFS-a
2. Kreirati novi projekat “LabModelLoader” i uključiti kreirani .DLL u njega.
3. Kreirati međusobno povezane instance definisanih *concrete* klasa.
4. Kreirati model koji sadrži 20 različitih instanci transformatora sa pripadajućim elementima.
5. Odštampati .TXT file sa izveštajem sadržaja kreiranog modela, tako da se za svaki objekat minimalno prikaže:
 - vrednost *mRID* atributa
 - vrednost *name* atributa
 - vrednost dodatnog atributa (po izboru)



Standardi i modeliranje elektroenergetskih sistema

VEŽBA 6:
Model loader

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Model loader 1/2

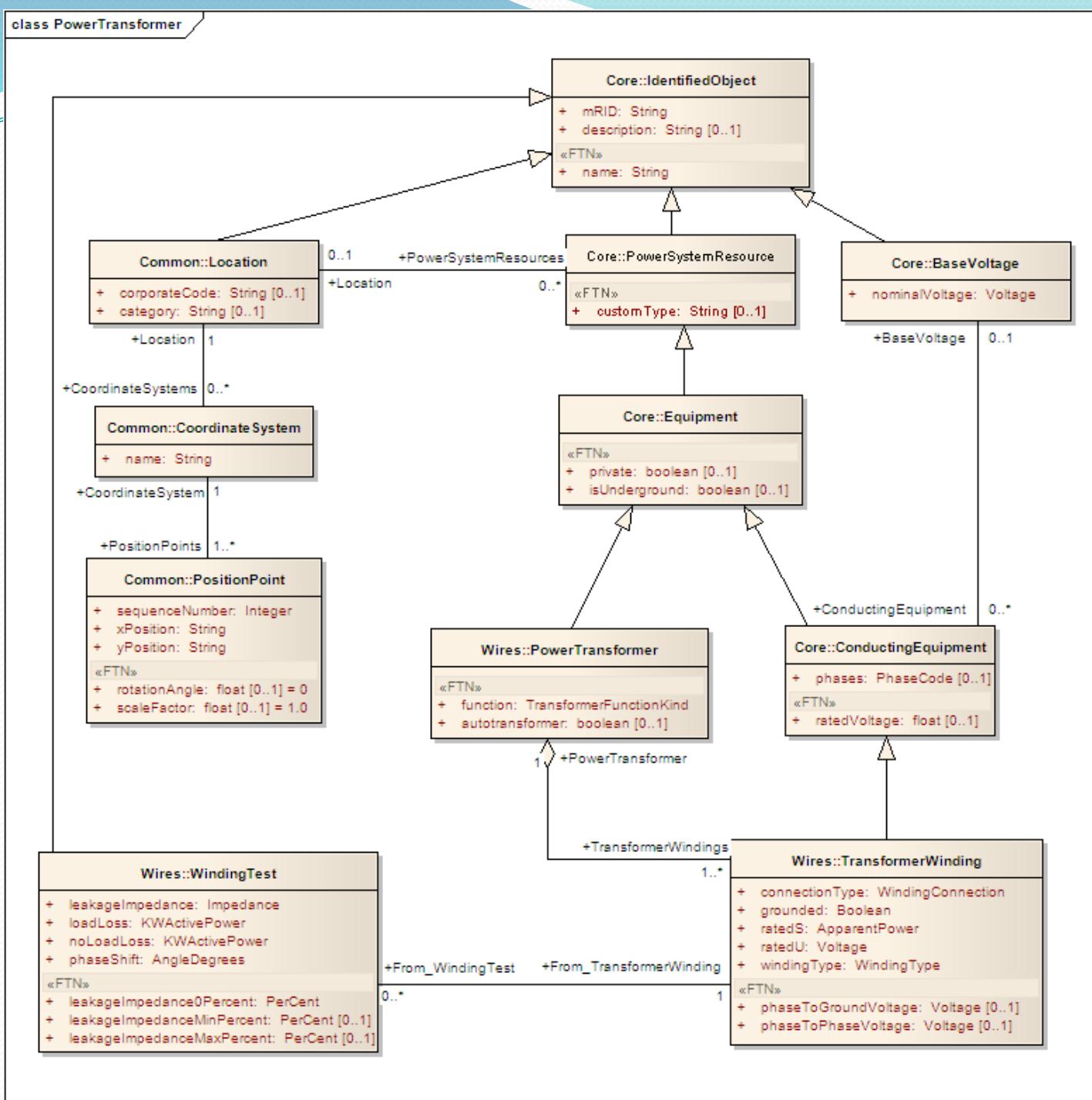
- ConcreteModel klasa
 - Model podataka koji se kreira na osnovu CIM/XML-a
 - Sadrži mapu svih entiteta
 - Obezbeđuje metode za upis i čitanje entiteta

Model loader 2/2

- ... **ConcreteModel** klasa
 - Organizacija entiteta po tipu
 - `SortedDictionary<string, SortedDictionary<string, object>> modelMap`
 - “Spolja” mapa kao ključ ima tip entiteta, dok je vrednost nova (“unutrašnja”) mapa koja sadrži entitete istog tipa
 - “Unutrašnja” mapa kao ključ ima mrID entiteta, dok je vrednost sam entitet

Zadaci

1. Dodati ispis broja entiteta svakog tipa
2. Ispisati vrednost atributa *LoadLoss* entiteta tipa *WindingTest* čiji je *mrID* jednak "939140759_TW1_WT".



Standardi i modeliranje elektroenergetskih sistema

VEŽBA 7:
Uvod u Network Model Servis

prof.: dr Milan Gavrić; dr Darko Čapko
asistenti: Stanislava Selena, MSc.; Nemanja Nedić, MSc.

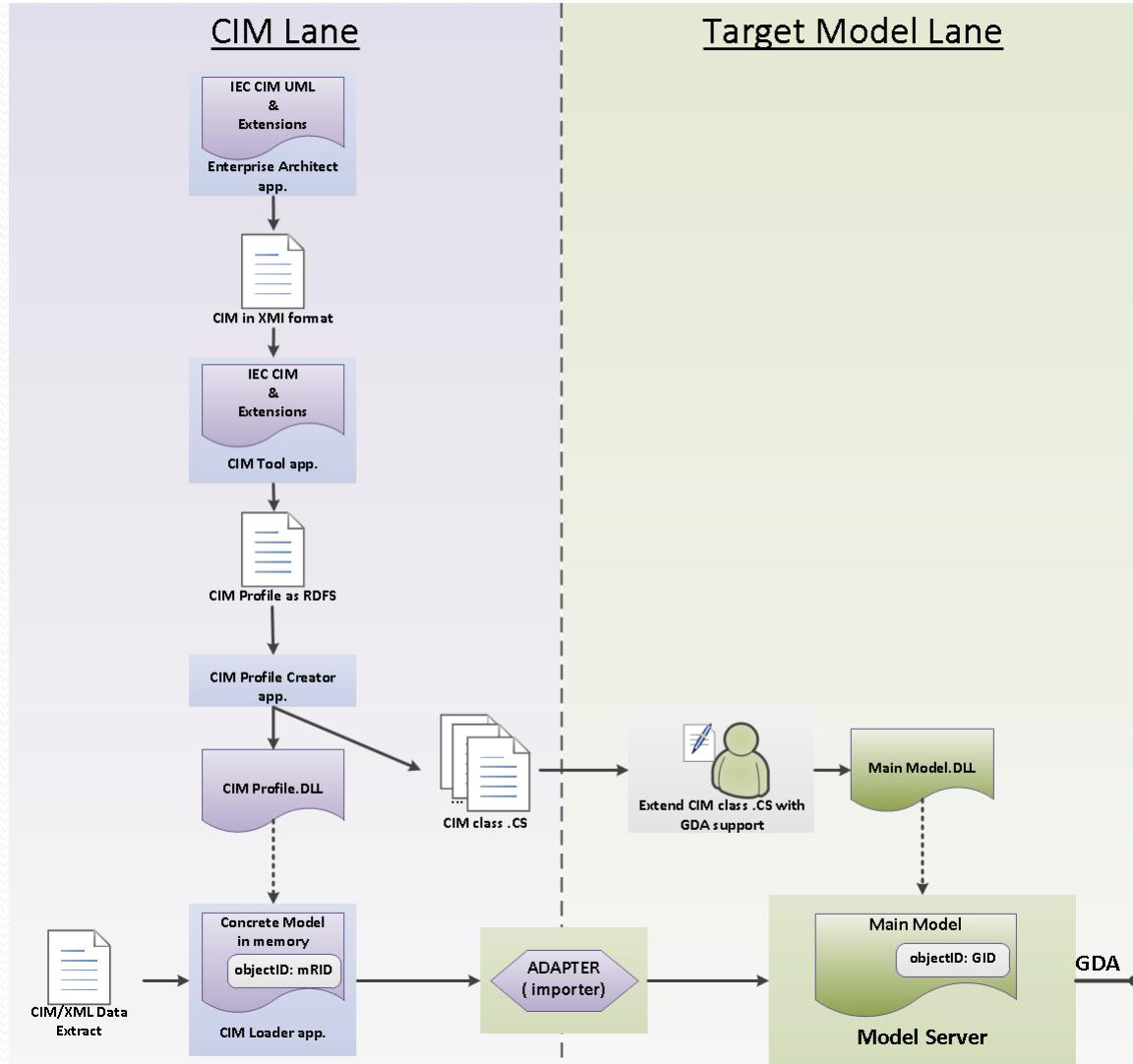
Kontakti:

- stanislava.selena@schneider-electric-dms.com
- nemanja.nedic@schneider-electric-dms.com

Konsultacije:

- zakazati putem e-maila

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Kreiranje Network Model servisa

- Model zasnovan na CIM-u
- Identifikacija objekata
- Referenciranje objekata

Implicitno definisanje konkretnih klasa 1/2

- *DMSType* enumeracija

```
public enum DMSType : short
{
    BASEVOLTAGE          = 0x0001,
    LOCATION              = 0x0002,
    POWERTR               = 0x0003,
    POWERTRWINDING        = 0x0004,
    WINDINGTEST           = 0x0005,
}
```

- 16-bitna enumeracija
- Svaka klasa čije seinstanciranje očekuje dobija odgovarajuću vrednost u *DMSType* enumeraciji

Implicitno definisanje konkretnih klasa 2/2

- ... *DMSType* enumeracija
 - *BaseVoltage* - *DMSType.BASEVOLTAGE*
 - *Location* - *DMSType.LOCATION*
 - *PowerTransformer* - *DMSType.POWERTR*
 - *TransformerWinding* - *DMSType.POWERTRWINDING*
 - *WindingTest* - *DMSType.WINDINGTEST*

Implicitno definisanje hijararhije nasleđivanja 1/2

- ...*ModelCode* enumeracija

- 64-bitna enumeracija

Nasleđivanje	DMSType	Opis atributa
32 bita	16 bita	16 bita

- Svaki tip resursa u modelu se jednoznačno identificuje odgovarajućim *ModelCode*-om:

- Svaka klasa (apstraktna ili konkretna) dobija svoj *ModelCode*
PowerSystemResource - *ModelCode.PSR*
PowerTransformer - *ModelCode.POWERTR*
 - Svaki atribut (svake klase) dobija svoj *ModelCode*
PowerTransformer.Function - *ModelCode.POWERTR_FUNC*
PowerSystemResource.CustomType - *ModelCode.PSR_CUSTOMTYPE*

Implicitno definisanje hijararhije nasleđivanja 2/2

- *ModelCode* enumeracija

```
public enum ModelCode : long
{
    IDOBJ                         = 0x1000000000000000,
    IDOBJ_GID                      = 0x1000000000000104,
    IDOBJ_DESCRIPTION              = 0x1000000000000207,
    IDOBJ_MRID                     = 0x1000000000000307,
    IDOBJ_NAME                     = 0x1000000000000407,

    PSR                           = 0x1100000000000000,
    PSR_CUSTOMTYPE                = 0x1100000000000107,
    PSR_LOCATION                   = 0x1100000000000209,

    ...
    POWERTRWINDING_PHASETOGRNDVOLTAGE = 0x1111000000040605,
    POWERTRWINDING_PHASETOPHASEVOLTAGE = 0x1111000000040705,
    POWERTRWINDING_POWERTRW          = 0x1111000000040809,
    POWERTRWINDING_TESTS            = 0x1111000000040919,
}
```

ModelCode semantika 1/8

- Vrednost ModelCode-a nosi u sebi niz informacija vezanih za resurs
- *ModelCode* enumeracija – nasleđivanje
 - Najviših 32 bita opisuje nasleđivanje.
 - “Child” klasa UVEK ima jednu cifru više od “parent” klase
 - Ukoliko više klasa nasleđuje istog parent-a, dodatna cifra se uveća za svaku “child” klasu.

ModelCode.IDOBJ = 0x**10000000**oooooooo;

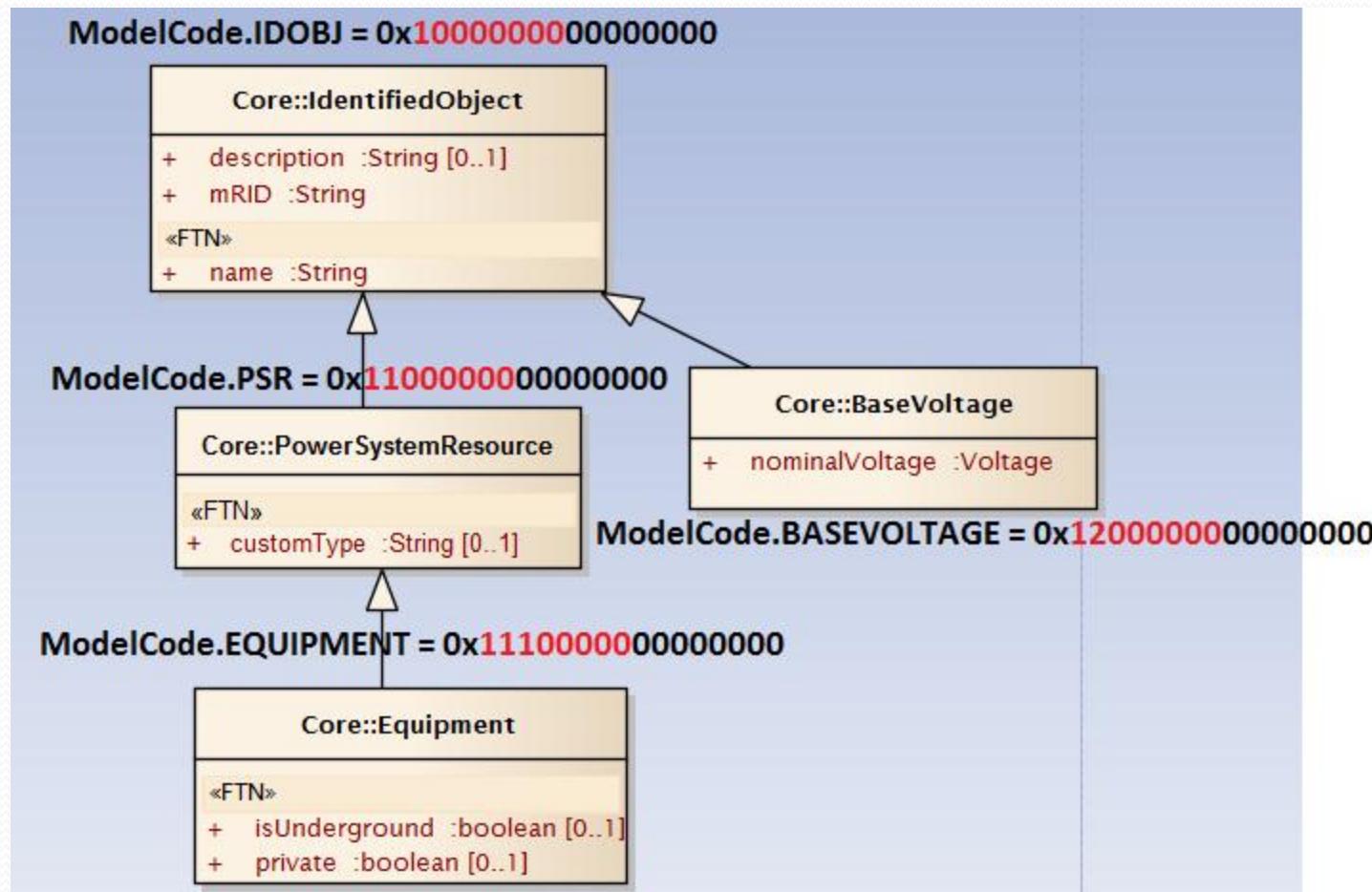
ModelCode.PSR = 0x**11000000**oooooooo;

