INSPIRE

Infrastructure for Spatial Information in Europe

Consolidation Team

Tutorial: Using Enterprise Architect with a central UML repository

Title	Training: Using Enterprise Architect with a central UML repository
Creator	M. Lutz & A. Friis-Christensen
Date (creation)	2008-03-28
Date (last update)	2012-02-07
Subject	Training on Enterprise Architect and central repository
Publisher	JRC CT
Type	Text
Description	This document is a tutorial for the use of EA and the central UML repository
Contributor	JRC CT
Format	PDF
Source	None
Rights	Public
Identifier	UML_repository_tutorial.doc
Language	En
Relation	n/a
Coverage	Project duration

These are Dublin Core metadata elements. See for more details and examples http://www.dublincore.org/.

Table of contents

Preface

This document is a tutorial on how to work with the consolidated UML repository used by the INSPIRE Thematic Working Groups (TWGs) for the data specifications work. It covers three main topics:

Section 1 gives a short introduction on creating UML static models in Enterprise Architect (EA) and on some conventions on how to document UML models in INSPIRE. Participants familiar with using EA can skim this part, but should pay attention to the sections on INSPIRE modeling conventions.

Section 2 deals with setting up version control in EA with subversion and introduces the central subversion repository set up at JRC.

Section 3 describes how to work with the repository.

At the end of each section, a number of exercises are included. These exercises refer to a test repository that can be used freely to get familiar with the setup and tools. To get access to this test repository, please contact Michael Lutz at michael.lutz@jrc.ec.europa.eu.

The screenshots in the text are based on EA version 7.5 on Windows XP.

1 Creating UML models in EA

This section gives a short introduction on creating UML static models in Enterprise Architect (EA) and on some conventions on how to document UML models in INSPIRE. Participants familiar with using EA can skim this part, but should pay attention to the sections on INSPIRE modeling conventions.

Convention: INSPIRE modelling conventions are highlighted like this.

1.1 Getting Started

Enterprise Architect (EA) starts with the screen shown in Figure 1.1. On the left, you see the *Toolbox* pane, on the upper right the *Project Browser / Model Views / Resources* pane and on the lower right the *Pan and Zoom/Notes/Properties/Tagged Values* pane. All of these panes can be hidden by clicking on the little pin on the upper right of the pane. They then become tabs on the right or left side of the EA window (see e.g. the *Layout Palette* and *EA – Getting Started* tabs on the left). You can also customize the layout of the screen by selecting the components to show in the VIEW menu item.

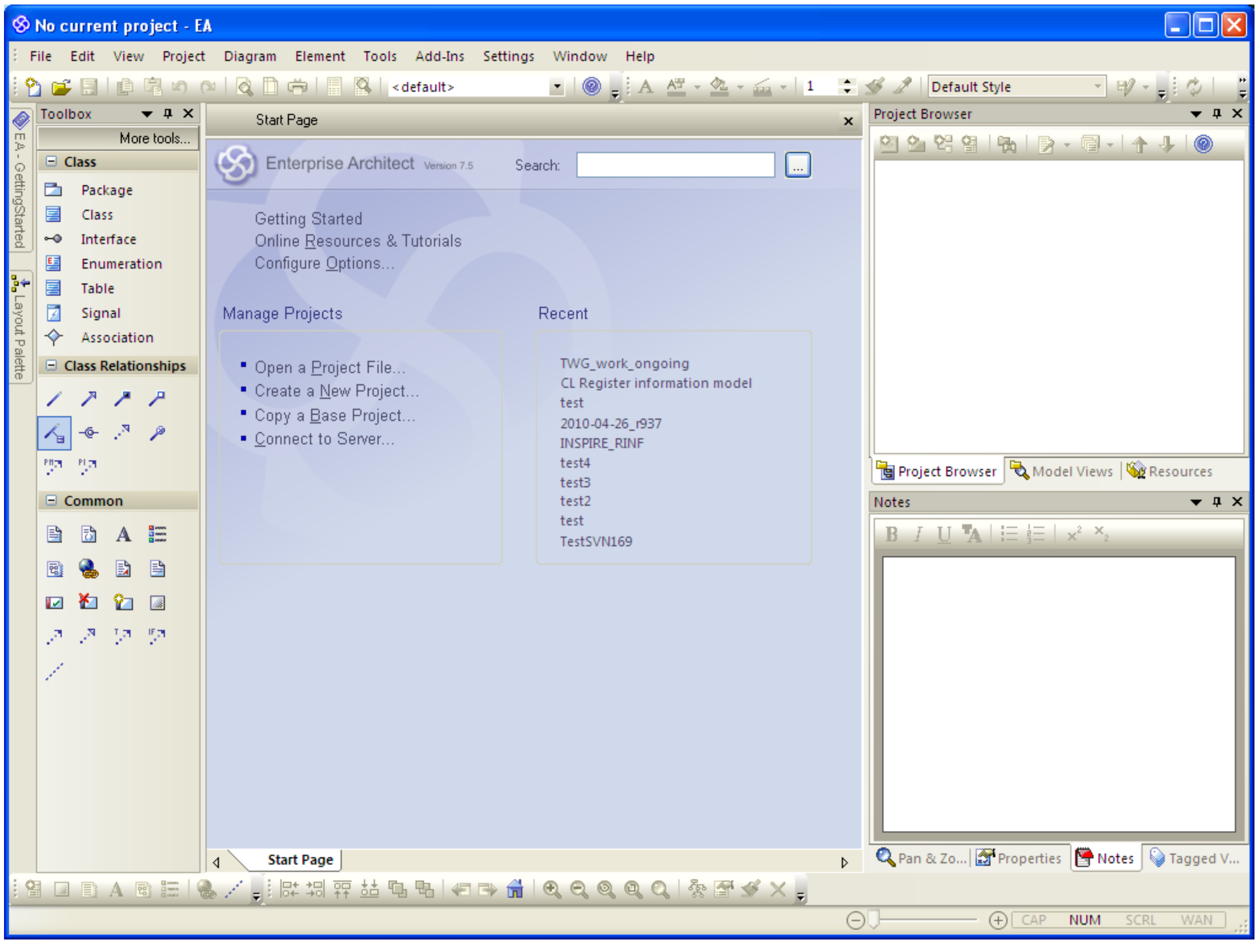


Figure 1.1: EA start screen.

You can open existing projects using MANAGE PROJECTS - OPEN A PROJECT FILE on the *Start Page* or the icon in the toolbar. Recently opened projects are shown in the *Recent* section of the *Start Page*. To create a new project, select MANAGE PROJECTS - CREATE A NEW PROJECT on the *Start Page*.

1.2 The UML INSPIRE profile

For INSPIRE, a number of specific stereotypes (e.g. «featureType» and «voidable») have been defined in a UML profile. The profile (current version 1.1) is available as an XML document from CIRCA at http://circa.europa.eu/Members/irc/jrc/imaco2000/library?l=/drafting_folders/data_specifications/thematic_working/common_area/templates/inspire_umlprofile/_EN_1.1_&a=d.

In order to use this profile in EA, download the XML document and import it by selecting IMPORT PROFILE from the *UML Profiles* folder in the *Resources* tab (Figure 1.2). In the following dialogue, keep all the default settings. After the import, the *UML Profiles* folder should look like shown in Figure 1.3.

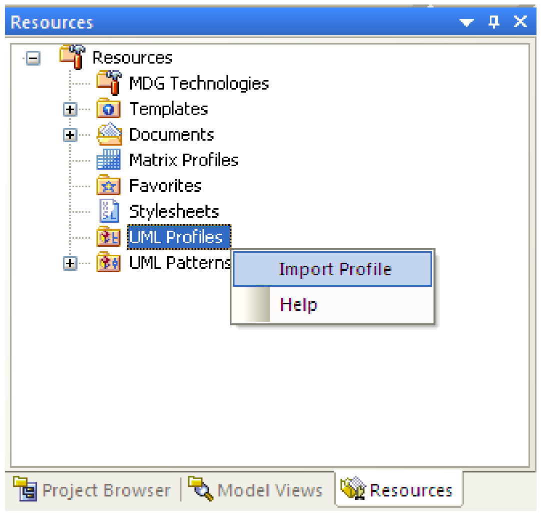


Figure 1.2: Importing a profile.

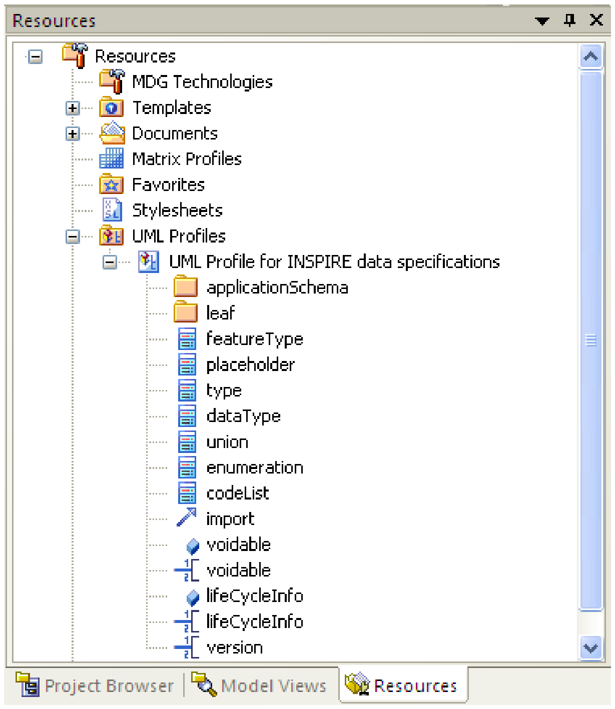


Figure 1.3: Resources tab after the import.

NOTE The stereotypes included in the INSPIRE UML profile reflect all the stereotypes that were required for the data specifications work in Annex I. If, during the work on the Annex II & III data specifications, further stereotypes are required, these should be added to the profile and documented in the Generic Conceptual Model.

1.3 Working with Packages and Classes

A new project contains a root package (or *model*) called "Model". The sub-packages of the model are called *views*, which represent the different UML views (e.g., the class view). Underneath views there are normal packages.

1.3.1 Working with models, views and packages

Models can be renamed by right-clicking on the name in the *Project Browser* and selecting RENAME MODEL (Figure 1.4). To create a new view, right-click on the model and select NEW VIEW in the context menu. Similarly, a package is created by right-clicking on a view or package and selecting ADD – ADD PACKAGE in the context menu. Views and packages can be renamed by double-clicking on them or by right-clicking on them and selecting PROPERTIES in the context menu.

Views and packages, like classes, attributes and other UML elements, can have stereotypes. Packages containing INSPIRE application schemas shall receive the stereotype «*applicationSchema*» (Figure 1.5). By clicking on the "..." icon on the right of the *Stereotype* combo box, you can open a dialogue that allows you to display only those stereotypes defined in the INSPIRE profile.

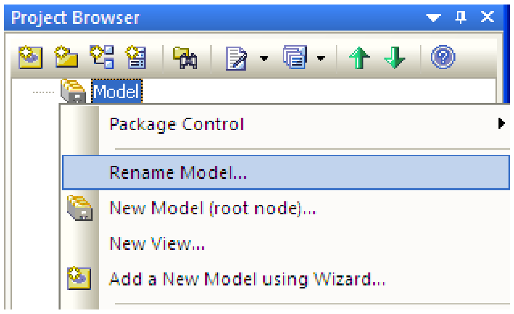
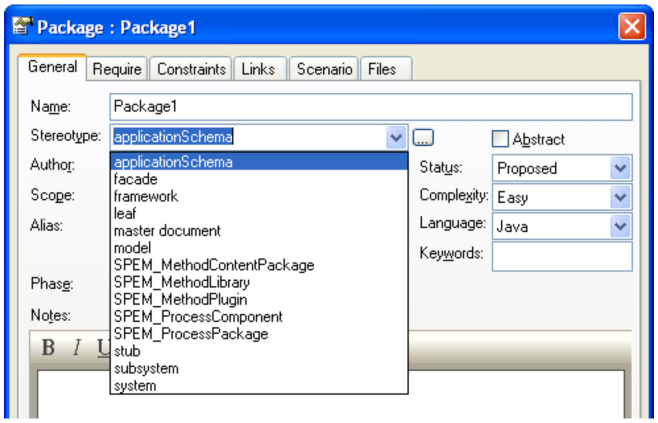


Figure 1.4: Renaming a model.



Name: Package1
Stereotype: applicationSchema
Author:
Scope:
Alias:
Phase:
Notes:
Status: Proposed
Complexity: Easy
Language: Java
Keywords:

applicationSchema
facade
framework
leaf
master document
model
SPEM_MethodContentPackage
SPEM_MethodLibrary
SPEM_MethodPlugin
SPEM_ProcessComponent
SPEM_ProcessPackage
stub
subsystem
system

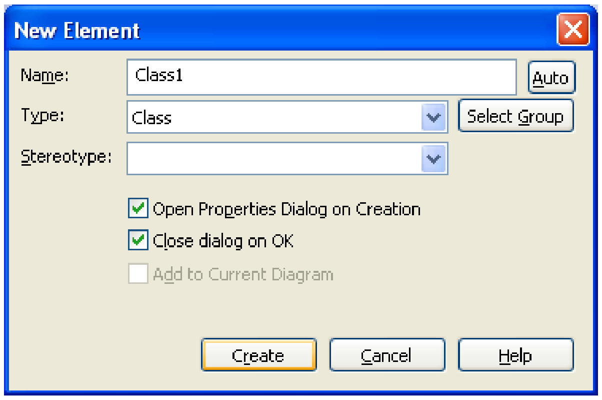
Figure 1.5: Specifying a package stereotype.

1.3.2 Working with classes

Classes can be added to a package either in the *Project Browser* or graphically in a diagram (see below). In the *Project Browser*, right-click on the package that is to contain the class and select ADD – ADD ELEMENT in the context menu. In the following dialogue (Figure 1.6), select "class" as a type and specify a name for the new class. When the *Open Properties Dialogue on Creation* checkbox is checked, a *Class Properties* window is opened, in which further details can be specified (Figure 1.7).

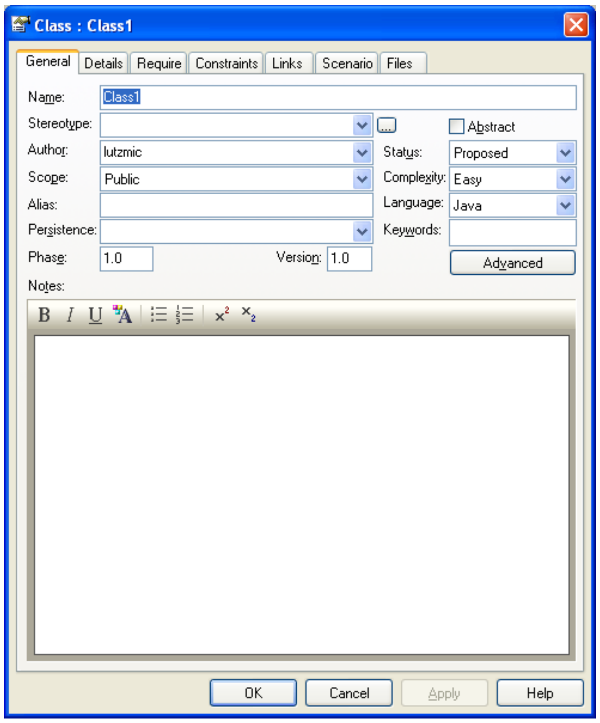
If a diagram is currently open you can select Add to Current Diagram (greyed out in Figure 1.6) to add the class to that diagram.

In the class *Class Properties* For example, you can use this dialogue to select a stereotype for this class, e.g. "featureType" (Figure 1.8).



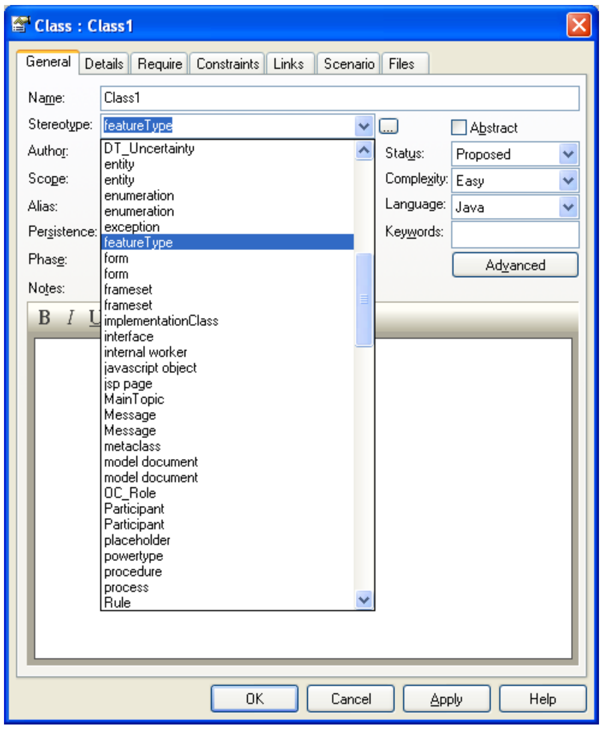
Name: Class1
Type: Class
Stereotype:
[✓] Open Properties Dialog on Creation
[✓] Close dialog on OK
[ ] Add to Current Diagram

Figure 1.6: Creating a class.



Name:	Class1
Stereotype:	
Abstract	☐
Author:	lutzmic
Status:	Proposed
Scope:	Public
Complexity:	Easy
Alias:	
Language:	Java
Persistence:	
Keywords:	
Phase:	1.0
Version:	1.0
Notes:	

Figure 1.7: Specifying the properties of a class.



