XCASE

User’s Guide

v 1.0

Table of Contents

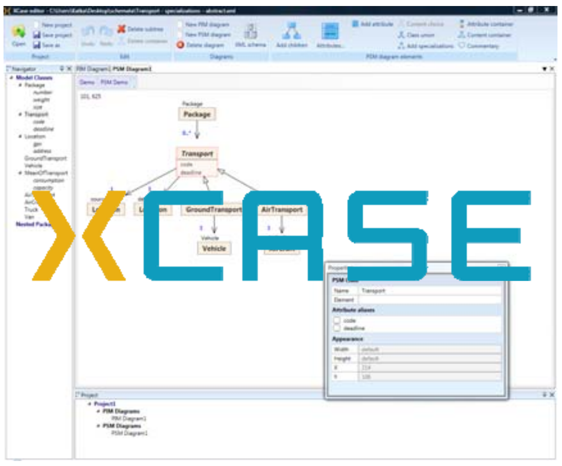
1. Introduction

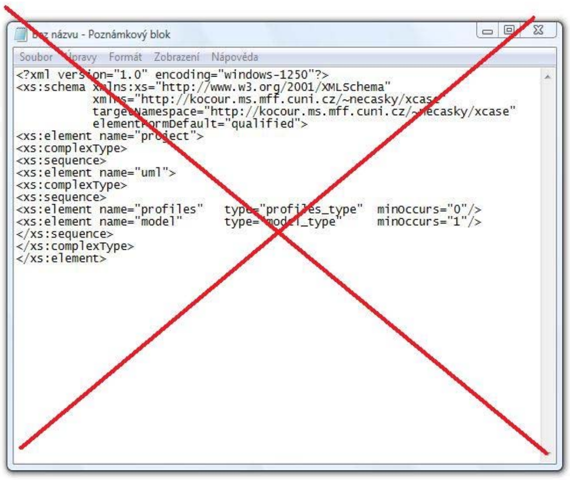
1.1.What is XCase

Conceptual modeling was successfully applied in the world of relational databases. It enables one to describe a problem domain in a way abstracted from particular user views and representations in various data models. Hence, it is a challenge to apply conceptual modeling to XML data model in a similar way. Since XML has some special features in comparison to, e.g. relational data model, current conceptual modeling approaches cannot be applied directly. We therefore need to augment them to fit requirements of the XML community.

In this project, we have developed a case tool for conceptual modeling for XML. The tool is called *XCase* and implements a new conceptual model for XML called *XSEM* [1]. The tool can be used for a comfortable, visually guided design of XML schemas at the conceptual level instead of writing these schemas by hand.

XSEM is based on the *Model-Driven Architecture* (MDA) [2] as it separates the conceptual modeling process to two levels. At the first level, you create a conceptual diagram in a *Platform-Independent Model* (PIM) that provides an XML-independent description of your problem domain. At the second level, you create one or more diagrams in a *Platform-Specific Model* (PSM) on the base of the PIM diagram. At this level you specify representations of your data in target XML formats. Finally, you can derive an XML schema from each PSM diagram. In the current version of XCase, XML Schema [3] is applied to express XML schemas.




Bez názvu - Poznámkový blok
Soubor Úpravy Formát Zobrazení Nápověda

```
<?xml version="1.0" encoding="windows-1250"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://kocour.ms.mff.cuni.cz/~necasky/xcase"
           targetNamespace="http://kocour.ms.mff.cuni.cz/~necasky/xcase"
           elementFormDefault="qualified">
<xs:element name="project">
<xs:complexType>
<xs:sequence>
<xs:element name="uml">
<xs:complexType>
<xs:sequence>
<xs:element name="profiles"   type="profiles_type"  minOccurs="0"/>
<xs:element name="model"      type="model_type"     minOccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

1.2. XCase Overview

XCase enables one to model XML data at PIM and PSM level as required by XSEM. It offers a standard environment commonly offered by other tools for conceptual modeling such as ER or UML class diagram editors.

Platform-Independent Model Level (PIM)

PIM enables one to design conceptual diagrams describing the problem domain independently of intended representations in various XML formats. As PIM, XSEM applies the well-known UML class model [4,5]. A sample PIM diagram is displayed in *Figure 1*.

The main XCase features at the PIM level are the following:

- Designing a model of the problem domain
- Structuring the model to packages
- Visualizing the model in various PIM diagrams
- Editing the model through a model navigator or PIM diagrams
- Support of all well-known UML constructs:
    - Classes (with attributes and operations), binary associations (aggregations, compositions), n-ary associations, association classes and generalizations

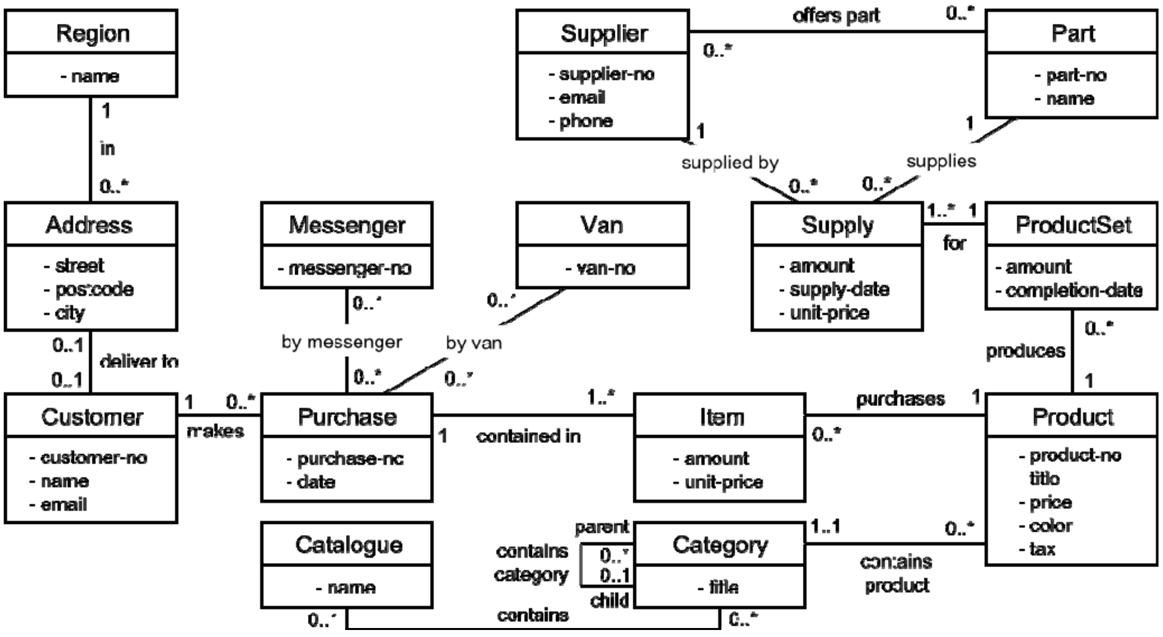


Figure 1: Company PIM diagram

Platform-Specific Model Level (PSM)

PSM enables one to describe how the data modeled by PIM diagrams is represented in various XML formats. As PSM, XSEM applies the UML class model extended with constructs for modeling special XML features such as, e.g. representing the data as XML elements vs. attributes, hierarchical and irregular structure, and ordering.

Each PSM diagram models a particular XML format. It contains classes from PIM diagrams and organizes them into a hierarchy by binary associations directed from the parent to child. Briefly, a PSM class models an XML element (its name is given by an *element label* displayed above the class). Each its attribute models an XML attribute and each its child models a child XML element. The extending constructs allow further augmenting this basic hierarchical representation as follows:

- *Attribute container.* Contains attributes of a PSM class and specifies that these attributes model XML elements instead of XML attributes.
- *Content choice.* Models variants in the content of a PSM class.
- *Content container.* Models an XML element that has no semantic equivalent at the PIM level (i.e. no equivalent PIM class)
- *Class union.* Models a mixture (union) of instances of two or more PSM classes.

After a PSM diagram is completed, it can be mechanically expressed in an XML schema language that describes the XML format at the logical level. Since the problem domain is modeled at the PIM level, PSM diagrams are derived from the PIM diagrams. Two sample PSM diagrams are depicted in *Figure 2.*

The main XCase features at the PSM level are the following:

- Designing basic XML representations by derivation from PIM diagrams
- Support of the extending constructs proposed in [1] to model more complex XML representations. In a concrete the constructs are
  - o Attribute container, content choice, content container and class union
- Automatic translation of PSM diagrams to a representation in XML Schema

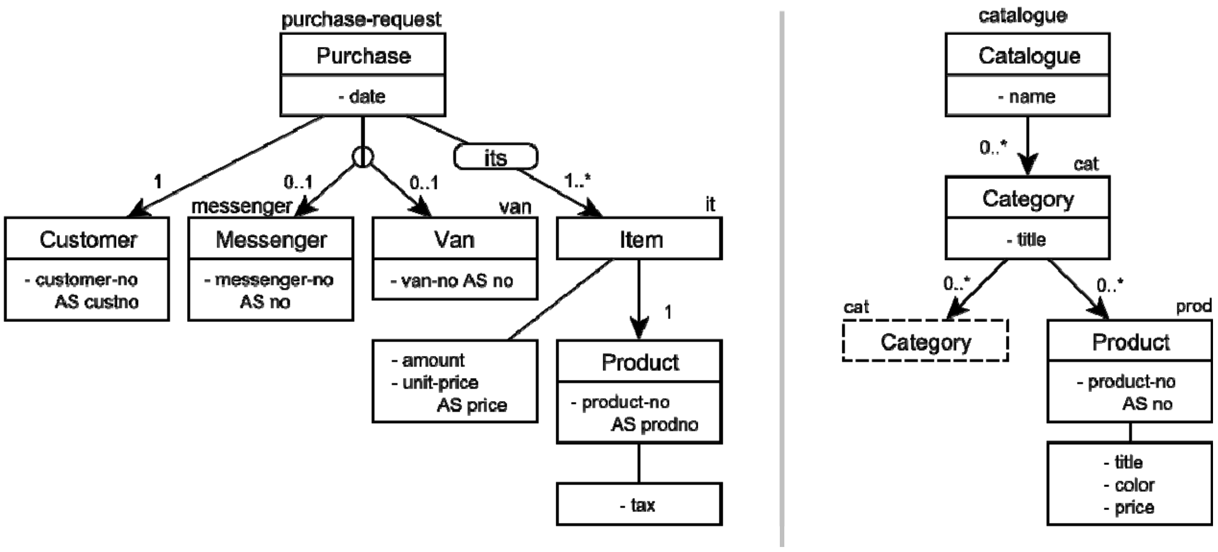


Figure 2: Purchase and Catalogue PSM diagram

Transition from PIM level to PSM level

In addition to common features provided by UML class diagram editors, the tool puts the accent on the transition between PIM and PSM diagrams. This transition cannot be performed automatically as we show in [1]. It must be driven by the designer and the tool provides assistance for the transition.

The main XCase PIM to PSM transition features include:

- Deriving PSM classes and PSM associations from PIM classes and paths in PIM diagrams, respectively, on the base of designers’ requirements
- More PSM diagrams can be derived from the same part of the model
- Checks of consistency between PIM and derived PSM diagrams.
    - Dynamic propagation of changes in PIM diagrams to corresponding PSM diagrams and back

Translation of PSM diagrams to XML schemas

Each PSM diagram models an XML format that represents a particular user view on the problem domain. The XML format is described at the logical level by an XML schema that is mechanically generated from the PSM diagram. We describe the details of the translation later in this documentation.

To demonstrate the power of the proposed PSM, we show two sample XML documents of the XML formats modeled by the above sample PSM diagrams. We do not explain the XML documents in detail since they are self-descriptive. Their structure directly conforms to the structure of the PSM diagrams.

