

UISOL

Utility Integration Solutions, Inc.

A Methodology for Managing CIM Extensions

Terry Nielsen, UISOL

Lee King, EPRI

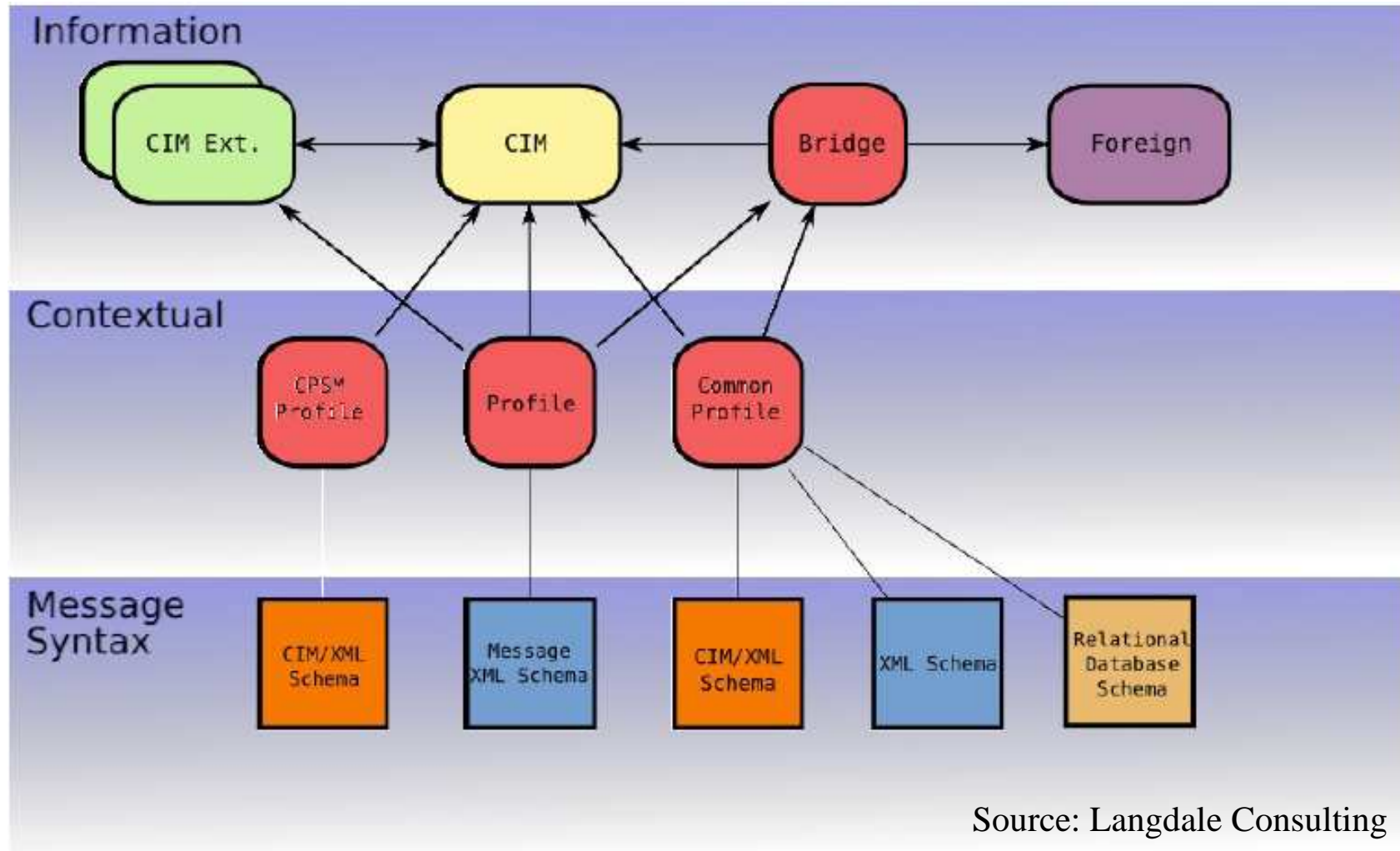
Scott Neumann, UISOL

July 30, 2009

Introduction

- The IEC Common Information Model (CIM) is commonly used for utility integration projects
- Any utility or vendor using the CIM will likely need to extend the CIM for any given project or product
- CIM itself is subject to change
 - New standard CIM versions are typically augmentations of previous versions
- Many approaches to extending the CIM, this presentation describes a flexible and practical approach using a combination of freely available and low cost tools

Models and Dependencies



Information Models and Extensions

- Need to develop or identify a base information model, this is usually a version of the IEC CIM
- Need to extend the model to address:
 - Vendor-specific and or proprietary features that require model extensions
 - Extensions that are widely useful and are good candidates for formal adoption in a future CIM version
 - Extensions required for local needs that are company or project specific
- There are two basic extension scenarios:
 - An attribute is added to an existing CIM class
 - New classes, with relationships and attributes are added

Extension Management

Several different extension techniques (and combinations thereof) have been used to date:

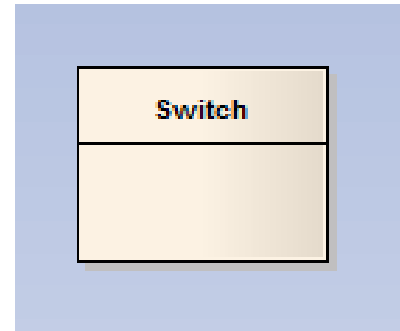
- Maintain a single logical model that includes extensions and is manually edited as needed to add extensions
- Placing extensions involving only new classes in a separate package
- Addition of a new class (containing extension attributes) with 0|1:1 association to existing class
- Tagging of extension elements
- Use of multiple inheritance, where a parent class identifies those classes that are extensions
- Tracking extensions in a database or spreadsheet
- ***Using an extension schema***

Overview of Extension Schemas

- Extensions are placed in a separate schema, using a UML tool where an XMI file can be generated
- More than one extension schema can be defined
- Each extension schema can have its own namespace
- CIM Schema and extension schema(s) are automatically merged when using CIMTool to define profiles
- Extensions do not need to be re-entered or re-merged when a new CIM release is available
- Extensions should not affect current implementations, so long as they are 'optional' (*as they should be for a logical information model*)
- Complete, end-to-end traceability can be achieved

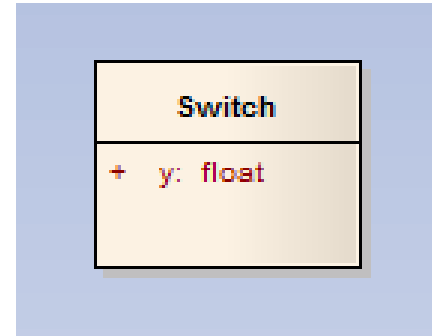
New Attribute to a CIM Class – Step 1

- In the extension model, add the class of interest
- For the purposes of this discussion, this is called a 'shadow' class
- The full definition of the shadow class remains in the base information model, with inheritance, associations and standard attributes