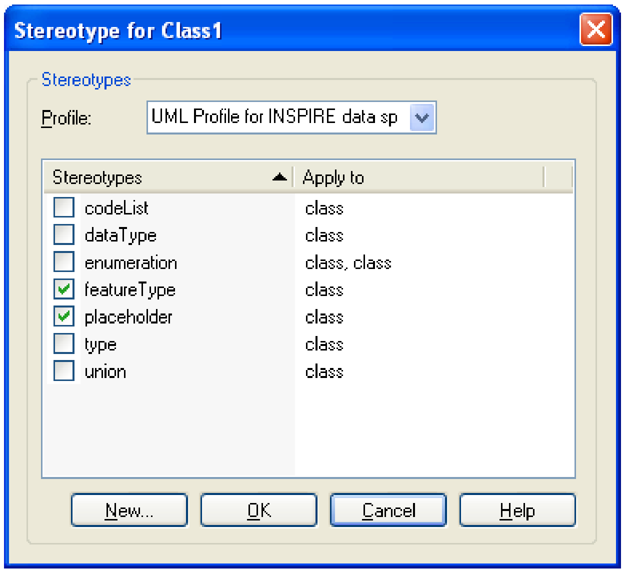
Name:	Class1
Stereotype:	featureType
Abstract	☐
Author:	
Status:	Proposed
Complexity:	Easy
Language:	Java
Keywords:	
Phase:	
Notes:	

DT_Uncertainty
entity
entity
enumeration
enumeration
exception
featureType
form
form
frameset
frameset
implementationClass
interface
internal worker
javascript object
jsp page
MainTopic
Message
Message
metaclass
model document
model document
OC_Role
Participant
Participant
placeholder
powertype
procedure
process
Rule

Figure 1.8: Specifying a stereotype.

By clicking on the "..." icon on the right of the *Stereotype* combo box, you can open a dialogue that allows you to display only those stereotypes defined in the INSPIRE profile (Figure 1.9). In this dialogue you can also select more than one stereotype, e.g. «placeholder» and «featureType» as shown in Figure 1.9.

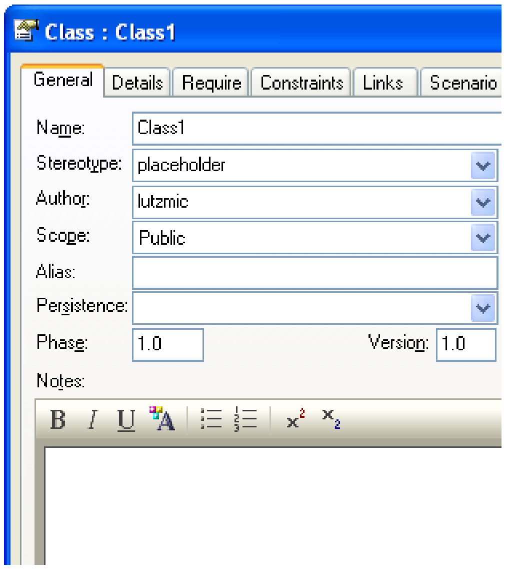
NOTE After closing the stereotypes dialogue, *only one* of the selected stereotypes will be displayed in the combo box of the class dialogue (Figure 1.10)! In order to see all stereotypes of a class (or attribute, see below), you need to open the stereotypes dialogue by clicking on the "…" button.



Profile: UML Profile for INSPIRE data sp

Stereotypes	Apply to
☐ codeList	class
☐ dataType	class
☐ enumeration	class, class
✓ featureType	class
✓ placeholder	class
☐ type	class
☐ union	class

Figure 1.9: Specifying more than one stereotype.



Name:	Class1
Stereotype:	placeholder
Author:	lutzmic
Scope:	Public
Alias:	
Persistence:	
Phase:	1.0
Version:	1.0
Notes:	

Figure 1.10: Only one of the specified stereotypes is shown in the combo box.

The Notes text field shall be used to document the class. The documentation shall include:

- a natural language name for that class, e.g. “speed limit” for the class SpeedLimit;
- a definition; and
- (optionally) a description, including further explanations, examples or references for the definition.

The documentation in the Notes field needs to follow a certain convention (see below). Based on this convention, the feature catalogue in the INSPIRE data specification and much of the content of the Implementing Rule are auto-generated.

Following this convention as closely as possibly greatly reduces the effort required for manual editing.

Convention: The three elements shall be documented in the Notes field as follows:

- The natural language name
    - o is introduced by the tag “-- Name --"
    - o shall be given in lower case.

- The definition
    - o is introduced by the tag “-- Definition --"
    - o shall be as concise as possible and not contain examples or further explanations (these shall go into the description)
    - o shall not start with “An address component is …”
    - o shall end with a full stop (“.”)
- If the definition uses another term that needs to be defined, this additional definition shall be included using the keyword DEFINITION (all upper case).

- The description:
    - o is introduced by the tag “-- Description --"
    - o is optional. If there is not further description, do not include the -- Description -- tag.
- The source reference of the definition can be included using the keyword SOURCE, after which a short name of the Source shall be included. The full reference shall be included in the INSPIRE data specification.
- Additional explanations in the descriptions shall be introduced by the keyword NOTE (all upper case). If there is more than one note, the notes shall be numbered: NOTE 1, NOTE 2, etc.
- Examples shall be introduced by the keyword EXAMPLE (all upper case). If there is more than one example, the examples shall be numbered: EXAMPLE 1, EXAMPLE 2, etc.

The following (partly fictional) example for the spatial object type “AddressComponent” shows how the convention should be applied:

> -- Name –
> address component
>
> -- Definition --
> Identifier or geographic name of a specific geographic area, location, or other spatial object which defines the scope of an address.
>
> DEFINITION Geographic name: A proper noun applied to a real world entity.
>
> -- Description --
> SOURCE [UPU-S21]

NOTE 1 Four different subclasses of address components are defined:
- Administrative unit name, which may include name of country, name of municipality, name of district
- Address area name like e.g. name of village or settlement
- Thoroughfare name, most often road name
- Postal descriptor

In order to construct an address, these subclasses are often structured hierarchically.

NOTE 2 It is the combination of the address locator and the address components, which makes a specific address spatial object readable and unambiguous for the human user.

EXAMPLE The combination of the locator "13" and the address components "Calle Mayor" (thoroughfare name), "Cortijo del Marqués" (address area name), "41037" (postal descriptor), "Écija", "Sevilla" and "España" (administrative unit names) makes this specific address.

Attributes can be created by selecting the *Attributes* button in the *Details* tab of the *Class Properties* dialogue (Figure 1.11) or in the context menu of the class in the *Project Browser*. In the example shown in Figure 1.12, a public attribute called *att1* of type *String* is specified.

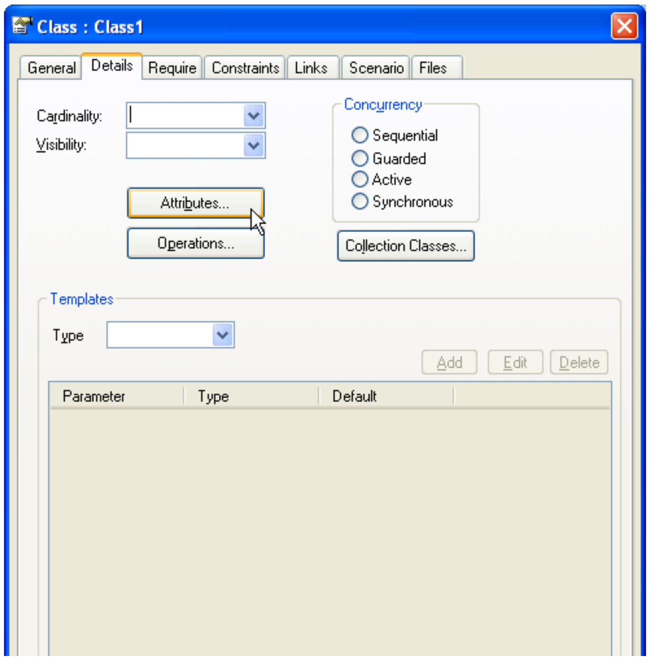
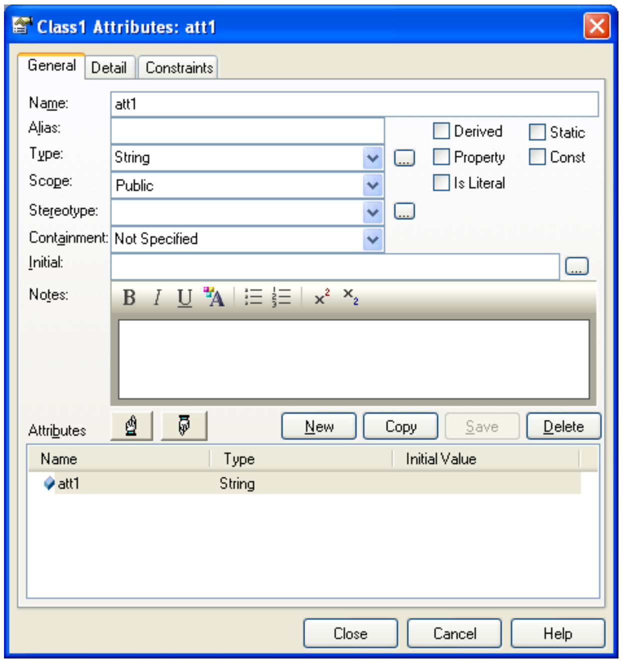


Figure 1.11: Opening the attribute dialogue.



Name: att1
Type: String
Scope: Public
Containment: Not Specified

Name	Type	Initial Value
att1	String	

Figure 1.12: Specifying a new attribute.

The type of an attribute can be specified in different ways: it can be typed in the *Type* combo box, it can be selected from the drop down list of the combo box, or it can be selected in a dialogue by clicking on the "…" button on the right of the combo box.

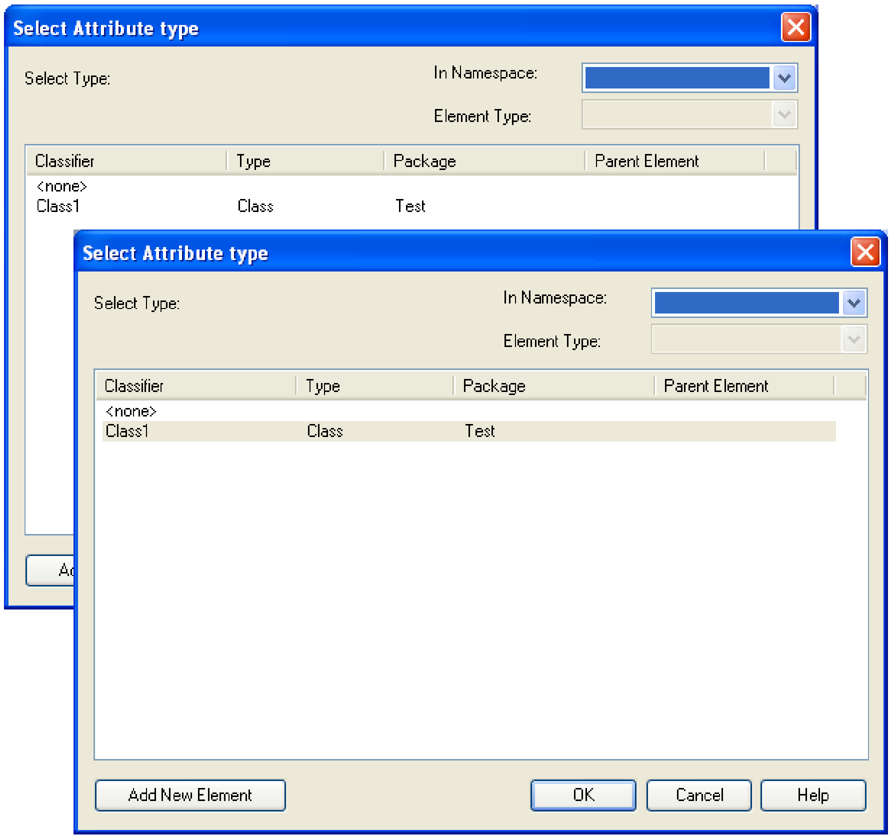
NOTE When just typing in the type name, you need to make sure that EA actually connects the entered type name with an existing type. You can verify this by clicking on the "…" button on the right of the combo box. If there is a connection to an existing type, the type is highlighted in grey (Figure 1.13, lower window). If not, no type is highlighted (Figure 1.13, upper window).

As for classes, stereotypes can be also be specified for both attributes and operations. If necessary, more than one stereotype can be specified in the same way as for classes (i.e., by selecting the "..." button next to the *Stereotype* combo box). Again, only one stereotype will be shown in the combo box.

The convention for documenting classes also applies for documenting attributes.

NOTE Since the INSPIRE data specifications specify a data model for data exchange, operations are typically not specified. If this is required for some reason, they can be specified in the same way as attributes.

To specify the multiplicity of the attribute, specify the lower and upper bound in the *Detail* tab.



Classifier	Type	Package	Parent Element
<none>			
Class1	Class	Test	

Figure 1.13: Ensuring that an attribute type is an existing type.

Classes can also contain constraints. These can be defined using the *Constraints* tab in the *Class Properties* dialogue.

Convention: For constraints in INSPIRE, select *OCL* in the *Type* drop down box and specify a name in the text field *Constraint.* The name should ideally give some indication on the content of the constraint.

The actual constraint is specified in the larger text box below. According to the GCM, OCL constraints should also be reported in natural language. In OCL, this can be done using the comment syntax: `/* ... */`. The OCL constraint itself should be specified below, starting with `inv:` for invariants. For example, Figure 1.14 specifies that an address shall have exactly one default geographic position. During the generation of the feature catalogue (Section 4), the natural language and the OCL expressions are extracted from this text.

For more information on OCL, see e.g. the *OCL 2.2 specification*[1], a *brief OCL 2.0 syntax*[2] or this *OCL tutorial*[3] (including an *OCL syntax checker*[4]).

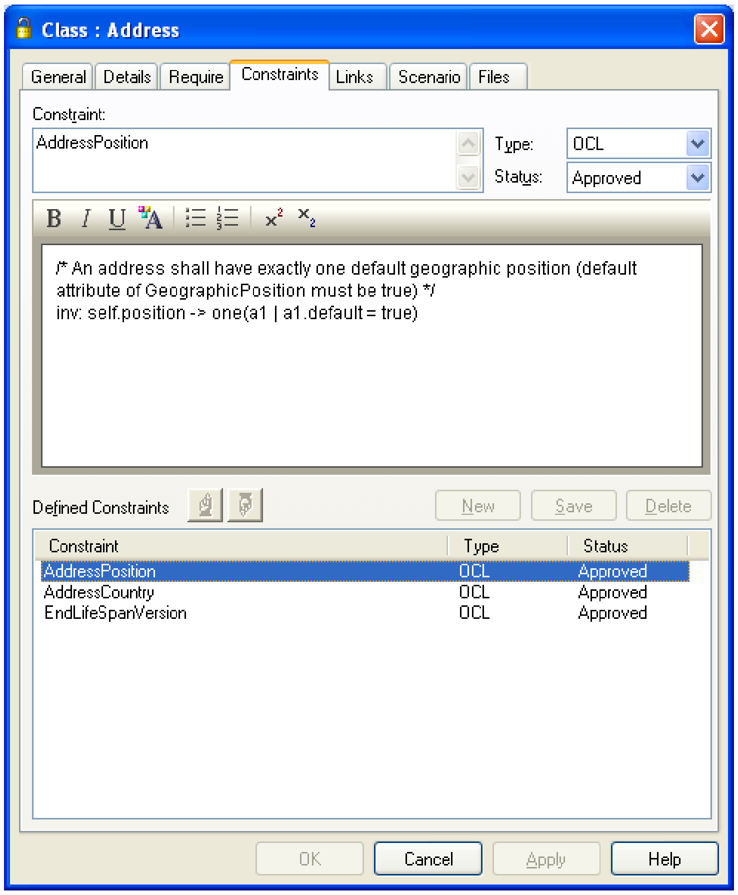
Convention: When referring to spatial objects of a specific type, use one of the following options:

- (preferred) you can refer to a FeatureTypeName spatial object, e.g. an AddressComponent (UpperCamelCase) spatial object, or
- your can refer to the natural language name of the feature type, e.g. an address component.

When referring to an attribute or association role of a spatial object, use one of the following options:

- (preferred) you can refer to the attributeName (lowerCamelCase) attribute or associationRoleName association (role), e.g. the name attribute of the Address spatial object, or
- you can refer to the attributeName (lowerCamelCase) or associationRoleName, e.g. the name of the address.

If the choice is between an elegant but ambiguous and a slightly awkward, but unambiguous phrasing, go for the latter. The people having to interpret and translate your text will thank you!



Constraint: AddressPosition

Type: OCL

Status: Approved

```
/* An address shall have exactly one default geographic position (default
attribute of GeographicPosition must be true) */
inv: self.position -> one(a1 | a1.default = true)
```

Constraint	Type	Status
AddressPosition	OCL	Approved
AddressCountry	OCL	Approved
EndLifeSpanVersion	OCL	Approved

Figure 1.14: Adding a constraint to a class.

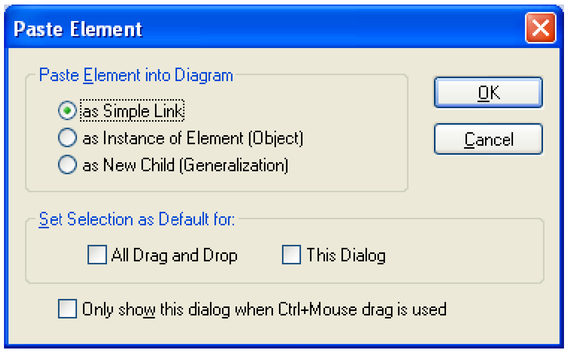
[1] http://www.omg.org/spec/OCL/2.2/PDF/
[2] http://www.csci.csusb.edu/dick/samples/ocl.html
[3] http://atlanmod.emn.fr/atldemo/oclturorial/
[4] http://atlanmod.emn.fr/atldemo/oclturorial/simpleOCLWebTester

1.4 Working with Diagrams

Diagrams can be added to a package by selecting ADD - ADD DIAGRAM in its context menu and providing a name and selecting the type of diagram (e.g. *Class*) in the following dialogue. By default, the name of the package is suggested as a name for the diagram.

1.4.1 Adding elements

You can add elements (e.g. classes) to a diagram by dragging and dropping them from the *Project Browser* into a diagram. Usually, classes should be pasted into a class diagram by checking the *as Simple Link* option (Figure 1.15). Classes can also be created directly in the diagram by clicking on the *Class* icon in the *Toolbox* on the left and then clicking on the diagram. This opens the *Class Properties* dialogue previously shown in Figure 1.7.