```
<purchase-request
  date=”2008/12/05” custno=”C8212”>
 <messenger no=”M211” />
 <its>
  <it prodno=”P32”>
   <amount>5</amount>
   <price>435</price>
   <tax>19</tax>
  </it>
  <it prodno=”P821”>
   <amount>1</amount>
   <price>9182</price>
   <tax>19</tax>
  </it>
 </its>
</purchase-request>
```

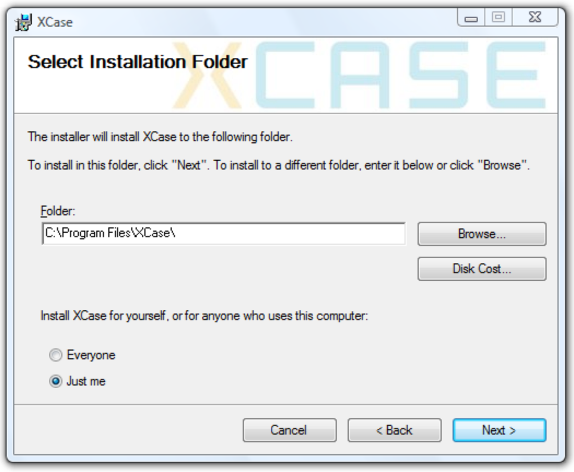
```
<catalogue name=”Winter 2008”>
 <cat title=”C1”>
  <cat title=”C2”>
   <product no=”P32”>
    <title>P32</title>
    <color>white</color>
    <price>435</price>
   </product>
  </cat>
  <cat title=”C3”>
   <product no=”P32”>
    <title>P32</title>
    <color>white</color>
    <price>435</price>
   </product>
  </cat>
 </cat>
</catalogue>
```

2. XCase Installation

2.1. Installation Wizard

To install XCase editor on your PC, run *setup.exe* from the enclosed XCase CD. The installation wizard then guides you through the installation process.

If you do not already have Windows Installer 3.1 and .NET Framework 3.5 SP1 installed, you need to be **connected to the Internet** while installing XCase. New or missing .NET Framework parts will be downloaded during the installation process.



The installer will install XCase to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files\XCase\

Install XCase for yourself, or for anyone who uses this computer:

Everyone [✓ Just me]

1] Click *Next* to begin the installation process

2] Enter the name of the installation folder on your computer

3] Click *Next* to complete the installation process

4] Click *Close* to end the installation wizard.

2.2. System Requirements

- Microsoft Windows XP SP2 or newer
- For off-line installation: .NET Framework 3.5 SP1, Windows Installer 3.1

The installation of XCase needs about 20MB of free hard disk space. Since XCase runs on .NET framework, your computer should meet its requirements for a proper run.

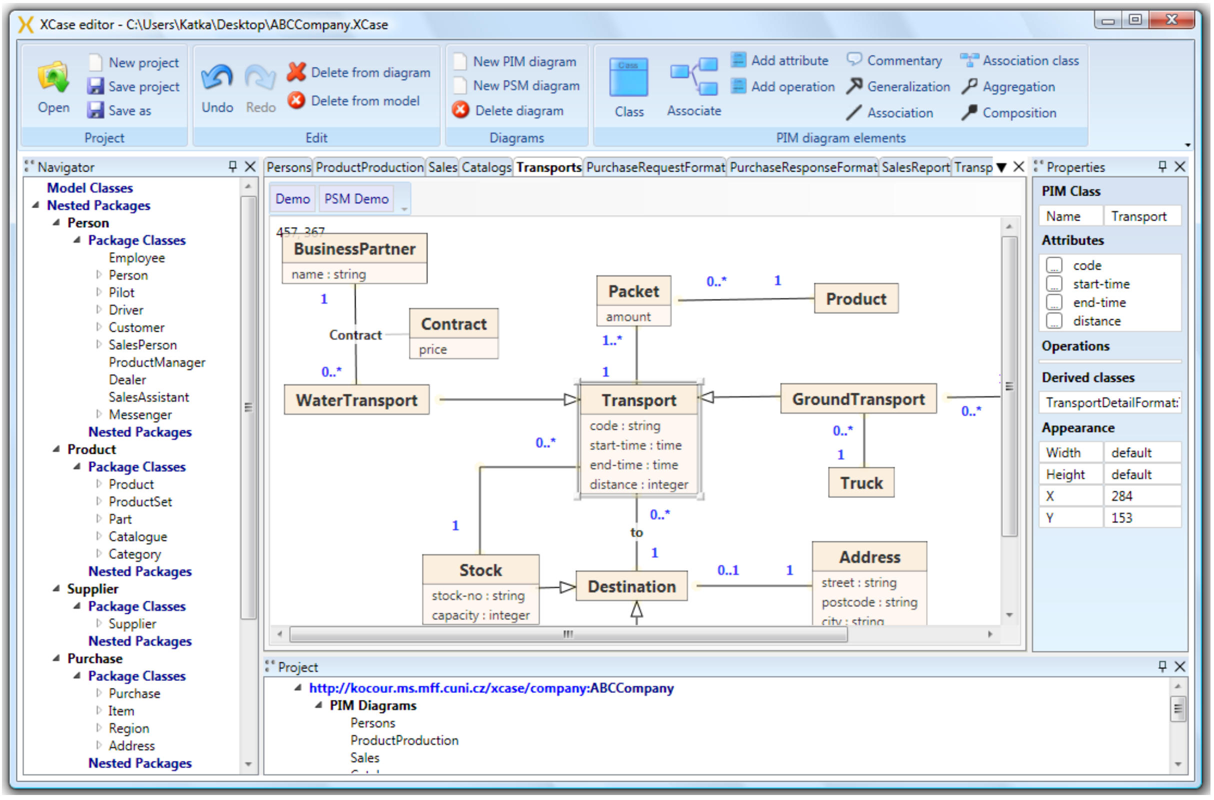
.NET Requirements:

- Processor: 400 MHz Pentium processor or equivalent (Minimum);
  1,6GHz Intel Core 2 Duo processor or equivalent (Recommended)
- RAM: 96 MB (Minimum); 2 GB (Recommended)
- Hard Disk: Up to 500 MB of available space may be required

3. XCase GUI Overview

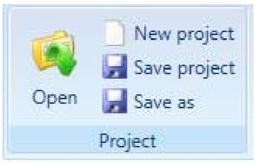
XCase editor comprises the following basic components:

- Main toolbar
- PIM and PSM diagram canvases
- Project window
- Navigator window
- Properties window



3.1. Main Toolbar

Main toolbar offers tools needed for modeling at both PIM and PSM levels. These tools are divided into several sections. Visibility of some sections depends on the type (PIM or PSM) of the active diagram selected.

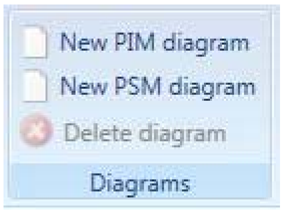


The sections available for both PIM and PSM diagrams are Project, Diagrams, Edit and Windows.

- **Project**

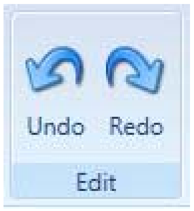
o **New Project** (CTRL+N) - Creates a new XCase project with one empty PIM diagram
o **Open Project** (CTRL+O) - Opens a previously saved XCase project
o **Save Project** (CTRL+S) - Saves changes in the current XCase project
o **Save As** (SHIFT+S) - Saves the current XCase project as another one

- **Diagrams**



  - **New PIM Diagram**
    Adds a new empty PIM diagram to the project
  - **New PSM Diagram**
    Adds a new empty PSM diagram to the project
  - **Delete Diagram**
    Deletes an active PIM or PSM diagram from the project

- **Edit**



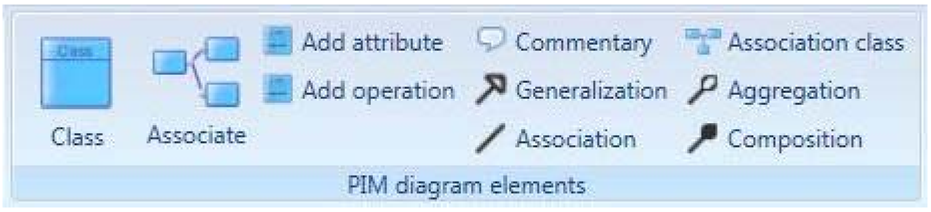
  - **Undo** - Undoes the last executed command
  - **Redo** - Redoes the last undone command

- **Windows**
  - **Projects -** Displays the Projects window in GUI if not visible before
  - **Properties -** Displays the Properties window in GUI if not visible before
  - **Navigator -** Displays the Navigator window in GUI if not visible before

If a PIM diagram is active, main toolbar offers the following additional sections:

- **Edit (partially)**
  - **Delete from diagram** (DELETE) - Deletes the currently selected component from the PIM diagram (it is still present in the model after deleting)
  - **Delete from model** (SHIFT+DELETE) - Deletes the currently selected component from the model (it is removed from the model as well as all diagrams)

- **PIM diagram elements**

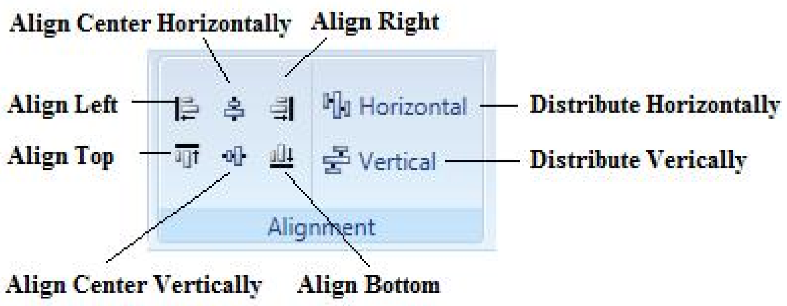


  - **Class** – Creates a new PIM class in the model and adds its visualization to the PIM diagram
  - **Associate** – Associates the currently selected PIM classes
  - **Add attribute –** Adds an attribute with a default name to the currently selected class
  - **Add operation –** Adds an operation with a default name to the currently selected class
  - **Commentary –** Adds a new floating commentary to the PIM diagram
  - **Association class -** Associates the currently selected PIM classes by an association class

- **Generalization, Association, Composition, Aggregation**
  When toggled, these buttons enable one to create generalizations | associations | compositions | aggregations between two classes. You click on a class in the PIM diagram and drag a connecting line to another class in the PIM diagram.

- **Derive**
  - **Derive to new PSM diagram –** Derives a new PSM class from the currently selected PIM class. The new PSM class is inserted into a new PSM diagram.
  - **Derive to existing diagram –** Derives a new PSM class from the currently selected PIM class. The new PSM class is inserted into an existing PSM diagram. If there are more existing PSM diagrams you are asked to select one.

- **Alignment**
  Alignment buttons become available when two or more classes in the PIM diagram are selected



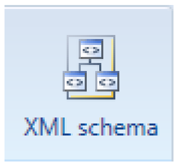
  - **Align Left –** Aligns the classes to the most left one
  - **Align Center Horizontally -** Aligns the classes to the horizontal middle one
  - **Align Right** Aligns the classes to the most right one
  - **Align Top -** Aligns the classes to the top one
  - **Align Center Vertically-** Aligns the classes to the vertical middle one
  - **Align Bottom -** Aligns the classes to the bottom one

  - **Horizontal** (Distribute Horizontally) - Uniform horizontal distribution of the classes
  - **Vertical** (Distribute Vertically) - Uniform vertical distribution of the classes

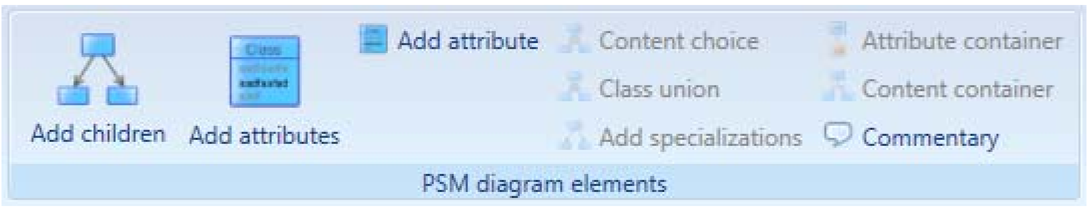
If a PSM diagram is active, main toolbar offers these additional sections

- **Edit (partially)**
  - **Delete subtree –** Deletes the entire visual sub-tree of the currently selected PSM class or container
  - **Delete container –** Deletes just the currently selected PSM container (content container, content choice, class union or attribute container) from the PSM diagram. The content of the container is not deleted.

- **Diagram (partially)**
  - **XML Schema –** Generates an XML schema from the currently active PSM diagram. The XML schema is displayed in a new window and can be saved to a file.



- **PSM Diagram elements**



  - **Add children** – Adds child classes under the currently selected PSM class
  - **Attributes** – Adds attributes to the currently selected PSM class. Only the attributes of the related PIM class are offered to add

  - **Add attribute** – Adds a new attribute, which has no relation to the PIM model, to the currently selected PSM Class. We call such an attribute *PIM-less*.
  - **Content Choice** – Encloses the currently selected neighboring PSM associations to a content choice
  - **Class Union –** Unifies two or more neighboring PSM associations
  - **Attribute Container** – Moves some attributes from the currently selected PSM class to a new attribute container.
  - **Content Container** – Encloses the currently selected neighboring PSM associations to a content container
  - **Add specifications –** Adds specializations of the currently selected PSM class according to specializations of the PIM class represented by the PSM class.
  - **Commentary –** Adds a free commentary to the diagram

- **Ordering**
  - **Move Left** – Swaps the selected PSM element with the component on its left-hand side
  - **Move Right** - Swaps the selected PSM element with the component on its right-hand side

3.2. PIM and PSM diagram canvases

PIM diagram canvas

PIM diagram canvas provides space for modeling PIM diagrams which are in fact UML class diagrams. The following visual components can be present on a PIM diagram canvas:

- PIM class
- Association class
- Association, aggregation, composition
- Generalization
- Comment

Hidden dependent element

If you delete an element from PIM diagrams but not from the model, you still need to find the element somewhere and have a possibility to make it visible in a diagram again. All PIM classes from the model are listed in the Navigator window. All other elements (e.g.

associations, comments etc.) can be found among hidden elements of a PIM diagram. To make such an element visible again, right-click on the canvas of a selected PIM diagram and select *Include hidden elements* from the context menu.

From a list offered, select a previously deleted element you want to get back to the diagram.

PSM diagram canvas

PSM diagram canvas provides space for modeling PSM diagrams. They are hierarchical diagrams that can be comprised of the following visual elements:

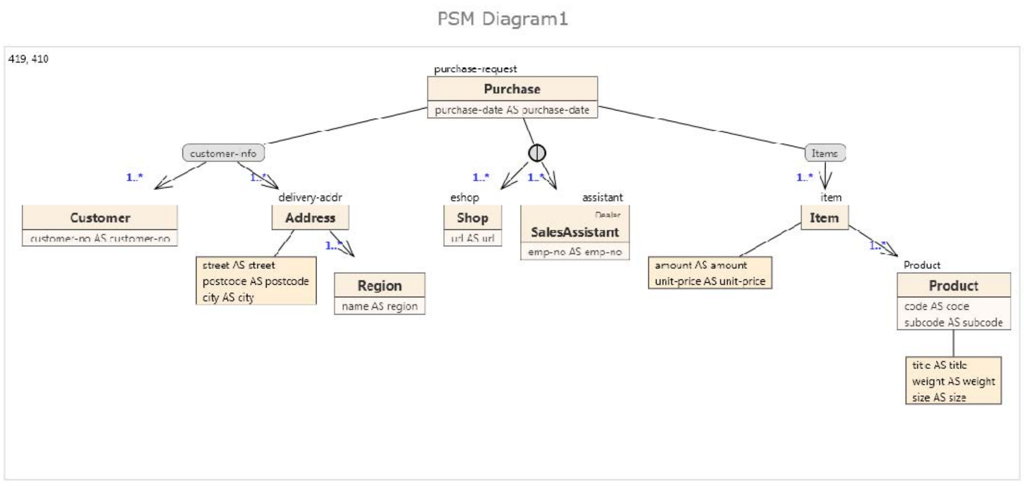
- PSM class
- PSM container
  - Attribute container
  - Content container
  - Content choice
  - Class union
- Association
- Specialization
- Comment

Diagram PNG exporting

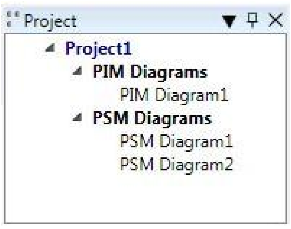
XCase enables one to export PIM and PSM diagrams to PNG picture format.

1] Right-click on the canvas of a diagram you want to export as a picture
2] Select *Export diagram to PNG*
3] Enter the name of the PNG file in the dialog box

The content of the diagram is then saved as a PNG picture.



3.3. Project window



An XCase project is composed of PIM and PSM diagrams. The Project window lists the diagrams in an unfolding tree layout.

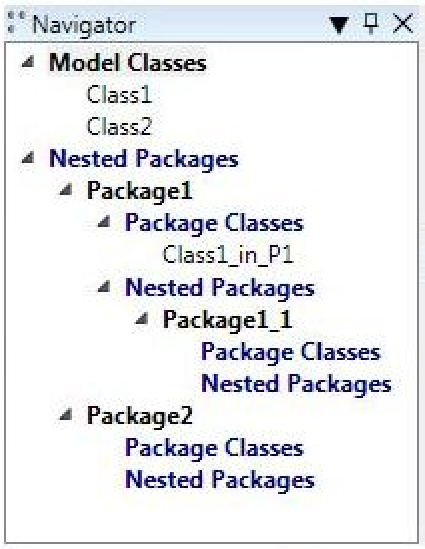
You can invoke a context menu by right mouse-button click on a project name. The context menu offers the following items:

- **Rename project** – Changes the name of the current XCase project
- **Change project's namespace** – Changes the target namespace for generated XML schemas
- **Add PIM Diagram** – Adds a new empty PIM diagram to the current project
- **Add PSM Diagram** – Adds a new empty PSM diagram to the current project

Context menu invoked on a PIM or PSM diagram offers:

- **Rename** – Renames selected PIM or PSM diagram
- **Remove** – Removes selected PIM or PSM diagram from the project

3.4. Navigator window



The Navigator window lists all packages and PIM classes present in the model for the current project. The listed classes do not have to be necessarily present in any PIM diagram, e.g. there can be classes without any visualization.

At the top level there are *model classes* and *model nested packages*. Nested packages can contain other nested packages and nested classes.

See the chapter 5 for how to create, rename and remove a package in XCase and the chapter 6 for how to add a new class to a package or move an existing class to another package.

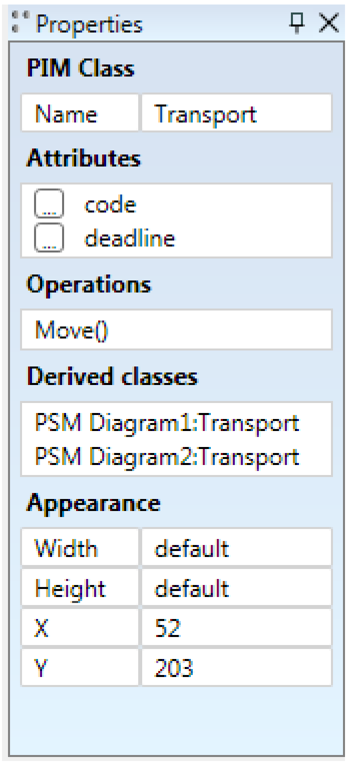
If you select a class in the *Navigator* window and this class has visualization in the current diagram, the visualization is selected.

3.5. Properties window

The *Properties* window displays properties of a selected element. These properties can be edited through this window. It displays the following properties depending on the type of the selected element:

On a PIM diagram canvas:

- **PIM Class**
  - name, attribute list, operation list, derived classes list, appearance
- **Association Class**
  - name, attribute list, operation list, appearance, label and multiplicity for each association end
- **Association**
  - name, label and multiplicity for each association end
- **Comment**
  - comment text, appearance



Properties

PIM Class

Name	Transport

Attributes

code
deadline

Operations

Move()

Derived classes

PSM Diagram1:Transport
PSM Diagram2:Transport

Appearance

Width	default
Height	default