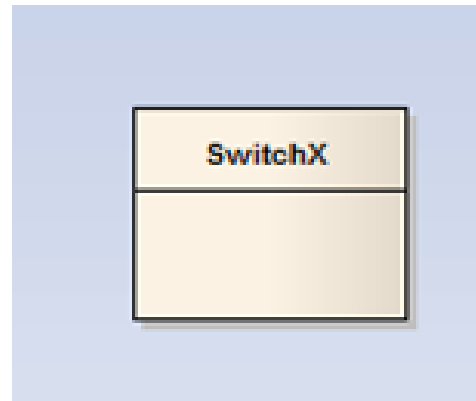
New Attribute to a CIM Class – Step 2

- Add the attribute to the shadow class
- Use CIM naming and type conventions



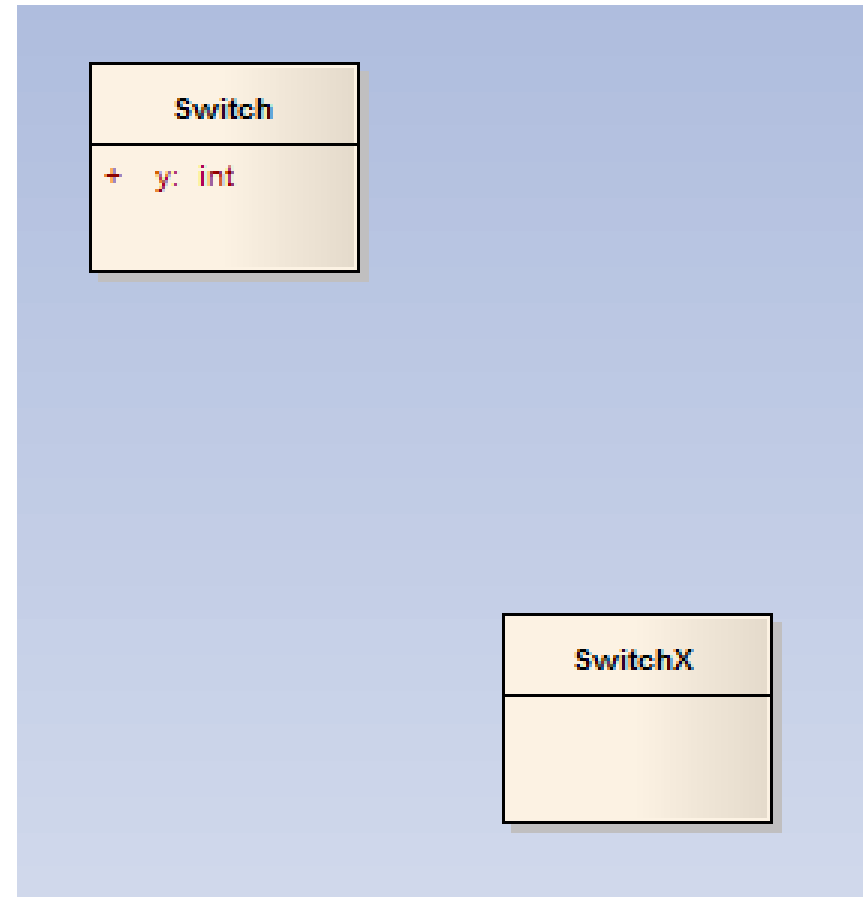
Creating a New Extension Class

- Open a package in an extension model
- Define the extension class with the desired name
- *This name obviously must not conflict with existing CIM class names*



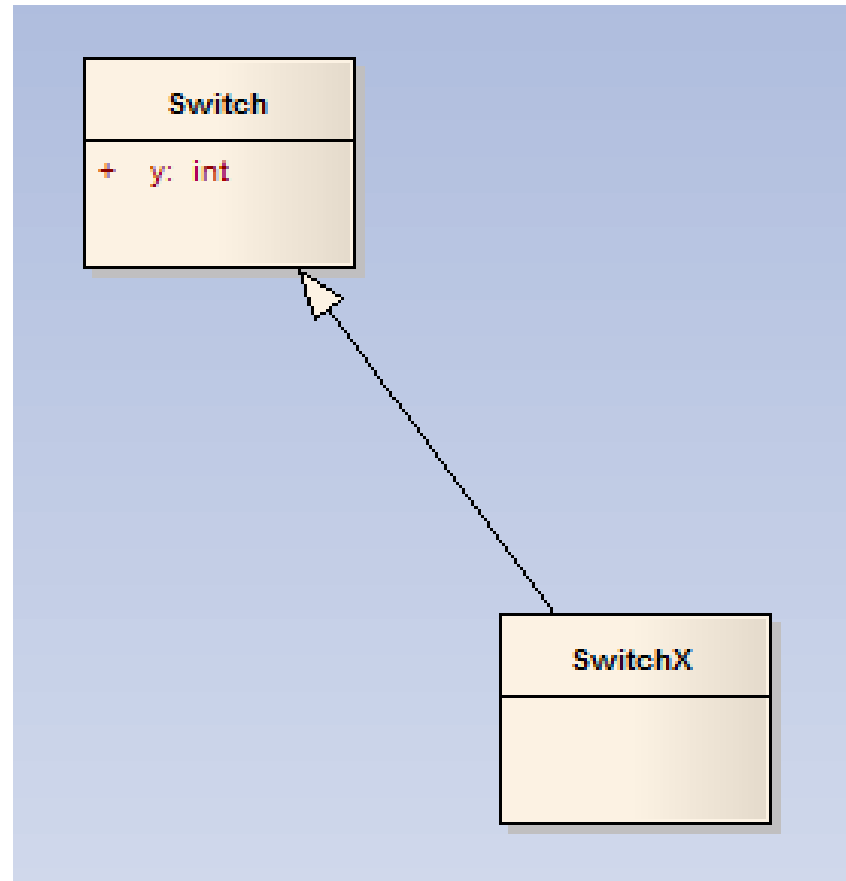
Defining Inheritance – Step 1

- Add the existing CIM class to the extension model that identifies the desired parent (or child) class (*note that this was already defined in the previous example*)
- This becomes a ‘shadow’ class in the extension model providing linkage to the CIM
- *Properties and relationships for the CIM class are maintained in the base information model*



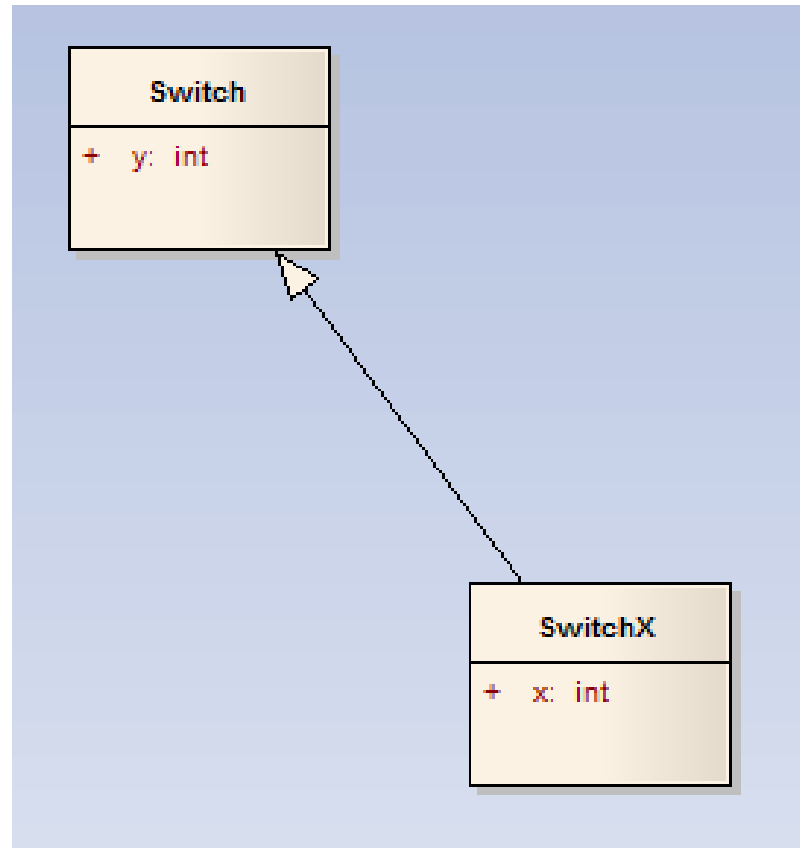
Defining Inheritance – Step 2

- Add the inheritance relationship from the extension class to the existing CIM class
- This could be either a new parent class or a new child class (*in this example the CIM class is a parent to the new extension class*)



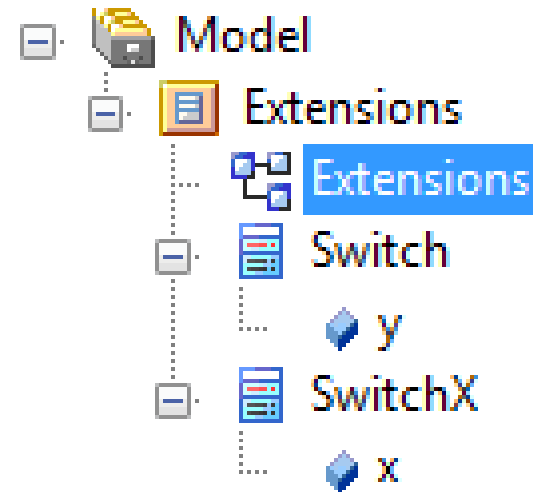
Adding Attributes to New Classes

- Add an extension attribute to the extension class, defining a name and type
- *This is the same procedure as adding attributes to CIM classes*