Paste Element

Paste Element into Diagram
[✓ as Simple Link]
as Instance of Element (Object)
as New Child (Generalization)

Set Selection as Default for:
☐ All Drag and Drop ☐ This Dialog

☐ Only show this dialog when Ctrl+Mouse drag is used

OK
Cancel

Figure 1.15: Pasting a class as a simple link.

You can also drag and drop classes from other packages into a diagram. These classes are shown in the diagram with their package name.

1.4.2 Adding associations

Relationships (or connectors) between classes can be created in two ways. When you select a class in a diagram, a small arrow appears next to it (Figure 1.16). You can drag and drop this arrow onto the class you want to connect to, and then select the type of relationship in a dialogue (Figure 1.17). Alternatively, you can select one of the Class Relationship icons in the *Toolbox* and then drag and drop a connection between two classes in the diagram. All relationships of a class are displayed in the *Links* tab of the *Class Properties* dialogue.

After creating an association, it is important to specify its properties (by double-clicking on the association), and in particular its direction (unspecified, bi-directional, source -> target or target -> source) (Figure 1.18) and its source and/or target roles (in the *Source Role* or *Target Role* tab, see Figure 1.19).

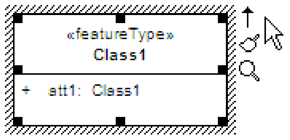


Figure 1.16: Selecting a class in a class diagram.

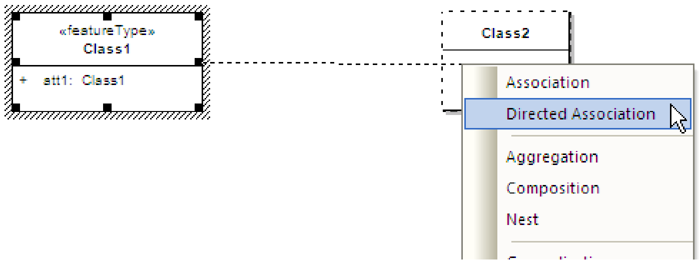
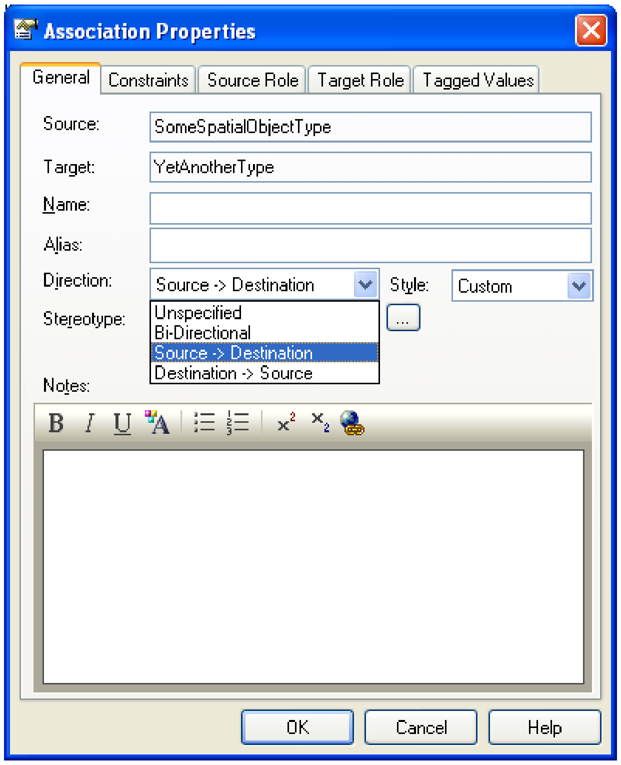


Figure 1.17: Selecting the type of connector.

Association roles should be documented in the same way as attributes, i.e. you should specify a name for the role using lowerCamelCase notation, a definition and description (in the *Role Notes* field) and the multiplicity. In addition, it is important to specify the *navigability* (navigable or non-navigable) using the corresponding drop down box.

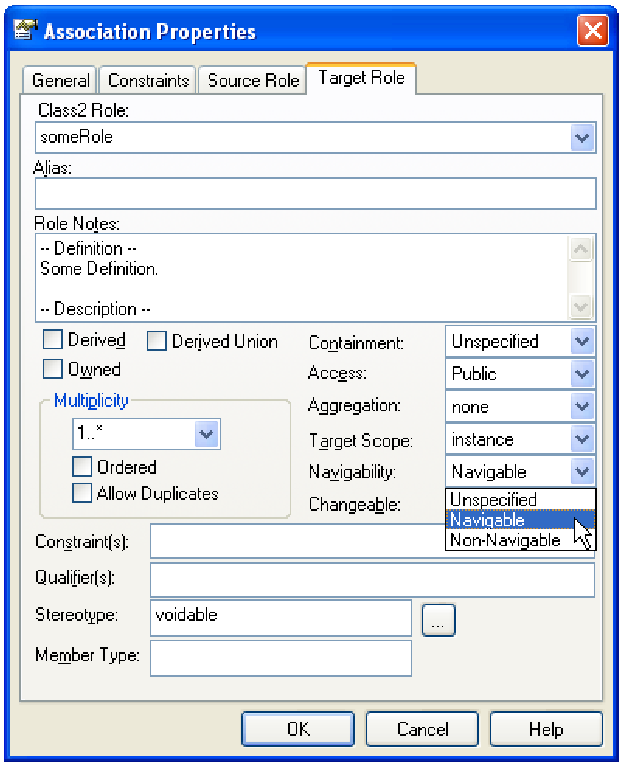
NOTE 1 The GCM recommends (in recommendation 10) that associations should be modelled as navigable in only one direction, unless no historic versions are maintained in the data sets for the theme or navigability in both directions is essential to meet the user requirements.

NOTE 2 Further, the GCM notes that role names, multiplicity, stereotype and tagged values are not relevant for non-navigable association roles. To quote ISO/TS 19103 D.7.2: "If this is important to the model, the association should be two-way navigable to make enforcement of the constraint more tenable. In other words, a one-way relation implies a certain 'don't care' attitude towards the non-navigable end."



Source: SomeSpatialObjectType
Target: YetAnotherType
Name:
Alias:
Direction: Source -> Destination
Style: Custom
Stereotype:
Unspecified
Bi-Directional
Source -> Destination
Destination -> Source
Notes:

Figure 1.18: Specifying the direction of an association.



Class2 Role: someRole
Alias:
Role Notes:
-- Definition --
Some Definition.

-- Description --

Derived
Derived Union
Owned
Multiplicity: 1..*
Ordered
Allow Duplicates
Containment: Unspecified
Access: Public
Aggregation: none
Target Scope: instance
Navigability: Navigable
Unspecified
Navigable
Non-Navigable
Changeable:
Constraint(s):
Qualifier(s):
Stereotype: voidable
Member Type:

Figure 1.19: Specifying properties of an association role.

1.4.3 Deleting elements

All elements can be deleted from a diagram by selecting them and pressing the *Delete* key or selecting *Delete* from the context menu.

NOTE Deleting classes in a diagram does not delete them from the model. When deleting a connector using *Delete*, you are asked in a dialogue whether to only hide it in the diagram or to delete it from the model. You can show a hidden connector again in a diagram by right-clicking on the link in the *Links* tab of the *Class Properties* dialogue of one of its classes and selecting SHOW RELATION in the context menu.

1.4.4 Displaying OCL Constraints

OCL constraints should be represented in the models as follows. The constraint should be assigned to the UML class (in the constraints tab of the *Class Properties* dialog) (see section 1.3). Enabling the display of constraints in the *Diagram Properties* or *Feature visibility* dialogue (see section 1.5.1) will cause the constraints to be shown in an extra compartment. However, here only the name of the constraint(s) is shown.

In order to depict the full constraint in a diagram, add a note (*not* a constraint!) to your diagram and create a link to the class containing the constraint to be shown. Right-click on the link and select LINK THIS NOTE TO AN ELEMENT FEATURE in the context menu (Figure 1.20). In the pop-up window you can select the constraint you wish to show in the note you just created (Figure 1.21). For further info go to the EA Help Topic *Link a Note to Internal Documentation*.

NOTE The constraint text that appears in the note after linking it to the constraint is *not* automatically updated when the constraint is! This means you have to repeat the procedure when you update a constraint.

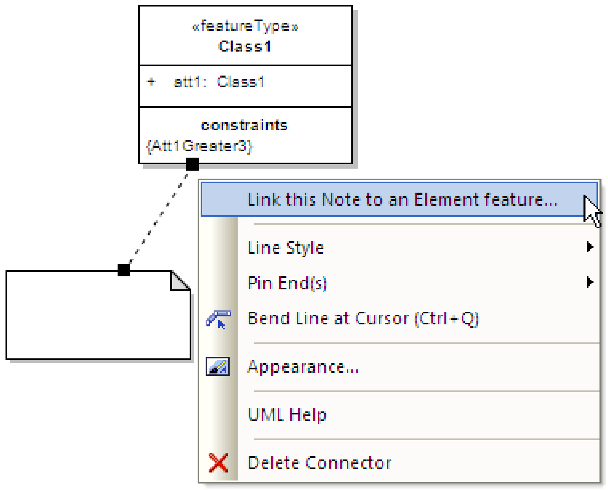
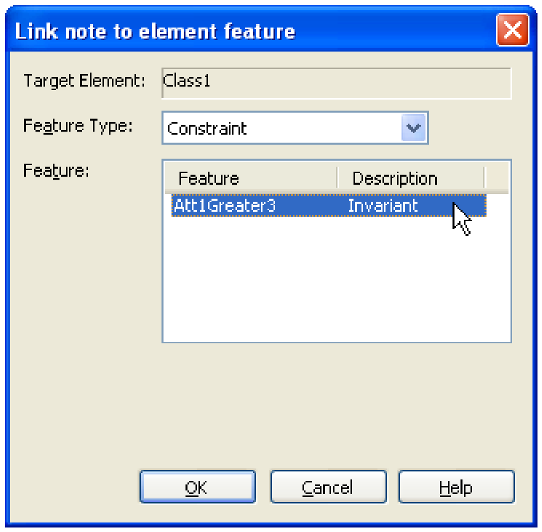


Figure 1.20: Linking a note to an element feature.



Target Element: Class1
Feature Type: Constraint
Feature:

Feature	Description
Att1Greater3	Invariant

Figure 1.21: Selecting the constraint to be displayed in the note.

1.5 Layout

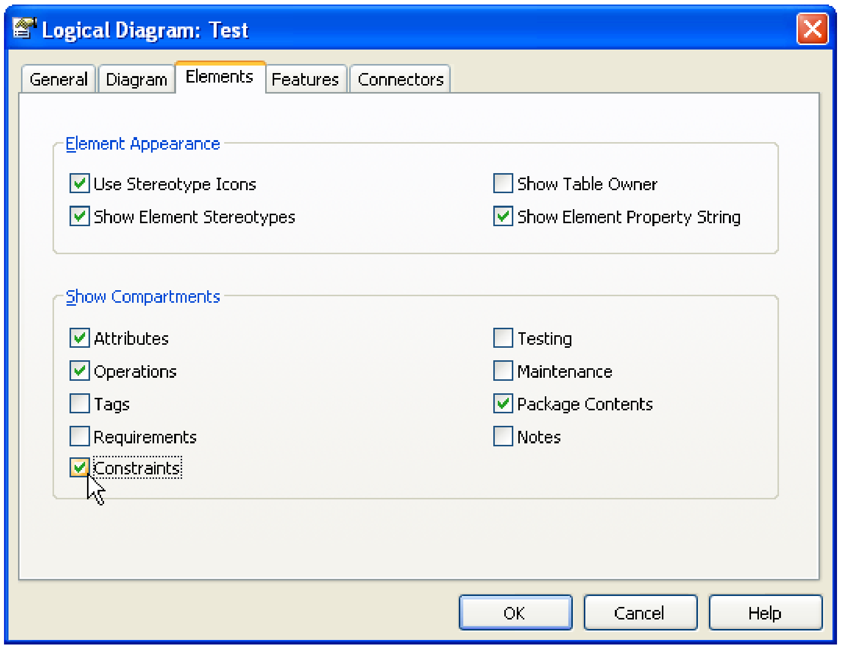
In order to have a common layout for all models and the diagrams in the data product specifications, the same colours, fonts and diagram properties should be used by all Thematic Working Groups.

1.5.1 Diagram properties

The layout of diagrams can be specified in a dialogue that can be brought up by right clicking on a diagram and selecting PROPERTIES in the context menu. In particular, it can be specified which properties of classes and attributes shall (not) be shown and how (in the *Elements* and *Features* tabs). For example, Figure 1.22 illustrates how to show the constraints of all classes in a diagram, while Figure 1.23 shows how to enable attribute and association role stereotypes to be displayed.

Convention: All INSPIRE TWGs shall use the diagram settings depicted in Figure 1.22 and Figure 1.23.

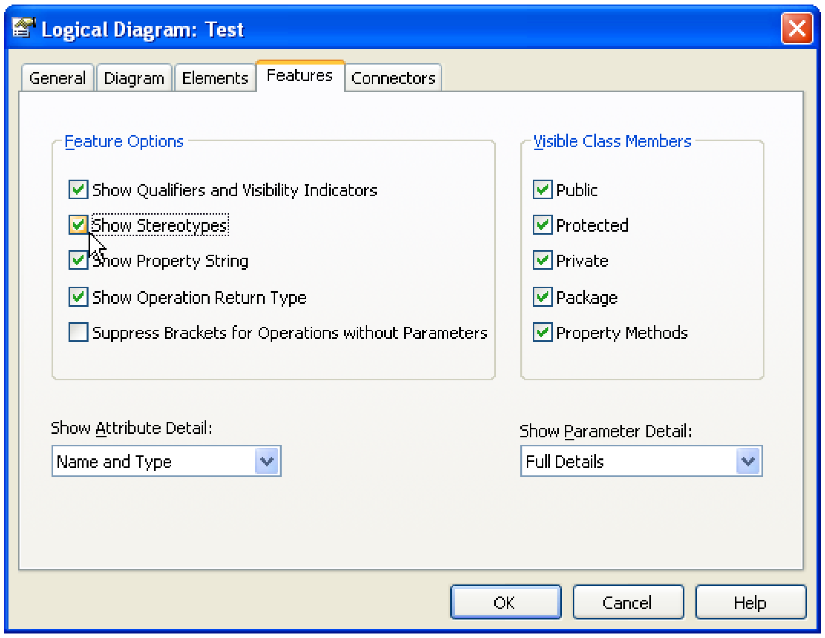
NOTE You can also choose to show specific elements only for a specific class by selecting FEATURE VISIBILITY in the class's context menu and checking the appropriate check box in the *Show element* compartments part. This shall only be used in exceptional cases, in order to emphasise or clarify a specific aspect of the model.



Element Appearance
- [✓] Use Stereotype Icons
- [✓] Show Element Stereotypes
- [ ] Show Table Owner
- [✓] Show Element Property String

Show Compartments
- [✓] Attributes
- [✓] Operations
- [ ] Tags
- [ ] Requirements
- [✓] Constraints
- [ ] Testing
- [ ] Maintenance
- [✓] Package Contents
- [ ] Notes

Figure 1.22: Enabling the display of constraints in a diagram.



Feature Options
- [✓] Show Qualifiers and Visibility Indicators
- [✓] Show Stereotypes
- [✓] Show Property String
- [✓] Show Operation Return Type
- [ ] Suppress Brackets for Operations without Parameters

Visible Class Members
- [✓] Public
- [✓] Protected
- [✓] Private
- [✓] Package
- [✓] Property Methods

Show Attribute Detail: Name and Type

Show Parameter Detail: Full Details

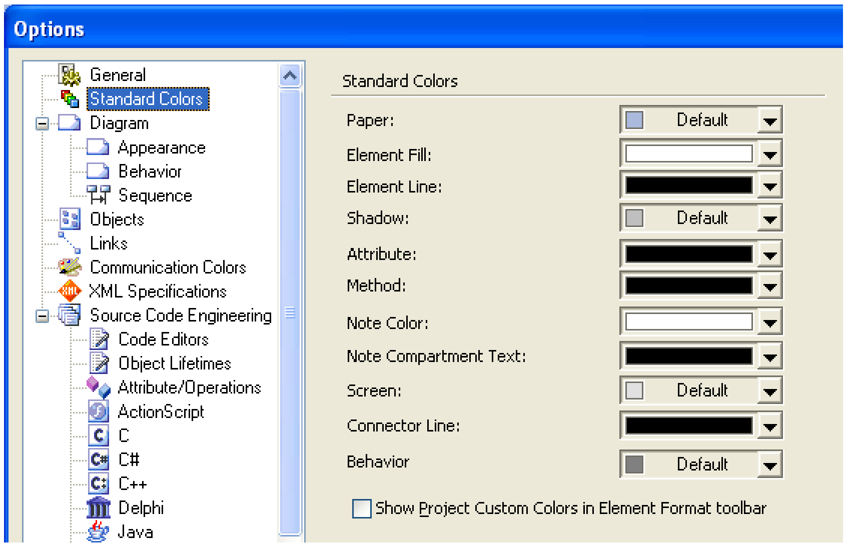
Figure 1.23: Showing attribute stereotypes.

1.5.2 Standard colours and fonts

The standard colours can be specified in the *Standard Colors* section of the *Options* dialogue (TOOLS – OPTIONS).

Convention: All INSPIRE TWGs shall use the standard colours depicted in Figure 1.24 (i.e. some colours shall be changed to white (RGB values 255,255,255) or black (0,0,0)) and the gradient fill setting (none) depicted in Figure 1.25.

The paper colour and gradient can be freely chosen as a white background is used in any case when copying (using Diagram – Copy Image or Crtl+B) and pasting a diagram into a document.