X	52
Y	203

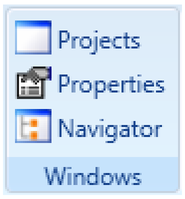
On a PSM diagram canvas:

- **PSM Class**
  - class name, element name, attribute list, appearance (non-editable)
- **Content Container**
  - element name, appearance (non-editable)
- **Attribute Container**
  - attribute list
- **PSM Association**
  - multiplicity, nesting joins
- **Comment**
  - comment text, appearance

3.6. Working with windows in GUI

Docking windows

Properties, *Project* and *Navigator* windows can be docked in several positions within the GUI. If you want to dock a window, just grab it and move it to the desired position indicated by a transparent window slot. You can also leave it undocked.



To open again a closed docked window, click on the proper button (Projects, Properties, Navigator) in Windows section of the main toolbar.

Auto hiding windows

To make a window *auto hiding*, click on 🖈 symbol on its label. The window is then collapsed to a small label stuck to the side of the editor. If you move mouse cursor over this label, the whole window becomes visible again. To make a window persistent (not auto hiding), click on ⊣ symbol of the auto hiding window.

3.7. XCase editor shortcuts

XCase editor supports some standard keyboard shortcuts which are listed in the following table.

Action	Shortcut
New Project	CTRL+N
Open Project	CTRL+O
Save Project	CTRL+S
Save As	SHIFT+S
About	CTRL+F1
Close active diagram	CTRL+W, CTRL+F4
Undo	CTRL+Z
Redo	CTRL+Y
Delete from diagram (PIM) /Delete subtree(PSM)	DELETE
Delete from model	SHIFT+DELETE
Rename element	F2
Rename label	SHIFT+F2
←↑→↓	Movement in PSM diagrams

PIM Level Modeling

4. PIM Diagrams

After XCase start-up, en empty environment is prepared with one empty PIM diagram opened. Its name is *PIM Diagram1*.

4.1. Creating PIM diagrams

There are several ways how to add a new empty PIM diagram to the project:

- Click on the *New PIM Diagram* button in Diagrams section in the main toolbar

or

- In the Project window, invoke a context menu on *PIM Diagrams* and then select *Add New PIM Diagram*

New empty PIM diagram is then added and opened in the project.

4.2. Renaming PIM diagrams

When a new PIM diagram is added to the project, it gets a default name. If you want to change this name:

1] Right-click on the PIM diagram name in the Project window
2] Select *Rename* from the context menu
3] Enter the desired name to the dialog box

The diagram is then renamed.

4.3. Removing PIM diagrams

There are several ways how to remove a PIM diagram from the project:

- Select the PIM diagram and press the *Delete diagram* button in the main toolbar (Diagrams section)

or

- In the Project window, invoke a context menu on the PIM diagram and choose *Remove* item

The PIM diagram and all its components are then removed from the project.

5. Packages

Packages allow further organization of PIM classes in an XCase project. They are handled via context menus in the Navigator window.

5.1. Creating packages

1] In the Navigator window, right-click on *Nested packages* of the package/model you want to add a new package to
2] Select *Add new package* from the context menu

A new package with default name *Package[n]* is then added to the project and visible in the Navigator window.

5.2. Renaming packages

1] In the Navigator window, double-click on the package you want to rename
2] Enter the name of the package in the dialog box and press OK

The package is then renamed.

5.3. Moving packages

1] In the Navigator window, double-click on the package you want to move
2] Select new parent package in the combo box and press OK

The package is then moved to the selected package.

5.4. Removing packages

1] In the Navigator window, right-click on the package you want to rename
2] Select *Remove* from the context menu

The package and all its classes are then removed.

6. PIM Classes

A PIM class represents one modeled entity. The Navigator window shows all PIM classes present in the model. You can also see a PIM class visualized in respective PIM diagrams.

6.1. Creating PIM class

Adding PIM class to both model and diagram

If you want to create a PIM class and its visualization in a PIM diagram at once, just click on the *Class* button in the main toolbar and drag the class object to the desired place on the canvas of a PIM diagram.

The class is then present in the model as well as in the PIM diagram.

Adding PIM class to model only

1] Right-click on *Model classes* in the Navigator window
2] Select *Add new class* from the context menu

A new PIM class with default name *Class[n]* is then added to the model but it has no visualization in any PIM diagram.

If you later decide to visualize this class, just select it in the Navigator window and drag it to the desired position in a PIM or PSM diagram.

6.2. Renaming PIM class

You can rename each PIM class by following these steps:

1] Double-click on the PIM class in the Navigator window.
2] Write the desired name into the dialog box.

If a PIM class has visualization in some PIM diagram, you can also use some other ways to rename it:

1] Select the PIM class on the canvas.
2] Edit its name in the Properties window

or

1] Right-click on the PIM class on the canvas.
2] Choose *Rename* from the context menu or press F2.

or

1] Double-click on the PIM class on the canvas to open the Class dialog box
2] Change class name
3] Press OK to confirm the change and close the dialog box

6.3. Moving PIM class to another package

1] Double-click on the PIM class on the canvas to open the Class dialog box
2] In *Package* combo box select the package to move this class to
3] Press OK to confirm the change and close the dialog box

6.4. Removing PIM class

You can remove a PIM class completely or just from diagrams. In the second case the class is still present in the model after removal. If you decide to add it to a PIM diagram back again, you just use the same steps as when adding a PIM class without visualization to a PIM diagram.

Removing PIM class from PIM diagram only

The class is still present in the model after removal.

1] Select the PIM class you want to remove
2] Press *SHIFT+DELETE* or click on the *Delete from diagram* button in the main toolbar or invoke its context-menu by right-click and then select *Remove*
3] If the PIM class is not present in any other PIM diagram, you are offered to remove it from the model as well. If you want to keep it in the model, uncheck the class name in the displayed dialog window

Removing PIM class from the model

The class is removed completely from the project.

1] Right-click on the PIM class in the Navigator window
2] Select *Remove* from the invoked context menu

or

1] Select the PIM class you want to remove on the canvas
2] Press *DELETE* or click on the *Delete from model* button in the main toolbar

6.5. PIM class attributes

Adding PIM class attributes

1] Right-click on the PIM class on the canvas or in the Navigator window
2] Select *Add new attribute* from the invoked context menu

or

1] Select the PIM class on the canvas
2] Click on *Add attribute* in the main toolbar (PIM diagram elements section)

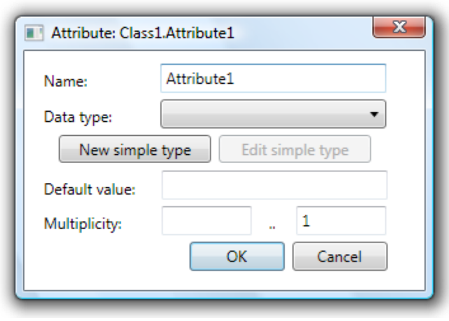
A new attribute with the default name *Attribute[n]* is then added to the class.

Editing PIM class attributes

1] Unroll the class with the desired attribute in the Navigator window
2] Double-click on the attribute name

or

1] Select the PIM class with the desired attribute on the canvas.
2] Right-click on the attribute name
3] Select *Properties* from the context menu.


Attribute: Class1.Attribute1

Name:	Attribute1
Data type:	
Default value:	
Multiplicity:	.. 1

Attribute pop-up window appears.
Here you can edit:

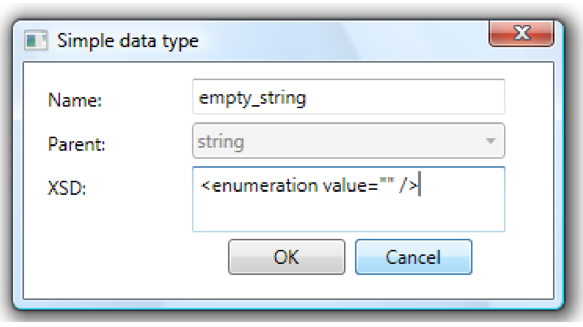
- Attribute name
- Attribute data type
- Attribute default value
- Attribute multiplicity

You can even create and edit your own simple data types and share them among attributes.

Creating and editing simple data types for PIM attributes

To create a new data type click *New simple type* in Attribute dialog box.
In the Simple data type dialog box you can fill in the following:

- **Data type name** – Identifier of the newly created data type
- **Parent data type** – Primitive data type this one is derived from
- **Implementation** – Implementation of the data type that will be used when creating XML schema.


Simple data type

Name:	empty_string
Parent:	string
XSD:	<enumeration value="" />

Since XML Schema is used as a target XML schema language, write here a proper XML schema code. The new type is created by restricting existing simple type (we do not support list and union derivation). Therefore, fill in only the facets applied for the restriction. The rest of the code for the simple type definition will be created mechanically for you. Also do not use a prefix for XSD namespace. It will be added automatically.

Click *OK* to add your new data type to the selected package.

You can then use this data type for any PIM attribute. To edit your new data type, select it in Attribute dialog and press *Edit simple type*.

Removing PIM class attributes

1] Select the PIM class with the desired attribute on the canvas
2] Right-click on the attribute name
3] Select *Remove attribute* from the context menu

6.6. PIM class operations

PIM class operations have any meaning neither in derived PSM diagrams nor in derived XML schemas. They are present just to support PIM level's functionality as a common UML class editor.

Adding PIM class operations

1] Right-click on the PIM class on the canvas
2] Select *Add new operation* from the invoked context menu

or

1] Select the PIM class on the canvas
2] Click on *Add operation* in the main toolbar (PIM diagram elements section)

A new operation with the default name *Operation[n]* is then added to the class.

Editing PIM class operations

To change a name of an operation, select its parent class on the canvas. In the Properties window, change its name to the new one.

Removing PIM class operations

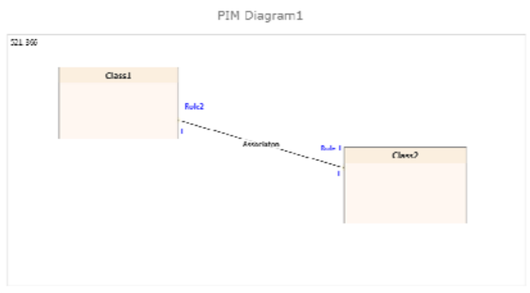
1] Click on the PIM class with the desired operation
2] Right-click on the operation name
3] Select *Remove operation* from the context menu

7. PIM Connections

7.1. PIM Associations

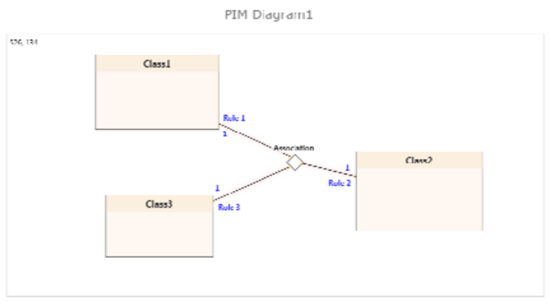
PIM associations describe relations among PIM classes. XCase supports binary as well as general n-ary associations.

Creating binary PIM Associations



To create a binary PIM association, toggle the *Association* button in the main toolbar.
Then click on the first class and drag the association line to the second class. The classes are then connected by a direct simple line.
You can repeat this step until untoggling the *Associate* button or right-mouse clicking on the canvas.

Creating n-ary PIM Association



To create an association among 2 or more classes, select these classes on the canvas and click on the *Associate* button in the main toolbar.

The classes are then connected via a connecting diamond if they are 3 or more.

Editing PIM Associations

The following properties can be edited for each PIM association:

- **Name**
  - Describes more precisely the meaning of the association
- **Role name** for each association end
  - Specifies the specific role of each class involved in the association
- **Cardinality** for each association end
  - Lower and upper limit for presence of each class involved in the association
- **Aggregation** type for each association type
  - None, shared or composite

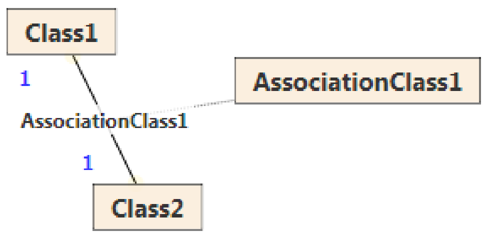
You can edit these properties in the dialog window invoked by double clicking on the desired association on the canvas. Name, role and cardinality can be edited directly via the Properties window.

7.2. PIM Association Class

Creating PIM association class

An Association class can be seen as an association that also has class properties, or as a class that also has association properties. Typically, it is used when you need to provide some more information for an association (by adding attributes to its class part). We refer to [4,5] for details.