ModelCode.BASEVOLTAGE = 0x**12000000**00010000;

ModelCode.EQUIPMENT = 0x**11100000**oooooooo;

ModelCode semantika 2/8

- ...ModelCode enumeracija – nasleđivanje



ModelCode semantika 3/8

- *ModelCode* enumeracija – apstraktni i konkretni tipovi
 - Narednih 16 bita opisuje da li je klasa apstraktna ili ne.
 - Ukoliko je klasa apstraktna vrednost je postavljena na nulu, u suprotnom vrednost odgovara vrednosti *DMType*-a za tu konkretnu klasu.

ModelCode.IDOBJ = 0x10000000**0000**0000;

ModelCode.PSR = 0x11000000**0000**0000;

ModelCode.BASEVOLTAGE = 0x12000000**0001**0000;

DMSType.BASEVOLTAGE = 0x**0001**;

ModelCode.LOCATION = 0x12000000**0002**0000;

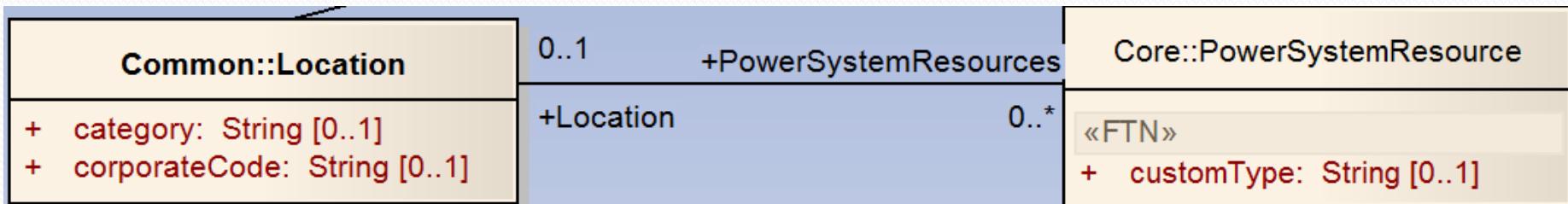
DMSType.LOCATION = 0x**0002**;

ModelCode semantika 4/8

- *ModelCode* enumeracija – atributi
 - Svaki atribut neke klase (apstraktne ili konkretne) ima svoj *ModelCode IdentifiedObject.Description* - *ModelCode.IDOBJ_DESCRIPTION*
BaseVoltage.NominalVoltage - *ModelCode.BASEVOLTAGE_NOMINALVOLTAGE*
 - Veći deo vrednosti *ModelCode*-a koji je dodeljen atributu (viših 48 bita) je identično *ModelCode*-u koji je dodeljen klasi kojoj atribut pripada
ModelCode.IDOBJ_DESCRIPTION = 0x**1000000000000207**
ModelCode.IDOBJ = 0x**1000000000000000**
 - Najnižih 16 bita rezervisani su za opis atributa
ModelCode.IDOBJ_DESCRIPTION = 0x100000000000**0207**
ModelCode.BASEVOLTAGE_NOMINALVOLTAGE = 0x100000000001**0105**
 - Ukoliko je *ModelCode* dodeljen klasi najnižih 16 bita su uvek nula
ModelCode.IDOBJ = 0x100000000000**0000**
ModelCode.BASEVOLTAGE = 0x120000000001**0000**

ModelCode semantika 5/8

- ...*ModelCode* enumeracija – atributi
 - Viših 8 bita označavaju redni broj atributa u klasi



ModelCode.PSR = 0x1100000000000000

ModelCode.PSR_CUSTOMTYPE = 0x110000000000**01**07

ModelCode.PSR_LOCATION = 0x110000000000**02**09

...

ModelCode.LOCATION = 0x1300000000020000

ModelCode.LOCATION_COORPORATECODE = 0x130000000002**01**07

ModelCode.LOCATION_CATEGORY = 0x130000000002**02**07

ModelCode.LOCATION_PSRS = 0x130000000002**03**19

ModelCode semantika 6/8

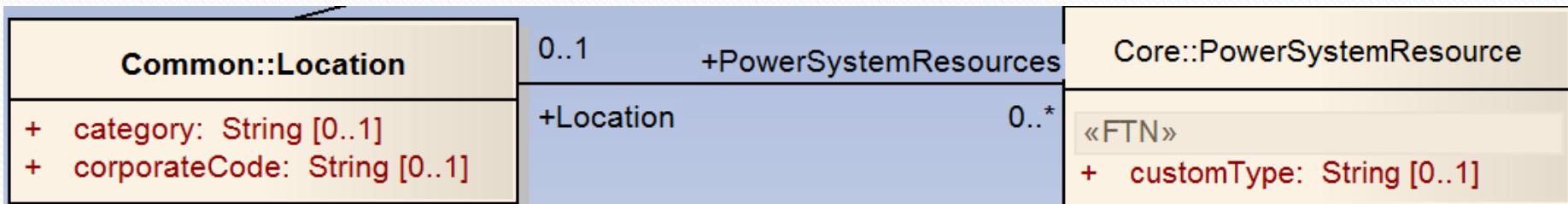
- ...ModelCode enumeracija – atributi

- Nižih 8 bita označavaju tip atributa u klasi
- Ukoliko je u pitanju prost tip: prva vrednost je nula, dok druga vrednost označava koji je prost tip u pitanju (*bool, int, long, string*)
- Ukoliko je u pitanju lista prostih tipova: prva cifra je uvek jedinica, a druga označava koje proste vrednosti sadrži lista.

```
public enum PropertyType : short
{
    Empty = 0,
    Bool      = 0x01,
    Byte      = 0x02,
    Int32     = 0x03,
    Int64     = 0x04,
    Float     = 0x05,
    Double    = 0x06,
    String    = 0x07,
    DateTime  = 0x08,
    Reference = 0x09,
    Enum      = 0x0a,
    TimeSpan  = 0x0c,
    BoolVector = 0x11,
    ByteVector = 0x12,
    Int32Vector = 0x13,
    Int64Vector = 0x14,
    FloatVector = 0x15,
    DoubleVector = 0x16,
    StringVector = 0x17,
    DateTimeVector = 0x18,
    ReferenceVector = 0x19,
    EnumVector = 0x1a,
    TimeSpanVector = 0x1c,
}
```

ModelCode semantika 7/8

- ...ModelCode enumeracija –atributi
 - Nižih 8 bita označavaju tip atributa u klasi (int, long, string, List<int>...)



ModelCode.PSR

= 0x1100000000000000

ModelCode.PSR_CUSTOMTYPE

= 0x11000000000001**07**

ModelCode.PSR_LOCATION

= 0x11000000000002**09**

...

ModelCode.LOCATION

= 0x1300000000020000

ModelCode.LOCATION_COORPOATECODE

= 0x13000000000201**07**

ModelCode.LOCATION_CATEGORY

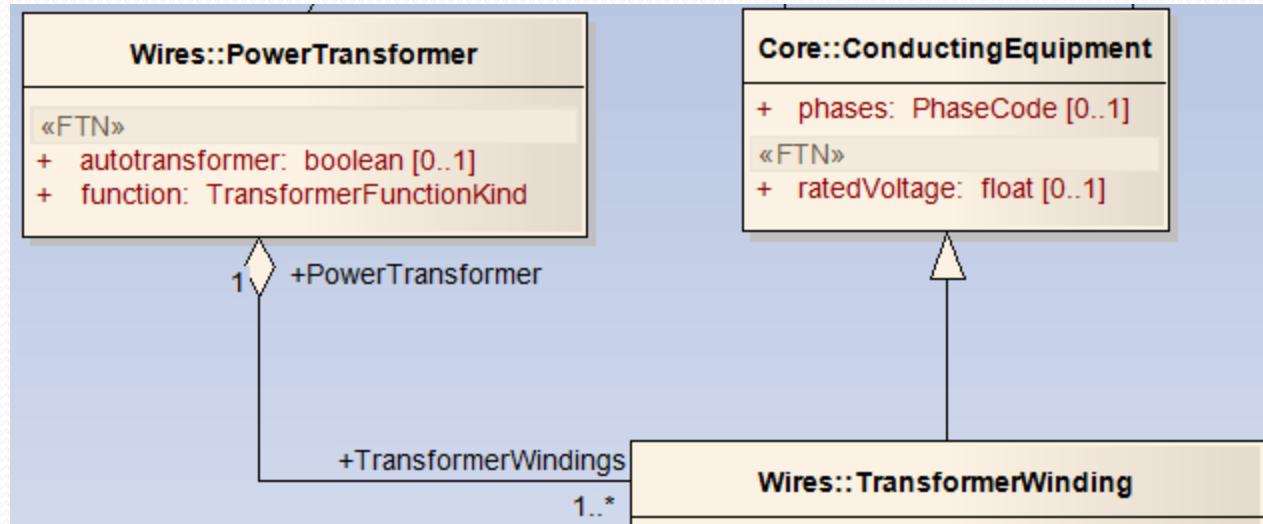
= 0x13000000000202**07**

ModelCode.LOCATION_PSRS

= 0x13000000000203**19**

ModelCode semantika 8/8

- ...*ModelCode* enumeracija – atributi



ModelCode. POWERTR

= 0X11120000000030000

ModelCode. POWERTR_FUNC

= 0X111200000000301**0a**

ModelCode. POWERTR_AUTO

= 0X111200000000302**01**

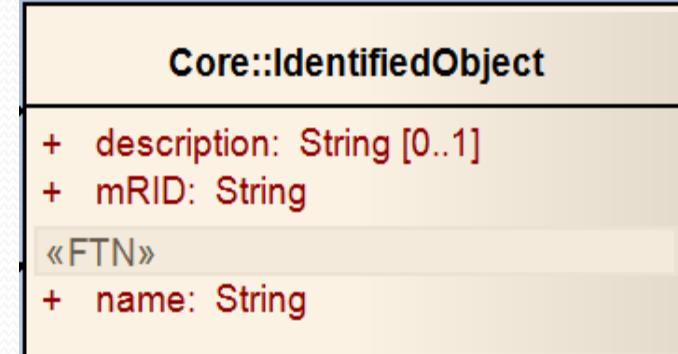
ModelCode. POWERTR_WINDINGS

= 0X111200000000303**19**

Globalni identifikator (GID) 1/2

- Globalni identifikator

- Iako je mRID jedinstven za svaki entitet koji se kreira, njegova upotreba za identifikaciju entiteta može da uspori rad servisa jer je u pitanju string
- Uvodi se novi jedinstveni generator tipa long (poređenje brojeva je daleko brža od poređenja string-ova)
- Globalni identifikator je definisan u klasi *IdentifiedObject* koja se nalazi u hijerarhiji nasleđivanja svih ostalih klasa. Iz tog razloga svaki entitet sadrži globalni identifikator