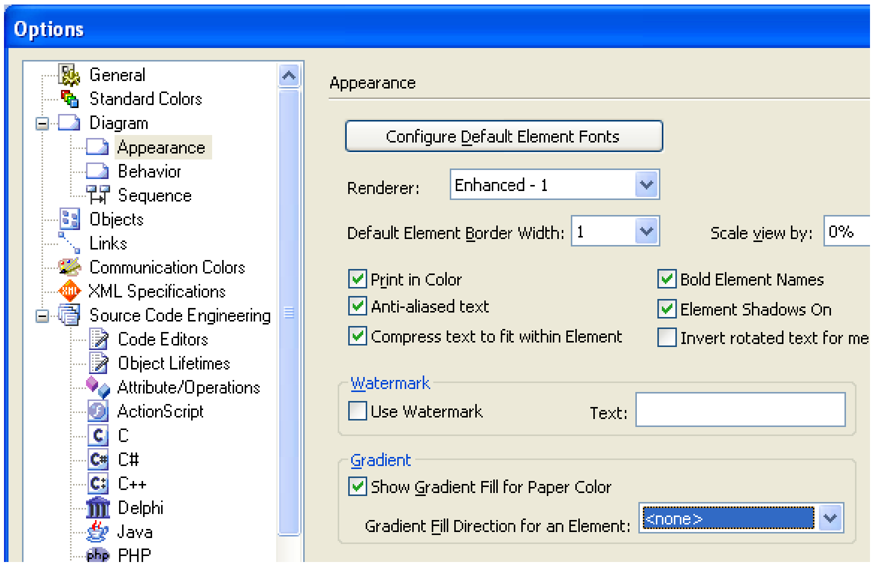


Standard Colors

Paper:	Default
Element Fill:	⬜
Element Line:	⬛
Shadow:	Default
Attribute:	⬛
Method:	⬛
Note Color:	⬜
Note Compartment Text:	⬛
Screen:	Default
Connector Line:	⬛
Behavior	Default

☐ Show Project Custom Colors in Element Format toolbar

Figure 1.24: Standard colours.



Appearance

Renderer: Enhanced - 1

Default Element Border Width: 1

Scale view by: 0%

☑ Print in Color
☑ Anti-aliased text
☑ Compress text to fit within Element
☑ Bold Element Names
☑ Element Shadows On
☐ Invert rotated text for me

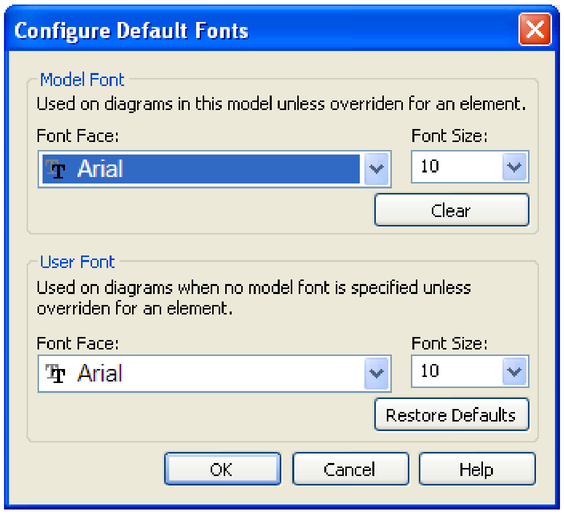
Watermark
☐ Use Watermark
Text:

Gradient
☑ Show Gradient Fill for Paper Color
Gradient Fill Direction for an Element: <none>

Figure 1.25: Gradient fill for elements.

The default fonts can be specified using the *Configure Default Element Fonts* button in the *Diagram – Appearance* section of the *Options* dialogue (TOOLS – OPTIONS). The *User Font* is the default font used by EA. It is a general setting that is applied to all EA projects. The *Model Font* is stored per EA project, i.e., different EA projects can have different *Model Fonts*. It overwrites the setting of the *User Font* and can be left empty (in which case the *User Font* is used).

Convention: Arial, 10pt for both the *Model Font* and the *User Font* (see Figure 1.26).


Configure Default Fonts

Model Font
Used on diagrams in this model unless overriden for an element.
Font Face: Arial
Font Size: 10

User Font
Used on diagrams when no model font is specified unless overriden for an element.
Font Face: Arial
Font Size: 10

Figure 1.26: Default fonts.

1.6 Exercises

Exercise 1.1. Create a new project called `uml_intro.eap`. In the first dialogue, do not select any models to add to your project.

Exercise 1.2. Download the INSPIRE UML profile from http://circa.europa.eu/Members/irc/jrc/imaco2000/library?l=/drafting_folders/data_specifications/thematic_working/common_area/templates/inspire_umlprofile/_EN_1.1_&a=d and import it into Enterprise Architect.

Exercise 1.3. Rename the model at the root to *UML Intro*. In this model, create a view called *View1* containing two packages: *Package1* and *Package2*. Add the «*applicationSchema*» stereotype to *Package1*. In *Package1*, create a class called *Class1*. Add a natural language name, a definition and a description following the convention described in section 1.3.2. In *Package1*, create a second class called *Class2.* Make *Class2* abstract.

Exercise 1.4. Add the stereotypes «*featureType*» and «*placeholder*» to *Class1*. Observe the stereotype that is displayed in the *Stereotype* combo box and in the *Project Browser*. Delete the placeholder displayed in the *Stereotype* combo box using the Delete key. Confirm using *OK*. Observe the stereotype that is displayed in the *Stereotype* combo box in the *Class properties* dialogue and in the *Project Browser*. Observe using the "…" button in the *Class properties* dialogue, which stereotype(s) is/are still selected.

Exercise 1.5. Create a public attribute called *att1* of type *Class2* in *Class1*. For this attribute, specify a multiplicity of *0..1* and add the «*voidable*» stereotype.

Exercise 1.6. In the *Constraints* tab in the *Class Properties* dialogue of *Class1*, add a new constraint called *Constraint1*. Select the type OCL. Specify a constraint saying in both OCL and natural language (as an OCL comment) that the attribute *att1* shall equal 17.

Exercise 1.7. Create a class diagram with the default name in *Package1* and add *Class1* and *Class2* to it as a simple link. Use the Toolbox icon to create one more class: *Class3*. Create a *Generalization* relationship between *Class3* and *Class2* and an association between *Class1* and *Class2*. Make the association navigable from *Class1* to *Class2*, specify a role name and a multiplicity for the *Class2* end of the association. Then hide the association from the diagram and verify in the *Class Properties* of *Class1* that it still exists in the model.

Exercise 1.8. Create a class called *Class4* in *Package2* in the Project Browser. Drag and drop that class into the *Package1* class diagram (as a simple link).

Exercise 1.9. In the *Diagram Properties* dialogue of the *Package1* class diagram, select to display constraints and use a note to display *Constraint1* of *Class1*.

Exercise 1.10. In the *Diagram Properties* dialogue of the *Package1* class diagram, select to display attribute stereotypes.

Exercise 1.11. Change the default colours as described in section 1.5.2. Change the Model Font to a different setting and observe the changes in your diagrams. Change the User Font to a different setting and observe the changes in your diagrams (nothing should change). Clear the setting for the Model Font and observe the changes in your diagrams.

2 Getting started with EA and Subversion

EA has integrated functionality for working with a version control system. However, since an EA stores its models in a binary, Microsoft Access, file, it cannot simply be uploaded to a version control repository. Thus, the handling of setting up a repository and checking out/in versions is not completely straightforward. Since the EA file itself is binary, EA will allow import/export of each package in XML Metadata Interchange format (which is an ASCII-file) that will then be checked out/in of EA. This means that the packages themselves are version controlled but the EA file is not. You may also import several packages from several different version control systems in one EA file or simply have some packages which are version controlled and some which are not.

In INSPIRE, subversion (SVN) will be used as a version control software. For a brief introduction to basic SVN operations, see Annex A.

NOTE EA requires a command line executable SVN client tool (such as the *CollabNet Subversion 1.6.9 client*[5]) to be installed on your computer. *TortoiseSVN*[6] does *not* provide such a command line tool.

2.1 Setting up Subversion on your computer

Figure 2.1 gives a schematic overview of the procedure required to set up EA with the INSPIRE UML repository. During this step, a local copy has to be created on the client machine using a SVN check-out (steps 1 and 2). The checked out XMI files are then imported into an EA project (steps 3 and 4).

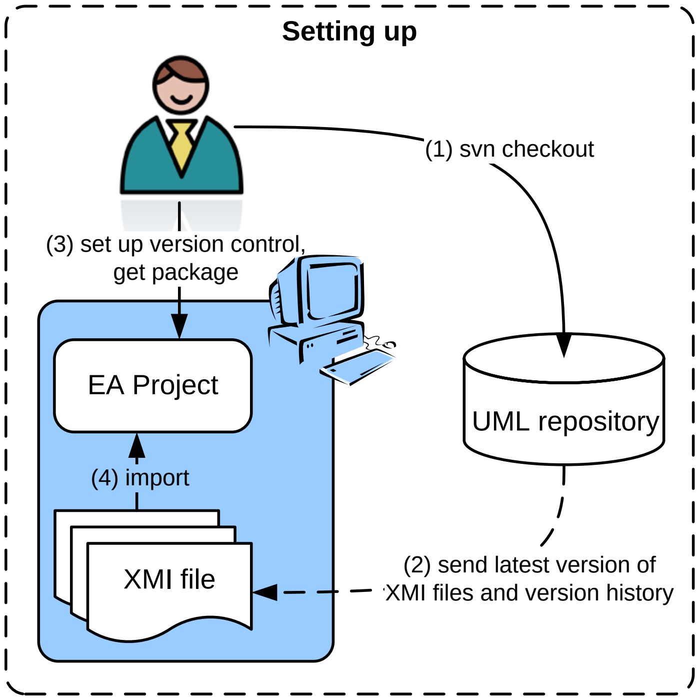

Setting up
(1) svn checkout
(3) set up version control, get package
EA Project
(4) import
XMI file
UML repository
(2) send latest version of XMI files and version history

Figure 2.1: Schematic overview of setting up EA with the UML repository.

[5] Available on CIRCA at http://circa.europa.eu/Members/irc/jrc/imaco2000/library?l=/drafting_folders/data_specifications/thematic_working/common_area/tools/collabnetsubversion-clie/_EN_1.0_&a=d

[6] Available on CIRCA at http://circa.europa.eu/Members/irc/jrc/imaco2000/library?l=/drafting_folders/data_specifications/thematic_working/common_area/tools&vm=detailed&sb=Title

The repository to be used for the developments of the UML application schemas for the TWGs is installed at https://inspire-twg.jrc.it/svn/inspire-model/. The ISO models are not included in our INSPIRE repository. They are located in a separate repository: https://inspire-twg.jrc.it/svn/iso/. The reason for splitting up the INSPIRE and ISO models is that JRC has agreed to host the developments of the ISO models. In that way we avoid the problems of inconsistency etc. within the modelling work.

The following additional svn repositories are referred to by some themes:

- GeoSciML v3.0RC2: https://www.seegrid.csiro.au/subversion/GeoSciML/tags/3.0.0_rc2/model
- EarthResourceML v2.0RC1: https://www.seegrid.csiro.au/subversion/xmml/GGIC/branches/2.0_RC1

First, you need to create local copies of these repositories on your file system. Creating the local copy is done by an SVN checkout from the existing repository.

To check out the **INSPIRE model**, issue the following command in the directory where you would like to store the local copy of the repository.

```
svn checkout https://inspire-twg.jrc.ec.europa.eu/svn/inspire-model/ --
username <username>
```

If you have never connected to the `inspire-twg.jrc.ec.europa.eu` server, the output for this command should be the following:

```
Error validating server certificate for 'https://inspire-
twg.jrc.ec.europa.eu:443':
 - The certificate is not issued by a trusted authority. Use the
   fingerprint to validate the certificate manually!
Certificate information:
 - Hostname: inspire-twg.jrc.ec.europa.eu
 - Valid: from Mon, 23 Mar 2009 08:17:38 GMT until Thu, 21 Mar 2019 08:17:38
GMT
 - Issuer: SDI, JRC, Ispra, Varese, IT
 - Fingerprint: 4e:d4:dd:63:95:ee:ff:6a:12:f4:18:26:0e:3d:6e:7f:8d:0f:6d:21
(R)eject, accept (t)emporarily or accept (p)ermanently?
```

Press p for accepting and storing the certificate permanently on your computer and then enter your password.

NOTE SVN complains that the certificate is not from a trusted source because our certificate is self-signed.

The check out command will create a directory named `inspire-model` in the folder where you issued the command containing all subdirectories of the repository.

To check out the **ISO TC211 model**, issue the following command in the directory where you would like to store the local copy of the repository:

```
svn checkout https://inspire-twg.jrc.it/svn/iso/ --username <username>
```

The check out command will create a directory named `iso` in the folder where you issued the command containing all subdirectories of the repository.

Other models (e.g. GeoSciML) are checked out in the same way.

NOTE Since the command line utility is required by EA in any case, it is used here for illustration. However, alternatively the above procedure can be done by using a graphical user interface for subversion such as the *TortoiseSVN*. In this case it is important that the SVN client used supports the same SVN version (1.5.x or 1.6.x – the third digit can be different) as the command line utility (otherwise you risk receiving the following error message: *“This client is too old to work with working copy”*).

2.2 Repositories

The repositories are created and maintained by JRC. The server and repositories will be backed up following a specific plan. The repositories are set up using secure access control and, thus, will only be accessible for users that are admitted access.

You should have received your username and password for the repositories. If not, contact your JRC contact point or Michael Lutz at michael.lutz@jrc.ec.europa.eu.

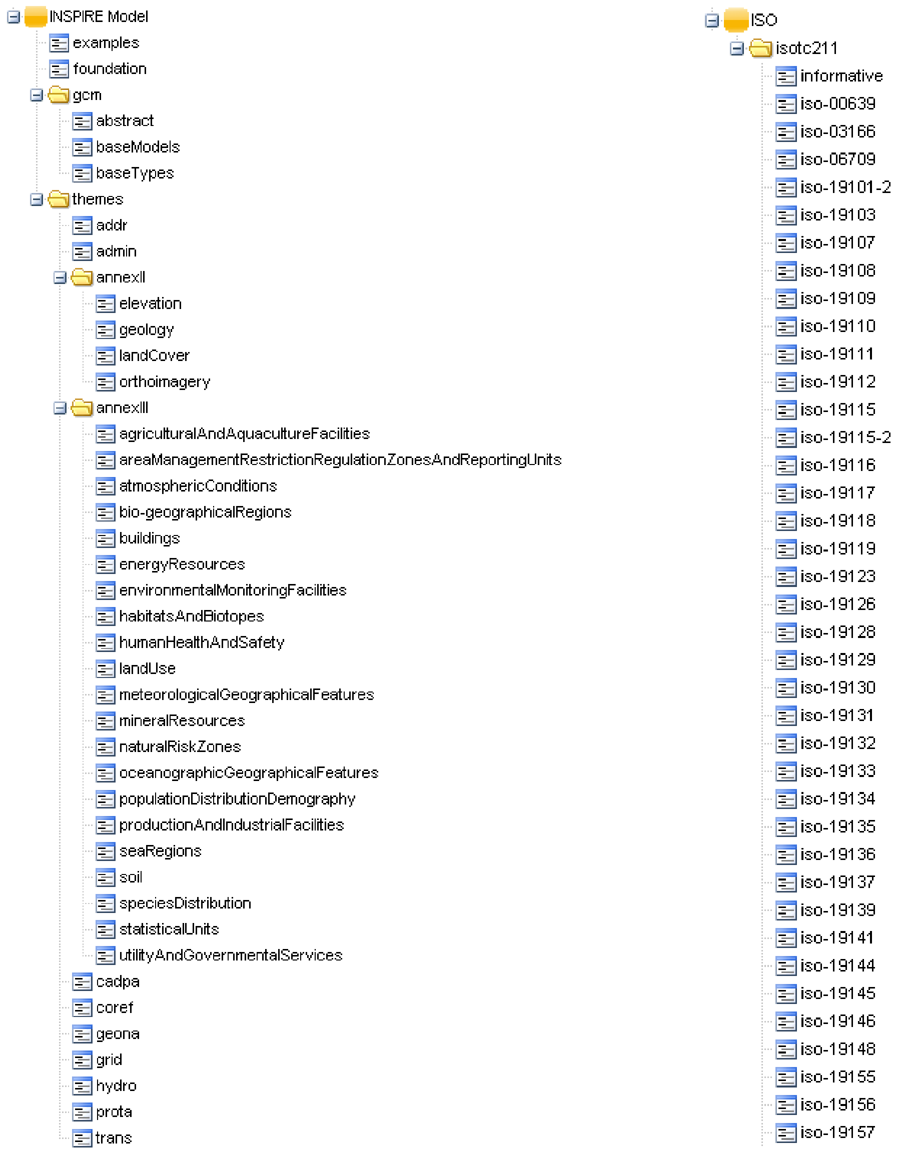
2.2.1 Structure

The structure of the repositories is shown in Figure 2.2.

The INSPIRE model contains a directory for the foundation packages (`/foundation`), which currently only contains the ISO TC211 19100 models. Furthermore, a directory for the generic conceptual model exists (`/gcm`) with three subdirectories: `abstract`, `baseModels`, `baseTypes`. The `abstract` directory contains nothing for the moment but is intended for abstract models of INSPIRE. The `baseModels` directory shall contain any cross-themes models, e.g., the Generic Network Model. The `baseTypes` directory is intended for all cross-themes data types. Finally, a `/themes` directory exists, which contains all the 34 INSPIRE annex I, II and III themes.

The structure of the INSPIRE repository does not currently follow the common SVN pattern to include three sub-directories called `trunk`, `branch` and `tags`[7], which are used when branching or tagging the content of the repository. There is, however, a branches folder on the top level of the repository, which currently contains a specific snapshot of the model development corresponding to the content of the Implementing Rule on interoperability of spatial data sets and services for the Annex I spatial data themes. This snapshot is available for public read access.

[7] The trunk is for the main developments of a given repository content. A branch is a "copy" of the content that exists and is developed independently of another development line. Tags are a snapshot on a certain point in time.