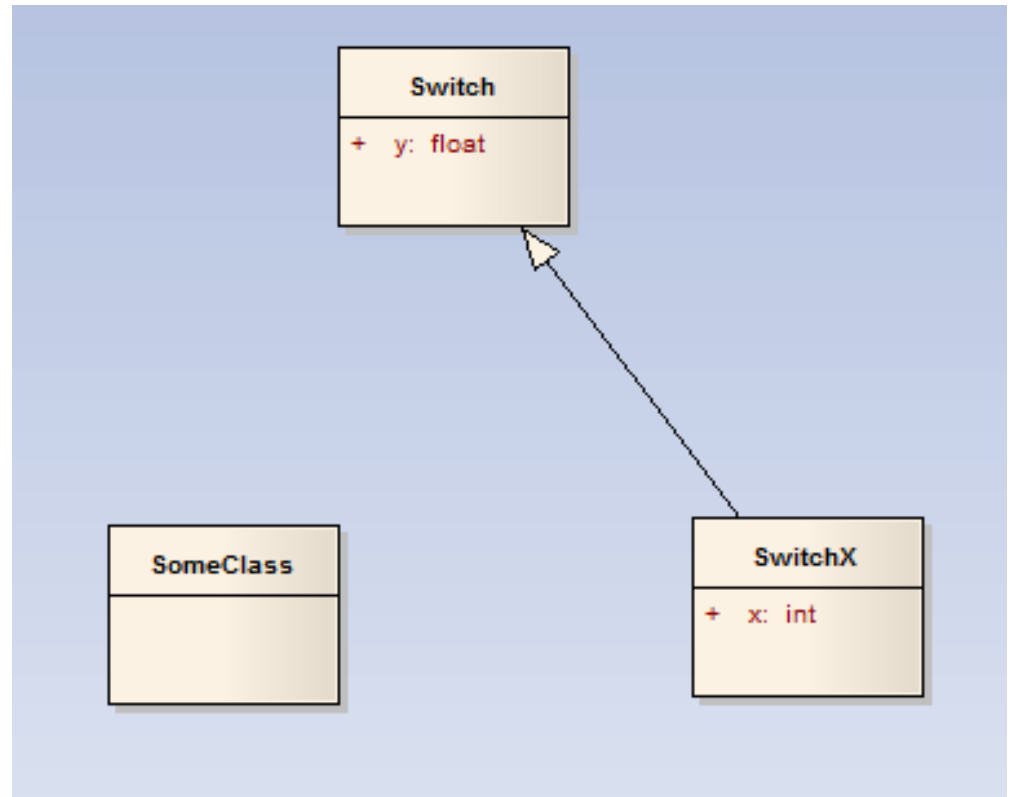
Extension Model ... so far

- CIM Switch was extended with attribute 'y'
- 'SwitchX' class created as a subclass of CIM Switch
- Attribute 'x' added to SwitchX



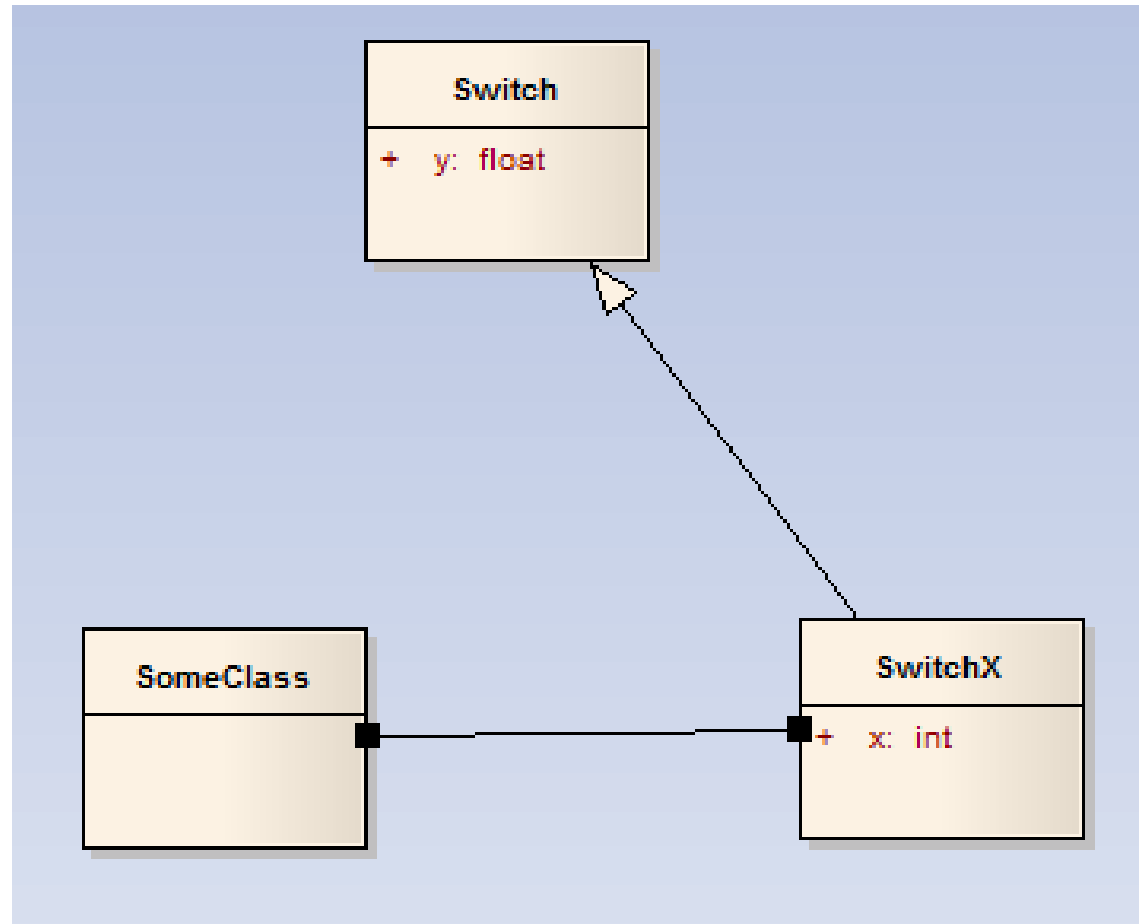
Adding a Relationship – Step 1

- Class to be related is added to the extension model
- This may be a new extension class or an existing CIM class
- *If a new class, there would typically be the need to also include a CIM parent class (e.g. IdentifiedObject) and an inheritance relationship*