ModelCode.IDOBJ

ModelCode. IDOBJ_GID

ModelCode. IDOBJ_DESCRIPTION

ModelCode. IDOBJ_MRID

ModelCode. IDOBJ_NAME

= 0x1000000000000000

= **0x100000000000104**

= 0x100000000000207

= 0x100000000000307

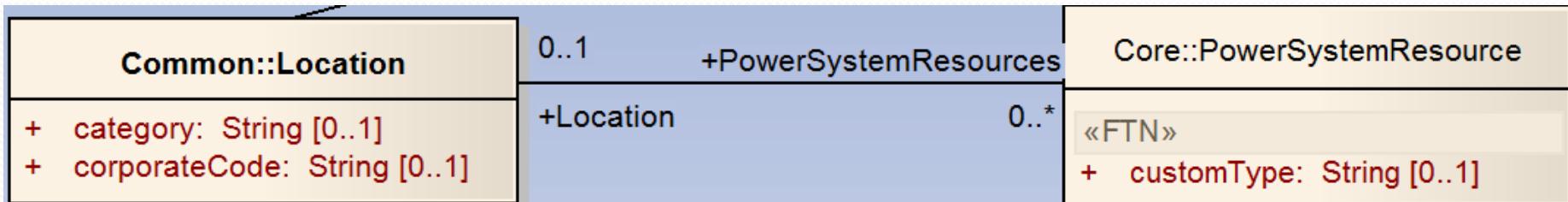
= 0x100000000000407

Globalni identifikator (GID) 2/2

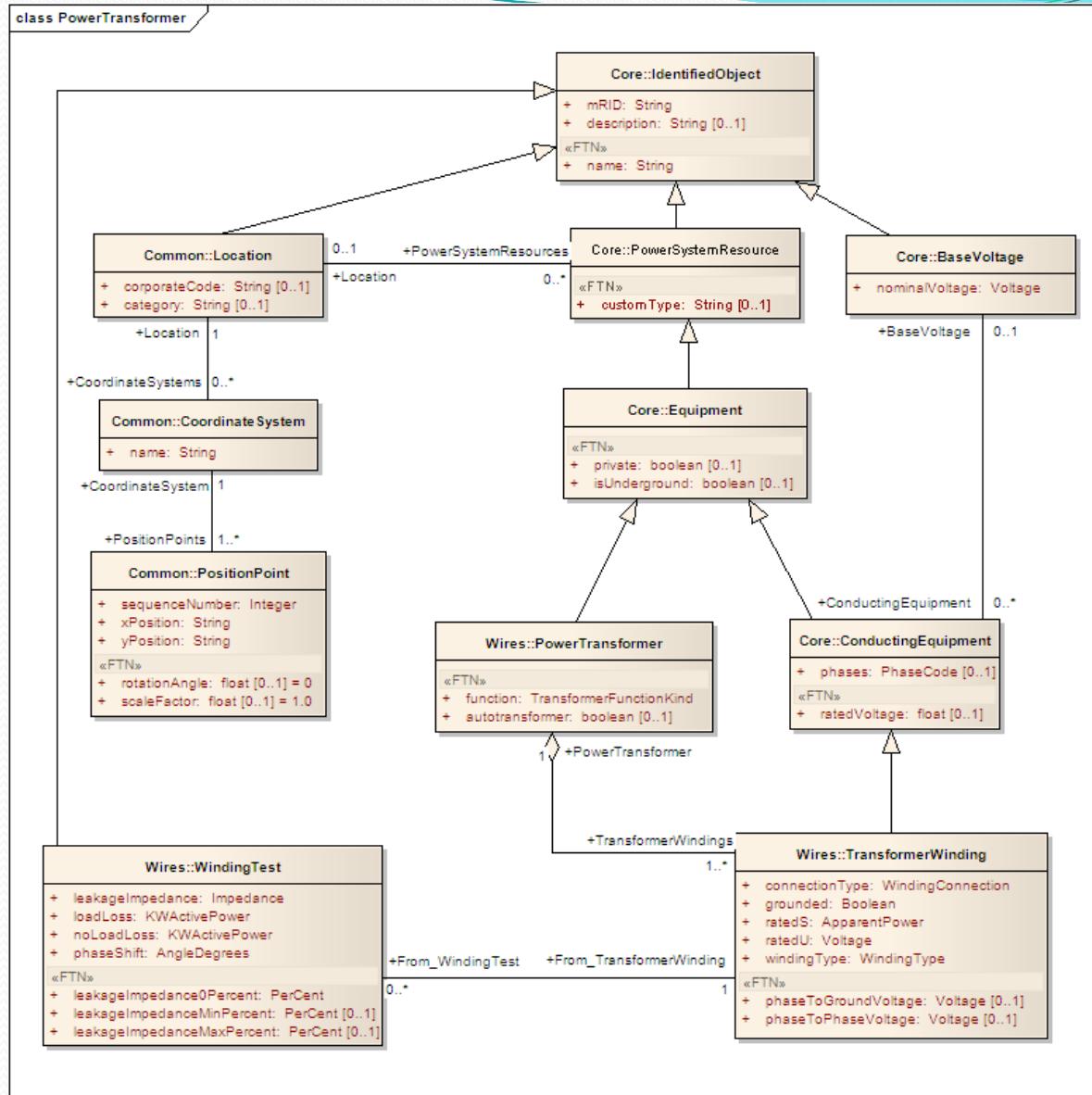
- Globalni identifikator
 - Globalni identifikator je 64 bitni
 - Kreira na osnovu tri podatka
 - 16 bitni *Sistem id* – za nas je uvek nula, jer koristimo jedan sistem
 - *DMSType* – odgovara tipu entiteta za koji se kreira globalni identifikator
 - *Brojač* – za svaki tip entitata postoji odgovarajući brojač koji obezbeđuje jedinstvenost globalnog identifikatora po tipu.



Reference – veze između entiteta



- Reference se modeluju globalnim identifikatorom
 - Za razliku od modela u *loader*-u koji je za referencu imao “pravu referencu” na drugi entitet, na servisu koristimo globalne identifikatore
 - Entitet koja referencira neki drugi entitet u stvari sadrži globalni identifikator tog entiteta
- Implementaciona specifičnost: Reference su dvosmerne na servisu: Ukoliko jedan entitet sadrži globalni identifikator drugog entiteta, onda i drugi entitet sadrži identifikator prvog.



Zadaci

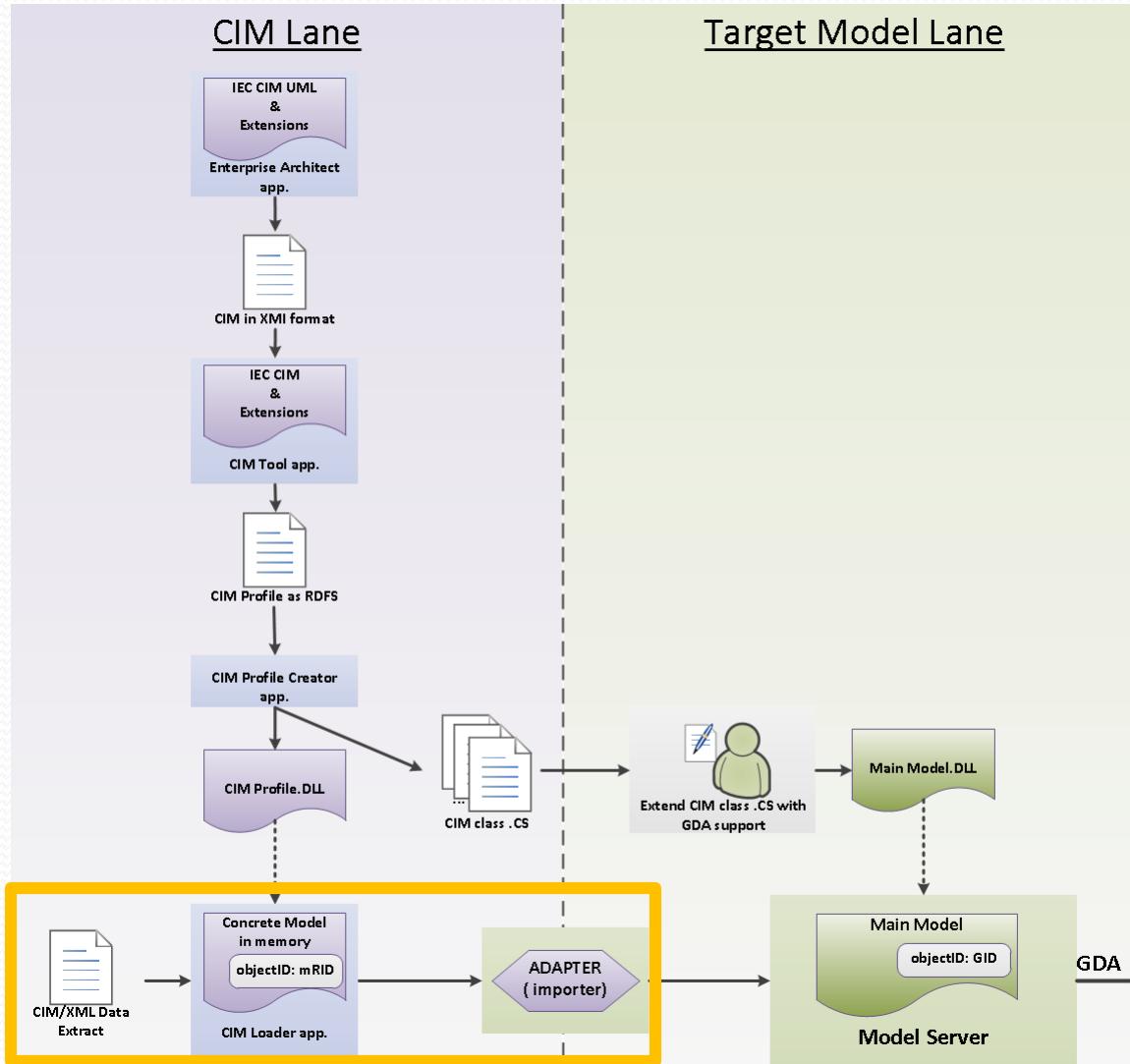
1. Kreirati *ModelCode*-ove za klase *TransformerWinding* i *WindingTest*

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 9:

Implementacija adaptera za import CIM-baziranog
modela podataka u Network Model Servis kroz GDA
interfejs

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Adapter 1/3

- Transformiše CIM-XML file u *Delta* objekat
- Primenuje *Delta* objekat na Network Model Servis kroz GDA interfejs

```
public Delta CreateDelta(Stream extract, SupportedProfiles extractType, out string log) ...  
public string ApplyUpdates(Delta delta) ...
```

Adapter 2/3

- Metoda **CreateDelta**:
 - Load proces:
 - extract i profil koristi za kreiranje *ConcreteModel* instance
 - Transform proces:
 - na osnovu profila poziva odgovarajući importer
 - importer transformiše sadržaj *ConcreteModel* objekta u *Delta* objekat

```
public Delta CreateDelta(Stream extract, SupportedProfiles extractType, out string log)
{
    Delta nmsDelta = null;
    ConcreteModel concreteModel = null;
    Assembly assembly = null;
    string loadLog = string.Empty;
    string transformLog = string.Empty;

    if (LoadModelFromExtractFile(extract, extractType, ref concreteModel, ref assembly, out loadLog))
    {
        TransformModel(assembly, concreteModel, extractType, out nmsDelta, out transformLog);
    }
    log = string.Concat("Load report:\r\n", loadLog, "\r\nTransform report:\r\n", transformLog);

    return nmsDelta;
}
```

Adapter 3/3

- Metoda **ApplyDelta**:

- koristi GDA interfejs ka Network Model Service-u
- pripremljen *Delta* objekat prosleđuje na primenu

```
public string ApplyUpdates(Delta delta)
{
    string updateResult = "Apply Updates Report:\r\n";
    System.Globalization.CultureInfo culture = Thread.CurrentThread.CurrentCulture;
    Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("en-US");
    if ((delta != null) && (delta.NumberOfOperations != 0))
    {
        //// TO BE ADDED: NetworkModelService->ApplyUpdates
        //// updateResult = MMSHandler.ApplyUpdates(electricDelta);
    }
    Thread.CurrentThread.CurrentCulture = culture;
    return updateResult;
}
```

Importer 1/7

- Importer je implementiran za određeni CIM profil.
- Ima znanje kako se elementi definisani u profilu mapiraju na objekte ciljanog DMS modela:
 - mapiranje CIM klasa na DMS klasu
 - mapiranje CIM atribut na DMS atribut
 - **Pažnja: mapiranje među modelima ne mora da bude 1-na-1!**
- Implementira proces koji transformiše sadržaj *ConcreteModel* objekta u *Delta* objekat

Importer 2/7

- Metoda **CreateNMSDelta**:

- poziva konverziju

```
public TransformAndLoadReport CreateNMSDelta(ConcreteModel cimConcreteModel)
{
    LogManager.Log("Importing PowerTransformer Elements...", LogLevel.Info);
    report = new TransformAndLoadReport();
    concreteModel = cimConcreteModel;
    delta.ClearDeltaOperations();

    if ((concreteModel != null) && (concreteModel.ModelMap != null))
    {
        try
        {
            // convert into DMS elements
            ConvertModelAndPopulateDelta();
        }
        catch (Exception ex)
        {
            string message = string.Format("{0} - ERROR in data import - {1}", DateTime.Now, ex.Message);
            LogManager.Log(message);
            report.Report.AppendLine(ex.Message);
            report.Success = false;
        }
    }
    LogManager.Log("Importing PowerTransformer Elements - END.", LogLevel.Info);
    return report;
}
```

Importer 3/7

- Konverzija modela:

- kreiranje *ResourceDescription* instanci na osnovu CIM objekata
- popunjavanje *Property*-a u okviru svakog *ResourceDescription* objekta
- vrednost referenci medju objektima: bitan je redosled kojim se generisu globalni identifikatori

```
private void ConvertModelAndPopulateDelta()
{
    LogManager.Log("Loading elements and creating delta...", LogLevel.Info);

    /// import all concrete model types (DMSType enum)
    ImportBaseVoltages();
    ImportLocations();
    ImportPowerTransformers();
    ImportTransformerWindings();
    ImportWindingTests();

    LogManager.Log("Loading elements and creating delta completed.", LogLevel.Info);
}
```

Importer 4/7

- **Import<Type>** metode:

- selektuju listu CIM objekata datog tipa iz *ConcreteModel*-a
- za svaki objekat iz liste (mape) kreira se *ResourceDescription* instanca i popunjava sa *Property* podacima

```
private void Import<Type>()
{
    SortedDictionary<string, object> cimObjects = concreteModel.GetAllObjectsOfType("FTN.Type");
    if (cimObjects != null)
    {
        foreach (KeyValuePair<string, object> cimObjectPair in cimObjects)
        {
            FTN.Type cimObject = cimObjectPair.Value as FTN.Type;
            ResourceDescription rd = CreateTypeResourceDescription(cimObject);
            if (rd != null)
            {
                delta.AddDeltaOperation(DeltaOpType.Insert, rd, true);
                report.Report.Append("Type ID = ").Append(cimObject.ID).Append(" SUCCESSFULLY
converted to GID = ").AppendLine(rd.Id.ToString());
            }
        }
    }
}
```

Importer 5/7

- Prilikom transformacije u *ResourceDescription* potrebno je:
 - generisati GID
 - upamtiti mapiranje CIM identifikatora na GID
- ImportHelper*

```
public class ImportHelper
{
    private Dictionary<DMSType, int> typeCounter = new Dictionary<DMSType, int>();
    private Dictionary<string, long> rdfIDtoGIDMapping = new Dictionary<string, long>();

    /// <summary> ...
    public int CheckOutIndexForDMSType(DMSType dmsType) ...

    /// <summary> ...
    public void DefineIDMapping(string rdfID, long gid) ...

    /// <summary> ...
    public long GetMappedGID(string rdfID) ...
}
```

Importer 6/7

- mapirati atributi CIM objekta na ModelCode
- ispravno postaviti vrednost svakog Property-ja