- INSPIRE Model
  - examples
  - foundation
  - gcm
    - abstract
    - baseModels
    - baseTypes
  - themes
    - addr
    - admin
    - annexII
      - elevation
      - geology
      - landCover
      - orthoimagery
    - annexIII
      - agriculturalAndAquacultureFacilities
      - areaManagementRestrictionRegulationZonesAndReportingUnits
      - atmosphericConditions
      - bio-geographicalRegions
      - buildings
      - energyResources
      - environmentalMonitoringFacilities
      - habitatsAndBiotopes
      - humanHealthAndSafety
      - landUse
      - meteorologicalGeographicalFeatures
      - mineralResources
      - naturalRiskZones
      - oceanographicGeographicalFeatures
      - populationDistributionDemography
      - productionAndIndustrialFacilities
      - seaRegions
      - soil
      - speciesDistribution
      - statisticalUnits
      - utilityAndGovernmentalServices
    - cadpa
    - coref
    - geona
    - grid
    - hydro
    - prota
    - trans

- ISO
  - isotc211
    - informative
    - iso-00639
    - iso-03166
    - iso-06709
    - iso-19101-2
    - iso-19103
    - iso-19107
    - iso-19108
    - iso-19109
    - iso-19110
    - iso-19111
    - iso-19112
    - iso-19115
    - iso-19115-2
    - iso-19116
    - iso-19117
    - iso-19118
    - iso-19119
    - iso-19123
    - iso-19126
    - iso-19128
    - iso-19129
    - iso-19130
    - iso-19131
    - iso-19132
    - iso-19133
    - iso-19134
    - iso-19135
    - iso-19136
    - iso-19137
    - iso-19139
    - iso-19141
    - iso-19144
    - iso-19145
    - iso-19146
    - iso-19148
    - iso-19155
    - iso-19156
    - iso-19157

Figure 2.2: Structure of the INSPIRE and ISO repositories.

2.2.2 Access control

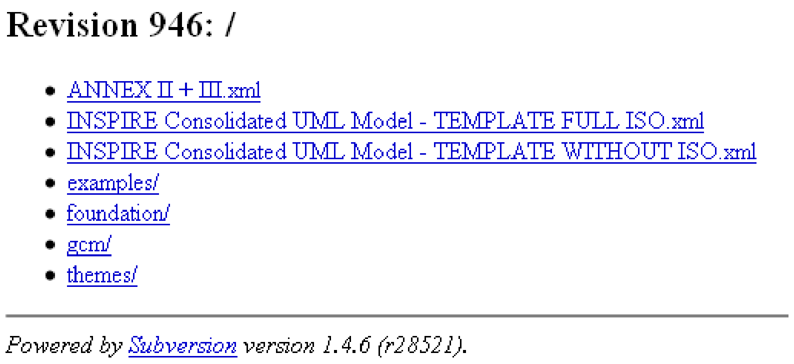
The repository is set up using secure access control. Each user that is created to have access to the repository will be assigned one (or more) role(s). These roles include:

- one role for each thematic working group,
- a role for the data specification drafting team,
- a role for the consolidation team,
- a model administrator role.

Roles can easily be added to the system. The access permissions are set up such that each thematic working group role has write access to their respective theme directory or directories. The data specification drafting team role has write access to the generic conceptual model directory. And finally, the model administrator role has write access to all directories. All roles have read access to the whole repository.

2.2.3 Web Interface – Browsing the repository and changing your password

It is possible to access and browse the newest revision of all files in the repository also for users that do not have subversion clients installed on their machine. The simplest way to browse the repository content is simply pointing a web browser to https://inspire-twg.jrc.ec.europa.eu/svn/inspire-model (this requires authentication, see also Section 2.2.2).



Revision 946: /

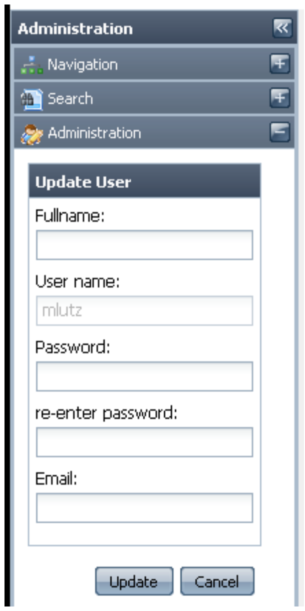
- ANNEX II + III.xml
- INSPIRE Consolidated UML Model - TEMPLATE FULL ISO.xml
- INSPIRE Consolidated UML Model - TEMPLATE WITHOUT ISO.xml
- examples/
- foundation/
- gcm/
- themes/

Powered by Subversion version 1.4.6 (r28521).

Figure 2.3: Simple web interface.

A more elaborate web interface is available at https://inspire-twg.jrc.ec.europa.eu/voilaSVN/. This interface also allows you to change your password: Select any repository and log in with your username and password. Find the *Administration* tab, open it and fill in form seen in Figure 2.4.

The web interface can also be used for browsing and searching the repositories (in this case, you have to select “INSPIRE Model” or “ISO” when logging in). In the search tab you can choose if you want to search through the latest version or if you want to search through all versions (Figure 2.5).



Fullname:

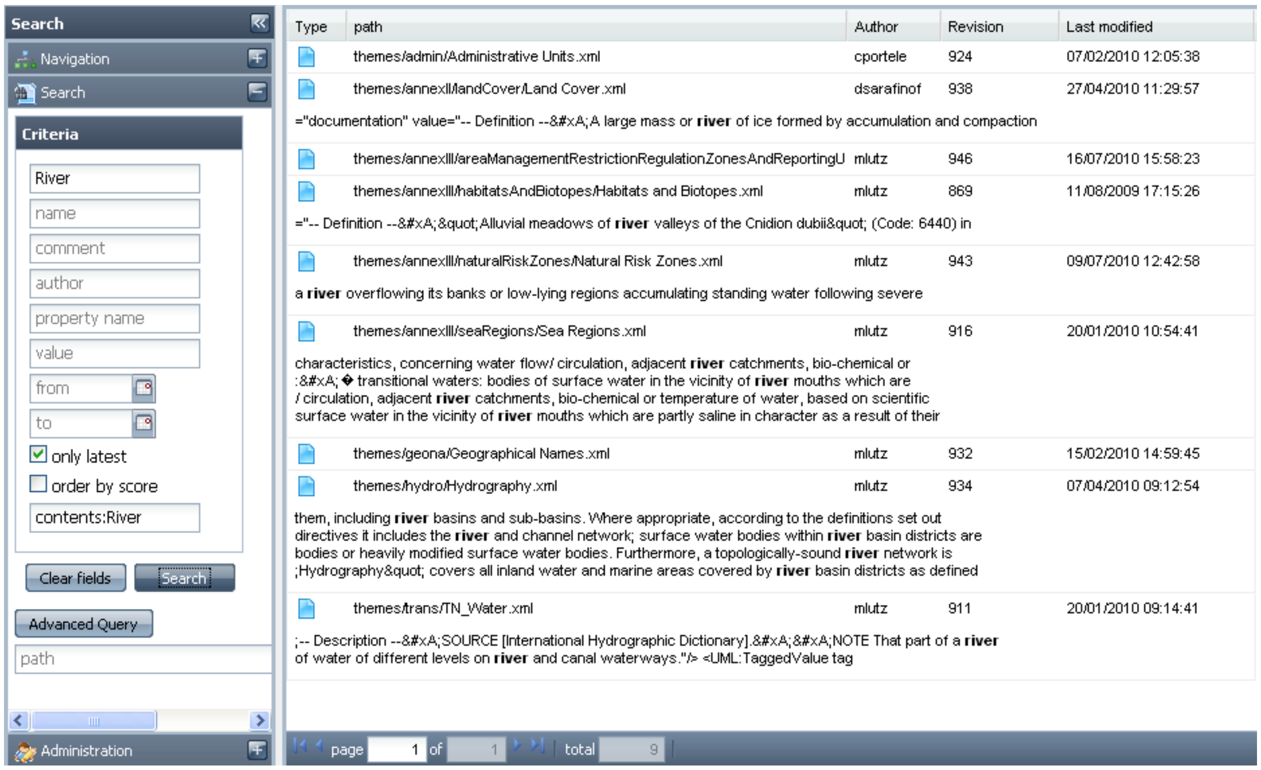
User name: mlutz

Password:

re-enter password:

Email:

Figure 2.4: Changing password for the repository.



River
name
comment
author
property name
value
from
to
[✓] only latest
[ ] order by score
contents:River
path

Type	path	Author	Revision	Last modified
	themes/admin/Administrative Units.xml	cportele	924	07/02/2010 12:05:38
	themes/annexII/landCover/Land Cover.xml	dsarafinof	938	27/04/2010 11:29:57
	="documentation" value="-- Definition --&#xA;A large mass or **river** of ice formed by accumulation and compaction			
	themes/annexIII/areaManagementRestrictionRegulationZonesAndReportingU	mlutz	946	16/07/2010 15:58:23
	themes/annexIII/habitatsAndBiotopes/Habitats and Biotopes.xml	mlutz	869	11/08/2009 17:15:26
	="-- Definition --&#xA;&quot;Alluvial meadows of **river** valleys of the Cnidion dubii&quot; (Code: 6440) in			
	themes/annexIII/naturalRiskZones/Natural Risk Zones.xml	mlutz	943	09/07/2010 12:42:58
	a **river** overflowing its banks or low-lying regions accumulating standing water following severe			
	themes/annexIII/seaRegions/Sea Regions.xml	mlutz	916	20/01/2010 10:54:41
	characteristics, concerning water flow/ circulation, adjacent **river** catchments, bio-chemical or :&#xA; � transitional waters: bodies of surface water in the vicinity of **river** mouths which are / circulation, adjacent **river** catchments, bio-chemical or temperature of water, based on scientific surface water in the vicinity of **river** mouths which are partly saline in character as a result of their			
	themes/geona/Geographical Names.xml	mlutz	932	15/02/2010 14:59:45
	themes/hydro/Hydrography.xml	mlutz	934	07/04/2010 09:12:54
	them, including **river** basins and sub-basins. Where appropriate, according to the definitions set out directives it includes the **river** and channel network; surface water bodies within **river** basin districts are bodies or heavily modified surface water bodies. Furthermore, a topologically-sound **river** network is ;Hydrography&quot; covers all inland water and marine areas covered by **river** basin districts as defined			
	themes/trans/TN_Water.xml	mlutz	911	20/01/2010 09:14:41
	;-- Description --&#xA;SOURCE [International Hydrographic Dictionary].&#xA;&#xA;NOTE That part of a **river** of water of different levels on **river** and canal waterways."/> <UML:TaggedValue tag			

page 1 of 1 | total 9

Figure 2.5: Searching for content mentioning “River” in the repository.

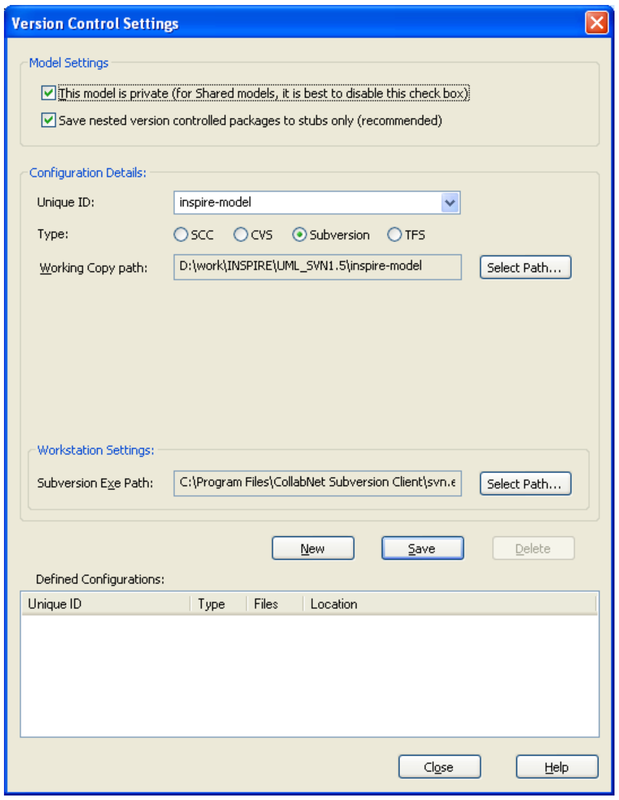
2.3 Creating the Structure in EA

Now that your system is connected with the repository, you can set up EA to work with it. When this is done you can import each package contained in the repository into EA. It is important to mention that EA considers a file in the repository ending with `.xml` containing XMI a UML package. To import packages from the repository you need to have an EA project file – it can be empty (newly created) or it can be an existing file already containing model/packages. EA does not care whether it imports packages from a version control system in an EA file already containing models.

2.3.1 Connecting EA to the repository

EA can work with a number of different repositories. It handles all connections giving them separate unique IDs. It is IMPORTANT that all users **use the same identifier** for a given repository (subset). If different identifiers are used with packages that contain version-controlled sub-packages, these sub-packages will lose their connection to the repository. For the INSPIRE work, you shall use the following identifiers:

- *inspire-model* for the repository with the INSPIRE application schemas
- *isotc211* for the ISO repository
- *GeoSciML* for GeoSciML v3.0
- *GGICWorld* for EarthResourceML v2.0



Model Settings
[✓] This model is private (for Shared models, it is best to disable this check box)
[✓] Save nested version controlled packages to stubs only (recommended)

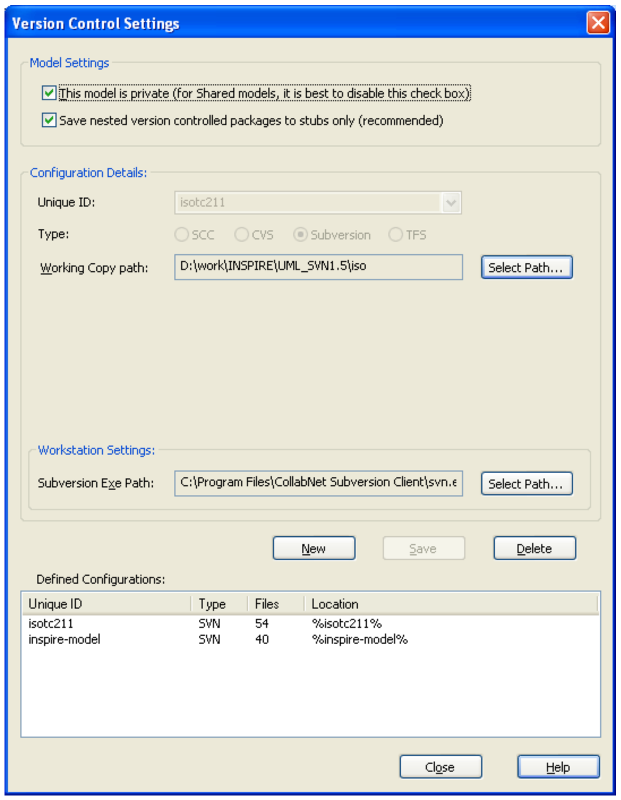
Configuration Details:
Unique ID: inspire-model
Type: SCC CVS [✓ Subversion] TFS
Working Copy path: D:\work\INSPIRE\UML_SVN1.5\inspire-model

Workstation Settings:
Subversion Exe Path: C:\Program Files\CollabNet Subversion Client\svn.e

Defined Configurations:

Unique ID	Type	Files	Location

Figure 2.6: Creating an identifier called for the INSPIRE repository.



Model Settings
[✓] This model is private (for Shared models, it is best to disable this check box)
[✓] Save nested version controlled packages to stubs only (recommended)

Configuration Details:
Unique ID: isotc211
Type: SCC CVS [✓ Subversion] TFS
Working Copy path: D:\work\INSPIRE\UML_SVN1.5\iso

Workstation Settings:
Subversion Exe Path: C:\Program Files\CollabNet Subversion Client\svn.e

Defined Configurations:

Unique ID	Type	Files	Location
isotc211	SVN	54	%isotc211%
inspire-model	SVN	40	%inspire-model%

Figure 2.7: Final version control settings.

Working with a repository and EA can follow certain schemes. The main ones are private and shared models. We will follow the private model scheme which has several advantages to the shared one. The shared model scheme is highly dependent on 100% server up-time and high speed internet connection. The private model scheme allows for flexibility and that each editor works with his own EA file. Settings related to the EA projects connection to version control systems are found in PROJECT – VERSION CONTROL – VERSION CONTROL SETTING (Figure 2.6).

Here, you need to specify the Unique ID and the type of repository (select *Subversion*). You also need tp specify the path to which you have checked out the repository, e.g. `D:\work\INSPIRE\UML_SVN1.5\inspire-model` in the example shown in Figure 2.6. Finally, you need to specify the path to the SVN executable `svn.exe`, e.g. `C:\Program Files\CollabNet Subversion Client\svn.exe` in the example.

NOTE The path for the isotc211 model is `<some path>\iso`, *not* `<some path>\iso\isotc211`.

After adding the identifier for the INSPIRE repository, repeat these steps for the ISO repository. The final settings are shown in Figure 2.7.

2.3.2 Manually importing packages

One way to import the packages existing in the repository is to import each one separately. To do this, right-click in the *Project Browser* on the package into which you want to import the package. It could be the top view. Then select PACKAGE CONTROL – GET PACKAGE as shown in Figure 2.8.