To associate two or more classes with an association class, select these classes on the canvas and click on the *Association class* button in the main toolbar.



Like with a simple association, the selected classes are then either connected by a direct simple line if they are just 2, or connected via a connecting diamond if they are 3 or more. There is also the class part attached to the association by a thin dotted line.

In general, you can use association class as a usual PIM class. You can add attributes and operations to it and connect it to some other classes.

Editing PIM association class

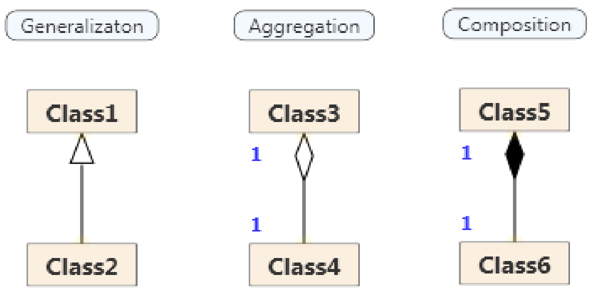
Association properties of an association class can be edited in the same way as properties of an association. To learn more, see the chapter 7.1

Class properties of an association class can be edited in the same way as properties of a PIM class. To learn more about it, see the chapter 6.5.

7.3. Creating PIM Generalizations, Aggregations and Compositions

XCase also allows you to work with more specific kinds of UML connections, namely generalizations, aggregations and compositions.

To create the above mentioned relations between 2 classes, follow these steps:



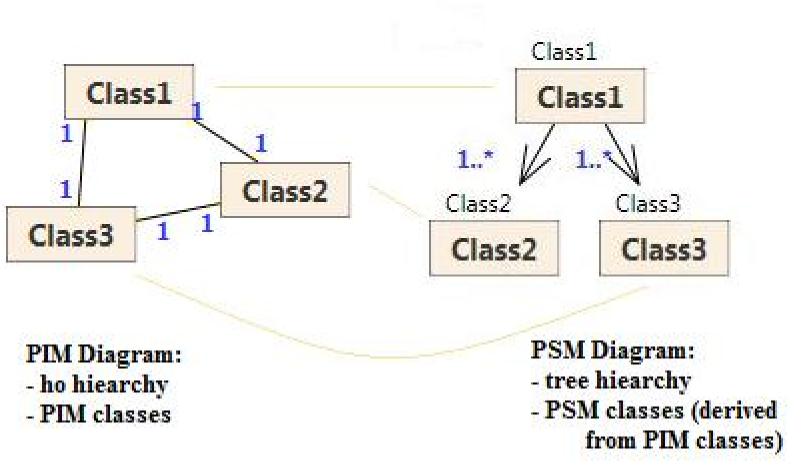
1] Toggle the appropriate [*Generalization*, *Aggregation*, *Composition*] button in the main toolbar
2] Select the first (child, specific, derived) class on the canvas a drag the connecting line to the second (parent, general) one.

You can continue to connect classes by *Generalization* | *Aggregation* | *Composition* line until untoggling the button in the main toolbar or right-clicking on the canvas.

Modeling on PSM Level

8. PSM Diagrams

A PSM diagram models a particular XML format in terms of PIM diagrams in the project. Its components represent components of the PIM diagrams and organize them to the required hierarchical structure of the XML format. XCase supports all PSM constructs proposed in [1].



8.1. Creating new empty PSM Diagram

There are several ways how to add a new empty PSM diagram to the project:

- Click on the *New PSM Diagram* button in the main toolbar (Diagrams section)

or

- In the Project window, invoke a context menu on *PSM Diagrams* and then select *Add new PSM Diagram*

A new empty PSM diagram is then added and opened in the project.

8.2. Renaming PSM Diagrams

When a new PSM diagram is added to the project, it gets a default name. If you want to change this name:

1] Right-click on the appropriate diagram in the Project window
2] Select *Rename* from the context menu
3] Enter the desired name in the dialog box

The selected PSM diagram is then renamed.

8.3. Removing PSM Diagrams

There are several ways how to remove a PSM diagram from the project:

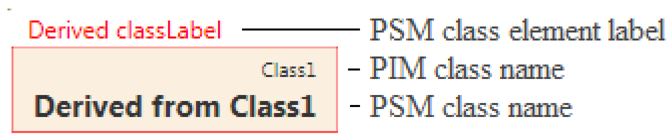
- Select the PSM diagram, you want to delete, so it becomes the active one. Then press the *Delete diagram* button in the main toolbar (Diagrams section)
- In the Project window, invoke a context menu on the PSM diagram, you want to delete. Then choose *Remove* item.

PSM diagram and all its elements are then removed from the project.

9. PSM Classes

Each PSM class in a PSM diagram must be derived from a PIM class. We say that the PSM class *represents* this PIM class. A PSM class models how instances of the represented PIM class are expressed in the corresponding XML format.

The PSM class has a name and element label as depicted in the following picture. When translating the PSM class to an XML schema representation, the name is used as a name of the generated XML type, while the element label is used as a name of the generated XML element. If the PSM class name differs from PIM class name, the PIM class name is displayed together with the PSM class as well.



9.1. Creating root PSM Class from PIM Class

A root PSM class can be derived from a PIM class to a new or existing PSM diagram as follows:

1] Select the PIM class in the PIM diagram
2] Click on *Derive to new PSM Diagram* or on *Derive to existing Diagram* in the main toolbar

9.2. Adding children to PSM Class

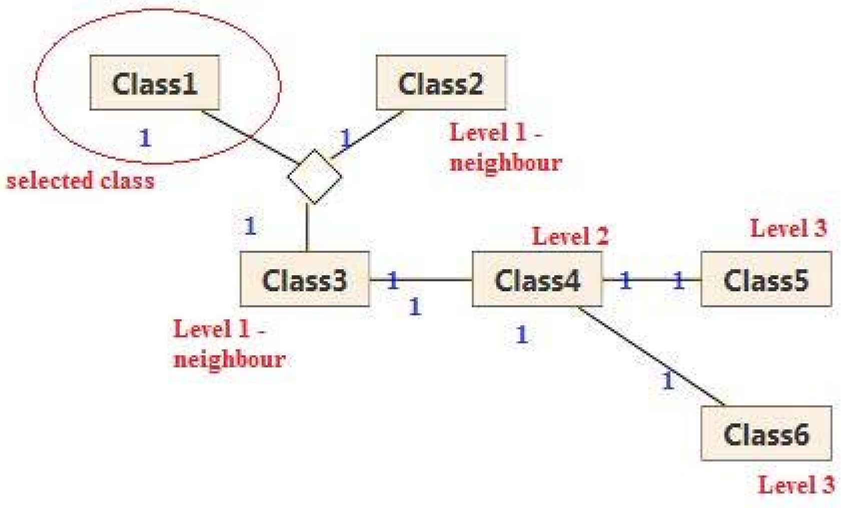
You can add children under an existing PSM class to model the requested XML tree hierarchy.

1] Click on the PSM class in the PSM diagram.
2] Click on the *Add children* button in the main toolbar.

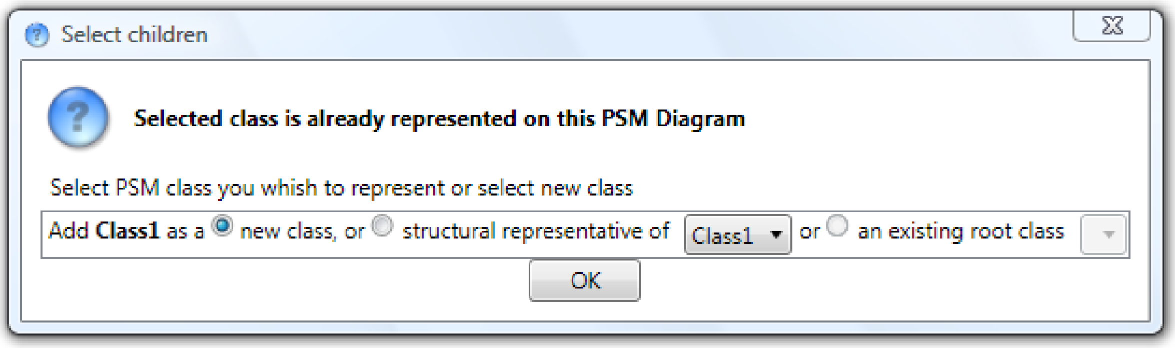
A new dialog box appears where you select one or more PIM classes. The tool then automatically creates for each selected PIM class a new PSM class, which represents the selected PIM class, and inserts it as a child of the actual PSM class.

PIM classes offered in the dialog box are organized according to the association paths connecting them with the PIM class represented by the actual PSM class. Just associations (aggregations, compositions) are considered to form the paths, not generalizations.

At the first level, there are neighbors of the PIM class, e.g. there are all PIM classes directly associated with the PIM class. At the second level, there are PIM classes which are in distance 2 from the PIM class and so on.



After specifying the requested PIM classes to insert as children of the actual PSM class, the following dialog box appears. You can then choose from three possibilities for each specified PIM class.


Select children

Selected class is already represented on this PSM Diagram

Select PSM class you whish to represent or select new class

Add **Class1** as a [✓ new class], or structural representative of Class1 or an existing root class

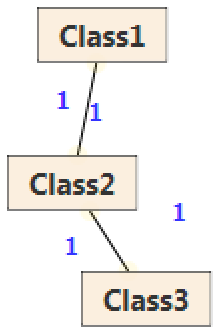
OK

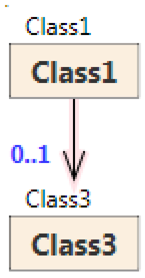
- The first possibility allows creating a completely new PSM class that represents the PIM class.
- The second possibility enables one to create so called *structural representative*. This possibility is explained in the chapter 9.3.
- The last possibility is designed for connecting an existing root PSM class, which represents the PIM class, as a child of the actual PSM class.

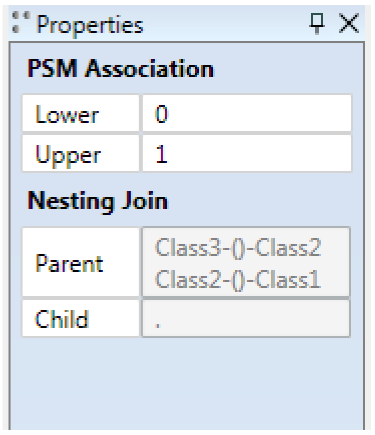
Nesting joins

The resulting child PSM class is connected with the parent PSM class by a PSM association. As each PSM class represents a corresponding PIM class, there is also a binding of each PSM association to the PIM level. This binding is called *nesting join*. We say that the PSM association represents the nesting join.

In a basic version, nesting join comprises two parts called *parent* and *child*. Parent is the path of PIM associations used to construct the PSM association. Child is empty. You can see the parent and child for each PSM association in the Properties window. An appropriate nesting join is constructed mechanically by the tool when creating the PSM association (i.e. as a result of *Add child* operation). Nesting joins can be more complex as we show later in the chapter 9.6. For a detailed description of nesting joins, we refer to [1].






Properties

PSM Association

Lower	0
Upper	1

Nesting Join

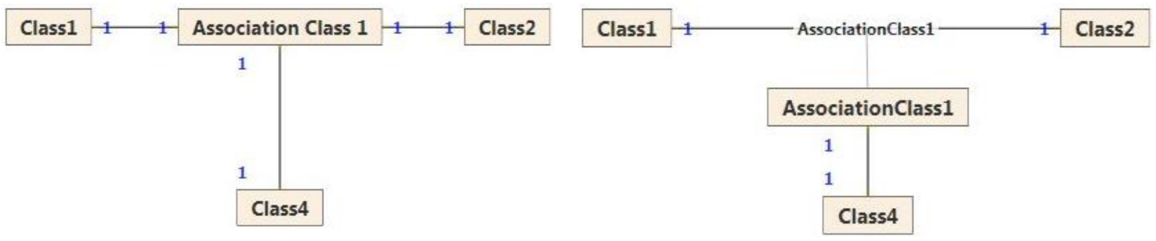
Parent	Class3-()-Class2 Class2-()-Class1
Child	.

PIM **PSM**

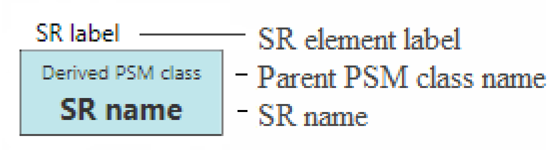
Association classes as children in PSM diagrams

There can also be an association class on a path instead of a simple association. In that case it must be possible to somehow insert not only an endpoint of that association class into the PSM diagram but also the association class itself. For this purpose, we suppose the association class to be a normal class connected with the original endpoints by two simple associations.

As a consequence, there is no difference between the two situations displayed in the following self-describing figure.



9.3. Creating Structural representatives of PSM Classes

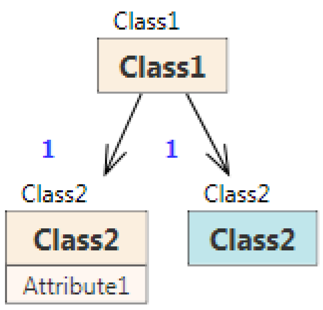


You can insert a PIM class into a PSM diagram not only as an ordinary PSM class but also as so called *structural representative*. You are offered this possibility when you are deriving a PSM class from a PIM class that is already represented in the actual PSM diagram.

A structural representative is a specific kind of PSM class that refers to another PSM class and obtains automatically its attributes and content. The structural representative extends these obtained components by its own attributes and content. Only these extending components are displayed for the representative. However, the obtained components are taken into account during the translation to the final XML schema.

Therefore, structural representatives allow reusing an already modeled content at more places in a PSM diagram at once. Since PSM diagrams must have a tree structure, we also use structural representatives for modeling recursive structures.

Visualization of structural representatives differs in color (powder blue) from visualization of ordinary PSM classes. The picture shows a PSM class *Class2* (on the left) and its structural representative *Class2* (on the right). The structural representative inherits *Attribute1* of *Class2*.



9.4. PSM Class Attributes

Adding PSM class attributes related to PIM class attributes

A PSM class can receive attributes of its PIM class.

1] Click on the PSM class
2] Click on the *Attributes* button in the main toolbar

In the dialog box select which attributes you want to add. There are all the attributes of the original PIM classes listed. You cannot change their names but you can use different aliases for them to be displayed in the PSM class. These aliases are then used when generating XML schema.

Adding independent ('PIM-less') PSM class attributes

A PSM class can have completely new attributes which have no associated counterparts in the appropriate PIM class. To add such a PIM-less attribute, do the following:

1] Click on the PSM class
2] Click on the *Add attribute* button in the main toolbar

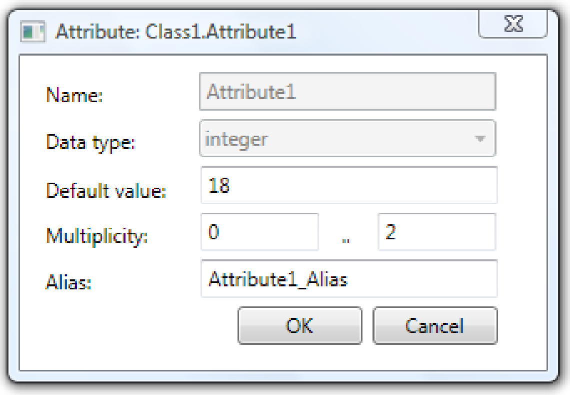
Editing PSM class attributes

1] Select the PSM class with the desired PSM attribute on the canvas.
2] Right-click on the attribute's name
3] Select *Properties* from the context menu.

An Attribute pop-up window appears.

Here you can edit:

- Attribute data type
- Attribute default value
- Attribute multiplicity
- Attribute alias


Attribute: Class1.Attribute1

Name: Attribute1
Data type: integer
Default value: 18
Multiplicity: 0 .. 2
Alias: Attribute1_Alias

The initial values of all these properties are taken right from its PIM attribute counterpart. The name and data type are received as well, but they are non editable.

If you change a multiplicity of a PIM attribute, you can decide if you want to propagate this change also to related PSM attributes. A change in a data type and default value is propagated automatically.

Another way how to edit a PSM class attribute is via the Properties window. There are all the attributes listed. You can change an alias for a PSM attribute directly. If you click on the button next to its name, the previously mentioned PSM Attribute dialog box is displayed.

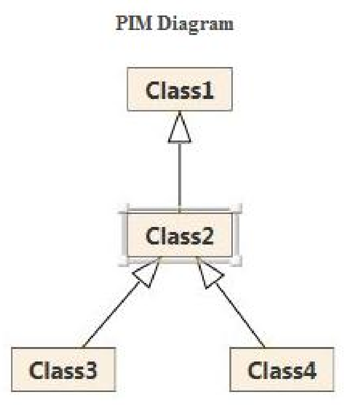
Removing PIM class attributes

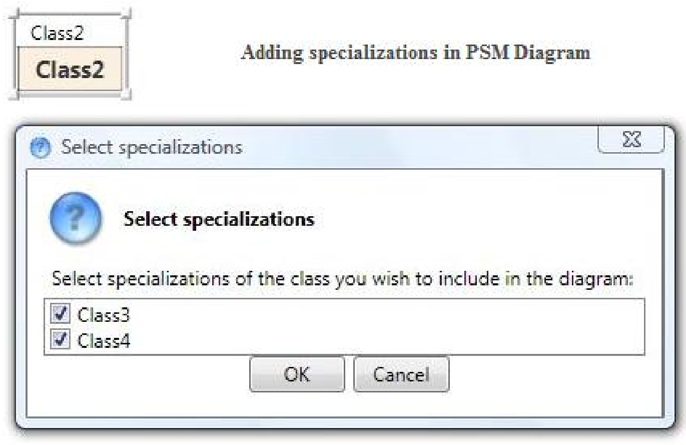
1] Select the PIM class with the desired attribute on the canvas.
2] Right-click on the attribute's name
3] Select *Remove attribute* from the context menu.

9.5. Adding specializations to PSM Classes

Correspondingly to specializations at the PIM level, you can specialize given PSM class by PSM classes that represent specializations of the corresponding PIM class. To add a specialization of a PSM class:

1] Click on the PSM class
2] Click on the *Add specialization* button in the main toolbar.
Note: If this button is not enabled, there are no available specializations for this class.
3] The dialog box shows a list of specializations of the corresponding PIM class. You can select one or more specializations that will be represented in the PSM diagram as specializations of the PSM class.





Select specializations

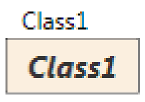
Select specializations of the class you wish to include in the diagram:

A PSM class with specializations specifies that instances of the PSM class are represented in the XML format at the corresponding location. Moreover, instances of the specializations are represented at this location. These specialized representations comprise of a part described by the PSM class and a part described by a corresponding specialization.

Abstract PSM class

It is also possible to denote a PSM class as abstract. In that case only instances of non-abstract specializations of the PSM class can be represented at the given location. The details of the corresponding XML structures modeled by PSM classes and their specializations are explained in the chapter 12.3.

A PSM class can be denoted as abstract as follows:



1] Right-click on the PSM class on the canvas
2] Tick *Abstract class* in the invoked context menu

The PSM Class name is then displayed in italics to express that it is an abstract class.

9.6. PSM Class Grouping

Due to a hierarchical nature of XML, it is quite easy to represent grouping of given objects according to some criteria. To model such grouping, XCase allows grouping of a given root PSM class by one or more of its children. Other than root PSM classes can not be grouped.

After the grouping, the children are represented as direct ancestors of the grouped PSM class. At the instance level, the grouping specifies that two instances of the grouped PSM class are in the same group if they are related to the same instances of the grouping ancestors.

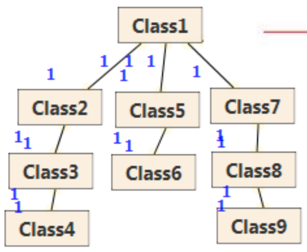
The formal semantics of the grouping is specified by nesting joins represented by the PSM classes connecting the ancestors. Each such PSM class represents a nesting join composed of the following:

- **parent:** corresponding PIM path (i.e. a path composed of PIM associations) going from the PIM class represented by the grouped PSM class to the PIM class represented by the parent of the PSM association
- **child:** corresponding PIM path going from the PIM class represented by the grouped PSM class to the PIM class represented by the child of the PSM association
- **context:** a set containing for each ancestor of the PSM association in the grouping a corresponding PIM path going from the PIM class represented by the grouped PSM class to the PIM class represented by this ancestor

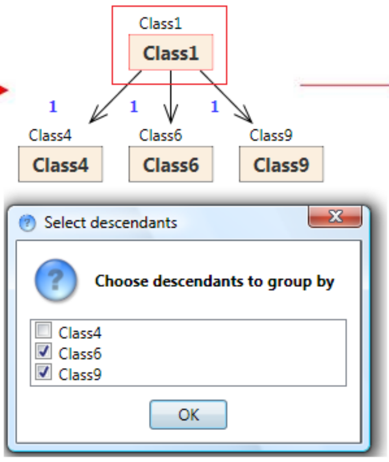
Corresponding nesting joins are created by the tool mechanically. You can see the nesting joins in the Properties window.

To group a root PSM class by one or more its children, follow these steps.

1] Right-click on the root PSM class in the PSM diagram
2] Select *Group by* from its context menu
3] Choose descendants to group by



PIM Diagram



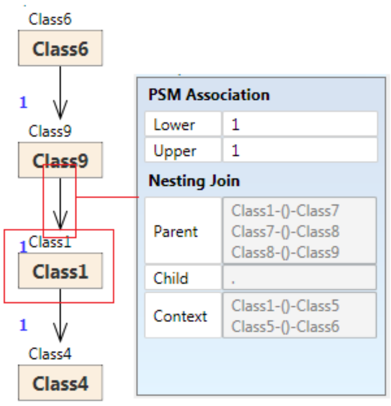
Select descendants

Choose descendants to group by

- [ ] Class4
- [✓] Class6
- [✓] Class9

OK

PSM Diagram



PSM Association

Lower	1
Upper	1

Nesting Join

Parent	Class1-()-Class7 Class7-()-Class8 Class8-()-Class9
Child	.
Context	Class1-()-Class5 Class5-()-Class6

PSM Diagram after Group by

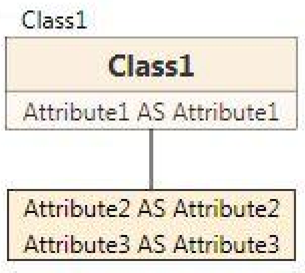
10. PSM Containers

The following PSM containers are available in PSM diagrams to help to model the tree structure and specify the output XML:

- PSM attribute container
- PSM class union
- PSM content container
- PSM content choice

10.1. PSM Attribute Container

Basically, a PSM attribute of a PSM class is expressed in XML documents as an XML attribute. To specify that it is expressed as an XML element we use *attribute containers*.



An attribute container is assigned to a PSM class and takes one or more of its attributes. It is displayed as a box connected to the parent class by a solid line. It is considered as a component of the content of the parent class and specifies the location of the modeled XML elements.

A PSM class can have more PSM attribute containers.

Moving PSM class attributes to PSM attribute container

1] Click on the PSM class with some PSM attributes
2] Click on the *Attribute container* button in the main toolbar (PSM diagram elements section)
3] In the dialog box, check which attributes to move to the attribute container

Attribute container with the selected attributes is then created and connected to the PSM class.

Moving PSM attributes from PSM attribute container back to PSM class

1] Right-click on the PSM attribute in PSM attribute container
2] Select *Move back to class* in the context menu

PSM attributes are then moved back to the PSM class.

Editing PSM attributes in PSM attribute container

PSM attributes displayed in a PSM attribute container are edited the same way as PSM attributes displayed directly in a PSM class. To edit a PSM attribute directly, right-click on it in a PSM attribute container and select *Properties* from its context menu.

Removing PSM attribute container

1] Select a PSM attribute container on the canvas
2] Click on the *Delete container* button in the main toolbar.

PSM attribute container is then deleted from the diagram and its attributes are returned back to the original PSM class.

10.2. PSM Content Container

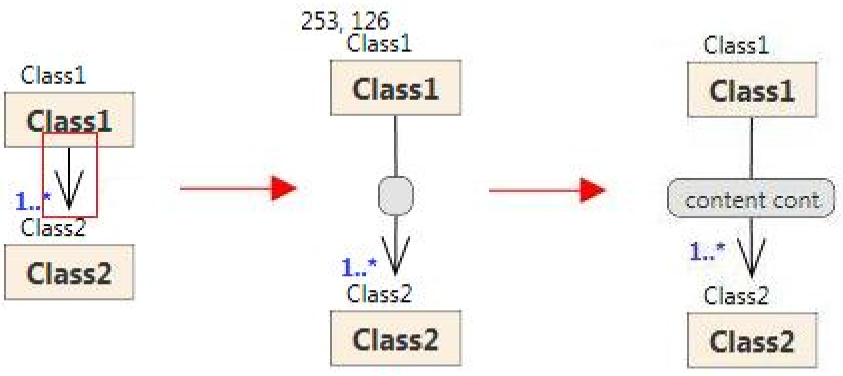
A content container allows modeling an XML element that does not have any semantics in terms of PIM diagrams.

PSM content container has a name and is assigned to a PSM class. It contains a part of the content of this class. It is displayed by a narrow rounded rectangle with the name inside and is connected to the parent class by a solid line. The content container models that for each instance of the PSM class, the XML code modeled by the components of the container is enclosed in a separate XML element named by the name of the container.

Adding PSM content container

1] Select the PSM association coming from a PSM class or a PSM container
2] Click on the *Content container* button in the main toolbar

A new unnamed PSM content container is added.



Editing PSM content container

PSM content container has no name by default. To enter a name (further denoted as XML element name), right-click on it and select *Rename content container*. You can also edit this name via the Properties window.

Deleting PSM content container (Revert)

1] Select the PSM content container
2] Click on the *Delete container* button in the main toolbar

PSM content container is then removed, so the diagram looks exactly the same as before it was added.

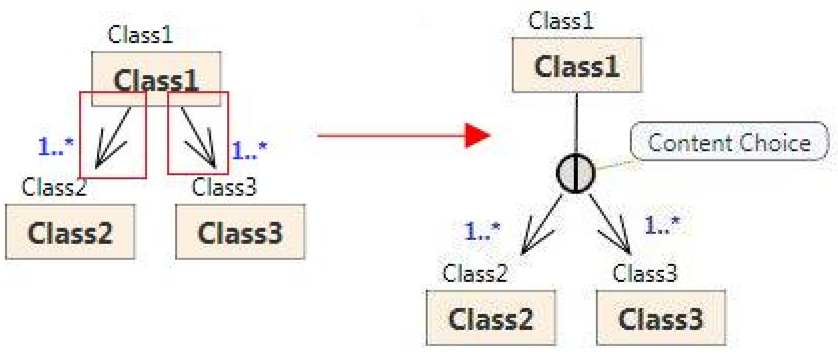
10.3. PSM Content Choice

A *PSM content choice* models variants in the content of a PSM class. It is assigned to a PSM class and contains two or more PSM associations coming from it.

It is displayed by a circle with an inner | and is connected to the PSM Class by a solid line. It models that for each instance of the PSM class, only one of the associations is instantiated.

Creating PSM content choice

1] Select two or more neighboring PSM associations
2] Click on the *Content choice* button in the main toolbar



Selected associations are then joined into a content choice

Deleting PSM content choice (Revert)

1] Select the PSM content choice
2] Click on the *Delete container* button in the main toolbar

Content choice is removed, so the diagram looks exactly the same as before it was added.

10.4. PSM Class Union

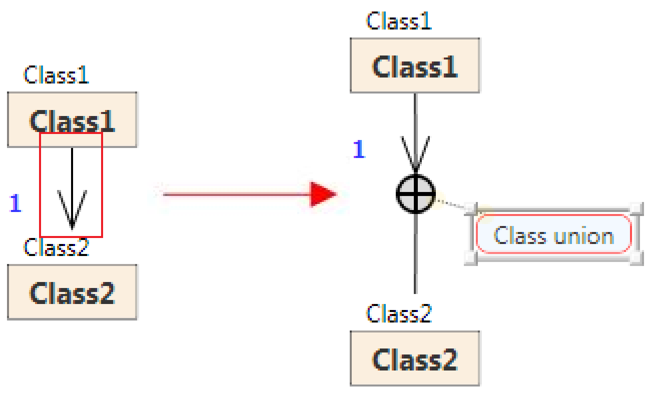
A *PSM class union* is an endpoint of a PSM association and contains one or more PSM classes.

It is displayed by a circle with an inner + at the end of a PSM association. The PSM classes in the union are displayed beneath the circle and are connected to the circle by solid lines. It models a mixture (i.e. union) of contained PSM classes. At the instance level, it models a mixture of their instances.

Creating PSM class union

1] Select the PSM association
2] Click on the *Class union* button in the main toolbar

A new PSM class union is the added.



Removing PSM class union (Revert)

1] Select the PSM class union you want to delete
2] Click on the *Delete container* button in the main toolbar

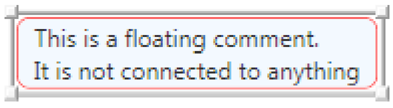
The PSM class union is removed, so the diagram looks exactly the same as before it was added.

11. Comments

Comment elements can contain additional information about whole modeled diagrams or specific elements. There are two types of comments: floating and connected.

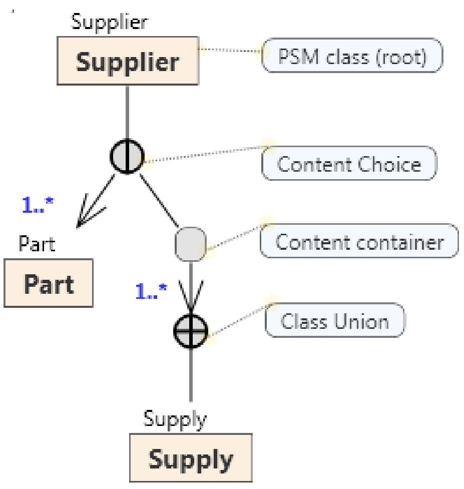
11.1. Floating comments

Floating comments can be placed freely on a PIM or PSM diagram. They are not connected to any elements and they can carry general information. To add such a comment to the current diagram, unselect all elements on the canvas and click on the *Comment* button in the main toolbar.



11.2. Connected comments

Connected comments are connected to some elements on the canvas. Elements that can be commented this way are the following:



In a PIM diagram:

- PIM class
- Association class
- Association
- Generalization

In a PSM diagram:

- PSM class
- Class union
- Content container
- Content choice
- Attribute container

To add a comment to any of these elements, right-click on it and select *Add commentary* from its context menu. If you drag a new comment to the canvas, the new comment is automatically connected to the selected element.

Both types of comments can be edited via the Properties window or directly by right-clicking on it and selecting *Change*. If you want to remove it, select *Remove from diagram*.

12. Generating XML schemas

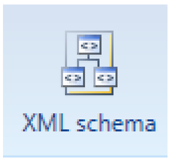
12.1. How PSM Diagram is translated into schema

Each PSM diagram can be translated to a representation in XML Schema [3]. Currently, we support a fully mechanical translation. It is not possible to customize the translation by any parameters.

The translation starts in root PSM classes and proceeds recursively to their descendants. Details of the translation of each separate PSM construct are depicted in Section 11.3. Briefly, a PSM class is translated to a sequence of element declarations (`element` construct) and a set of attribute declarations (`attribute` construct). The element declarations result from the content of the class. The attribute declarations result from its attributes. If the class has an element label, the declarations are inserted into the resulting XML schema as a complex type (`complexType` construct). Otherwise, they are inserted as a model and attribute group (`group` and `attributeGroup` construct, respectively).

This basic translation is further influenced by PSM structural constructs, i.e. attribute container, content container, content choice and class union as demonstrated by the table. The table also explains how structural representatives and specializations in PSM diagrams are expressed in XML Schema.

12.2. Generating XML schema in XCase



To generate an XML schema for a current PSM diagram, click on the *XML schema* button in the main toolbar (Diagrams section).

XML Schema window with generated XML schema is then opened. Generated schema is written in W3C XML Schema language [3].

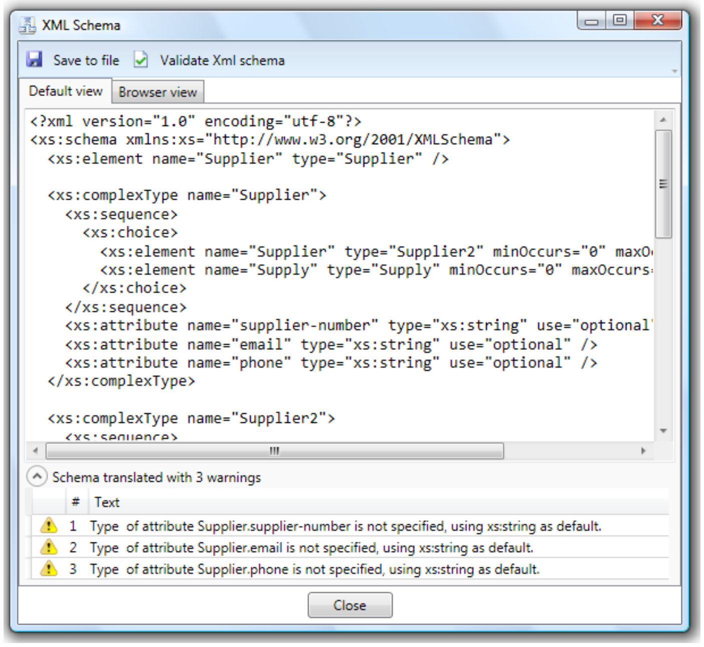
XML Schema window tabs

- **Default view** - Editable text of generated XML schema
- **Browser view** - Non editable collapsible text of generated schema

XML Schema window buttons

- **Save to file** - Saves text content of *Default view* to a named *.xsd file

- **Validate XML schema** – Checks whether the XML schema is a valid XML Schema document



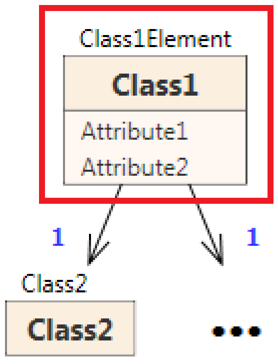
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Supplier" type="Supplier" />

  <xs:complexType name="Supplier">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Supplier" type="Supplier2" minOccurs="0" maxO
        <xs:element name="Supply" type="Supply" minOccurs="0" maxOccurs
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="supplier-number" type="xs:string" use="optional"
    <xs:attribute name="email" type="xs:string" use="optional" />
    <xs:attribute name="phone" type="xs:string" use="optional" />
  </xs:complexType>

  <xs:complexType name="Supplier2">
    <xs:sequence>
```