Converter

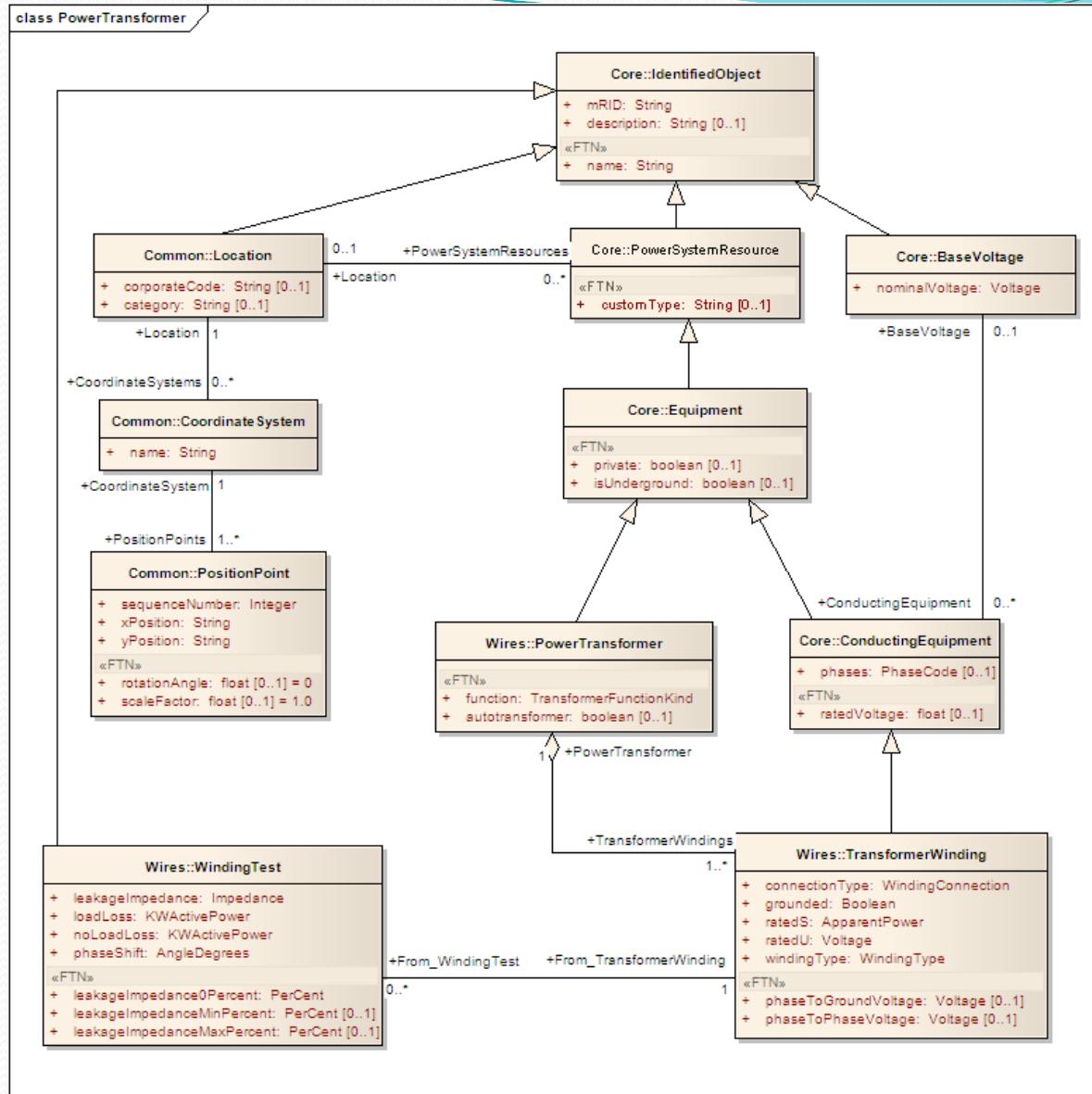
```
public static class PowerTransformerConverter
{
    #region Populate ResourceDescription
    public static void PopulateIdentifiedObjectProperties(FTN.IdentifiedObject cimIdentifiedObject, ResourceDescription rd)...
    public static void PopulateLocationProperties(FTN.Location cimLocation, ResourceDescription rd)...
    public static void PopulatePowerSystemResourceProperties(FTN.PowerSystemResource cimPowerSystemResource, ResourceDescription rd,
    public static void PopulateBaseVoltageProperties(FTN.BaseVoltage cimBaseVoltage, ResourceDescription rd)...
    public static void PopulateEquipmentProperties(FTN.Equipment cimEquipment, ResourceDescription rd, ImportHelper importHelper, Tra
    public static void PopulateConductingEquipmentProperties(FTN.ConductingEquipment cimConductingEquipment, ResourceDescription rd,
    public static void PopulatePowerTransformerProperties(FTN.PowerTransformer cimPowerTransformer, ResourceDescription rd, ImportHel
    public static void PopulateTransformerWindingProperties(FTN.TransformerWinding cimTransformerWinding, ResourceDescription rd, Imp
    public static void PopulateWindingTestProperties(FTN.WindingTest cimWindingTest, ResourceDescription rd, ImportHelper importHelpe
    #endregion Populate ResourceDescription

    #region Enums convert
    public static PhaseCode GetDMSPPhaseCode(FTN.PhaseCode phases)...
    public static TransformerFunction GetDMSTransformerFunctionKind(FTN.TransformerFunctionKind transformerFunction)...
    public static WindingType GetDMSWindingType(FTN.WindingType windingType)...
    public static WindingConnection GetDMSWindingConnection(FTN.WindingConnection windingConnection)...
    #endregion Enums convert
}
```

Importer 7/7

```
public static void PopulatePowerSystemResourceProperties(FTN.PowerSystemResource cimPowerSystemResource,
                                                       ResourceDescription rd, ImportHelper importHelper)
{
    if ((cimPowerSystemResource != null) && (rd != null))
    {
        PowerTransformerConverter.PopulateIdentifiedObjectProperties(cimPowerSystemResource, rd);

        if (cimPowerSystemResource.CustomTypeHasValue)
        {
            rd.AddProperty(new Property(ModelCode.PSR_CUSTOMTYPE, cimPowerSystemResource.CustomType));
        }
        if (cimPowerSystemResource.LocationHasValue)
        {
            long gid = importHelper.GetMappedGID(cimPowerSystemResource.Location.ID);
            rd.AddProperty(new Property(ModelCode.PSR_LOCATION, gid));
        }
    }
}
```



Zadaci

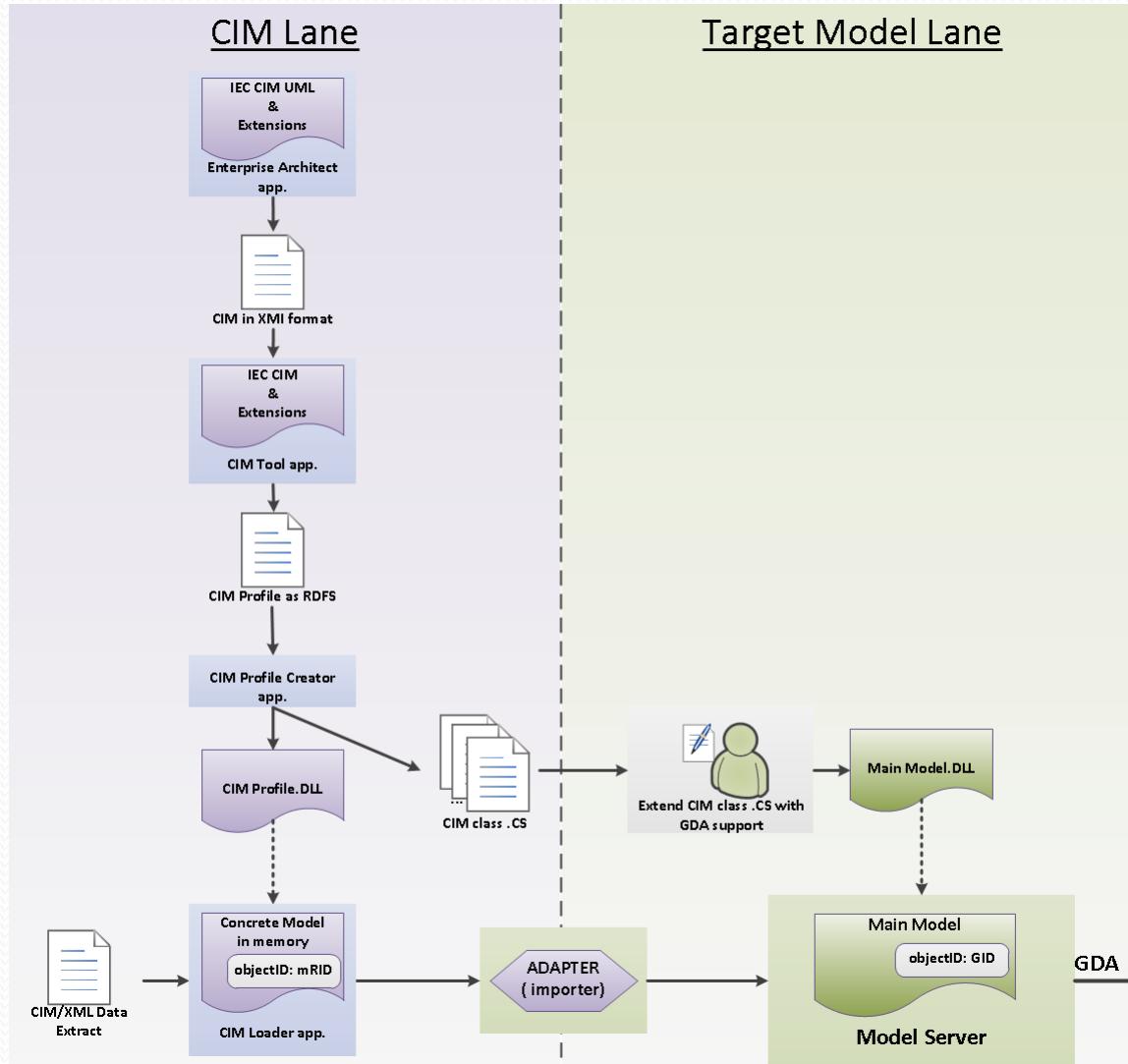
1. U okviru klase *PowerTransformerImporter.cs* podržati import CIM tipova:
 1. TransformerWinding
 2. WindingTest

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 10:

Generic Data Access podrška za čitanje
podataka sa Network Model Service-a

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Generic Data Access (GDA) 1/2

- Omogućava pristup podacima bez ikakvog znanja o logičkoj šemi koja se koristi za unutrašnje skladištenje - implementaciji. Dovoljno je poznavanje definisanog modela, a u ovom slučaju je to CIM.
- Prvobitno je standardizovan u okviru OMG-a kao Data Access Facility (DAF) koji je pružao osnovnu sposobnost postavljanja upita nad nekim modelom. Daljom nadogradnjom nastaje GDA koji obezbeđuje naprednije funkcionalnosti (npr. filtriranje), čitanje i upis podataka.
- Bilo koja klasa i njeni atributi iz internog modela prikazuju se u formi resursa pridržavajući se određenih pravila.

Generic Data Access (GDA) 2/2

- Standard definiše upite i odgovore u vidu resursa, propertija i vrednosti.
- Resurs je svaki objekat koji poseduje jedinstveni identifikator
- Properti je neki aspekt resursa koji se može opisati. Kada se pojavi u upitu predstavljen je preko *ModelCode-a*. Svaki properti ima svoj domen – niz resursa na koji se može primeniti.
- Asocijacije između resursa se kreiraju preko propertija koji su tipa *Reference*.

Metode za čitanje podataka

- Resource Query Service

- GetValues(...) – čitanje jednog resursa

```
public ResourceDescription GetValues(long resourceId, List<ModelCode> propIds)
```

- GetExtentValues(...) – čitanje niza resursa za isti tip entiteta

```
public int GetExtentValues(ModelCode code, List<ModelCode> propIds)
```

- GetRelatedValues(...) – čitanje referenciranog resursa

```
public int GetRelatedValues(long source, List<ModelCode> propIds, Association association)
```

- GetDescendentValues(...) – čitanje niza referenciranih resursa

```
public int GetDescendentValues(List<long> sources, List<ModelCode> propIds,  
List<Association> path, List<Association> tail)
```

GetValues(...)

- Čitanje jednog resursa (entiteta)

```
public ResourceDescription GetValues(long resourceld, List<ModelCode> proplds)
```

- Metodi se prosleđuje globalni identifikator entiteta i lista *ModelCode*-ova koji predstavljaju identifikatore propertija vezanih za entitet
- Rezultat je *ResourceDescription* koji sadrži prosleđeni identifikator i listu zahtevanih propertija

```
long globalId = 0x0000000300000001;

List<ModelCode> propertyIDs = new List<ModelCode>();
propertyIDs.Add(ModelCode.IDOBJ_MRID);
propertyIDs.Add(ModelCode.IDOBJ_DESCRIPTION);
propertyIDs.Add(ModelCode.EQUIPMENT_ISPRIVATE);
propertyIDs.Add(ModelCode.POWERTR_FUNC);

rd = GdQueryProxy.GetValues(globalId, propertyIDs);
```

GetExtentValues(...)

- Prva iz grupe metoda koje vraćaju niz resursa
- Čitanje niza resursa za isti tip entiteta

```
public int GetExtentValues(ModelCode code, List<ModelCode> propIds)
```

- Metodi se prosleđuje *ModelCode* koji predstavlja tip entiteta i lista *ModelCode*-ova koji predstavljaju identifikatore propertija vezanih za entitet
- Rezultat je identifikator upita (identifikator iterаторa) na osnovu koga klijent može da pročita rezultujuće *ResourceDescription*-e

```
ModelCode resourceType = ModelCode.POWERTR;  
  
List<ModelCode> propertyIDs = new List<ModelCode>();  
propertyIDs.Add(ModelCode.IDOBJ_MRID);  
propertyIDs.Add(ModelCode.IDOBJ_DESCRIPTION);  
propertyIDs.Add(ModelCode.EQUIPMENT_ISPRIVATE);  
propertyIDs.Add(ModelCode.POWERTR_FUNC);  
  
int iteratorId = GdqProxy.GetExtentValues(resourceType, propertyIDs);
```

Iteriranje kroz rezultat upita 1/2

- Metode čiji je rezultat niz *ResourceDescription*-a, ne vraćaju niz već identifikator upita. Niz može biti veoma velik i nije pogodno da klijent sve podatke prihvati od jednom. Servis čuva rezultate upita, a klijent je u mogućnosti da iterira kroz rezultujući niz.
- Da bi ovo omogućio, pored metoda koje predstavljaju upit, neophodno je da servis obezbedi metode za iteriranje kroz rezultate upita:
 - List<*ResourceDescription*> IteratorNext(int n, int id); - pozivom ove metode klijent zahteva od servisa da mu prosledi narednih *n* *ResourceDescription*-a za upit čiji je identifikator *id*
 - bool IteratorRewind(int id); - zahtev servisu da se vrati na početak rezultujućeg niza
 - int IteratorResourcesTotal(int id); - koliko ukupno postoji *ResourceDescription*-a u rezultujućem nizu
 - int IteratorResourcesLeft(int id); - koliko je ostalo nepročitanih *ResourceDescription*-a
 - bool IteratorClose(int id); - oslobođanje resursa, poziva se kada se završi rad nad upitom čiji je identifikator *id*

Iteriranje kroz rezultat upita 2/2

```
ModelCode resourceType = ModelCode.POWERTR;

List<ModelCode> propertyIDs = new List<ModelCode>();
propertyIDs.Add(ModelCode.IDOBJ_MRID);
propertyIDs.Add(ModelCode.IDOBJ_DESCRIPTION);
propertyIDs.Add(ModelCode.EQUIPMENT_ISPRIVATE);
propertyIDs.Add(ModelCode.POWERTR_FUNC);

int iteratorId = GdaQueryProxy.GetExtentValues(resourceType, propertyIDs);

int resourcesLeft = GdaQueryProxy.IteratorResourcesLeft(iteratorId);

int numberOfResources = 500;
List<ResourceDescription> result = new List<ResourceDescription>();

while (resourcesLeft > 0)
{
    List<ResourceDescription> rds = GdaQueryProxy.IteratorNext(numberOfResources, iteratorId);
    result.AddRange(rds);
    resourcesLeft = GdaQueryProxy.IteratorResourcesLeft(iteratorId);
}

GdaQueryProxy.IteratorClose(iteratorId);
```