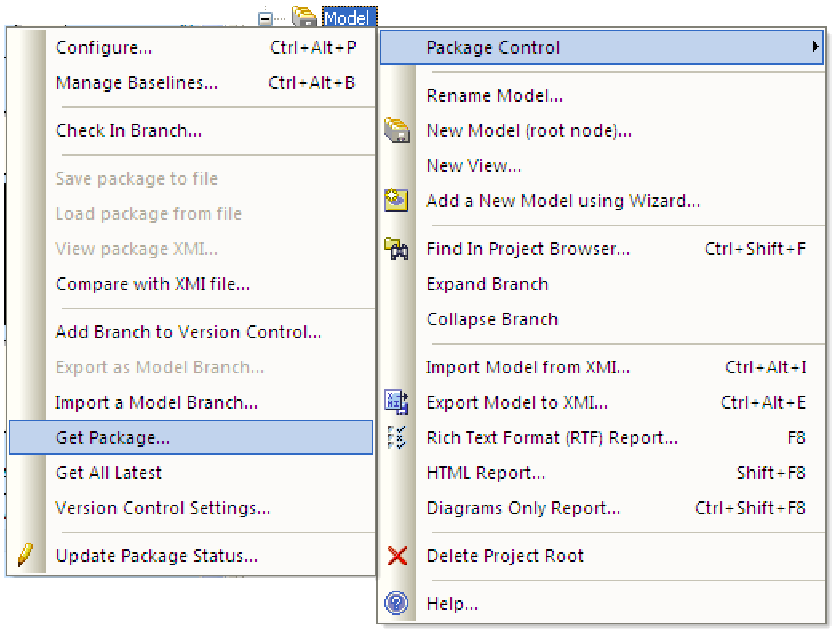
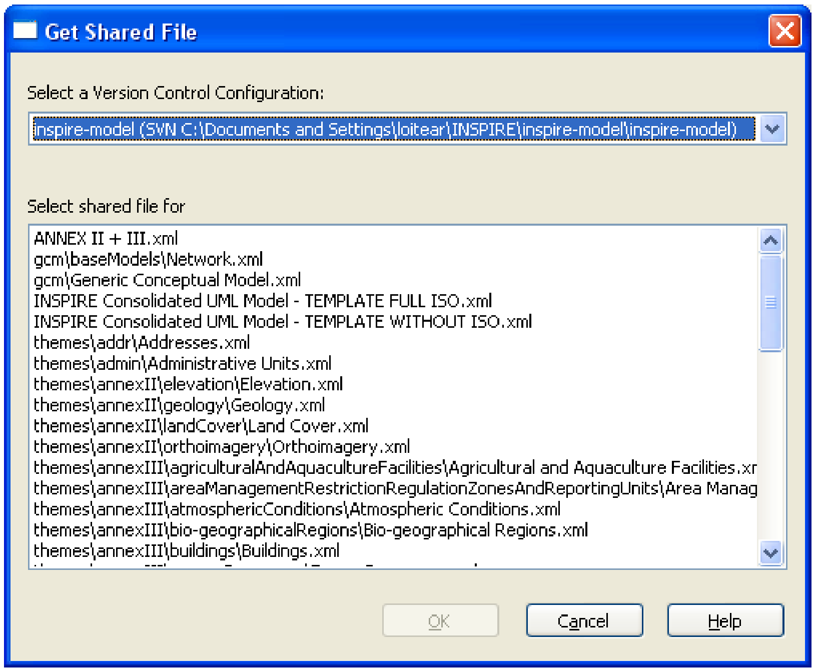


Figure 2.8: Get package.

This will open a dialog as shown in Figure Figure 2.9. Select the identifier for the version control system: (*inspire-model* or *isotc211*), which will then show all UML packages available in the repository. Unfortunately, in this dialogue it is only possible to select one package at a time, so that this procedure is rather cumbersome. On the other hand, it allows you to have all packages imported in a structure that you decide yourself. You could, e.g., import all theme packages in the root of your model.



Select a Version Control Configuration:

inspire-model (SVN C:\Documents and Settings\loitear\INSPIRE\inspire-model\inspire-model)

Select shared file for

```
ANNEX II + III.xml
gcm\baseModels\Network.xml
gcm\Generic Conceptual Model.xml
INSPIRE Consolidated UML Model - TEMPLATE FULL ISO.xml
INSPIRE Consolidated UML Model - TEMPLATE WITHOUT ISO.xml
themes\addr\Addresses.xml
themes\admin\Administrative Units.xml
themes\annexII\elevation\Elevation.xml
themes\annexII\geology\Geology.xml
themes\annexII\landCover\Land Cover.xml
themes\annexII\orthoimagery\Orthoimagery.xml
themes\annexIII\agriculturalAndAquacultureFacilities\Agricultural and Aquaculture Facilities.xr
themes\annexIII\areaManagementRestrictionRegulationZonesAndReportingUnits\Area Manag
themes\annexIII\atmosphericConditions\Atmospheric Conditions.xml
themes\annexIII\bio-geographicalRegions\Bio-geographical Regions.xml
themes\annexIII\buildings\Buildings.xml
```

Figure 2.9: Select package to import.

2.3.3 Automatically importing each package

In order to make the import process easier for the editors, we have created three "template" files containing the whole structure in EA. These correspond to different approaches for how to work with the ISO models and create your overall structure. We recommend using the first template.

- `TEMPLATE - INSPIRE Consolidated UML Model - ANNEX I-II-III - full ISO.xml`
  This template contains all INSPIRE themes packages and all the ISO models, i.e. a rather large number of packages. So the import will take some time (around 5-10 minutes) – but you only have to do this once. The advantage is that you have all the ISO models and can refer to them in your models.

- `TEMPLATE - INSPIRE Consolidated UML Model - ANNEX I-II-III - with ISO+GeoSciML.xml`
  This template contains all INSPIRE themes packages, all the ISO models and GeoSciML.

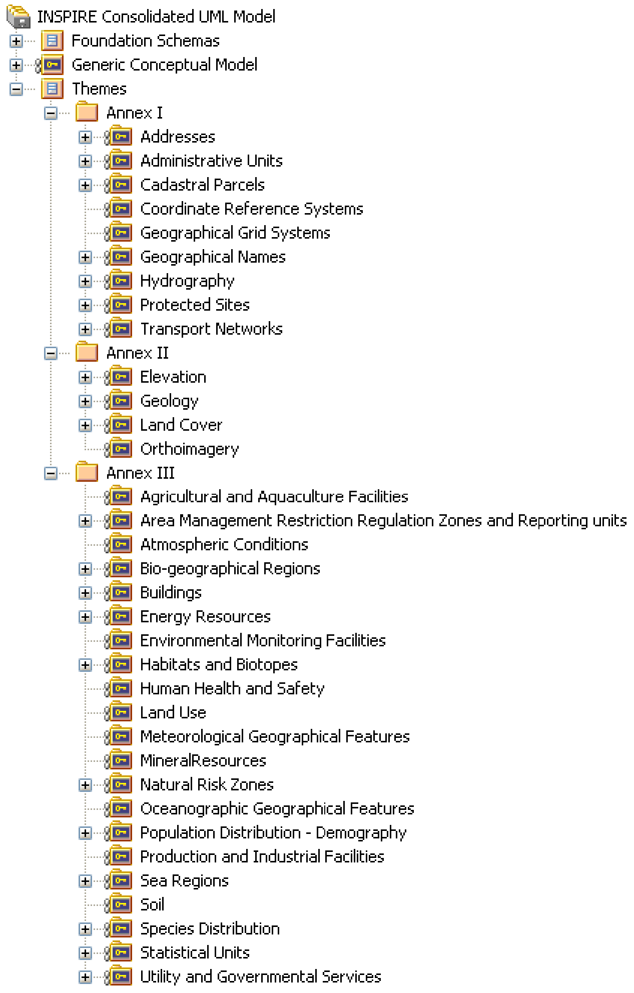
- `TEMPLATE - INSPIRE Consolidated UML Model - ANNEX I-II-III - without ISO.xml`
  This template contains all INSPIRE themes packages, but none of the ISO packages. This means that you have to import those ISO packages (standards) that you need in your model work manually. This template is appropriate if you need to use only few ISO standards and want to keep your EA model small. After importing the template, your EA project will contain a top package for the ISO models: *INSPIRE Consolidated UML Model/Foundation Schemas/ISO TC211/*. To import a specific ISO model, go to the *ISO TC211* package, choose PACKAGE CONTROL – GET PACKAGE from the *isotc211* repository. In the list of packages you select the one you need. If necessary, repeat this process until you have imported all required packages.

  NOTE Some ISO standards have more than one package. These typically represent different versions of the standard (e.g. ISO 19115:2003 and ISO:19115:2006 Corrigendum). When importing standards manually, make sure that you include the latest version of a particular standard.

- `ISO TC211.xml`
  This template contains only all the ISO models, but not the INSPIRE themes packages. You can use this template to include the ISO models if you have previously imported only the INSPIRE themes packages. To add all the ISO models, go to the *ISO TC211* package, choose PACKAGE CONTROL – GET PACKAGE from the *inspire-model* repository and select the `ISO TC211.xml` package.

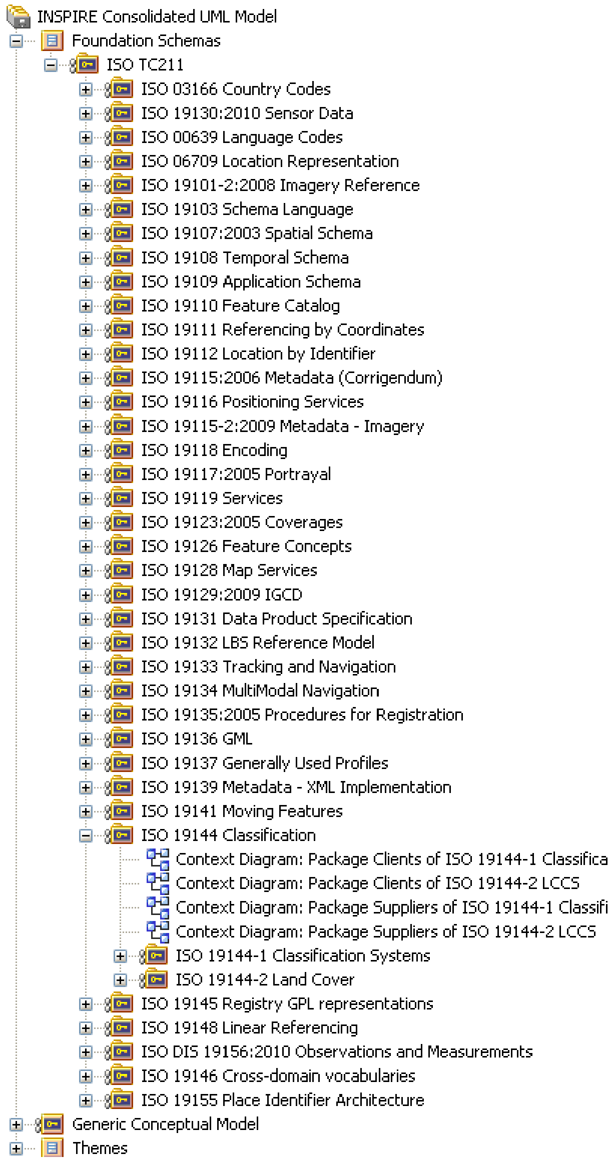
After importing one or several of the templates, all sub-packages can be retrieved by right-clicking on a package in the *Project Browser* and selecting PACKAGE CONTROL – GET ALL LATEST in the context menu. Figure 2.10 shows the Project Browser after this step (with all INSPIRE-related packages expanded on the left, and all ISO packages expanded on the right).

INSPIRE



- INSPIRE Consolidated UML Model
  - Foundation Schemas
  - Generic Conceptual Model
  - Themes
    - Annex I
      - Addresses
      - Administrative Units
      - Cadastral Parcels
      - Coordinate Reference Systems
      - Geographical Grid Systems
      - Geographical Names
      - Hydrography
      - Protected Sites
      - Transport Networks
    - Annex II
      - Elevation
      - Geology
      - Land Cover
      - Orthoimagery
    - Annex III
      - Agricultural and Aquaculture Facilities
      - Area Management Restriction Regulation Zones and Reporting units
      - Atmospheric Conditions
      - Bio-geographical Regions
      - Buildings
      - Energy Resources
      - Environmental Monitoring Facilities
      - Habitats and Biotopes
      - Human Health and Safety
      - Land Use
      - Meteorological Geographical Features
      - MineralResources
      - Natural Risk Zones
      - Oceanographic Geographical Features
      - Population Distribution - Demography
      - Production and Industrial Facilities
      - Sea Regions
      - Soil
      - Species Distribution
      - Statistical Units
      - Utility and Governmental Services

ISO

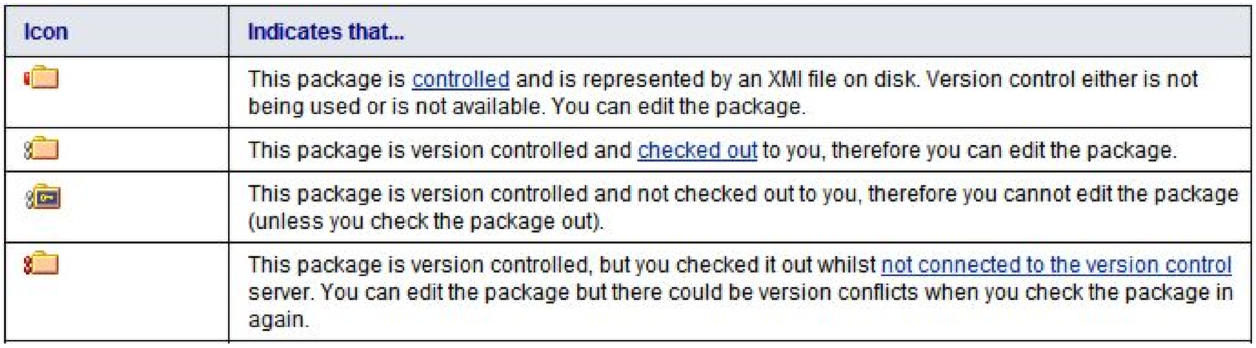


- INSPIRE Consolidated UML Model
  - Foundation Schemas
    - ISO TC211
      - ISO 03166 Country Codes
      - ISO 19130:2010 Sensor Data
      - ISO 00639 Language Codes
      - ISO 06709 Location Representation
      - ISO 19101-2:2008 Imagery Reference
      - ISO 19103 Schema Language
      - ISO 19107:2003 Spatial Schema
      - ISO 19108 Temporal Schema
      - ISO 19109 Application Schema
      - ISO 19110 Feature Catalog
      - ISO 19111 Referencing by Coordinates
      - ISO 19112 Location by Identifier
      - ISO 19115:2006 Metadata (Corrigendum)
      - ISO 19116 Positioning Services
      - ISO 19115-2:2009 Metadata - Imagery
      - ISO 19118 Encoding
      - ISO 19117:2005 Portrayal
      - ISO 19119 Services
      - ISO 19123:2005 Coverages
      - ISO 19126 Feature Concepts
      - ISO 19128 Map Services
      - ISO 19129:2009 IGCD
      - ISO 19131 Data Product Specification
      - ISO 19132 LBS Reference Model
      - ISO 19133 Tracking and Navigation
      - ISO 19134 MultiModal Navigation
      - ISO 19135:2005 Procedures for Registration
      - ISO 19136 GML
      - ISO 19137 Generally Used Profiles
      - ISO 19139 Metadata - XML Implementation
      - ISO 19141 Moving Features
      - ISO 19144 Classification
        - Context Diagram: Package Clients of ISO 19144-1 Classifica
        - Context Diagram: Package Clients of ISO 19144-2 LCCS
        - Context Diagram: Package Suppliers of ISO 19144-1 Classifi
        - Context Diagram: Package Suppliers of ISO 19144-2 LCCS
        - ISO 19144-1 Classification Systems
        - ISO 19144-2 Land Cover
      - ISO 19145 Registry GPL representations
      - ISO 19148 Linear Referencing
      - ISO DIS 19156:2010 Observations and Measurements
      - ISO 19146 Cross-domain vocabularies
      - ISO 19155 Place Identifier Architecture
  - Generic Conceptual Model
  - Themes

Figure 2.10: Project browser after importing all packages.

2.3.4 Icons used for version control

EA uses specific icons that tell about the status of a version controlled package. An overview of these icons is seen in Figure 2.11.



Icon	Indicates that...
📁	This package is controlled and is represented by an XMI file on disk. Version control either is not being used or is not available. You can edit the package.
📁	This package is version controlled and checked out to you, therefore you can edit the package.
📁	This package is version controlled and not checked out to you, therefore you cannot edit the package (unless you check the package out).
📁	This package is version controlled, but you checked it out whilst not connected to the version control server. You can edit the package but there could be version conflicts when you check the package in again.

Figure 2.11: EA icons for version controlled packages.

2.4 Exercises

For the exercises in sections 2 and 3 we have created a test repository at https://inspire-twg.jrc.ec.europa.eu/svn/inspire-test/. This repository is a copy of a snapshot of the production repository, has the same structure and access permissions and should be used with the same identifier (*inspire-model*). For simplicity, the exercises do not include the ISO repository.

Exercise 2.1. Select a directory on your computer which you will use as the base directory for the repository. Open a command prompt in that directory and check out the repository at https://inspire-twg.jrc.ec.europa.eu/svn/inspire-test/ using the `svn checkout` command (section 2.1).

Exercise 2.2. Create a new project as described in section 1. Cancel to select any type of model. Set up the version control settings as described in section 2.3.1 (using *inspire-model* as the identifier).

Exercise 2.3. Create a package in the model and call it *Themes*. Right-click on this folder and import a package (e.g. one of the INSPIRE thems) as described in section 2.3.2. Repeat this step with another one or two packages.

Exercise 2.4. Delete all packages you imported in the previous exercise from the EA project leaving the root package (this cannot be deleted). Import the template package `ANNEX I-II-III.xml`. You will be asked whether to import package as root model. Answer "yes" and the import will start. Now you should have two root models: *INSPIRE Consolidated UML Model* and *Model*. Right-click on *Model* and delete it.

Exercise 2.5. Select the *INSPIRE Consolidated UML Model* package and remove it from version control (section 2.3.3). Then select PACKAGE CONTROL - GET ALL LATEST in the context menu. This will retrieve all sub-packages under version control.

3 Working with the repository

Now that the repository is retrieved on your local computer, you can start developing the model. A version control system works with check-out and check-in. The simple process is as follows: you check out (i.e., retrieve the newest version from the repository), you make your modifications, and then you check in (i.e., uploading your modifications to the repository).

Normally when you are working with a version control system, several users can concurrently work on the same (ASCII) file. When users check in their file, the version control system automatically merges the changes. However, when checking out a package in EA, it locks the file/package in the repository, and thus, it cannot be modified by anyone until it is checked in again. This can of course cause some problems, but it should not be a problem, as rarely several users will work on the same package.

If there are multiple non-versioned packages under one version controlled package, each sub-package can be versioned as well to allow multiple users working in the same package structure. An example is the *Generic Conceptual Model*, which is in one versioned package with multiple sub-packages. Within the *Base models* package, there is another versioned package: the *Network* application schema. This set-up allows only one person to work on the *Network* model, while other people can work on the other sub-packages of the *Generic Conceptual Model* at the same time.