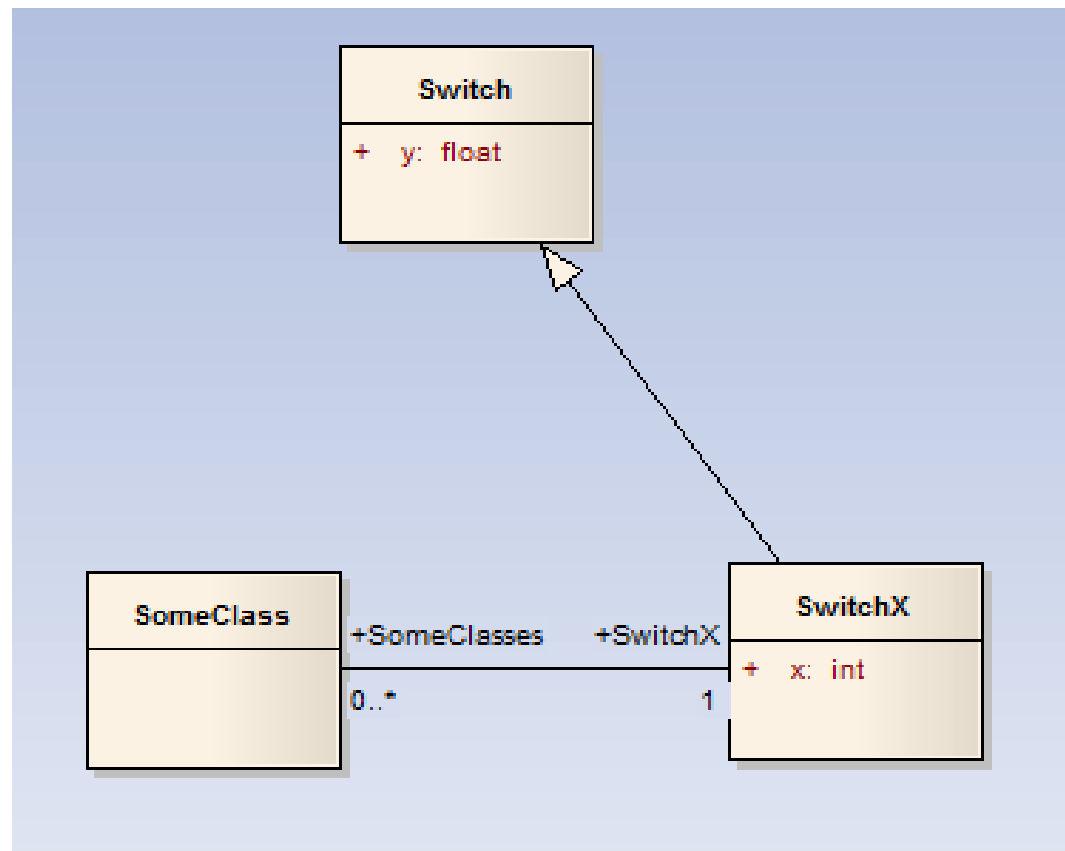
Adding a Relationship - Step 2

- Draw the relationship between the classes
- Could be inheritance, association or aggregation
- Relationship may be between existing or new classes

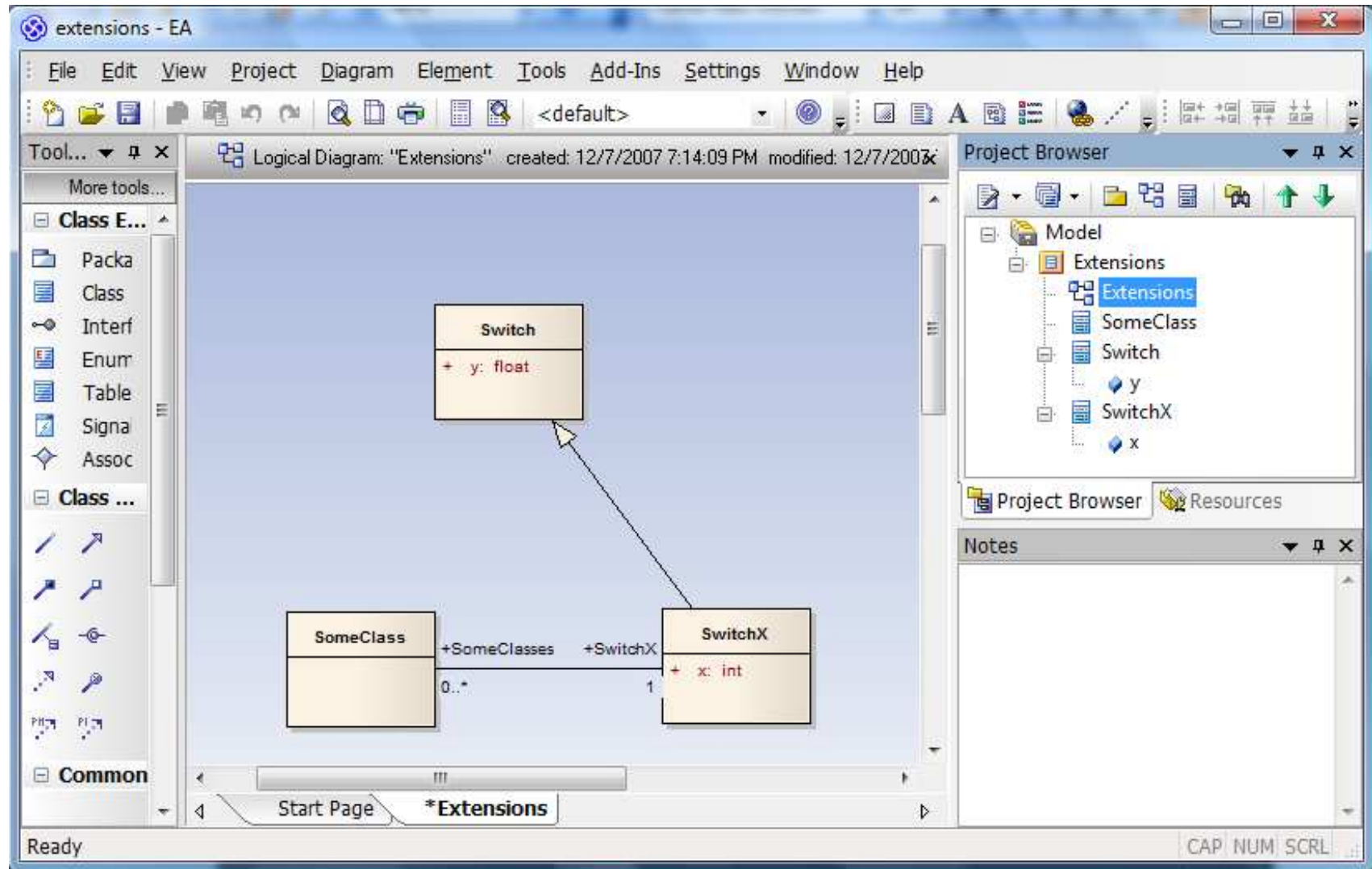


Adding a Relationship – Steps 3, 4

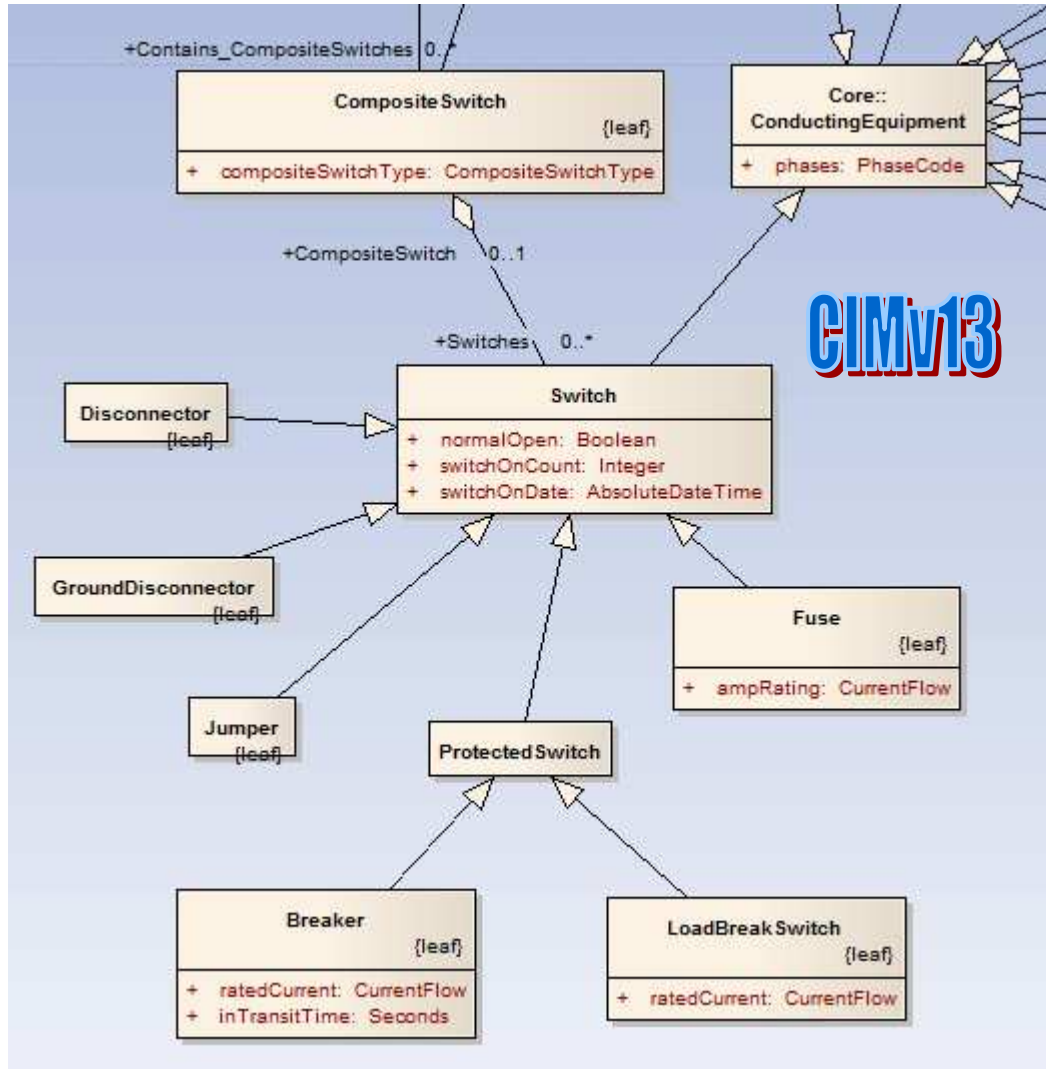
- If the relationship is an association or aggregation, name each end (role) of the association using CIM conventions (e.g. class name in singular or plural form)
- Define multiplicities