GetRelatedValues(...) 1/3

- Čitanje niza resursa vezanih za neki resurs (entitet)

```
public int GetRelatedValues(long source, List<ModelCode> propIds, Association association)
```

- Metodi se prosleđuje *source* koji predstavlja globalni identifikator entiteta za koji želimo da nađemo povezane resurse(entitete), lista *ModelCode*-ova koji predstavljaju identifikatore propertija rezultujućih resursa i *association* koja predstavlja vezu između *source*-a i rezultujućih resursa
- Rezultat je identifikator upita (identifikator iteratora) na osnovu koga klijent može da pročita rezultujuće *ResourceDescription*-e

GetRelatedValues(...) 2/3

- Association – objekat kojim se definišu veze između entiteta u GDA upitima
- Sadrži tri atributa: *inverse*, *propertyId*, *type*
- *bool Inverse* – nije od interesa na ovom kursu
- *ModelCode propertyId* – identifikator propertija tipa *reference* koji sadrži *source*. Vrednost tog propertija sadrži globalne identifikatore resursa koji su povezani sa *source*-om preko ovog propertija.
- *ModelCode type* – služi za filtriranje resursa koji se dobiju na osnovu *propertyId*-a. Ukoliko mu je vrednost nula ne radi se filtriranje, u suprotnom kao rezultat primene asocijacije vraćaju se samo resursi tog tipa.

```
[DataContract]
public class Association
{
    private bool inverse;
    private ModelCode propertyId;
    private ModelCode type;

    public Association()
    {
        this.inverse = false;
        this.propertyId = 0;
        this.type = 0;
    }
}
```

GetRelatedValues(...) 3/3

```
long source = 0x0000000200000001;

Association association = new Association();
association.PropertyId = ModelCode.LOCATION_PSRS;
association.Type = ModelCode.POWERTR;

List<ModelCode> propertyIDs = new List<ModelCode>();
propertyIDs.Add(ModelCode.IDOBJ_MRID);
propertyIDs.Add(ModelCode.IDOBJ_DESCRIPTION);
propertyIDs.Add(ModelCode.EQUIPMENT_ISPRIVATE);
propertyIDs.Add(ModelCode.POWERTR_FUNC);

int iteratorId = GdaQueryProxy.GetRelatedValues(source, propertyIDs, association);
int resourcesLeft = GdaQueryProxy.IteratorResourcesLeft(iteratorId);

int numberOfResources = 500;
List<ResourceDescription> result = new List<ResourceDescription>();

while (resourcesLeft > 0)
{
    List<ResourceDescription> rds = GdaQueryProxy.IteratorNext(numberOfResources, iteratorId);
    result.AddRange(rds);
    resourcesLeft = GdaQueryProxy.IteratorResourcesLeft(iteratorId);
}

GdaQueryProxy.IteratorClose(iteratorId);
```

GetDescendentValues(...) 1/2

- Čitanje niza resursa vezanih za niz prosleđenih resursa (entiteta)

```
public int GetDescendentValues(List<long> sources, List<ModelCode> propIds,  
                               List<Association> path, List<Association> tail)
```

- Metodi se prosleđuje lista *source*-eva koji predstavljaju globalne identifikator entiteta za koje želimo da nađemo povezane resurse(entitete), lista *ModelCode*-ova koji predstavljaju identifikatore propertija rezultujućih resursa i lista asocijacija (*path*) koje predstavljaju veze između *source*-eva i rezultujućih resursa
- Rezultat je identifikator upita (identifikator iteratora) na osnovu koga klijent može da pročita rezultujuće *ResourceDescription*-e

GetDescendentValues(...) 2/2

```
List<ModelCode> properties = new List<ModelCode>();
properties.Add(ModelCode.WINDINGTEST_LOADLOSS);
properties.Add(ModelCode.WINDINGTEST_NOLOADLOSS);
properties.Add(ModelCode.IDOBJ_MRID);

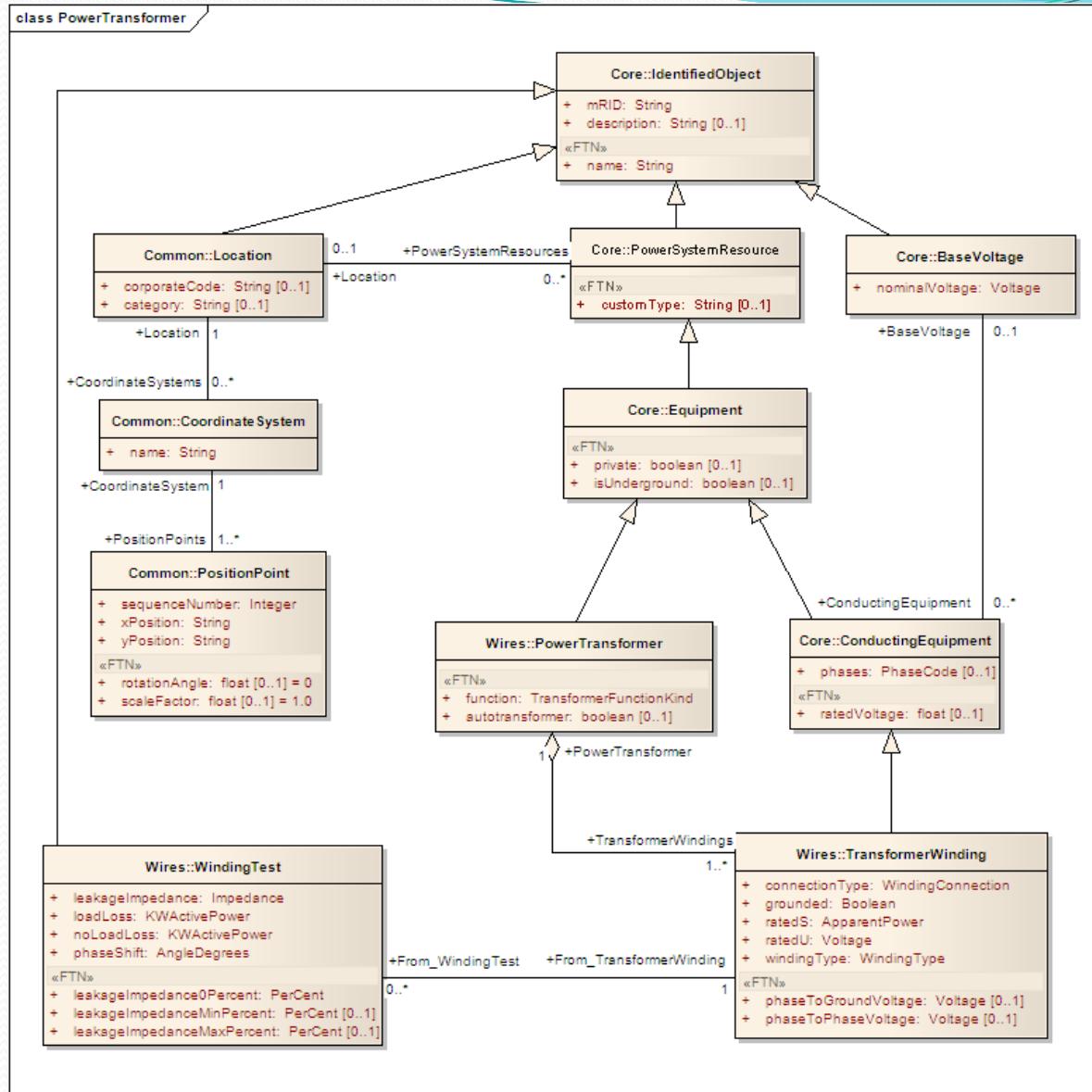
Association firstAssociation = new Association();
firstAssociation.PropertyId = ModelCode.BASEVOLTAGE_CONDEQS;
firstAssociation.Type = ModelCode.POWERTRWINDING;

Association secondAssociation = new Association();
secondAssociation.PropertyId = ModelCode.POWERTRWINDING_TESTS;
secondAssociation.Type = 0;

List<Association> path = new List<Association>() { firstAssociation, secondAssociation };
List<Association> tail = new List<Association>();

List<long> sources = new List<long>() { 0x0000000100000001, 0x0000000100000002, 0x0000000100000003 };

int iteratorId = GdaQueryProxy.GetDescendentValues(sources, properties, path, tail);
```



Zadaci

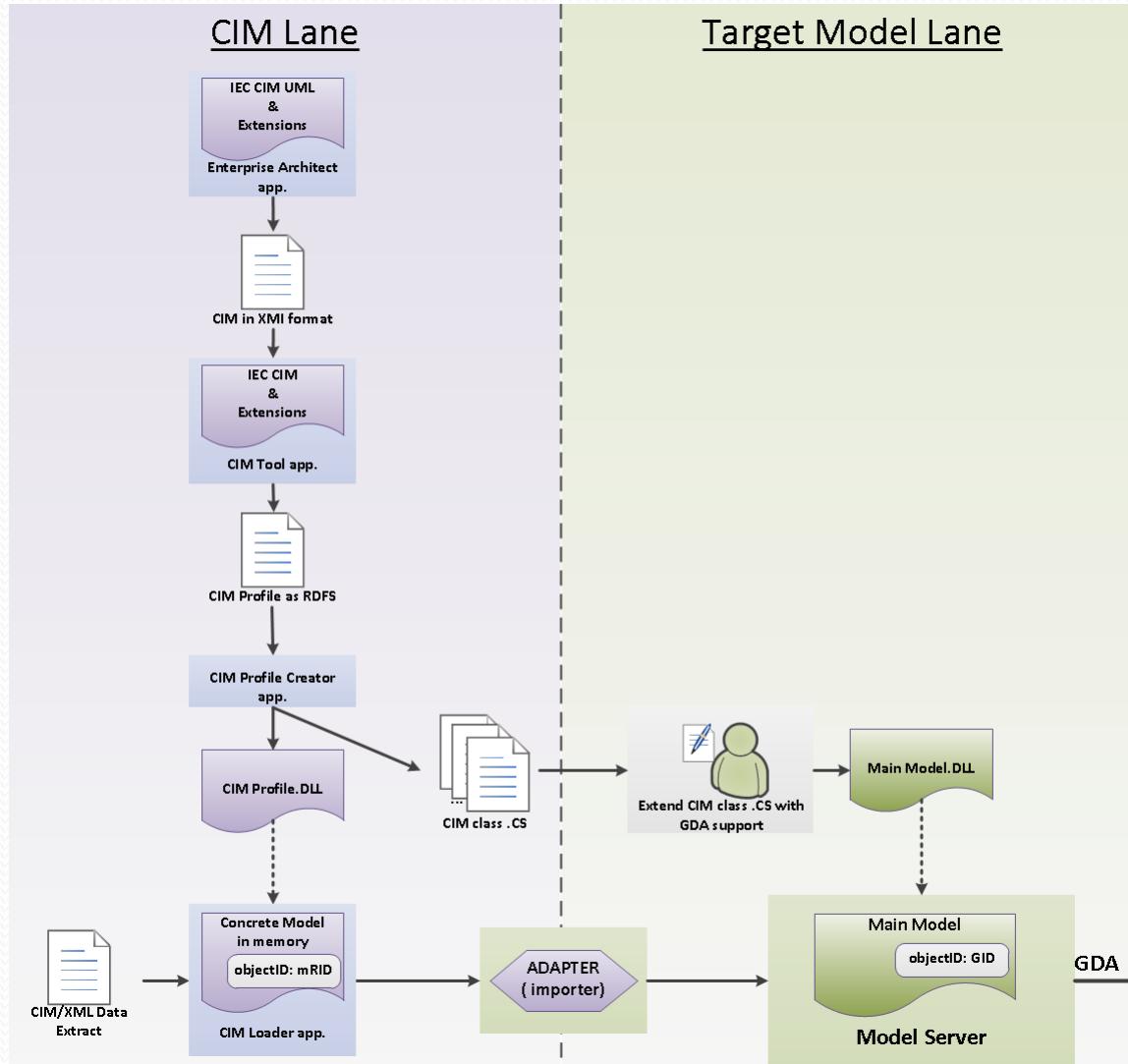
1. Pročitati sve *PowerTransformer*-e sa servisa. Za svaki od njih tražiti sve propertije koje *PowerTransformer* sadrži. Rezultate prikazati na proizvoljan način (štampati u XML, prikazati na formi...).
2. Za *PowerTransformer* koji ima najmanji globalni identifikator pročitati *TransformerWinding*-e koji su vezani za njega. Tražiti sve propertije koje *TransformerWinding* sadrži. Rezultate prikazati na proizvoljan način (štampati u XML, prikazati na formi...).

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 11:

Dodavanje nove klase (bez dodatnih referenci)
u Network Model Service

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Osnovne informacije o Network Model Service-u (NMS) 1/2

- Expose-uje WCF interfejs *INetworkModelGDAContract* na adresi *net.tcp://localhost:10000/NetworkModelService/GDA/*
- Podešavanja vezana za *binding* i *serviceBehavior* nalaze se u konfiguracionom fajlu servisa
- Obezbeđuje log; nivo logovanja i ime log fajla je moguće moguće menjati u konfiguracionom fajlu
- Zahtevi za izmenom modela (*delta objekte*) koji stignu na servis čuvaju se u posebnoj datoteci definisanoj u konfiguracionom fajlu. **Ukoliko dođe do spuštanja servisa, pri narednom podizanju učitavaju se svi podaci koji su do tada pristigli od strane adaptera.**

Osnovne informacije o Network Model Service-u (NMS) 2/2

- Deo konfiguracionog fajla servisa

```
<system.diagnostics>
  <trace autoflush="true">
    <listeners>
      <add type="System.Diagnostics.TextWriterTraceListener" name="TextWriter" initializeData="..../NetworkModelService.log" />
    </listeners>
  </trace>
  <switches>
    <!-- 0 - Disabled
        1 = Error - Gives error messages
        2 = Warning - Gives errors and warnings
        3 = Info - Gives more detailed error information
        4 = Verbose - Gives verbose trace information. -->
    <add name="TraceLevel" value = "Info" />
  </switches>
</system.diagnostics>

<connectionStrings>
  <add name="networkModelConnectionString" connectionString="..../NetworkModelData.data"/>
</connectionStrings>
```

Pravila implementacije Common-a 1/4

- Učitava ga svaka aplikacija u sistemu
- Opisuje model koji postoji na NMS (sadrži metapodatke o modelu)
- *CommonTrace* klasa se koristi za logovanje stanja aplikacije: statickoj metodi *WriteTrace()* prosleđuje se nivo log-a i poruka koju želimo da zapišemo.

```
string message = "Starting Network Model Service...";  
CommonTrace.WriteTrace(CommonTrace.TraceInfo, message);
```