In order to be sure to have the latest model you can always right-click on any package and select PACKAGE CONTROL - GET ALL LATEST in the context menu. Make sure to select *Import changed files only* as otherwise EA will re-import the whole repository, which is time consuming.

3.1 Checking out packages

The principle of checking out packages in EA is illustrated in Figure 3.1. When a user does a check-out on a package (step 1), EA sends a SVN update request to the repository and locks the package, so that it cannot be modified (step 2). The updated XMI files are then imported into an EA project (steps 3 and 4).

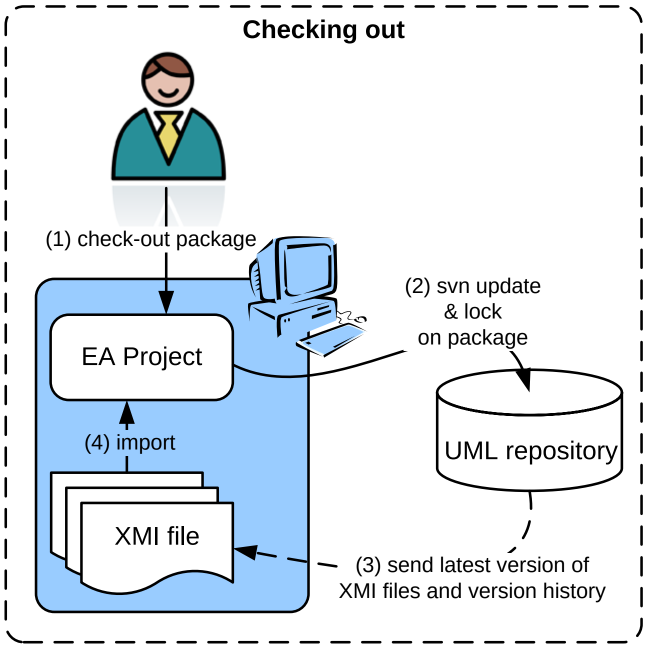

Checking out
(1) check-out package
EA Project
(2) svn update & lock on package
UML repository
(4) import
XMI file
(3) send latest version of XMI files and version history

Figure 3.1: Schematic overview of checking out packages in EA.

Checking out a package in EA is very simple. The icon next to a version controlled package means that it is locked and can be checked out (if you have the appropriate permissions). To check out, simply right-click on the package you wish to modify as shown in Figure 3.2.

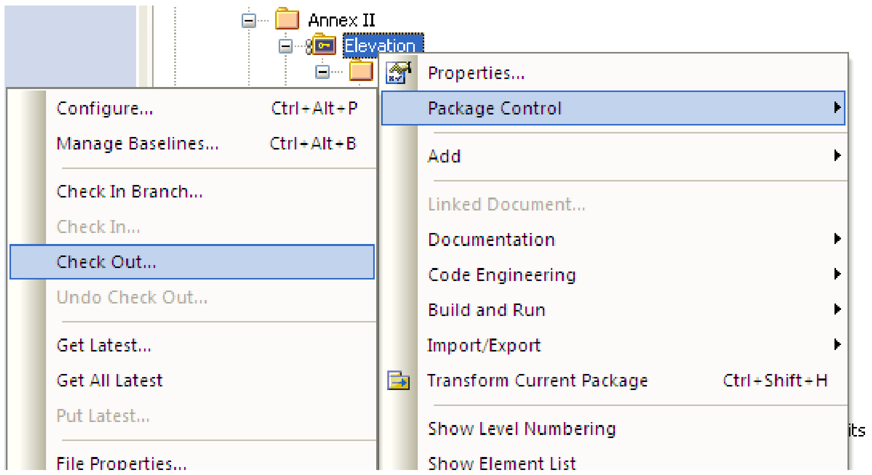
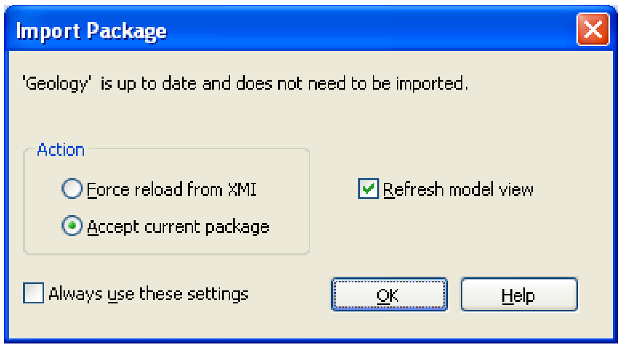


Figure 3.2: Checking out a package.

If a dialog pops up and says that the package is up-to-date and does not have to be imported, accept the current package as shown in Figure 3.3. You can specify that EA remembers this decision. For small packages it is not a problem to reload even if not necessary, but for large packages it might be annoying to wait for a reload that is not necessary.


Import Package

'Geology' is up to date and does not need to be imported.

Action
- ( ) Force reload from XMI
- (•) Accept current package

[✓] Refresh model view

[ ] Always use these settings

Figure 3.3: Import package dialog.

You can always undo a check-out by right-clicking on a package and selecting Package Control – Undo check-out in the context menu. Note that this will discard all the changes you did since the check-out.

3.2 Checking in packages

The principle of checking in packages in EA is illustrated in Figure 3.4. When a user does a check-in on a package (step 1), EA exports the updated package to XMI (step 2). Then EA sends a SVN commit request to the repository and releases the lock on the package, so that it can be checked out and modified again by someone else (steps 3 and 4).

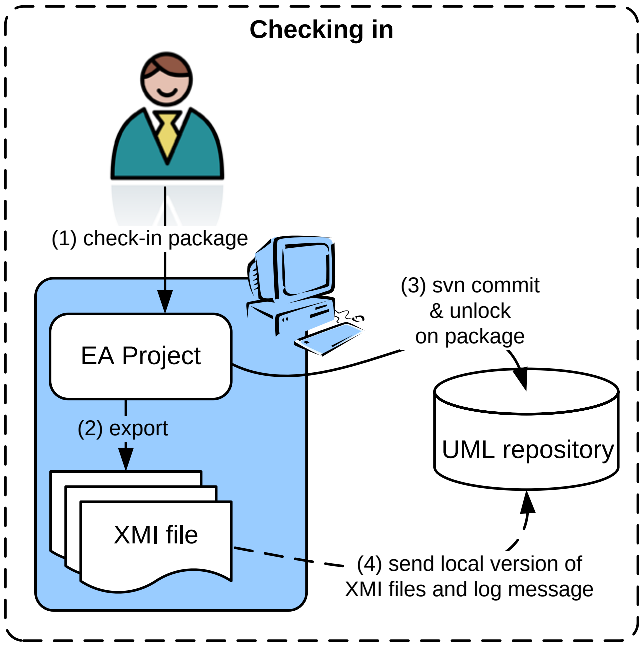

Checking in
(1) check-in package
EA Project
(2) export
XMI file
(3) svn commit & unlock on package
UML repository
(4) send local version of XMI files and log message

Figure 3.4: Schematic overview of checking in packages in EA.

Checking in a package from EA is as easy as checking one out. The icon next to a version controlled package means that it is checked out and can be checked in. To check in, simply right-click on the package you wish to upload to the repository and select Package Control – Check In in the context menu. (Figure 3.5). This will open a comments box where you can add a comment for your revision as shown in Figure 3.6. A default comment containing a date and time is automatically inserted. Also your SVN user name will be added to the comment (even if this is not shown in the comment dialogue).

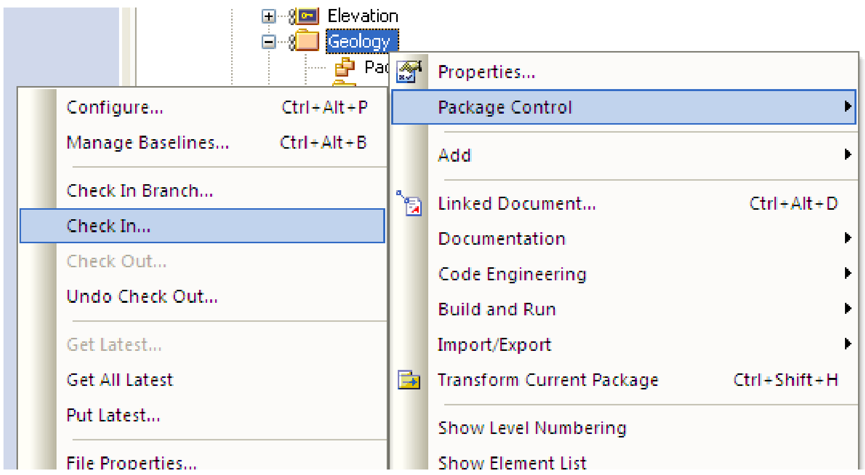
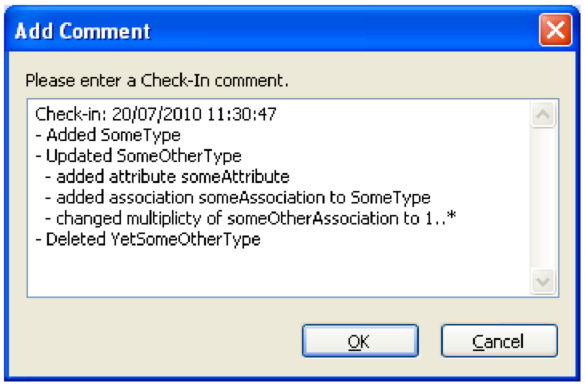


Figure 3.5: Checking in a package.

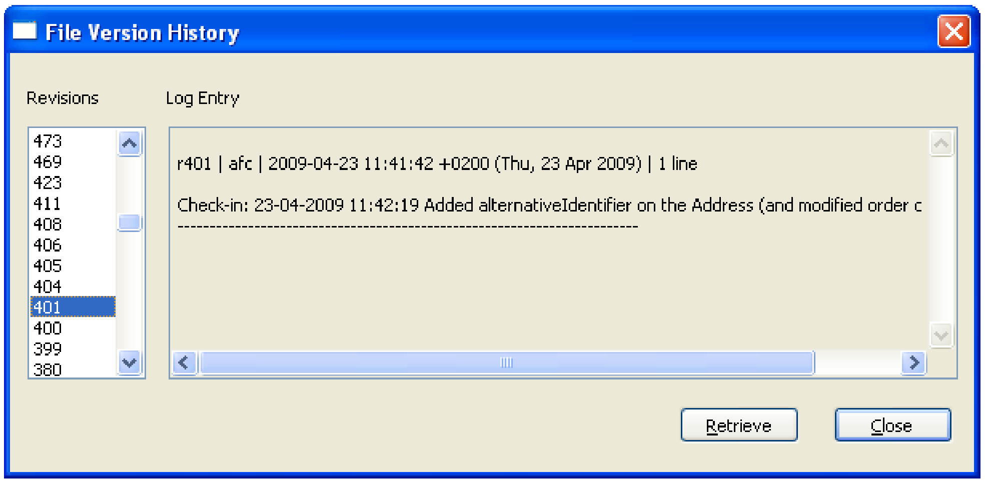
Convention: In order to improve the understanding of the content of revisions made in the repository, make a sensible comment describing the changes made. Be as precise as possible, e.g. by listing the classes that were added or updated, and the changes made to attributes and associations.



```
Check-in: 20/07/2010 11:30:47
- Added SomeType
- Updated SomeOtherType
  - added attribute someAttribute
  - added association someAssociation to SomeType
  - changed multiplicty of someOtherAssociation to 1..*
- Deleted YetSomeOtherType
```

Figure 3.6: Comments on revision.

The check-in comments of a package can be viewed in the package's history. To bring this up, right-click on a package and select PACKAGE CONTROL – FILE HISTORY , which will bring up the File Version History dialogue (Figure 3.7).



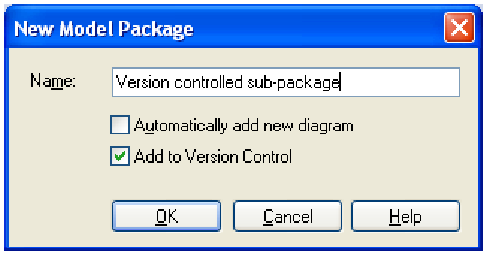
```
r401 | afc | 2009-04-23 11:41:42 +0200 (Thu, 23 Apr 2009) | 1 line

Check-in: 23-04-2009 11:42:19 Added alternativeIdentifier on the Address (and modified order c
------------------------------------------------------------------------
```

Figure 3.7: A package's history.

3.3 Sub-packages under version control

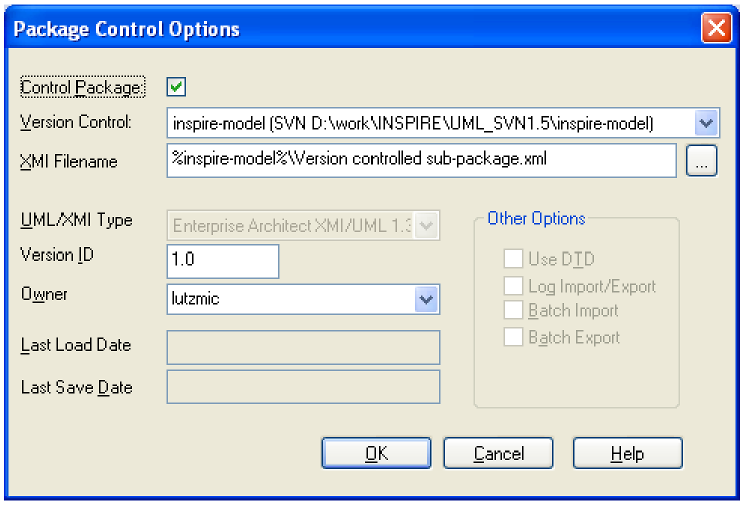
You can version control sub-packages in packages that are already version controlled. The advantage is that several users can work concurrently in the same area, e.g., with multiple sub-packages under the same theme. Either you can version control an already existing sub-package which is not under version control, or you can version control it already at creation time. I you want to version control a package already at creation time, you simply right-click on the versioned package in which you want the sub-package. This will open a dialog as shown in Figure 3.8. You can check the box *Add to Version Control*, which when set will open the dialog in Figure 3.9. Here you select the repository to use and in which directory of your local copy of the repository you want to store the XMI file.


New Model Package

Name: Version controlled sub-package

☐ Automatically add new diagram
☑ Add to Version Control

Figure 3.8: Dialog for a new package.


Package Control Options

Control Package: ☑
Version Control: inspire-model (SVN D:\work\INSPIRE\UML_SVN1.5\inspire-model)
XMI Filename: %inspire-model%\Version controlled sub-package.xml
UML/XMI Type: Enterprise Architect XMI/UML 1.3
Version ID: 1.0
Owner: lutzmic
Last Load Date:
Last Save Date:

Other Options
☐ Use DTD
☐ Log Import/Export
☐ Batch Import
☐ Batch Export

Figure 3.9: Version control settings for a package.

If you want to version control an already existing sub-package, right-click on it and select PACKAGE CONTROL – CONFIGURE in the context menu, and the dialogue shown in Figure 3.9 will appear.

If you want to check in a whole branch of packages this can be done by right-clicking on the topmost package and select PACKAGE CONTROL - CHECK IN BRANCH. This will open a dialog where you can specify which packages to check in.

3.4 Working offline

You have the possibility to work offline. This is done by selecting the menu PROJECT – VERSION CONTROL – WORK OFFLINE. This allows you to check out any package without being connected to the repository. A package that has been checked out while offline is marked with the icon. You can check in the package later when connected to the repository again. It can help in the situation where, e.g., you are not connected to the internet or that the repository is (unexpectedly) down.

NOTE 1 You can check out any package when you are offline, also those you do not have permissions to check in again. You will get a permission error when trying to check in again after working online.

NOTE 2 When performing a check-out while being offline, EA does not lock the package in the UML repository, and it can be modified by someone else. Thus, you may risk overwriting any changes made by another person working online. Thus, if you plan to work offline, coordinate with the people that might work with the same package.

3.5 Concurrent usage

Working with the repository allows concurrent developments in different parts of the model, even if some parts being updated are reused in another package being at the same time updated. Example: assume that an editor of a theme is modelling dependencies to some classes of the generic conceptual model. What happens? Since the dependencies in EA are administered through ID, a class reused will simply be updated in the respective diagrams where it is used. A simple way to reuse a model element is simply to drag the element into a diagram where you need it. Then you can make associations etc. as required.

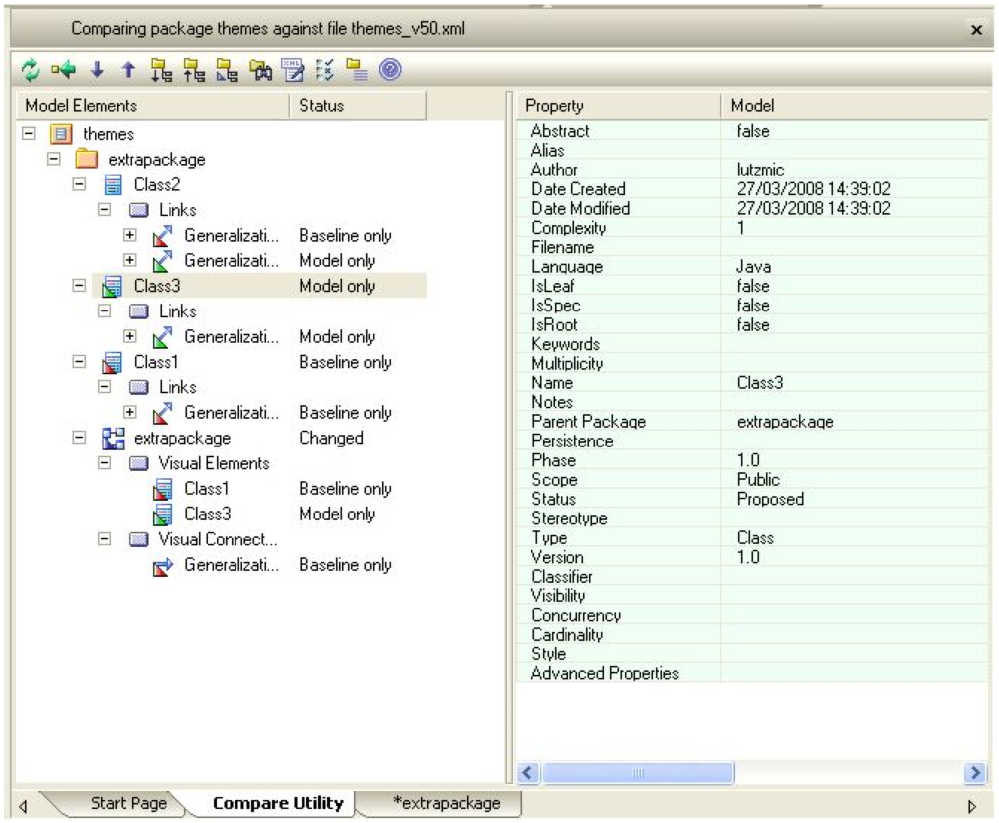
3.6 Comparing versions

EA allows comparing a package in the current model to versions of that package stored in the repository. It is possible to visualize the differences made between the two versions. You can use this functionality to see the changes that you yourself have made in a checked-out package in comparison to the last version in the repository. This can be useful when listing the changes made to a package since the last check out in the check-in comment (see Figure 3.6). Another use of this feature is to see the changes that have been committed by someone else to the repository in comparison to your own model (e.g. before updating the latest changes from the repository).