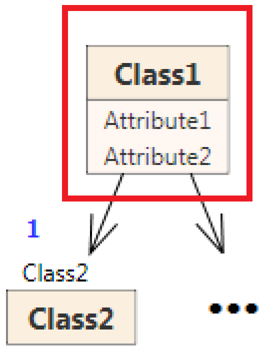
Schema translated with 3 warnings

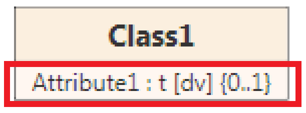
	#	Text
⚠	1	Type of attribute Supplier.supplier-number is not specified, using xs:string as default.
⚠	2	Type of attribute Supplier.email is not specified, using xs:string as default.
⚠	3	Type of attribute Supplier.phone is not specified, using xs:string as default.

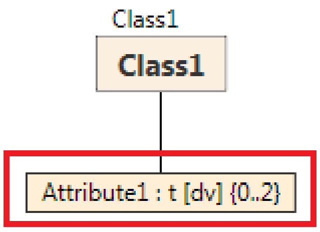
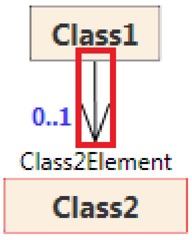
12.3. Translation table

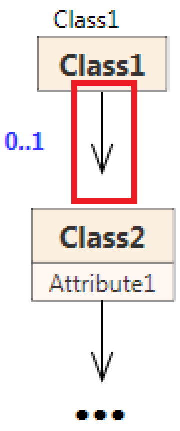
XCase PSM element	XML Schema element

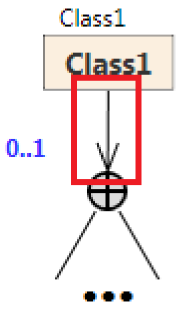
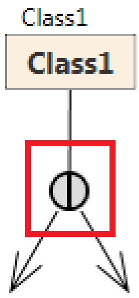
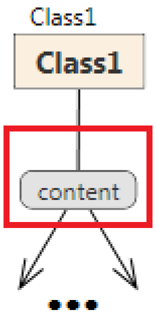
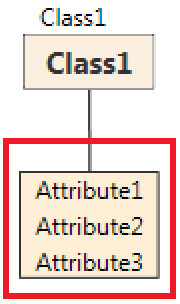
Class with an element label  Class1Element
Class1
Attribute1
Attribute2
1
1
Class2
Class2
•••
	<xs:complexType name="Class1">
	  <xs:sequence>
	    <xs:element name="Class2" type="Class2" />
	    …
	  </xs:sequence>
	  <xs:attribute name="Attribute1"… />
	  <xs:attribute name="Attribute2" .. />
	</xs:complexType>
	
	• If it is a root and is not abstract or is abstract, but has non-abstract specialization without an element label, then (additionally to the previous complex type):
	
	<xs:element name="Class1Element" type="Class1"/>

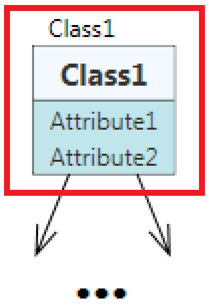
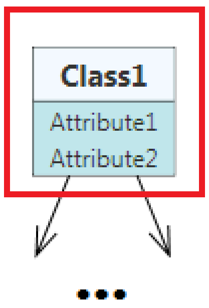
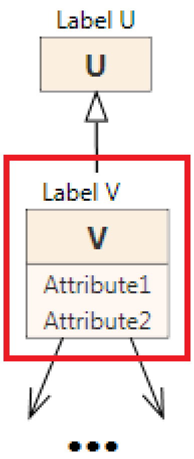
Class without an element label  Class1
Attribute1
Attribute2
1
Class2
Class2
•••
	<xs:group name="Class1-c">
	  <xs:sequence>
	    <xs:element name="Class2" type="Class2" />
	    ..
	  </xs:sequence>
	</xs:group>
	
	<xs:attributeGroup name="Class1-a">
	  <xs:attribute name="Attribute1"… />
	  <xs:attribute name="Attribute2" … />
	</xs:attributeGroup>
	
	<xs:attributeGroup name="Class1-a-opt">
	  <xs:attribute name="Attribute1" **use="optional"**/>
	  <xs:attribute name="Attribute2" **use="optional"**/>
	</xs:attributeGroup>

Attribute of a class (*type t, default value dv, multiplicity m…n*) 	• If m > 0 (minimal multiplicity is 1 or higher): **<xs:attribute name="Attribute1" type="t" default="dv" />** • If m=0 (minimal multiplicity is 0): **<xs:attribute name="Attribute1" type="t" default="dv" use="optional" />** A maximal multiplicity > 1 is ignored.

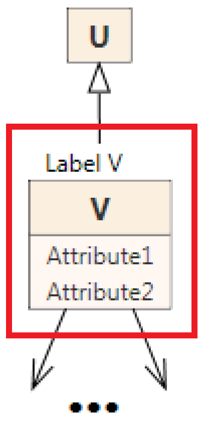
XCase PSM element	**XML Schema element**
Attribute in an attribute container **(*type t, default value dv, multiplicity m..n*)** 	**<xs:element name="Attribute1" type="xs:t" default="dv" minOccurs="0" maxOccurs="2" />** • minOccurs/maxOccurs attributes are generated according to Attribute1 multiplicity
Association going to a node with an element label 	**<xs:element name="Class2Element" type="Class2" minOccurs="0"/>** • minOccurs/maxOccurs attributes are generated according to Class2 multiplicity

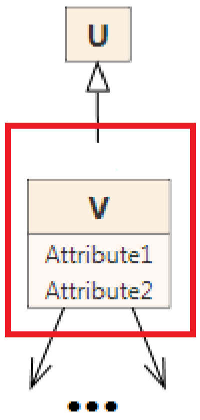
Association going to a class without an element label (multiplicity m..n) 	<xs:complexType name="Class1">
	  <xs:sequence>
	    <xs:group ref="Class2-c" minOccurs="0" />
	  </xs:sequence>
	
	If m=0:
	  <xs:attributeGroup ref="Class2-a-opt" />
	otherwise
	  <xs:attributeGroup ref="Class2-a " />
	
	</xs:complexType>
	• minOccurs/maxOccurs attributes of the group are generated according to the multiplicity

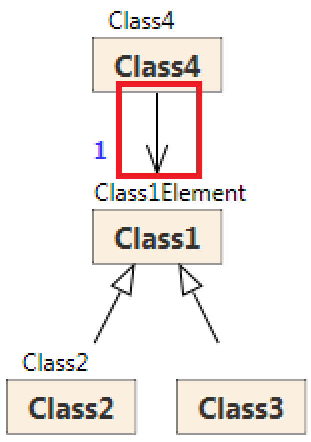
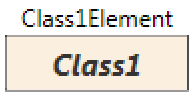
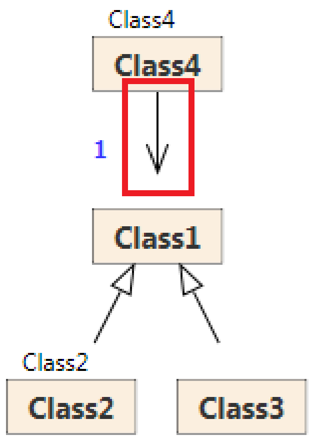
XCase PSM element	XML Schema element
Assoc. going to a class union 	**<xs:choice minOccurs="0" maxOccur="1"/>** … **</xs:choice>** • minOccurs/maxOccurs attributes of the choice are generated according to the class union multiplicity
Content choice 	**<xs:choice>** … **</xs:choice>** <-- for each PSM class Class2 without an element label in the content of the container --> **<xs:attributeGroup ref="Class2-a-opt">**
Content container 	**<xs:element name="content">** <xs:complexType> <xs:sequence> … </xs:sequence> </xs:complexType> **</xs:element>**
Attribute container 	<xs:complexType name="Class1"> <xs:sequence> **<xs:element name="Attribute1" … />** **<xs:element name="Attribute2" … />** **<xs:element name="Attribute3" … />** </xs:sequence> </xs:complexType>

XCase PSM element	XML Schema element
Structural representative of a PSM class *Class1’* with an element label 	**<xs:complexType name="Class1">** <xs:sequence> <xs:group ref="Class1‘-c" /> ... </xs:sequence> <xs:attributeGroup ref="Class1‘-a" /> <xs:attribute name=“Attribute1“ ... /> <xs:attribute name=“Attribute2“ ... /> **</xs:complexType>**
Structural representative of a PSM class *Class1’* without an element label 	**<xs:group name="Class1">** <xs:sequence> <xs:group ref="Class1‘-c" /> ... </xs:sequence> **</xs:group>** **<xs:attributeGroup name="Class1">** <xs:attributeGroup ref="Class1‘-a" /> <xs:attribute name=“Attribute1“ ... /> <xs:attribute name=“Attribute2“ ... /> **</xs:attributeGroup>**
Specialization V of a class U where U has an element label 	**<xs:complexType name="V">** <xs:complexContent> **<xs:extension base="U">** <xs:sequence> … </xs:sequence> <xs:attribute name="Attribute1" … /> <xs:attribute name="Attribute2" … /> **</xs:extension>** </xs:complexContent> **</xs:complexType>** • if U is a root and V has a different element label from V then **<xs:element name="Label V" type="V" />**