- *ModelDefines.cs* definiše SVE *ModelCode*-ove i *DMSType*-ove (svi koji postoje su navedeni u tom .cs fajlu)
- *Enums.cs* sadrži definicije svih enumeracija koje postoje u modelu. Ovo su enumeracije koje predstavljaju tip nekog atributa klase u modelu.

```
using System;  
  
namespace FTN.Common  
{  
    public enum PhaseCode : short...  
    public enum TransformerFunction : short...  
    public enum WindingConnection : short...  
    public enum WindingType : short  
    {  
        None = 0,  
        Primary = 1,  
        Secondary = 2,  
        Tertiary = 3  
    }  
}
```

Pravila implementacije Common-a 2/4

- *EnumDescs* klasa opisuje mapiranje enumeracija i atributa određene klase. Ukoliko je atribut neke klase tipa enumeracija potrebno je dodati mapiranje između *ModelCoda*-a koji je dodeljen tom atributu i tipa enumeracije.

```
public class EnumDescs
{
    private Dictionary<ModelCode, Type> property2enumType = new Dictionary<ModelCode, Type>();

    public EnumDescs()
    {
        property2enumType.Add(ModelCode.CONDEQ_PHASES, typeof(PhaseCode));
        property2enumType.Add(ModelCode.POWERTR_FUNC, typeof(TransformerFunction));
        property2enumType.Add(ModelCode.POWERTRWINDING_CONNTYPE, typeof(WindingConnection));
        property2enumType.Add(ModelCode.POWERTRWINDING_WINDTYPE, typeof(WindingType));
    }
    ...
}
```

- Klasa *PowerTransformer* sadrži atribut *function* koji je tipa enumeracija *TransformerFunction*. *EnumDescs MORA* da definiše mapiranje *ModelCode.POWERTR_FUNC* na tip *TransformerFunction*.

```
public class PowerTransformer : Equipment
{
    private bool autotransformer = false;

    private TransformerFunction function;

    private List<long> transformerWindings = new List<long>();

    public PowerTransformer(long globalId)
        : base(globalId)
    {
    }
```

Pravila implementacije Common-a 3/4

- *ModelResourcesDesc* klasa obezbeđuje niz metoda za manipulaciju *ModelCode*-ovima na osnovu informacija koje su definisane u samoj vrednosti *ModelCode*-a (nasleđivanje, da li je klasa apstraktna ili ne, tip podatka, itd.):
 - *public static ModelCode FindFirstParent(ModelCode typeId)* – vraća *ModelCode* roditeljske klase
 - *public static bool InheritsFrom(ModelCode parentModelCode, ModelCode childModelCode)* – da li se *parentModelCode* nalazi bilo gde u hijararhiji nasleđivanja *childModelCode*-a
 - *public List<ModelCode> GetAllPropertyIds(ModelCode code)* – vraća *ModelCode*-ove koji su dodeljeni atributima neke klase.
 - ...
- *ModelResourcesDesc* klasa definiše listu atributa (propertija) čija vrednost NE SME da se postavi od strane klijenta. Vrednosti ovih atributa određuje servis, a ukoliko klijent pokuša da postavi neku od ovih vrednosti (kroz *Delta* objekat) servis će to odbiti. Pored globalnog identifikatora ovde se navode svi atributi koji su tipa lista referenci.

HashSet<ModelCode> notSettablePropertyIds –
- sadrži *ModelCode*-ove atributa čije vrednosti određuje isključivo servis.

```
private void InitializeNotSettablePropertyIds()
{
    notSettablePropertyIds.Add(ModelCode.IDOBJ_GID);
    notSettablePropertyIds.Add(ModelCode.BASEVOLTAGE_CONDEQS);
    notSettablePropertyIds.Add(ModelCode.LOCATION_PSRS);
    notSettablePropertyIds.Add(ModelCode.POWERTRWINDING_TESTS);
}
```

Pravila implementacije Common-a 4/4

- Jedan od najvažnijih zadataka *ModelResourcesDesc* klase jeste da definiše kojim redosledom će se izvršavati operacije koje stignu na servis kao *Delta* objekat (insert, update, delete).
- Kada na servis stigne *Delta* objekat radi izmene modela (najčešće od strane adaptera) potrebano je da servis sortira operacije po tipu entiteta. Servis čita redosled iz *ModelResourcesDesc* klase i vrši soritiranje pre primene izmena.
private List<ModelCode> typeIdsInInsertOrder – definiše redosled sortiranja.

```
private void InitializeTypeIdsInInsertOrder()
{
    typeIdsInInsertOrder.Add(ModelCode.BASEVOLTAGE);
    typeIdsInInsertOrder.Add(ModelCode.LOCATION);
    typeIdsInInsertOrder.Add(ModelCode.POWERTR);
    typeIdsInInsertOrder.Add(ModelCode.POWERTRWINDING);
    typeIdsInInsertOrder.Add(ModelCode.WINDINGTEST);
}
```

- Lista **MORA** da sadrži *ModelCode*-ove svih konkretnih klasa.
- Pravilo za formiranje redosleda *ModelCode*-ova u ovoj listi je sledeće:

Ukoliko klasa A ima atribut tipa referenca čija je vrednost globalni identifikator klase B, tada klasa B ima atribut tipa lista referenci čija je vrednost lista globalnih identifikatora klase A. Tada kažemo da klasa A referencira klasu B, dok klasu B nazivamo target.

U listi UVEK mora da se navede target (klasa B) pre klase koja ga referencira (klasa A).

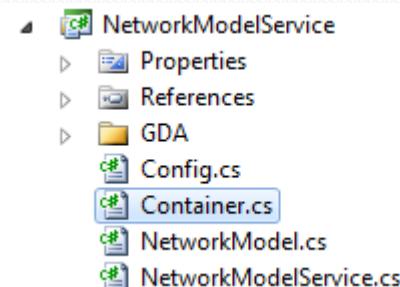
Implementacija Network Model Service-a 1/5

- Klasa *Container* grupiše entitete istog tipa.
- Sadrži mapiranje globalnih identifikatora na njihove entitete

```
public class Container
{
    /// <summary>
    /// The dictionary of entities. Key = GlobalId, Value = Entity
    /// </summary>
    private Dictionary<long, IdentifiedObject> entities = new Dictionary<long, IdentifiedObject>();

    /// <summary>
    /// Initializes a new instance of the Container class
    /// </summary>
    public Container() ...
}
```

- Zadužena je za kreiranje i pribavljanje entiteta koji ima odgovarajući globalni identifikator



Implementacija Network Model Service-a 2/5

- Metoda *CreateEntity()* klase *Container* MORA da obezbedi kreiranje bilo koje konkretne klase na osnovu globalnog identifikatora. Iz globalnog identifikatora se “izvuče” tip entiteta i kreira odgovarajući entitet.
- Ukoliko se u model dodaje nova konkretna klasa potrebno je proširiti *CreateEntity()* metodu kako bi klasa *Container* postala „svesna“ novog tipa entita.

```
/// <summary> ...
public IdentifiedObject CreateEntity(long globalId)
{
    short type = ModelCodeHelper.ExtractTypeFromGlobalId(globalId);

    IdentifiedObject io = null;
    switch ((DMSType)type)
    {
        case DMSType.BASEVOLTAGE:
            io = new BaseVoltage(globalId);
            break;

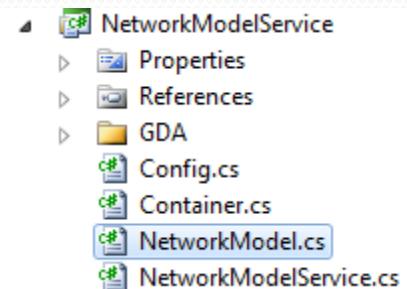
        case DMSType.LOCATION:
            io = new Location(globalId);
            break;
        case DMSType.POWERTR:
            io = new PowerTransformer(globalId);
            break;
        case DMSType.POWERTRWINDING:
            io = new TransformerWinding(globalId);
            break;
        case DMSType.WINDINGTEST:
            io = new WindingTest(globalId);
            break;

        default:
```

Implementacija Network Model Service-a 3/5

- *NetworkModel* singleton klasa sadrži instance *Container* klase za svaki tip entiteta.

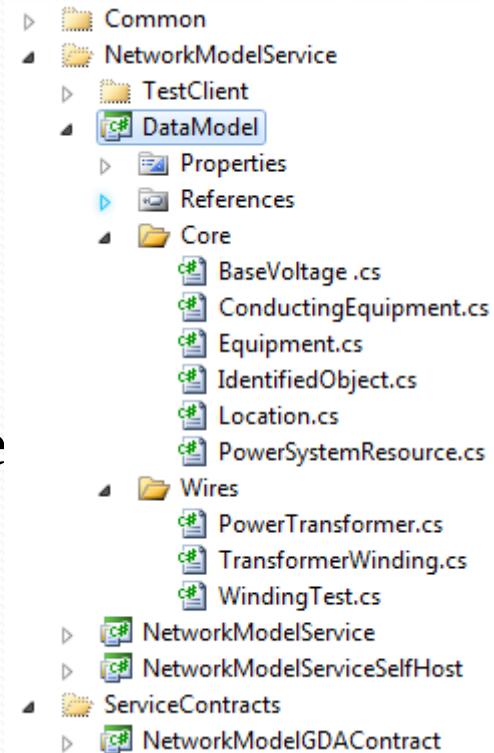
```
public class NetworkModel
{
    /// <summary>
    /// Dictionaru which contains all data: Key - DMSType, Value - Container
    /// </summary>
    private Dictionary<DMSType, Container> networkDataModel;
```



- Ukoliko je potrebno kreirati ili pribaviti neki entitet *NetworkModel* klasa pronalazi odgovarajuću instancu klase *Container* i prosleđuje joj zahtev.
- Implementacije *GDA* metoda nalaze se u *NetworkModel* klasi

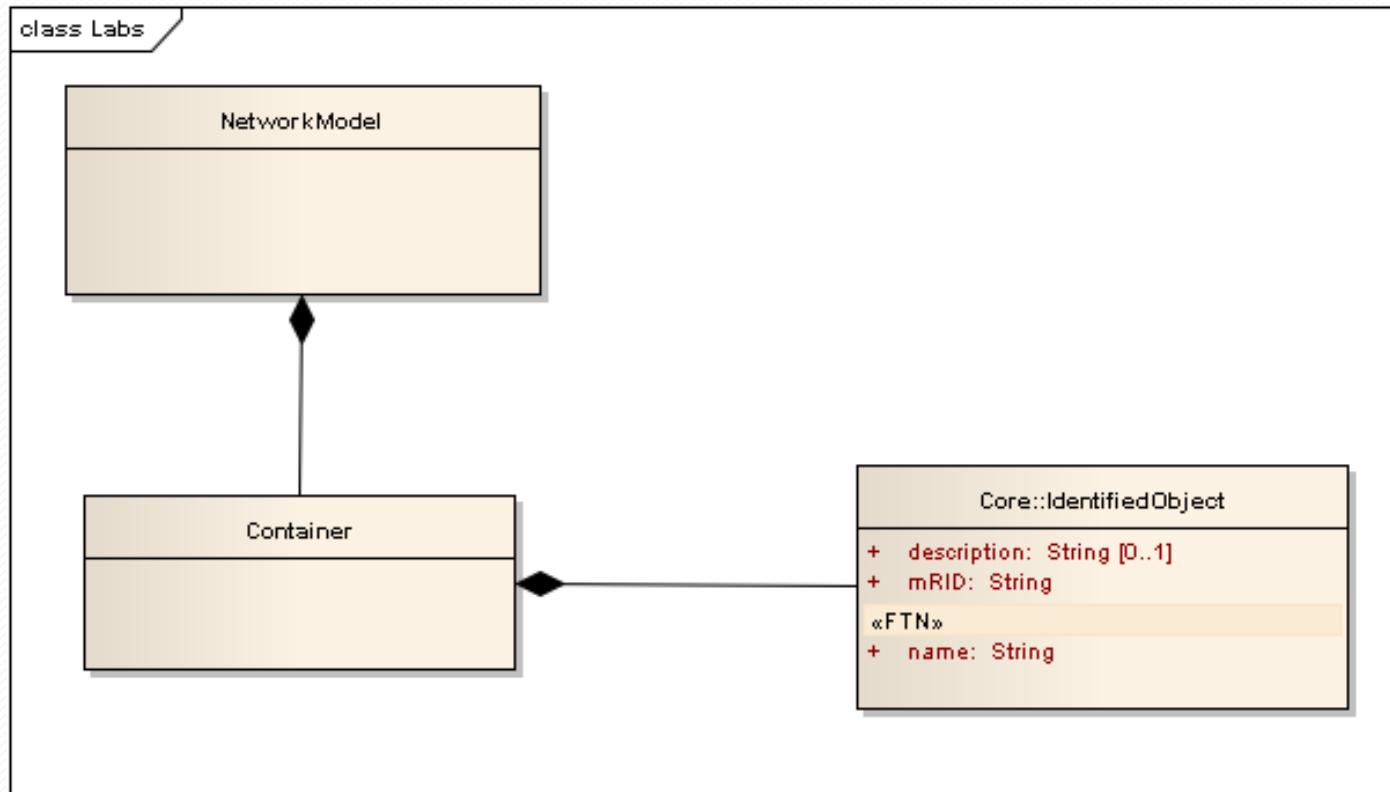
Implementacija Network Model Service-a 4/5

- *DataModel* projekat sadrži implementaciju samih klasa koje ulaze u model.
- Svaka klasa ima striktno definisanu strukturu: metode koje treba da implementira i kako da ih implementira. Na ovaj način bilo koja klasa (koja prati definisani način implementacije) može se dodati u model i servis će znati da manipuliše njom. (PODSETNIK: Ukoliko je klasa konkretna potrebno je proširiti *CreateEntity()* metodu klase *Container* kako bi servis mogao da kreira tu klasu)



Implementacija Network Model Service-a 5/5

- UML dijagram – organizacija *Network Model Service-a*



Struktura DataModel klase 1/6

- Potrebno je implementirati nasleđivanje na osnovu definisanog modela
- Klasa sadrži *private* atributе definisane modelom i za svaki od njih *public* propertije (NAPOMENA: U ovom slučaju se misli na C# propertije neke klase, a ne *Property* objekte koje koristi GDA standard)
- Svaka klasa implementira konstruktor koji kao parametar prima globalni identifikator

```
public class PowerTransformer : Equipment
{
    private bool autotransformer = false;

    private TransformerFunction function;

    private List<long> transformerWindings = new List<long>();

    public PowerTransformer(long globalId)
        : base(globalId)
    {
    }

    public bool Autotransformer
    {
        get { return autotransformer; }
        set { autotransformer = value; }
    }

    public TransformerFunction Function
    {
        get { return function; }
        set { function = value; }
    }

    public List<long> TransformerWindings
    {
        get { return transformerWindings; }
        set { transformerWindings = value; }
    }
}
```