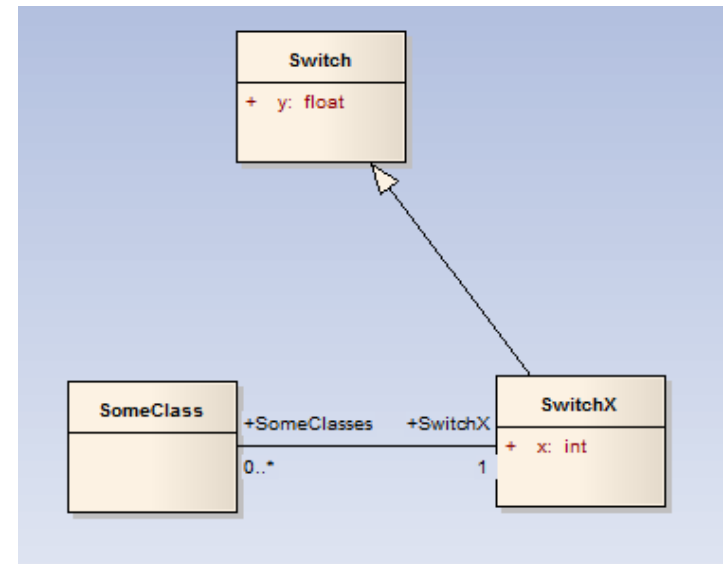
View within Modeling Tool



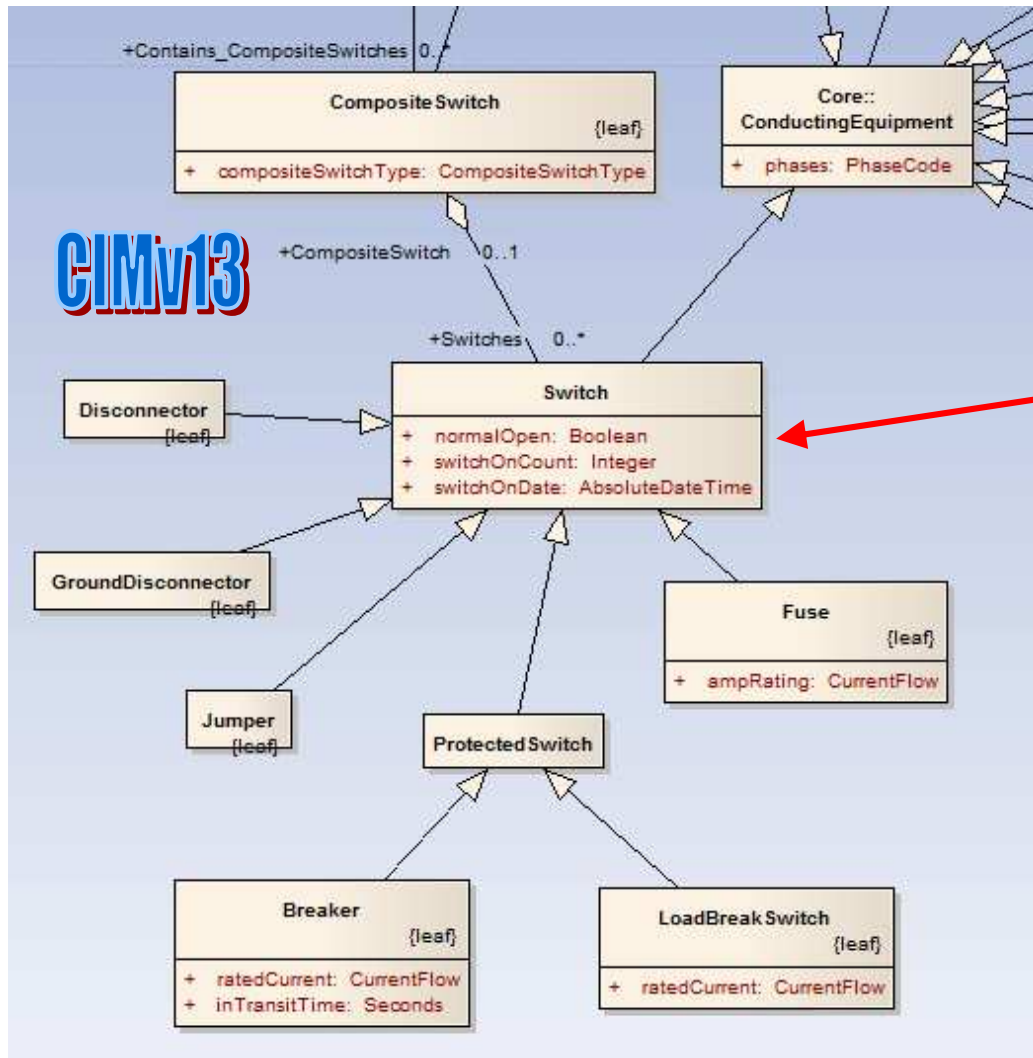
CIM + Extensions



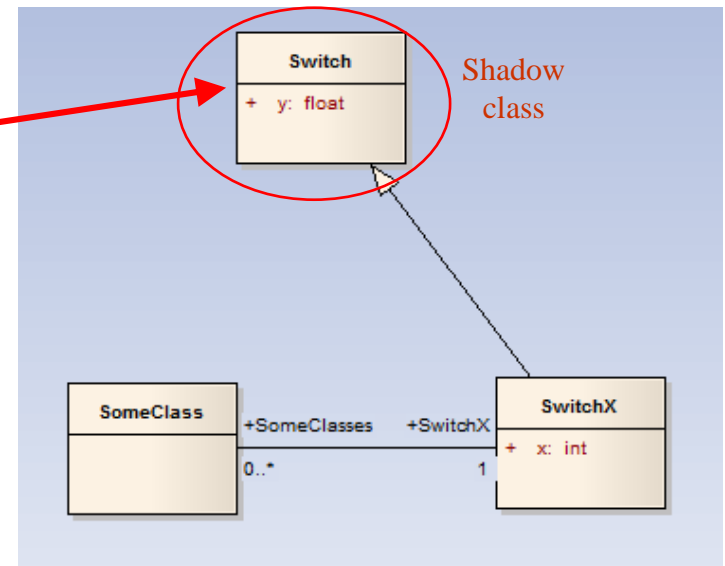
Extension Model



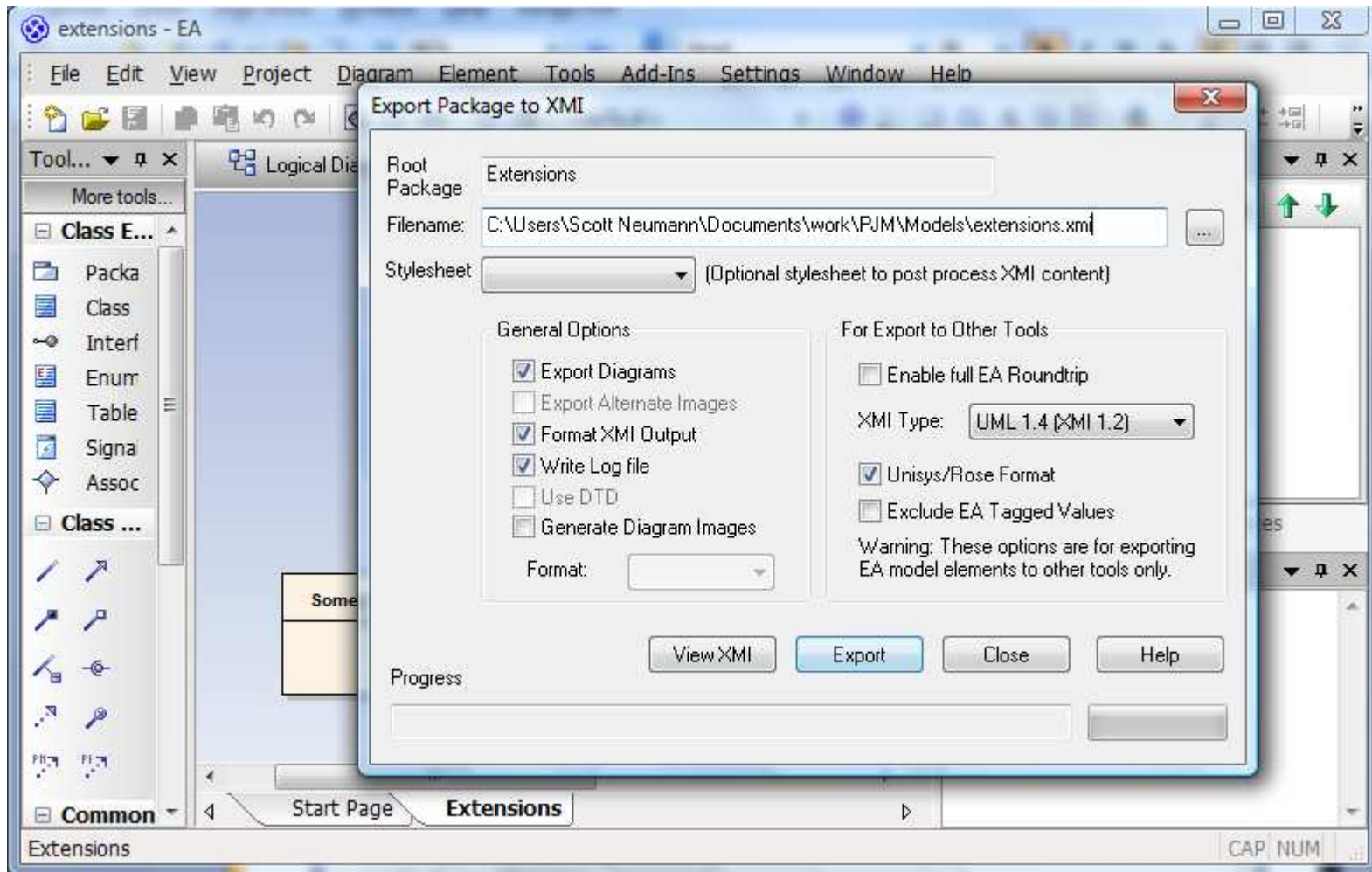
CIM + Extensions



Extension Model



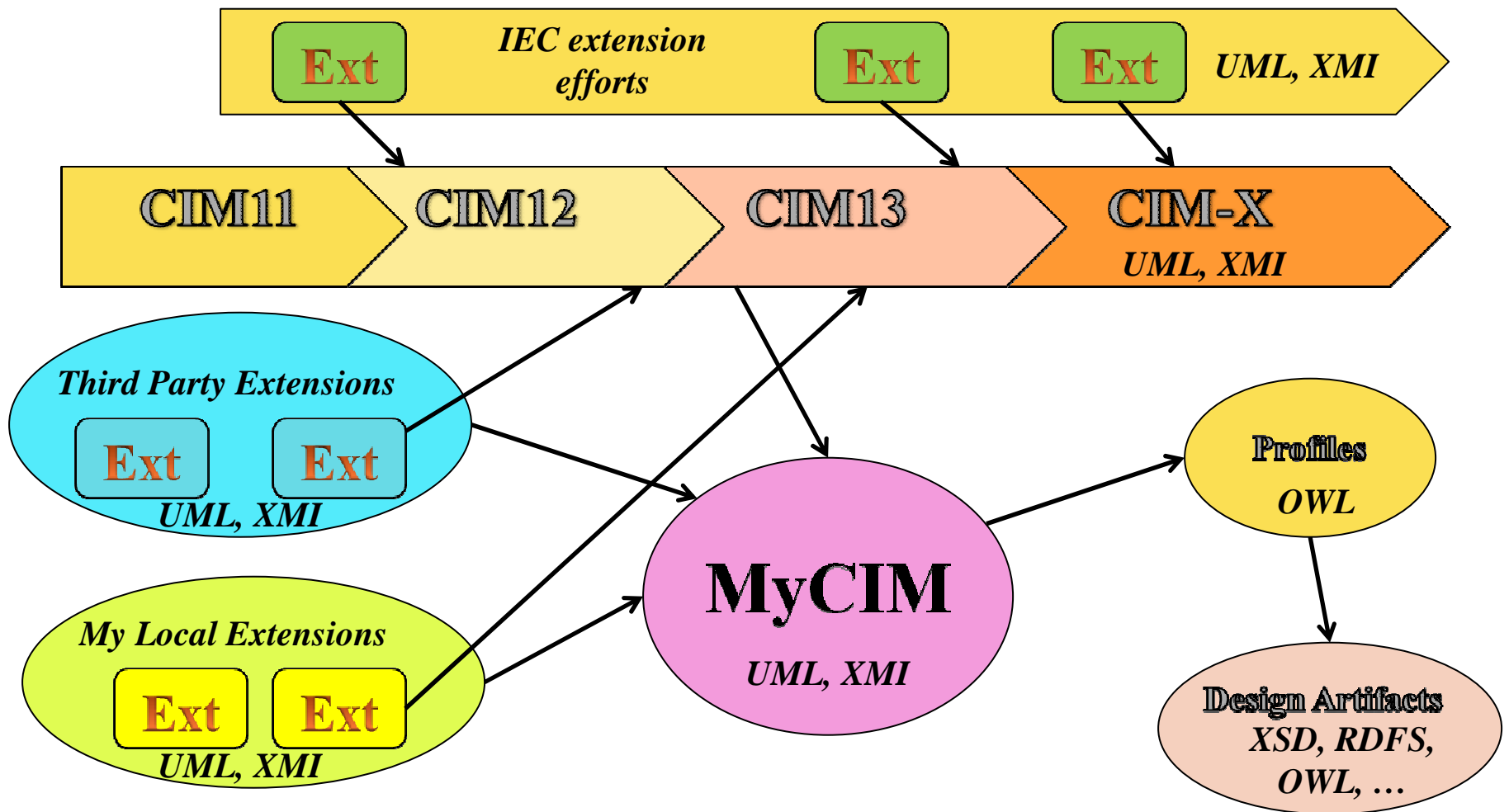
EA: Exporting the Extension Model



Impact of New CIM Versions

- From time to time a new CIM version will be available from the CIM Users Group web site
- There may be more than one source of extensions (e.g. standards efforts, vendors, project needs)
- Extension models would typically be compatible with new CIM versions
- Sometimes a new CIM version could impact extension models, as in the case where an extension model was directly adopted or provided the basis for an extension to the standard (*Typically all that would be needed is the deletion of classes, attributes or relationships from the extension model in cases where the class, attribute or relationship had been adopted by the new standard CIM release*)

MyCIM = CIM + Extensions



Merging Models

- Need the ability to either logically or physically merge one or more extension models together or to merge with a selected base model
- The shadow classes in the extension model provide the logical linkage points between the base and extension models
- Options for merging:
 - This can be done ‘logically’ using CIMTool, where the merging of the models occurs automatically when an extension model is loaded
 - This can be done ‘physically’, where an extensions are merged with a base model to generate a single model file that is the composite of the base and extensions
- In either case, the models should be in an XMI format before and after the merge