XCase PSM element	XML Schema element

Specialization V of a class U where U does not have an element label and V does

	<xs:complexType name="V">
	  <xs:sequence>
	    <!-- if the content of U is not empty -->
	    <xs:group ref="U-c" />
	      …
	  </xs:sequence>
	  <!-- if U has attributes -->
	  <xs:attributeGroup ref="U-a" />
	  <xs:attribute name="Attribute1" … />
	  <xs:attribute name="Attribute2" … />
	</xs:complexType>
	
	• if U is a root then
	<xs:element name="Label V" type="V" />

Specialization V of a class U where neither U nor V have an element label

	<xs:group name="V-c">
	  <xs:sequence>
	    <!-- if the content of U is not empty -->
	    <xs:group ref="U-c" />
	    …
	  </xs:sequence>
	</xs:group>
	<xs:attributeGroup name="V-a">
	  <!-- if U has attributes -->
	  <xs:attributeGroup ref="U-a" />
	  <xs:attribute name="Attribute1" … />
	  <xs:attribute name="Attribute2" … />
	</xs:attributeGroup>

XCase PSM element	XML Schema element
An association E going to a specialized class with an element label  Class4 Class4 1 Class1Element Class1 Class2 Class2 Class3	**<xs:choice>** <xs:element name="Class1Element" type="Class1" /> <xs:element name="Class2" type="Class2" /> **</xs:choice>** • If Class1 is abstract:  **<xs:choice>** <xs:element name="Class1Element" type="Class1" /> <xs:element name="Class2" type="Class2" /> **</xs:choice>** + **<xs:complexType name="Class1" abstract="true">**
An association E going to a specialized class without an element label  Class4 Class4 1 Class1 Class2 Class2 Class3	**<xs:choice>** <xs:group ref="Class1-c" /> <xs:element name="Class2" type="Class2" /> <xs:group ref="Class3-c" /> **</xs:choice>** **<!-- if Class3 has attributes -->** **<xs:attributeGroup ref="Class1-a" />** • If Class1 is an abstract class: **<xs:choice>** <xs:element name="Class2" type="Class2" /> <xs:group ref="Class3-c" /> **</xs:choice>**

13. Sample Diagrams

In this section, we provide sample PIM and PSM diagrams modeling a company that produces and sells products. PIM diagrams provide a conceptual description of the domain. PSM diagrams provide description of XML formats applied by the company to communicate with its suppliers and customers.

All the diagrams can be found in the project *ABCCompany.XCase* on XCase CD.

13.1. PIM Diagrams

Persons. Figure 3 depicts a PIM diagram modeling hierarchy of persons. It demonstrates how XCase allows modeling classes and their specializations using generalization connections. E.g., there is class *Person* modeling persons in general. It is specialized by classes *Customer* and *Employee* modeling customers and employees, respectively. *Employee* is further specialized to more specific types.

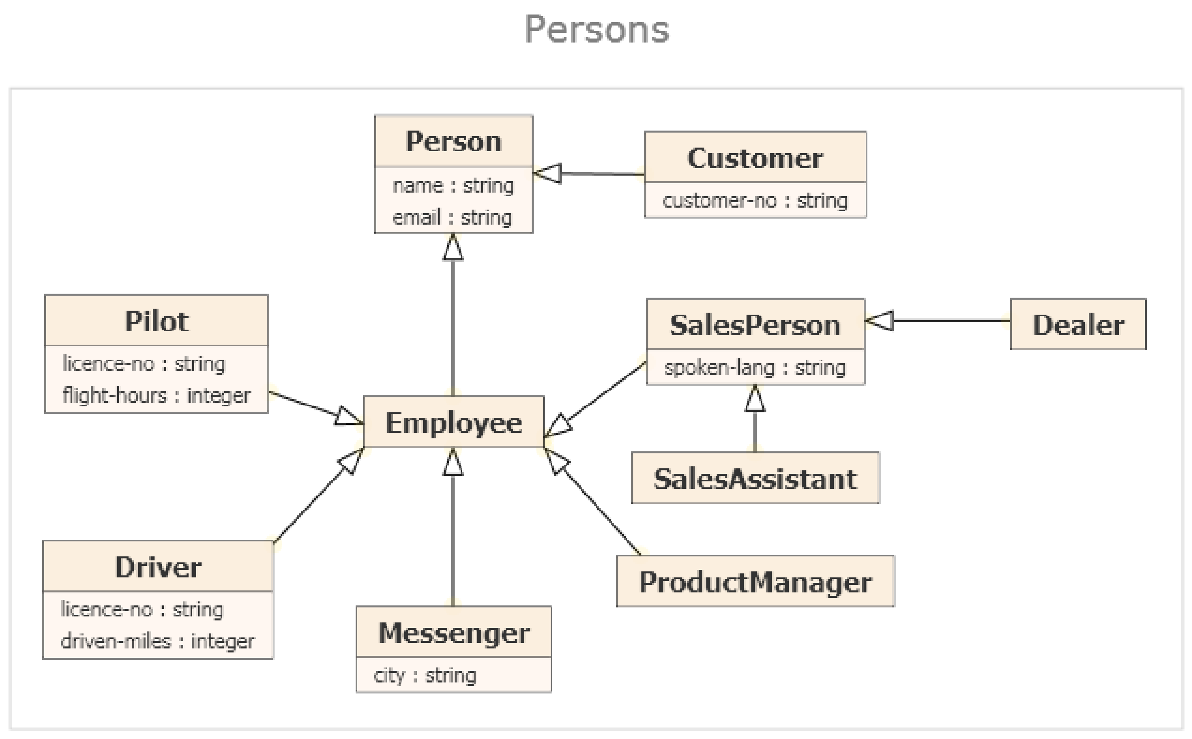


Figure 3

Sales. Figure 4 depicts a PIM diagram modeling product sales. It demonstrates modeling associations in XCase. The core class is *Purchase* modeling purchase orders. It is composed of items as modeled by a composition connecting *Purchase* and *Item*. Each item is related to the purchased product.

Production. The last PIM diagram, which is depicted in Figure 5, models supplies of parts for production. It demonstrates modeling n-ary association classes. An association class *Supply* has three endpoints, i.e. connects three classes. It models that parts are supplied by suppliers to product sets.

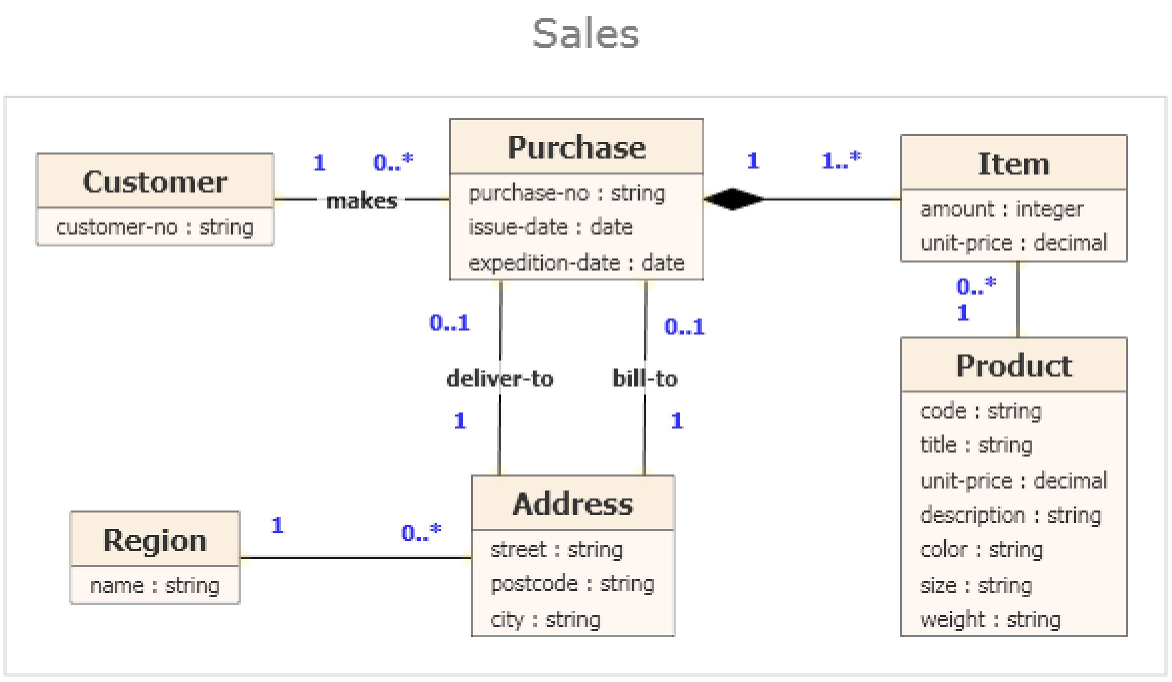


Figure 4

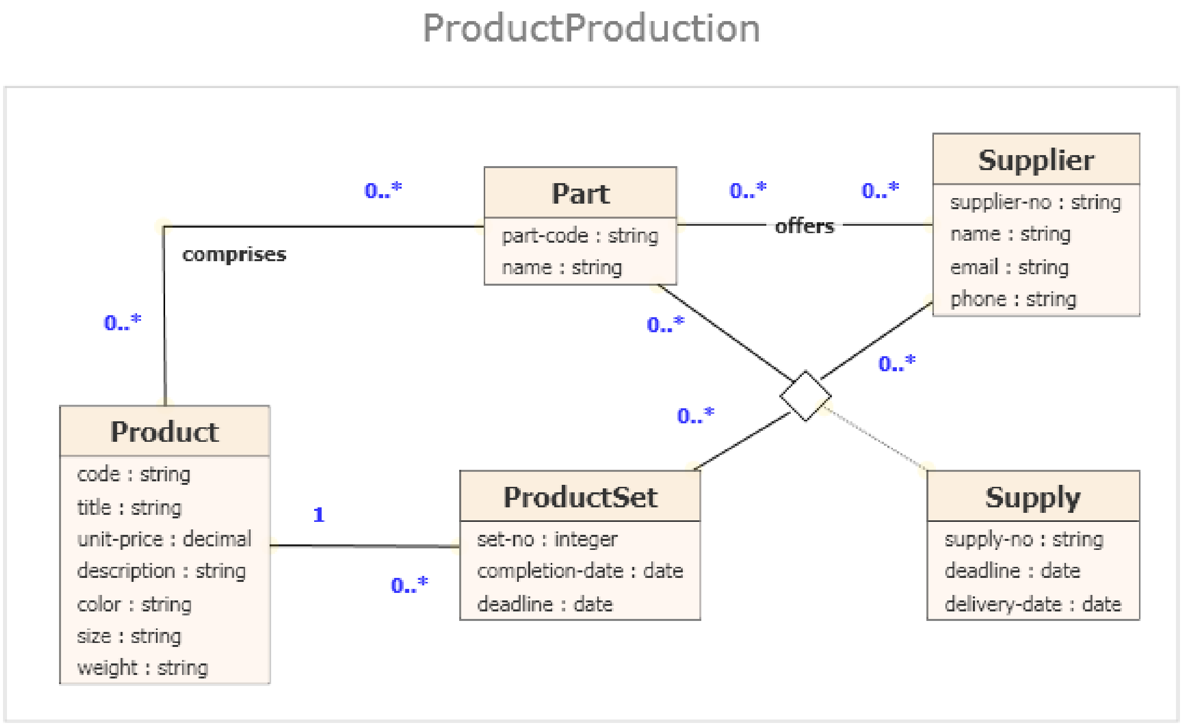


Figure 5

13.2. PSM Diagrams

Purchase Request and Response XML Formats. Figures 6 and 7 depict two PSM diagrams modeling XML formats for purchase requests and responses, respectively. *PurchaseRequest-Format* has two PSM classes representing *Customer*. *NewCustomer* is applied by non-registered customers while *RegCustomer* is applied by the others. Further, it contains two representations of *Address*, namely *DeliveryAddress* and *BillAddress*. Both model the same structure but have different element labels. The other is a structural representative of the former, i.e. its structure is modeled by *DeliveryAddress*. *PurchaseResponseFormat* represents *Purchase* as well but in a different structure.