Struktura DataModel klasa 2/6

- Implementirati *Equals()* i *GetHashCode()* metode
- *Equals()* metoda proverava jednakost atributa koji pripadaju roditeljskoj klasi. Ukoliko se dobije potvrdan odgovor proveravaju se atributi tekuće klase.

```
public override bool Equals(object obj)
{
    if (base.Equals(obj))
    {
        PowerTransformer x = (PowerTransformer)obj;
        return (x.function == this.function && x.autotransformer == this.autotransformer &&
            CompareHelper.CompareLists(x.TransformerWindings, this.TransformerWindings, true));
    }
    else
    {
        return false;
    }
}

public override int GetHashCode()
{
    return base.GetHashCode();
}
```

Struktura DataModel klasa 3/6

- Kako bi servis na jedinstven način mogao da radi sa svakim tipom entiteta, potrebno je da klasa implementira metode koje će servis koristiti za GDA manipulaciju (*IAccess implementation* region).
- *IReference implementation* region sadrži metode koje servis koristi kako bi samostalno mogao da manipuliše atributima koji su tipa lista referenci (*target*) - detaljnije na sledećim vežbama.

```
#region IAccess implementation

public override bool HasProperty(ModelCode t)...
public override void GetProperty(Property prop)...
public override void SetProperty(Property property)...

#endregion IAccess implementation

#region IReference implementation

public override bool IsReferenced...
public override void GetReferences(Dictionary<ModelCode, List<long>> references, TypeOfReference refType)...
public override void AddReference(ModelCode referenceId, long globalId)...
public override void RemoveReference(ModelCode referenceId, long globalId)...

#endregion IReference implementation
```

Struktura DataModel klase 4/6

- *HasProperty()* metoda daje odgovor na pitanje da li vrednost *ModelCode-a* odgovara nekom atributu klase. Prvo se proverava da li je to to atribut tekuće klase, ukoliko nije prelazi se na proveru da li je to atribut roditeljske klase.

```
public override bool HasProperty(ModelCode t)
{
    switch (t)
    {
        case ModelCode.POWERTR_AUTO:
        case ModelCode.POWERTR_FUNC:
        case ModelCode.POWERTR_WINDINGS:
            return true;

        default:
            return base.HasProperty(t);
    }
}
```

Struktura DataModel klasa 5/6

- *GetProperty()* metoda služi za konverziju atributa klase u *Property* objekat koji koristi GDA standard.
- Pročita se *propertyId* (*ModelCode*) prosleđenog properti objekta i proverava se da li on odgovara nekom od atributa tekuće klase. Ukoliko odgovara, vrednost tog atributa se zapiše kao vrednost prosleđenog propertija (*SetValue()* metoda). Ako *propertyId* ne odgovara nijednom atributu tekuće klase, poziva se *GetProperty()* metoda roditeljske klase.
- *GetProperty()* metoda MORA da obezbedi mogućnost čitanja svakog atributa klase.
- Ukoliko je atribut tipa enum MORA se konvertovati u *short* pre nego što se njegova vrednost zapiše u properti.

```
public override void GetProperty(Property prop)
{
    switch (prop.Id)
    {
        case ModelCode.POWERTR_FUNC:
            prop.SetValue((short)function);
            break;

        case ModelCode.POWERTR_AUTO:
            prop.SetValue(autotransformer);
            break;

        case ModelCode.POWERTR_WINDINGS:
            prop.SetValue(transformerWindings);
            break;

        default:
            base.GetProperty(prop);
            break;
    }
}
```

Struktura DataModel klasa 6/6

- *SetProperty()* metoda služi za postavljanje vrednosti atributa na osnovu *Property* objekta koji koristi GDA standard.
- Pročita se *propertyId* (*ModelCode*) prosleđenog properti objekta i proverava se da li on odgovara nekom od atributa tekuće klase. Ukoliko odgovara vrednost propertija se postavlja kao vrednost tog atributa. Ako *propertyId* ne odgovara nijednom atributu tekuće klase, poziva se *SetProperty()* metoda roditeljske klase.
- *SetProperty()* metoda NE SME da dozvoli postavljanje vrednosti *notSettable* atributa – atributa koji su navedeni u *ModelResourcesDesc* klasi kao attribute čiju vrednost postavlja servis (globalni identifikator i liste referenci (*target*))
- Ukoliko je atribut tipa enum, vrednost propertija je potrebno kastovati u odgovarajući tip enumeracije.

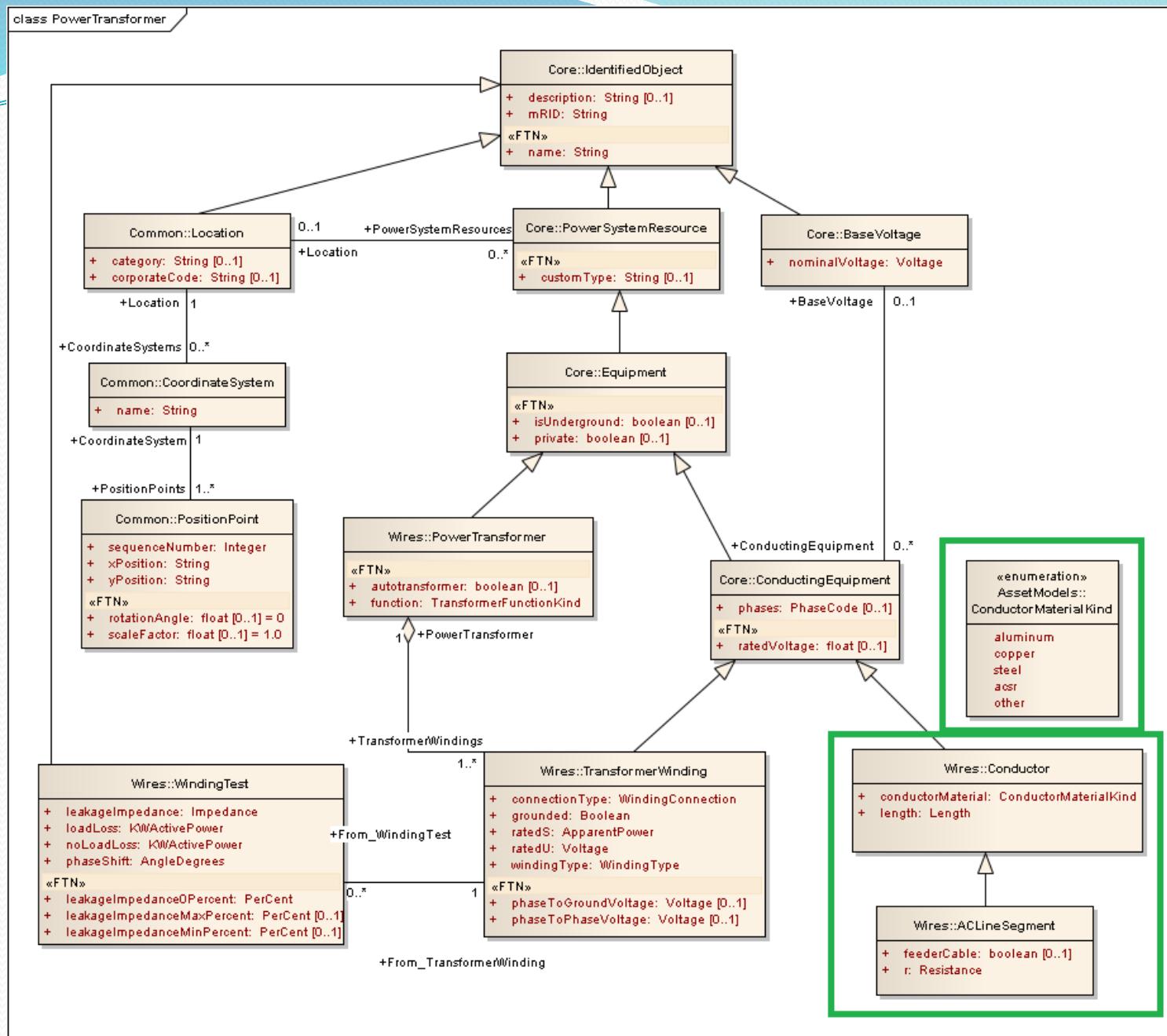
```
public override void SetProperty(Property property)
{
    switch (property.Id)
    {
        case ModelCode.POWERTR_AUTO:
            autotransformer = property.AsBool();
            break;

        case ModelCode.POWERTR_FUNC:
            function = (TransformerFunction)property.AsEnum();
            break;

        default:
            base SetProperty(property);
            break;
    }
}
```

Rezime

- Prilikom dodavanja nove klase (ili atributa postojeće klase) potrebno je pratiti sledeće korake:
 1. Dodati odgovarajuće *ModelCode*-ove za nove klase i atribute
 2. Ukoliko su dodate nove enumeracije proširiti *Enums.cs* fajl
 3. Ukoliko je neki od atributa tipa enumeracija proširiti *EnumDescs* klasu dodatnim mapiranjem
 4. Ukoliko imamo *notSettable* atributa proširiti njima *ModelResourcesDesc* klasu (doraditi metodu *InitializeNotSettablePropertyIds()*)
 5. Ukoliko je dodata konkretna klasa proširiti *ModelResourcesDesc* i *Container* klase (metode *InitializeTypeIdsInInsertOrder()* i *CreateEntity()*)
 6. Implementirati nove klase i po potrebi doraditi postojeće na osnovu prethodno definisanih pravila.



Zadaci

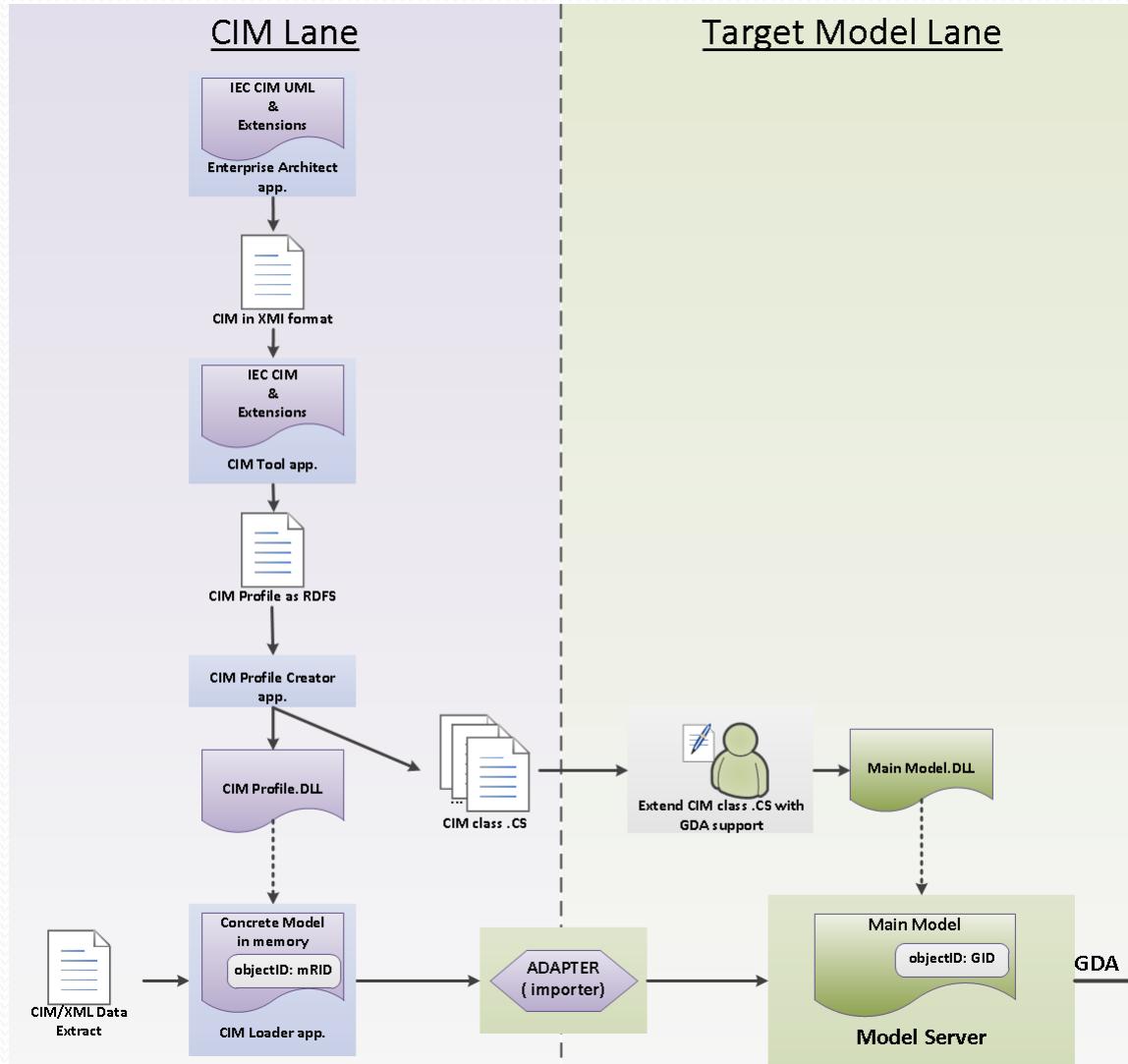
1. Dodati apstraktnu klasu *Conductor*
2. Dodati konkretnu klasu *ACLineSegment*

Standardi i modeliranje elektroenergetskih sistema

VEŽBA 12:

Dodavanje nove klase (sa dodatnim referencama)
u Network Model Service

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Struktura DataModel klase 1/2

- Atribut klase koji predstavlja referencu je tipa *long*, a njegova vrednost je globalni identifikator entiteta koga referencira
- Atribut klase koji predstavlja listu referenci (*target*) je tipa *Lista<long>* i njegova vrednost je lista globalnih identifikatora entiteta koji imaju referencu na pomenutu klasu
- Klase koje sadrže reference (jednu referencu ili listu referenci) implementiraju dodatne metode koje će omogućiti servisu manipulaciju nad referencama

```
public class TransformerWinding : ConductingEquipment
{
    private WindingConnection connectionType;
    private WindingType windingType;
    private bool grounded;
    private float ratedS;
    private float ratedU;
    private float phaseToGroundVoltage;
    private float phaseToPhaseVoltage;
    private long powerTransformer = 0;
    private List<long> windingTests = new List<long>();

    public TransformerWinding(long globalId)
        : base(globalId)
    {
    }
}
```

Struktura DataModel klasa 2/2

- *Equals()* metoda mora da proverava jednakost i ovih atributa (atributi tipa referenca i lista referenci (*target*))

```
public override bool Equals(object obj)
{
    if (base.Equals(obj))
    {
        TransformerWinding x = (TransformerWinding)obj;
        return (x.windingType == this.windingType && x.grounded == this.grounded && x.connectionType == this.connectionType &&
            x.ratedS == this.ratedS && x.ratedU == this.ratedU && x.phaseToGroundVoltage == this.phaseToGroundVoltage &&
            x.phaseToPhaseVoltage == this.phaseToPhaseVoltage && x.powerTransformer == this.powerTransformer &&
            CompareHelper.CompareLists(x.windingTests, this.windingTests));
    }
    else
    {
        return false;
    }
}
```