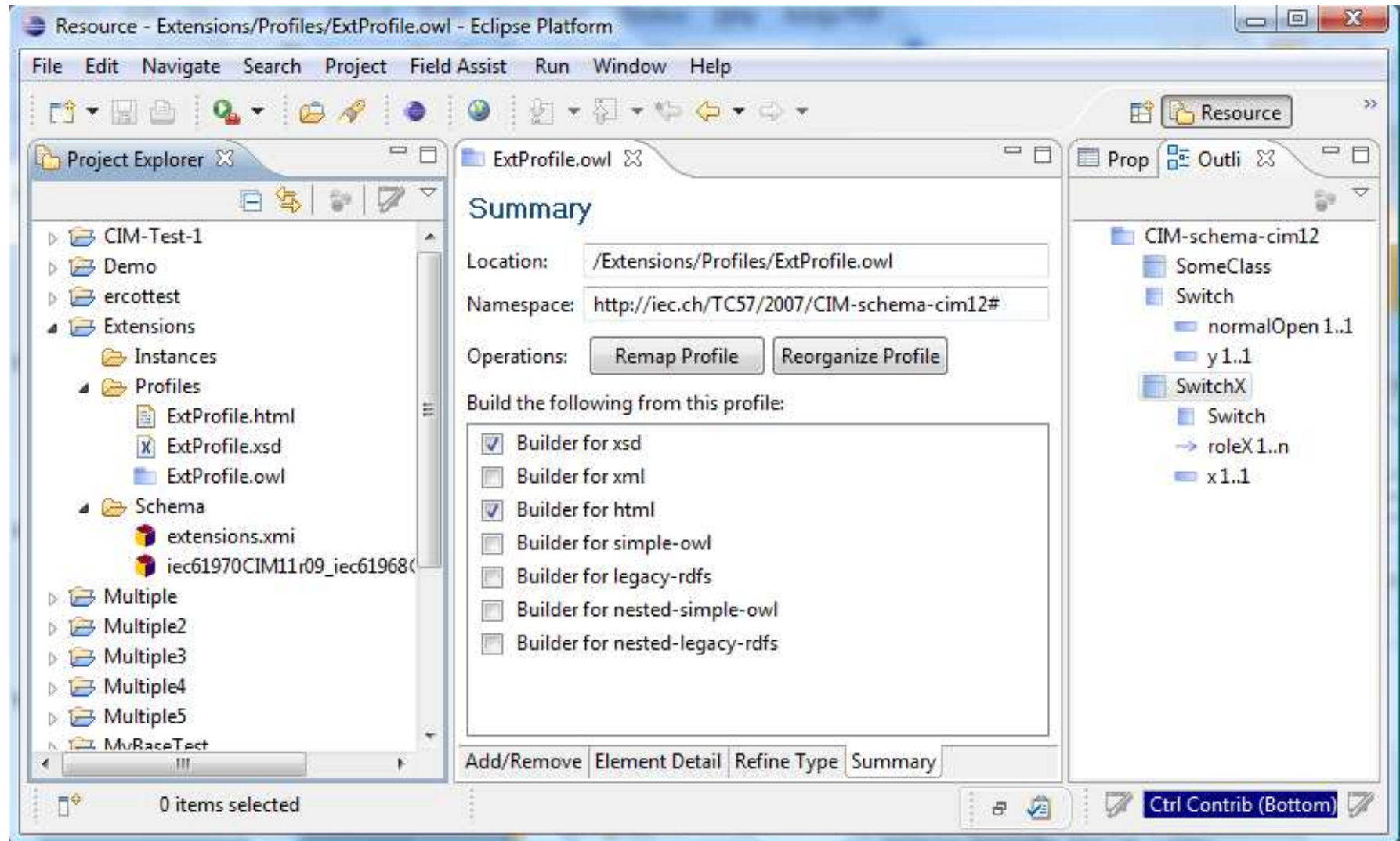
Contextual Modeling

- Next step in the process is to use the model and extensions to develop profiles
- Profiles are used to define the content of information exchanges
- Profiles identify a formal subset of the logical information model and its extensions with added constraints
- CIMTool is the recommended tool for contextual modeling

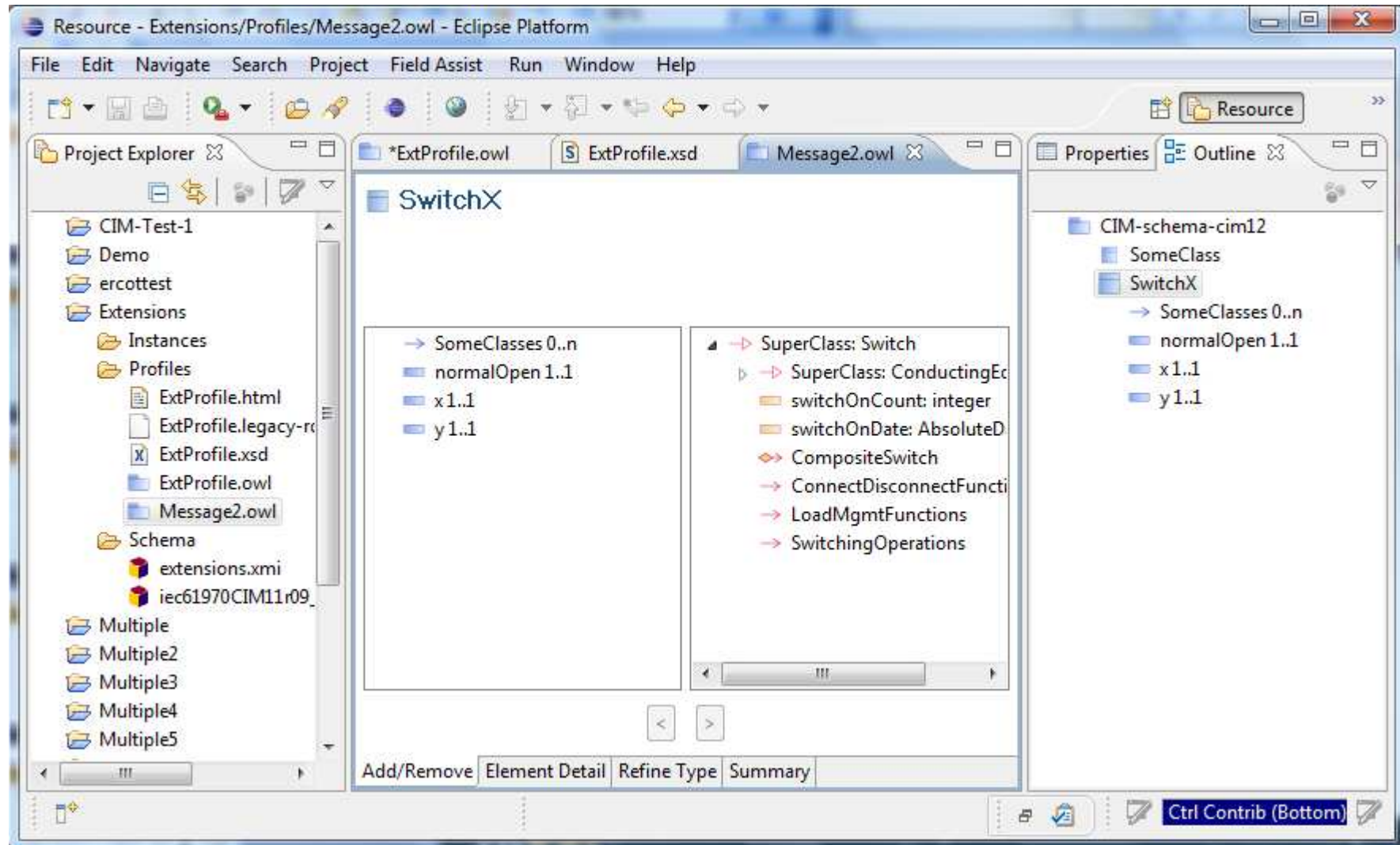
Building Profiles using Extensions

- Profiles are built using CIMTool
- CIMTool is freely available as an Eclipse plug-in
- High level steps:
 - Create a new CIMTool project, specifying a CIM XMI Schema file as a base model
 - Import additional Extension Schema files as needed
 - Create profiles as needed to describe information exchanges