PurchaseRequestFormat

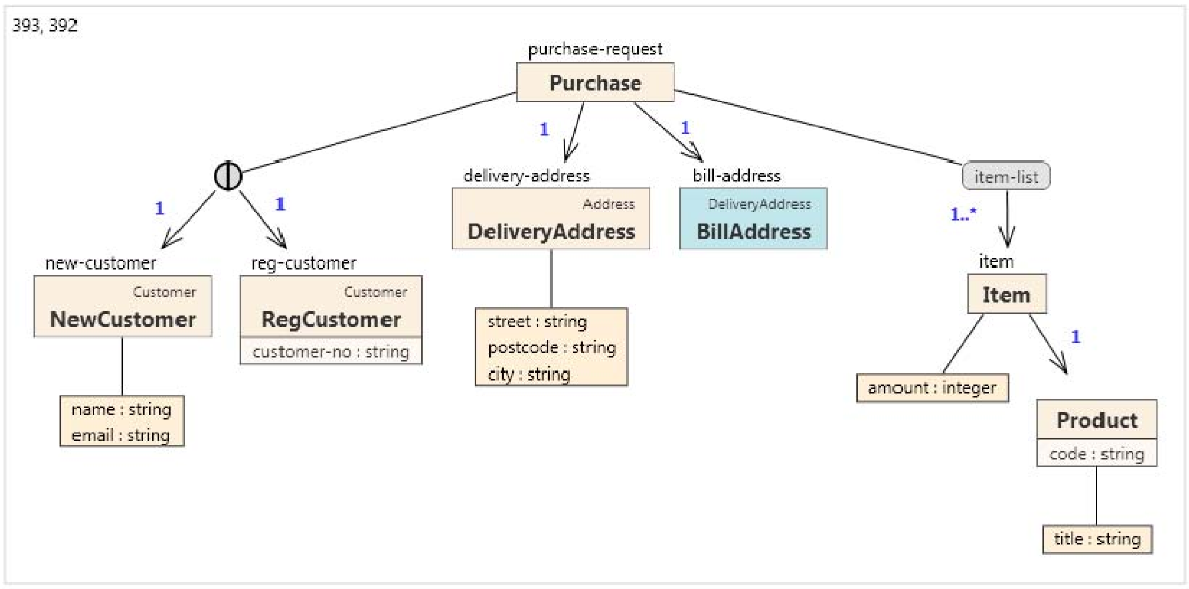


Figure 6

PurchaseResponseFormat

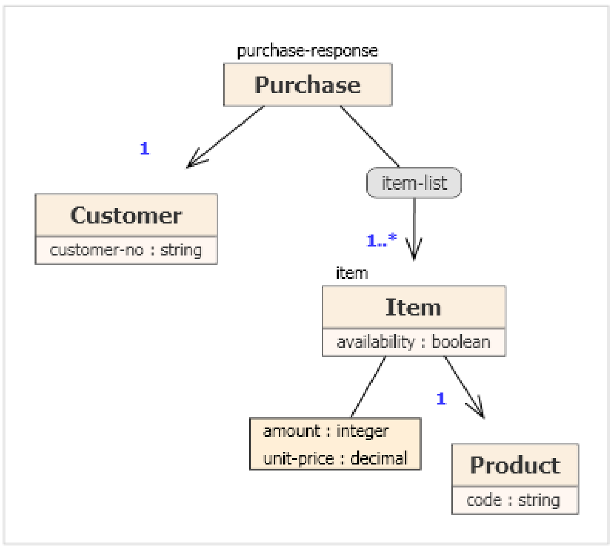


Figure 7

Purchase Request Format generated XML schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://kocour.ms.mff.cuni.cz/xcase/company/"
 targetNamespace="http://kocour.ms.mff.cuni.cz/xcase/company/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="purchase-request" type="Purchase" />

  <xs:complexType name="Purchase">
    <xs:sequence>
      <xs:choice>
        <xs:element name="new-customer" type="NewCustomer" />
        <xs:element name="reg-customer" type="RegCustomer" />
      </xs:choice>
      <xs:element name="delivery-address" type="DeliveryAddress" />
      <xs:element name="bill-address" type="BillAddress" />
      <xs:element name="item-list">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="item" type="Item" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="NewCustomer">
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="email" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="RegCustomer">
    <xs:attribute name="customer-no" type="xs:string" use="required" />
  </xs:complexType>

  <xs:complexType name="DeliveryAddress">
    <xs:sequence>
      <xs:group ref="DeliveryAddress-c" />
    </xs:sequence>
  </xs:complexType>

  <xs:group name="DeliveryAddress-c">
    <xs:sequence>
      <xs:element name="street" type="xs:string" />
      <xs:element name="postcode" type="xs:string" />
      <xs:element name="city" type="xs:string" />
    </xs:sequence>
  </xs:group>

  <xs:complexType name="BillAddress">
    <xs:sequence>
      <xs:group ref="DeliveryAddress-c" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Item">
    <xs:sequence>
      <xs:element name="amount" type="xs:int" />
      <xs:group ref="Product-c" />
    </xs:sequence>
    <xs:attributeGroup ref="Product-a" />
  </xs:complexType>

  <xs:group name="Product-c">
    <xs:sequence>
      <xs:element name="title" type="xs:string" />
    </xs:sequence>
  </xs:group>
  <xs:attributeGroup name="Product-a">
    <xs:attribute name="code" type="xs:string" use="required" />
  </xs:attributeGroup>

</xs:schema>
```

Purchase Response Format generated XML schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http//www.example.org/"
 targetNamespace="http//www.example.org/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="purchase-response" type="Purchase" />

  <xs:complexType name="Purchase">
    <xs:sequence>
      <xs:element name="item-list">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="item" type="Item" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attributeGroup ref="Customer-a" />
  </xs:complexType>

  <xs:attributeGroup name="Customer-a">
    <xs:attribute name="customer-no" type="xs:string" use="required" />
  </xs:attributeGroup>

  <xs:complexType name="Item">
    <xs:sequence>
      <xs:element name="amount" type="xs:int" />
      <xs:element name="unit-price" type="xs:decimal" />
    </xs:sequence>
    <xs:attribute name="availability" type="xs:boolean" use="required" />
    <xs:attributeGroup ref="Product-a" />
  </xs:complexType>

  <xs:attributeGroup name="Product-a">
    <xs:attribute name="code" type="xs:string" use="required" />
  </xs:attributeGroup>

</xs:schema>
```

Sales Report XML Format. Figure 8 depicts a PSM diagram modeling an XML format for product managers viewing sales reports. It represents purchases grouped by regions, they are billed to, and products sold. The formal semantics of the grouping is described by nesting joins viewed by the tool in the Properties window. Informally, the association going from *Region* to *Product* models that each region contains a list of products ordered from that region. The association going from *Product* to *Purchase* models that each product contains a list of purchase orders from that region and purchasing the product.

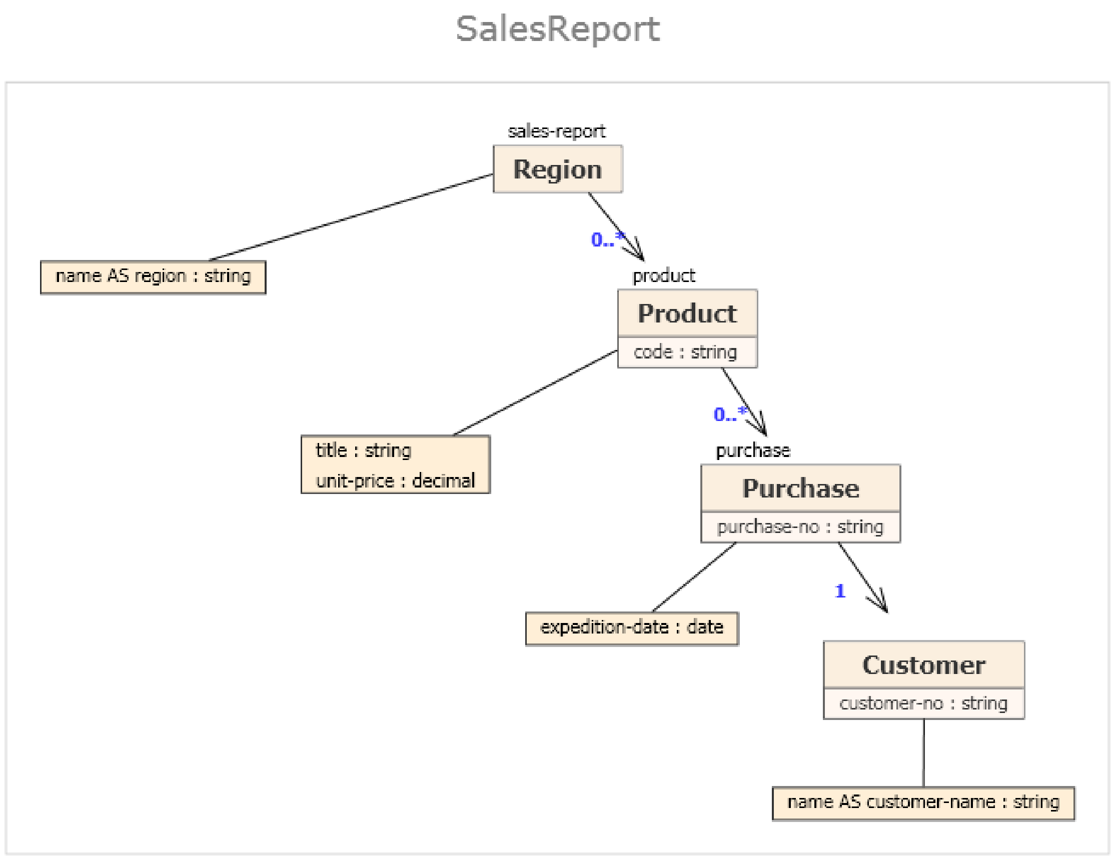


Figure 8

Sales Report generated XML schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http//www.example.org/"
 targetNamespace="http//www.example.org/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="sales-report" type="Region" />

  <xs:complexType name="Region">
    <xs:sequence>
      <xs:element name="region" type="xs:string" />
      <xs:element name="product" type="Product" minOccurs="0"
                                                maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Product">
    <xs:sequence>
      <xs:element name="title" type="xs:string" />
      <xs:element name="unit-price" type="xs:decimal" />
      <xs:element name="purchase" type="Purchase" minOccurs="0"
                                             maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="code" type="xs:string" use="required" />
  </xs:complexType>

  <xs:complexType name="Purchase">
    <xs:sequence>
      <xs:element name="expedition-date" type="xs:date" />
      <xs:group ref="Customer-c" />
    </xs:sequence>
    <xs:attribute name="purchase-no" type="xs:string" use="required" />
    <xs:attributeGroup ref="Customer-a" />
  </xs:complexType>

  <xs:group name="Customer-c">
    <xs:sequence>
      <xs:element name="customer-name" type="xs:string" />
    </xs:sequence>
  </xs:group>
  <xs:attributeGroup name="Customer-a">
    <xs:attribute name="customer-no" type="xs:string" use="required" />
  </xs:attributeGroup>

</xs:schema>
```

Transport Detail XML format. Figure 9 depicts a PSM diagram modeling an XML format for detailed information about transport source and target destinations. It demonstrates how PIM classes and their specializations can be represented in PSM diagrams. Each transport goes from a stock to a destination which is a stock or customer. Therefore, there is PSM class *Destination* with two specializations *DstStock* and *Customer*. *Destination* models that for each destination there is an address and region. *DstStock* extends this representation of destination with stock number and capacity (since it is a structural representative of *SrcStock*). *Customer* extends it with customer number and name. Moreover, *Destination* is abstract. It means that instances of *Destination* can not be represented in the modeled XML format. Only instances of the specializations can be represented since they are not abstract. Each stock is represented as an XML element *to-stock* while each customer as an XML element *to-customer*.

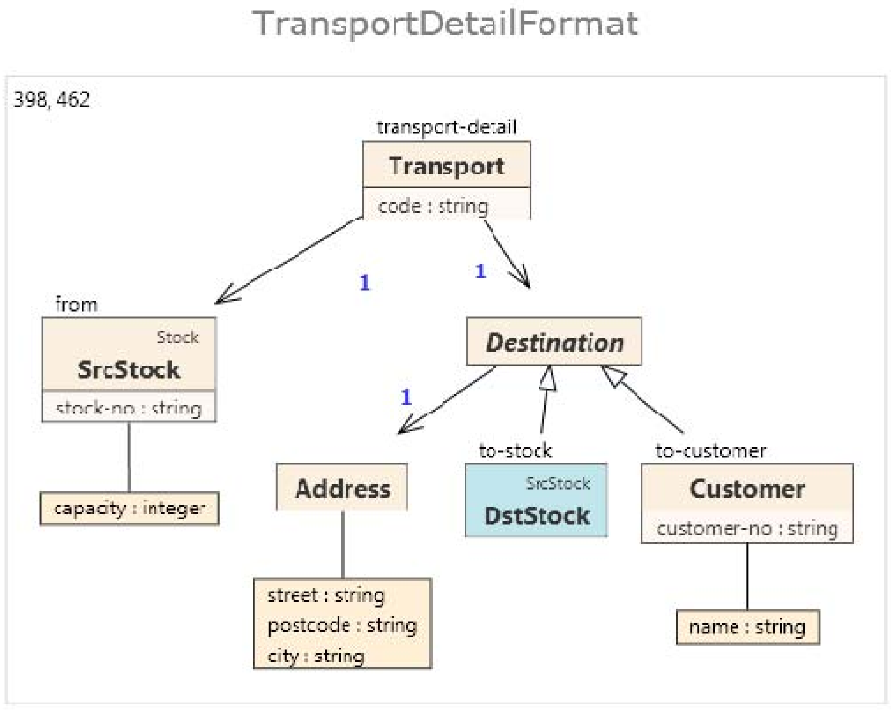


Figure 9

Transport Detail Format generated XML schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://kocour.ms.mff.cuni.cz/xcase/company/"
 targetNamespace="http://kocour.ms.mff.cuni.cz/xcase/company/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="transport-detail" type="Transport" />

  <xs:complexType name="Transport">
    <xs:sequence>
      <xs:element name="from" type="SrcStock" />
      <xs:choice>
        <xs:element name="to-stock" type="DstStock" />
        <xs:element name="to-customer" type="Customer" />
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="code" type="xs:string" use="required" />
  </xs:complexType>
  <xs:complexType name="SrcStock">
    <xs:sequence>
      <xs:group ref="SrcStock-c" />
    </xs:sequence>
    <xs:attributeGroup ref="SrcStock-a" />
  </xs:complexType>

  <xs:group name="SrcStock-c">
    <xs:sequence>
      <xs:element name="capacity" type="xs:int" />
    </xs:sequence>
  </xs:group>
  <xs:attributeGroup name="SrcStock-a">
    <xs:attribute name="stock-no" type="xs:string" use="required" />
  </xs:attributeGroup>

  <xs:group name="Destination-c">
    <xs:sequence>
      <xs:group ref="Address-c" />
    </xs:sequence>
  </xs:group>

  <xs:group name="Address-c">
    <xs:sequence>
      <xs:element name="street" type="xs:string" />
      <xs:element name="postcode" type="xs:string" />
      <xs:element name="city" type="xs:string" />
    </xs:sequence>
  </xs:group>

  <xs:complexType name="DstStock">
    <xs:sequence>
      <xs:group ref="Destination-c" />
      <xs:group ref="SrcStock-c" />
    </xs:sequence>
    <xs:attributeGroup ref="SrcStock-a" />
  </xs:complexType>

  <xs:complexType name="Customer">
    <xs:sequence>
      <xs:group ref="Destination-c" />
      <xs:element name="name" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="customer-no" type="xs:string" use="required" />
  </xs:complexType>

</xs:schema>
```

References

1] M. Necasky: Conceptual Modeling for XML. Ph.D. thesis.
Faculty of Mathematics and Physics, Charles University, Prague. May 2008.
http://www.necasky.net/thesis.pdf

2] J. Miller, J. Mukerji. MDA Guide Version 1.0.1. OMG.

3] W3C, XML Schema Part 0: Primer Second Edition, October 2004,
http://www.w3.org/TR/xmlschema-0/

4] Unified Modeling Language (UML), Superstructure, V2.1.2. OMG.
http://www.omg.org/docs/formal/07-11-02.pdf

5] Unified Modeling Language (UML), Infrastructure, V2.1.2. OMG.
http://www.omg.org/docs/formal/07-11-04.pdf

XCase CD Contents

- XCase Installer in the root folder
- User's and Programmer's Documentation in the *doc* folder
- Source code in the *src* folder
- Documentation generated from the source code in the *gendoc* folder

XCase Folder Contents (after installation)

- XCase
- Examples in the *Examples* folder
- User's and Programmer's Documentation in the *Documentation* folder