Metode za pristup podacima 1/4

- U *HasProperty()* metodi potrebno je navesti *ModelCode*-ove dodeljene ovim atributima.

```
public override bool HasProperty(ModelCode t)
{
    switch (t)
    {
        case ModelCode.POWERTRWINDING_CONNTYPE:
        case ModelCode.POWERTRWINDING_GROUNDED:
        case ModelCode.POWERTRWINDING_RATEDS:
        case ModelCode.POWERTRWINDING_RATEDU:
        case ModelCode.POWERTRWINDING_WINDTYPE:
        case ModelCode.POWERTRWINDING_PHASETOGRNDVOLTAGE:
        case ModelCode.POWERTRWINDING_PHASETOPHASEVOLTAGE:
        case ModelCode.POWERTRWINDING_POWERTRW:
        case ModelCode.POWERTRWINDING_TESTS:
            return true;

        default:
            return base.HasProperty(t);
    }
}
```

Metode za pristup podacima 2/4

- *GetProperty()* metoda omogućava čitanje ovih atributa.

```
public override void GetProperty(Property property)
{
    switch (property.Id)
    {
        case ModelCode.POWERTRWINDING_CONNTYPE:
            property.SetValue((short)connectionType);
            break;

        ...
        case ModelCode.POWERTRWINDING_PHASETOPHASEVOLTAGE:
            property.SetValue(phaseToPhaseVoltage);
            break;

        case ModelCode.POWERTRWINDING_POWERTRW:
            property.SetValue(powerTransformer);
            break;

        case ModelCode.POWERTRWINDING_TESTS:
            property.SetValue(windingTests);
            break;

        default:
            base.GetProperty(property);
            break;
    }
}
```

Metode za pristup podacima 3/4

- *SetProperty()* metoda omogućava upis samo atributa tipa referenca. Vrednost atribut tipa lista referenci (*target*) kreira sam servis i njegov upis se NE SME dozvoliti kroz *SetProperty()* metodu

```
public override void SetProperty(Property property)
{
    switch (property.Id)
    {
        case ModelCode.POWERTRWINDING_CONNTYPE:
            connectionType = (WindingConnection)property.AsEnum();
            break;

        ...
        case ModelCode.POWERTRWINDING_POWERTRW:
            powerTransformer = property.AsReference();
            break;
        default:
            base SetProperty(property);
            break;
    }
}
```

Metode za pristup podacima 4/4

- Pošto vrednost atribut tipa lista referenci ne može da se postavi od strane klijenta, već je određuje servis, potrebno je navesti njegov *ModelCode* u listu *notSettable* atributa u klasi *ModelResourceDescs*

```
private void InitializeNotSettablePropertyIds()
{
    notSettablePropertyIds.Add(ModelCode.IDOBJ_GID);
    notSettablePropertyIds.Add(ModelCode.BASEVOLTAGE_CONDEQS);
    notSettablePropertyIds.Add(ModelCode.LOCATION_PSRS);
    notSettablePropertyIds.Add(ModelCode.POWERTRWINDING_TESTS);
}
```

Metode za rad sa referencama 1/4

- *IsReferenced* implementiraju samo klase koje imaju listu referenci (*target-e*) kao neki od atributa. Metoda vraća indikaciju da li postoji neki entitet koji referencira instancu ove klase (referencira neki od *target-a*). Metoda vraća *true* ukoliko bar jedan entitet referencira instancu ove klase (proverava se da li neka lista referenci koje sadrži klasa ima bar jednu vrednost – globalni identifikator).

```
public override bool IsReferenced
{
    get
    {
        return windingTests.Count != 0 || base.IsReferenced;
    }
}
```

- Metodu koristi servis kada dobije zahtev za brisanjem odgovarajućeg entiteta. Ukoliko je entitet referenciran od strane nekog drugog entiteta, zahtev se odbija. **Smatra se da je entitet referenciran nekim drugim entitetom ukoliko neki od njegovih *target-a* ima bar jedan globalni identifikator u listi.**

Metode za rad sa referencama 2/4

- *GetReferences()* metoda se implementira za klasu koja ima atribut tipa referenca i/ili lista referenci (*target*). Kreira se mapa koja vraća vrednosti globalnih identifikatora koji su vrednosti atributa tipe referenca ili lista referenci (*target-a*).

```
public override void GetReferences(Dictionary<ModelCode, List<long>> references, TypeOfReference refType)
{
    if (powerTransformer != 0 && (refType == TypeOfReference.Reference || refType == TypeOfReference.Both))
    {
        references[ModelCode.POWERTRWINDING_POWERTRW] = new List<long>();
        references[ModelCode.POWERTRWINDING_POWERTRW].Add(powerTransformer);
    }

    if (windingTests != null && windingTests.Count != 0 && (refType == TypeOfReference.Target || refType == TypeOfReference.Both))
    {
        references[ModelCode.POWERTRWINDING_TESTS] = windingTests.GetRange(0, windingTests.Count);
    }

    base.GetReferences(references, refType);
}
```

Metode za rad sa referencama 3/4

- *AddReference()* metodu implementiraju samo klase koje imaju listu referenci (*target*) kao neki od atributa. Metodu koristi servis kako bi dodao odgovarajuću vrednost u atribut tipa lista referenci (*target*).
NAPOMENA: Treba primetiti da se u *switch* naredbi ne koristi *ModelCode* koji je dodeljen atributu tipa lista referenci (*target*) već *ModelCode* atributa koji pripada klasi koja referencira pomenuti target.

```
public override void AddReference(ModelCode referenceId, long globalId)
{
    switch (referenceId)
    {
        case ModelCode.WINDINGTEST_POWERTRWINDING:
            windingTests.Add(globalId);
            break;

        default:
            base.AddReference(referenceId, globalId);
            break;
    }
}
```

Metode za rad sa referencama 4/4

- *RemoveReference()* metodu implementiraju samo klase koje imaju listu referenci (*target*) kao neki od atributa. Metodu koristi servis kako bi uklonio odgovarajuću vrednost iz atributa tipa lista referenci (*target*) .
NAPOMENA: Treba primetiti da se u *switch* naredbi ne koristi *ModelCode* koji je dodeljen atributu tipa lista referenci (*target*) već *ModelCode* atributa koji pripada klasi koja referencira pomenuti target.

```
public override void RemoveReference(ModelCode referenceId, long globalId)
{
    switch (referenceId)
    {
        case ModelCode.WINDINGTEST_POWERTRWINDING:
            if (windingTests.Contains(globalId))
            {
                windingTests.Remove(globalId);
            }
            else
            {
                CommonTrace.WriteTrace(CommonTrace.TraceWarning, "Entity (GID = 0x");
            }
            break;

        default:
            base.RemoveReference(referenceId, globalId);
            break;
    }
}
```

InsertEntity() implementacija

- Razmotrićemo implementaciju *InsertEntity()* metode kako bi dodatno razjasnili ulogu metoda za kreiranje atributa tipa lista referenci (*target*).
- Prilikom primene *delte*, za svaku insert operaciju servis poziva metodu *InsertEntity()* i prosleđuje joj *ResourceDescription* instancu na osnovu koje je potrebno kreirati entitet i smestiti ga u odgovarajuću instancu klase *Container*.
- Za svaki *property* koji postoji u prosleđenom *ResourceDescription*-u se poziva metoda *SetProperty()*. Ukoliko je u pitanju *property* tipa referenca potrebna je dodatna obrada:
 - Pročita se vrednost *property*-a tipa referenca. Vrednost je globalni identifikator entiteta koji se referencira tim *property*-em.
 - Ukoliko je vrednost različita od nule, pronalazi se referencirani entitet i poziva se njegova *AddReference()* metoda.
 - Na ovaj način se *target* atributu referenciranog entiteta (referencira ga prethodno pomenuti *property*) doda globalni identifikator entiteta koji ga referencira.



```
/// <summary>
/// Inserts entity into the network model.
/// </summary>
/// <param name="rd">Description of the resource that should be inserted</param>
private void InsertEntity(ResourceDescription rd)
{
    if (rd == null)
    {
        CommonTrace.WriteTrace(CommonTrace.TraceVerbose, "Insert entity is not done because update operation is empty.");
        return;
    }

    long globalId = rd.Id;

    CommonTrace.WriteTrace(CommonTrace.TraceInfo, "Inserting entity with GID ({0:x16}).", globalId);

    // check if mapping for specified global id already exists
    if (this.EntityExists(globalId))
    {
        string message = String.Format("Failed to insert entity because entity with specified GID ({0:x16}) already exists");
        CommonTrace.WriteTrace(CommonTrace.TraceError, message);
        throw new Exception(message);
    }

    try
    {
        // find type
        DMSType type = (DMSType)ModelCodeHelper.ExtractTypeFromGlobalId(globalId);

        Container container = null;

        // get container or create container
        if (ContainerExists(type))
        {
            container = GetContainer(type);
        }
        else
        {
            container = new Container();
            networkDataModel.Add(type, container);
        }

        // create entity and add it to container
        IdentifiedObject io = container.CreateEntity(globalId);
    }
}
```

```

// create entity and add it to container
IdentifiedObject io = container.CreateEntity(globalId);

// apply properties on created entity
if (rd.Properties != null)
{
    foreach (Property property in rd.Properties)
    {
        // globalId must not be set as property
        if (property.Id == ModelCode.IDOBJ_GID)
        {
            continue;
        }

        if (property.Type == PropertyType.Reference)
        {
            // if property is a reference to another entity
            long targetGlobalId = property.AsReference();

            if (targetGlobalId != 0)
            {

                if (!EntityExists(targetGlobalId))
                {
                    string message = string.Format("Failed to get target entity with GID: 0x{0:X16}. {1}", targetGlobalId);
                    throw new Exception(message);
                }

                // get referenced entity for update
                IdentifiedObject targetEntity = GetEntity(targetGlobalId);
                targetEntity.AddReference(property.Id, io.GlobalId);

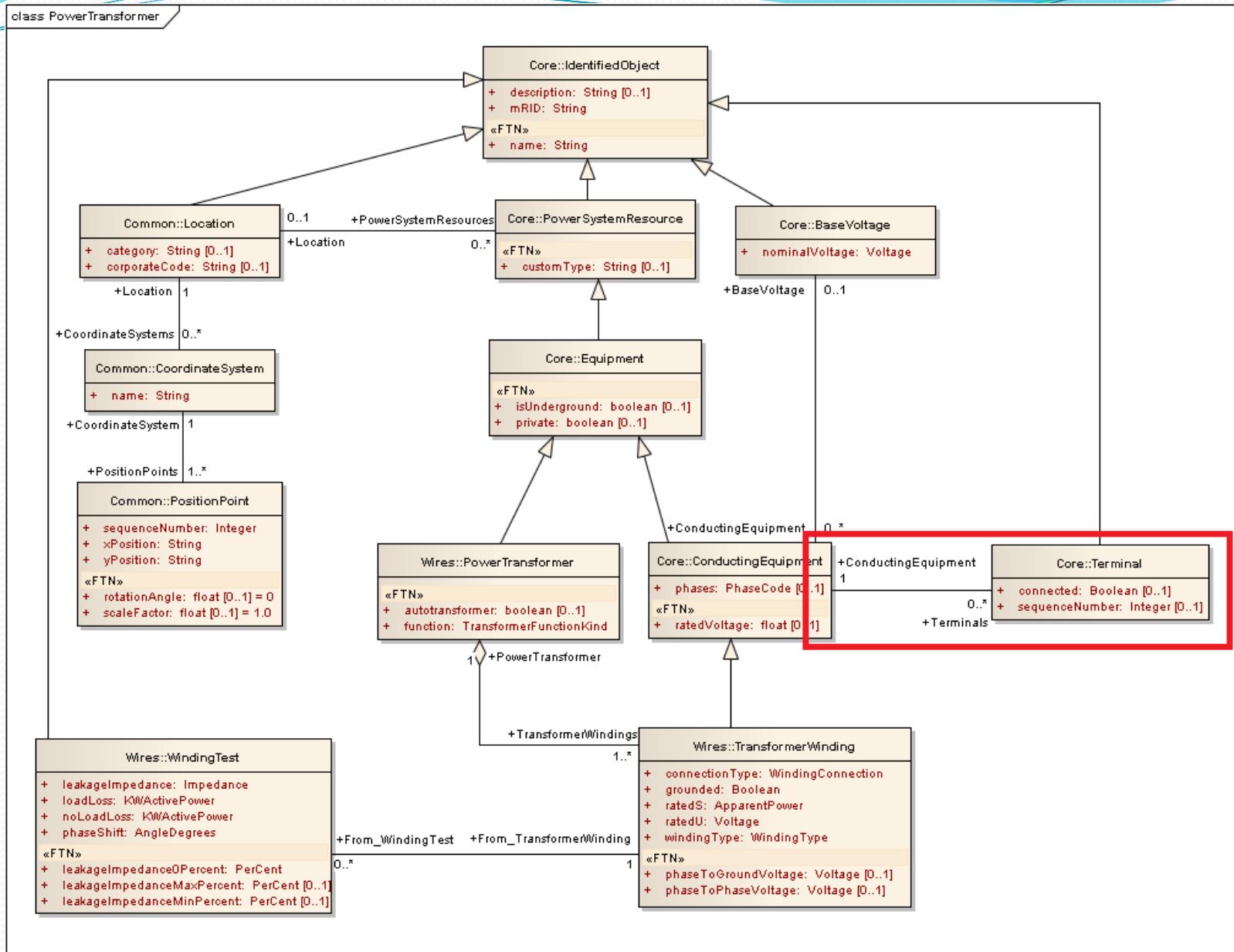
                io SetProperty(property);
            }
        }
        else
        {
            io SetProperty(property);
        }
    }
}

CommonTrace.WriteLine(CommonTrace.TraceVerbose, "Inserting entity with GID ({0:x16}) successfully finished.", globalId);
}
catch (Exception ex)

```

Rezime

- Prilikom dodavanja nove klase (ili atributa postojeće klase) potrebno je pratiti sledeće korake:
 1. Dodati odgovarajuće *ModelCode*-ove za nove klase i atribute
 2. Ukoliko su dodate nove enumeracije, proširiti *Enums.cs* fajl
 3. Ukoliko je neki od atributa tipa enumeracija proširiti *EnumDescs* klasu dodatnim mapiranjem
 4. Ukoliko imamo *notSettable* atributa proširiti njima *ModelResourcesDesc* klasu (doraditi metodu *InitializeNotSettablePropertyIds()*)
 5. Ukoliko je dodata konkretna klasa proširiti *ModelResourcesDesc* i *Container* klase (metode *InitializeTypeIdsInInsertOrder()* i *CreateEntity()*)
 6. Implementirati nove klase i po potrebi doraditi postojeće na osnovu prethodno definisanih pravila.



Zadaci

1. Dodati klasu *Terminal* koja ima referencu prema klasi *ConductingEquipment* (Napomena: Obratiti pažnju u koji paket se dodaje nova klasa i da su potrebne dodatne izmene nad klasom *ConductingEquipment*)