CIMTool: Using Extension Schema



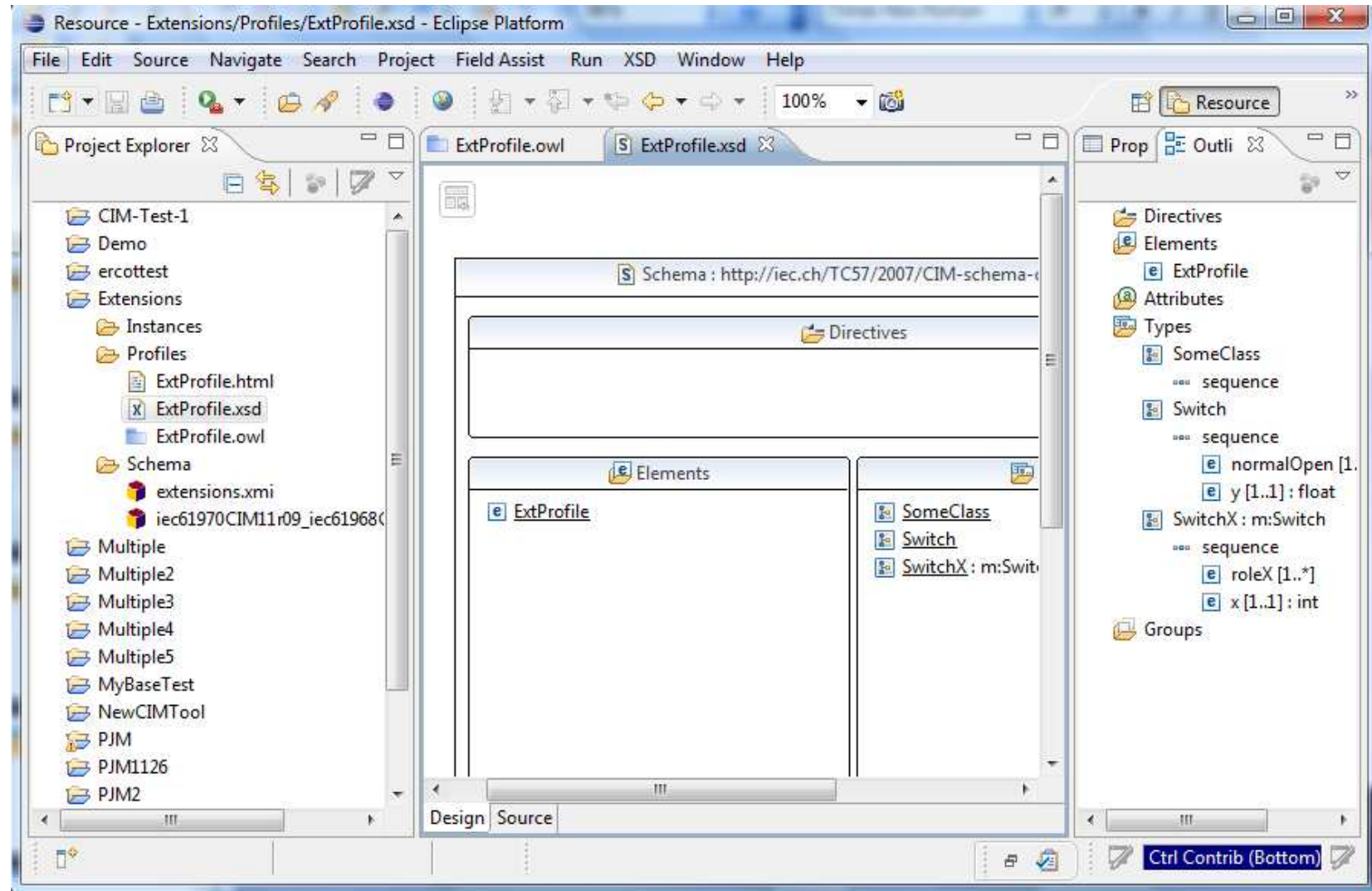
CIMTool: Building Profiles



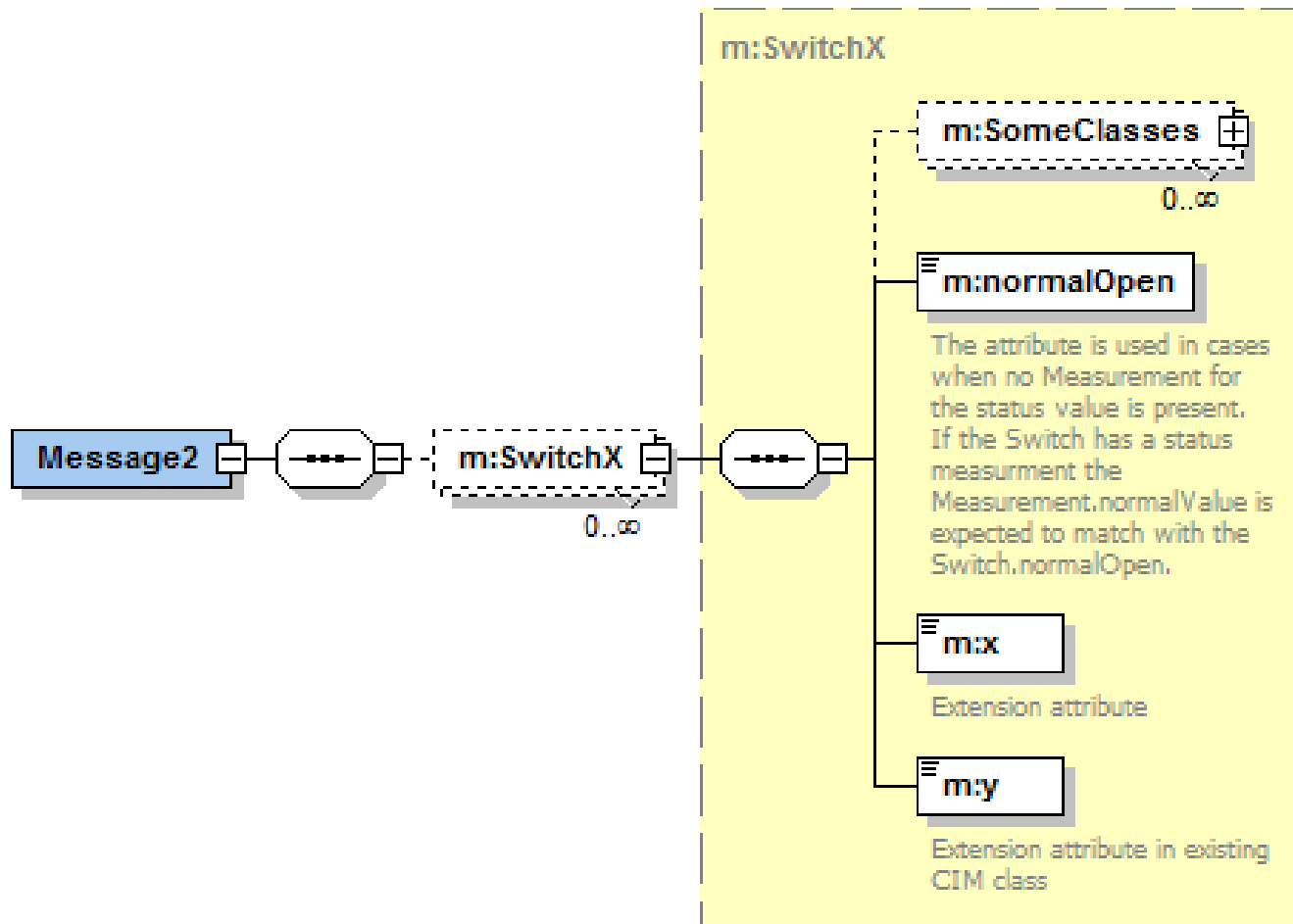
Message Syntax

- Assembly rules are used to construct 'artifacts' from profile definitions
- Message syntax can be in the form of:
 - XML Schema (XSD)
 - RDF
 - OWL
 - HTML
 - XML
- Using style sheets, many more forms are possible (e.g. Database DDL, or alternative XSD realizations)
- CIMTool regenerates the artifacts whenever the associated profile is modified and saved

Resulting XML Schema in Eclipse



(Same) XML Schema



Further Research

Useful extensions to current capabilities would include functionality to:

- Manage a 'source' namespace for each information model schema (base and extension files)
- Identify source namespace in annotations within resulting XML Schemas to provide traceability (*still only want a single 'target' namespace for XSDs realized from profile definitions*)
- Leverage version management capabilities provided by Eclipse
- Provide capabilities for physically merging extension schemas with base model (*useful for standards efforts*)

Benefits

- Can be done using a combination of the freely available Eclipse-based CIMTool and a variety of low cost UML tools
- Provides the means to simply define and describe extensions in a 'contained' manner, as opposed to a set of changes dispersed throughout a large UML model
- Much simpler to describe extensions than the current CIM change/issue spreadsheets, while at the same time providing a graphical description of the extension
- Provides the means to support and promote parallel extension efforts
- It is possible to 'pick and choose' extensions of interest from other sources, prior to formal inclusion within the CIM