The EA compare utility can also be used for models that are not version-controlled. In this case, so-called *Baselines* have to be defined that represent a snapshot of the model at a certain point in time. These baselines are stored in an `.xmi`-file, which is then used for comparisons. We will only use baselines for comparisons with older versions (see Section 3.6.2).

3.6.1 Comparing to the latest version

To compare the current model to the latest version in the repository, right-click on a version-controlled package in the *Project Browser* and select PACKAGE CONTROL – COMPARE WITH CONTROLLED VERSION in the context menu. This brings up a comparison window in the main pane of EA. Figure 3.10 shows an example of a comparison between a checked-out model and the latest version in the repository. Added elements are marked with a green and deleted elements with a red triangle. In the example, *Class3* and a generalization relationship to *Class2* have been added to the package *extrapackage* (status *Model only*), while *Class1* and its generalization relationship to *Class2* have been deleted (status *Baseline only*). Also, the class diagram has changed accordingly (status *Changed*). For elements that have been moved to a different package, the status *Moved* is shown.


Comparing package themes against file themes_v50.xml

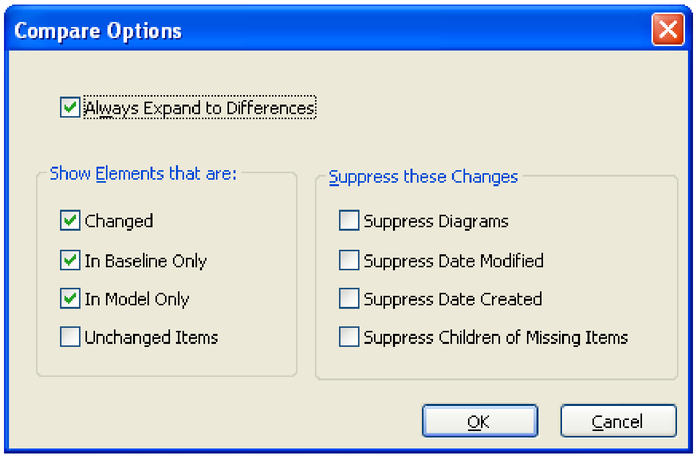
```
Model Elements                          Status
⊟ themes
  ⊟ extrapackage
    ⊟ Class2
      ⊟ Links
        ⊞ Generalizati...               Baseline only
        ⊞ Generalizati...               Model only
    ⊟ Class3                            Model only
      ⊟ Links
        ⊞ Generalizati...               Model only
    ⊟ Class1                            Baseline only
      ⊟ Links
        ⊞ Generalizati...               Baseline only
    ⊟ extrapackage                      Changed
      ⊟ Visual Elements
          Class1                        Baseline only
          Class3                        Model only
      ⊟ Visual Connect...
          Generalizati...               Baseline only
```

Property	Model
Abstract	false
Alias	
Author	lutzmic
Date Created	27/03/2008 14:39:02
Date Modified	27/03/2008 14:39:02
Complexity	1
Filename	
Language	Java
IsLeaf	false
IsSpec	false
IsRoot	false
Keywords	
Multiplicity	
Name	Class3
Notes	
Parent Package	extrapackage
Persistence	
Phase	1.0
Scope	Public
Status	Proposed
Stereotype	
Type	Class
Version	1.0
Classifier	
Visibility	
Concurrency	
Cardinality	
Style	
Advanced Properties	

Figure 3.10: Comparing the current model with the latest version in the repository.

Note that when you compare your own (not checked-out) model to the latest version in the repository, the changes *of your model in relation to the repository* will be shown by the compare tool. So if, for example, a class *Class1* has been added and a diagram *Diagram1* has been deleted in the repository, the compare tool will show that *your* model, *Class1* has been deleted and a diagram *Diagram1* has been added (in comparison to the version in the repository). As this can be confusing, it is helpful to look at the status property to see whether an element exists in the *Model only* or in the *Baseline only* (i.e. in the repository).

By selecting one of the elements in the left part of the pane, the changes are highlighted in green in the right part of the pane. By right-clicking on an element in the left pane and selecting COMPARE OPTIONS or selecting the third icon from the right in the *Compare* toolbar, you can bring up the *Compare Options* dialogue (Figure 3.11). Here, you can specify whether or not to automatically expand all the changes, which elements should be shown and which changes not to consider in the comparison (e.g. *Date Modified*).



[✓] Always Expand to Differences

Show Elements that are:
[✓] Changed
[✓] In Baseline Only
[✓] In Model Only
[ ] Unchanged Items

Suppress these Changes
[ ] Suppress Diagrams
[ ] Suppress Date Modified
[ ] Suppress Date Created
[ ] Suppress Children of Missing Items

Figure 3.11: Setting the compare options.

3.6.2 Comparing to an older version

Comparisons between the current model and older versions are not directly supported in EA, but they can be done using a small trick. You can use EA to find the version you want to compare with - by right-clicking on a package and selecting PACKAGE CONTROL – FILE HISTORY. In the dialogue that is brought up, you can select a version and retrieve it. Your model is then changed to this version. Note that this is only possible if the package is not checked out (to prevent overwriting changes that you might have made in a package that is checked out). Once you have discovered a version to compare with, you can right-click on the package and select PACKAGE CONTROL – COMPARE WITH CONTROLLED VERSION to make the comparison. Note that the comparison will be between the old version and the current version and thus "inverted" as described above.

3.7 Reverting to an Older Version

If you have (or someone else has) committed unwanted changes to the repository, you can revert to an older version stored in the repository. Unfortunately, while reverting to an older version is natively supported by EA (through PACKAGE CONTROL – FILE HISTORY – RETRIEVE), checking this version back in is not. Thus, an outside subversion tool has to be used to perform this activity. You simply retrieve the required version of a package from the repository and check it back in again. Then, in EA you can update the package to the latest version by PACKAGE CONTROL – GET LATEST.

Select your package in the *Project Browser* and browse its file history to find a version that you want to revert the current model to. For retrieving this revision and committing it as the current version to the repository, you can either use the *CollabNet subversion client* or *TortoiseSVN*. In the *CollabNet subversion client*, simply type

```
svn merge -r HEAD:<revision number> "<path to working copy on your PC>"
```

and

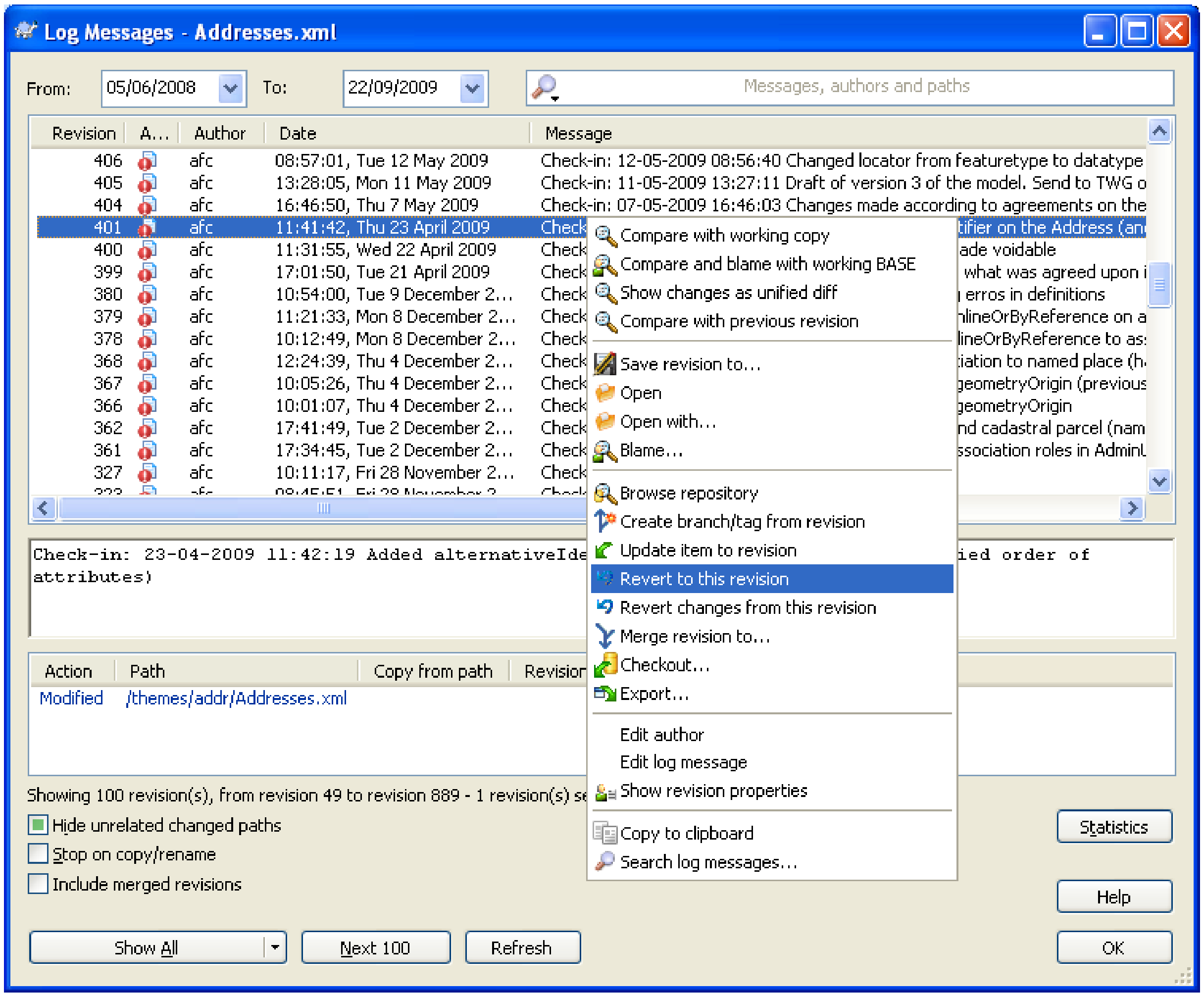
```
svn commit "<path to working copy on your PC>" -m "<commit message>"
```

For example, in order to revert your working copy of the *Addresses* package to revision 1, type

```
svn merge -r HEAD:1 "C:\inspire\inspire-model\themes\addr\Addresses.xml"
svn commit " C:\inspire\inspire-model\themes\addr\Addresses.xml" –m "was version 1"
```

Alternatively, you can use *TortoiseSVN*. Here, switch to *Explorer* and browse to the directory, where your XMI-files are stored. Right-click on the XMI-file of your package and select TORTOISESVN – SHOW LOG in the context menu. In the dialogue (Figure 3.12), right-click on the version you want to compare to and select REVERT TO THIS REVISION in the context menu. Close the dialogue, right-click on the XMI-file of your package and select SVN COMMIT in the context menu. You will see a status message that a new revision has been committed. Return to EA and use Package CONTROL – GET LATEST to update the model to the new (old) version. You can check that the new version is the same as the selected old version by using PACKAGE CONTROL – FILE HISTORY – RETRIEVE to retrieve this version. EA should tell you that this and the current version are the same.

NOTE When using a graphical user interface for subversion such as the *TortoiseSVN* in parallel with the command line utility used by EA, it is important that both SVN clients support the same SVN version (1.5.x or 1.6.x – the third digit can be different). Otherwise you risk receiving the following error message: *“This client is too old to work with working copy”*.



Revision	A…	Author	Date	Message
406		afc	08:57:01, Tue 12 May 2009	Check-in: 12-05-2009 08:56:40 Changed locator from featuretype to datatype
405		afc	13:28:05, Mon 11 May 2009	Check-in: 11-05-2009 13:27:11 Draft of version 3 of the model. Send to TWG o
404		afc	16:46:50, Thu 7 May 2009	Check-in: 07-05-2009 16:46:03 Changes made according to agreements on the
401		afc	11:41:42, Thu 23 April 2009	Check … tifier on the Address (an
400		afc	11:31:55, Wed 22 April 2009	Check … ade voidable
399		afc	17:01:50, Tue 21 April 2009	Check … what was agreed upon i
380		afc	10:54:00, Tue 9 December 2…	Check … erros in definitions
379		afc	11:21:33, Mon 8 December 2…	Check … lineOrByReference on a
378		afc	10:12:49, Mon 8 December 2…	Check … lineOrByReference to as
368		afc	12:24:39, Thu 4 December 2…	Check … iation to named place (h
367		afc	10:05:26, Thu 4 December 2…	Check … geometryOrigin (previous
366		afc	10:01:07, Thu 4 December 2…	Check … geometryOrigin
362		afc	17:41:49, Tue 2 December 2…	Check … nd cadastral parcel (nam
361		afc	17:34:45, Tue 2 December 2…	Check … ssociation roles in AdminU
327		afc	10:11:17, Fri 28 November 2…	Check

```
Check-in: 23-04-2009 11:42:19 Added alternativeIde … ied order of
attributes)
```

Action	Path	Copy from path	Revision
Modified	/themes/addr/Addresses.xml		

Showing 100 revision(s), from revision 49 to revision 889 - 1 revision(s) se

Figure 3.12: Retrieving an old version using TortoiseSVN.

3.8 Exercises

For the exercises in sections 2 and 3 we have created a test repository at https://inspire-twg.jrc.ec.europa.eu/svn/inspire-test/. This repository is a copy of a snapshot of the production repository, has the same structure and access permissions and should be used with the same identifier (*inspire-model*). For simplicity, the exercises do not include the ISO repository.

Exercise 3.1. Try to check out a package assigned to your TWG following the procedure described in section 3.1. Make modifications in your package. E.g., adding a class, package etc. Try to check out another package and observe the message that EA gives you.

Exercise 3.2. Try to check in your modified package following the procedure described in section 3.2. Give a comment to the revision. See the result by right-clicking on the newly checked-in package and selecting PACKAGE CONTROL – FILE HISTORY. If you are following a training session with other participants: After everybody has checked in their packages, retrieve all packages by right-clicking on a package and selecting PACKAGE CONTROL – GET ALL LATEST.

Exercise 3.3. Add a new package in your working package. Remember it has to be checked out, else you cannot modify the package. Check *Add to version control* and specify the repository and put the package in the directory of your main package. Check the package in using branch check in (alternatively the packages can be checked in one by one).

Exercise 3.4. Check out your model. Open the diagram in your package. Go to the *Base Types* application schema package in the *Generic Conceptual Model* and drag and drop the *Identifier* data type into your diagram. Create a class in your package and make a composition from the *Identifier* data type to your newly created class such that the *Identifier* is the composite and your class is the part. Observe the error message given by EA. Now create a composition where your class is the composite and the *Identifier* is the part.

Exercise 3.5. If you are following a training session, the instructor will change the *Identifier* class, e.g., by adding an attribute or renaming the class. After this is done select PACKAGE CONTROL – GET ALL LATEST at the package of the generic conceptual model or even at the root package (make sure to select import changed files only). The changed model element should be updated in your diagram as well. If time permits further similar experiments can be done, e.g., deleting the *Identifier* class (which will also delete the composition).

Exercise 3.6. Check out your package and modify it slightly, e.g. add a class to the package, add an attribute to an existing class, change the type or visibility of an existing attribute, delete a class from the model or from a diagram. Then compare the model with the latest version in the repository (do not check it in again before the comparison!).

Exercise 3.7. If you are following a training session, the instructor will check out a package, modify it and check it in again. Bring up the compare pane for this package and have a look at the changes. Classes that have been deleted should be shown as added classes and vice-versa. Select PACKAGE CONTROL - GET LATEST in the package's content menu. You should see that the changes previously shown in the compare pane.

Exercise 3.8. Select your package in the *Project Browser* and browse its file history to find a version that you want to compare the current model to. Retrieve that version into your model. Use PACKAGE CONTROL - COMPARE WITH CONTROLLED VERSION to compare the retrieved revision to the current model in the repository.

Annnex A: Working with Subversion

SVN can be used to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly-compatible successor to the widely used Concurrent Versions System (CVS). In the following the most basic SVN operations are described. For a comprehensive overview see the "Subversion book" at http://svnbook.red-bean.com/.

The core of a SVN system is a central repository that stores all the files and their revisions (as versions are called in SVN). In the case of the modular data specifications template, this repository is located at https://inspire-twg.jrc.ec.europa.eu/svn/ds_twg/. You can browse the repository content by simply using a web browser[8].

It is important to remember that SVN stores all historical versions of a file. So even if a file is deleted at one point, it is still possible to retrieve the latest (or indeed any previous) version of that file from the repository. It is also possible to revert to previous versions, e.g. in order to undo a change that was committed by accident.

A.1 Basic operations

The first thing you need to do (and if all goes well you need to do this only once) is to create a local copy of the contents of the SVN repository on your local machine (Figure 3.13). This is done using the `svn checkout` command. As a result, the latest revision (r114 in the example shown in Figure 3.13) of all files in the repository will be copied to the selected folder on your local machine.

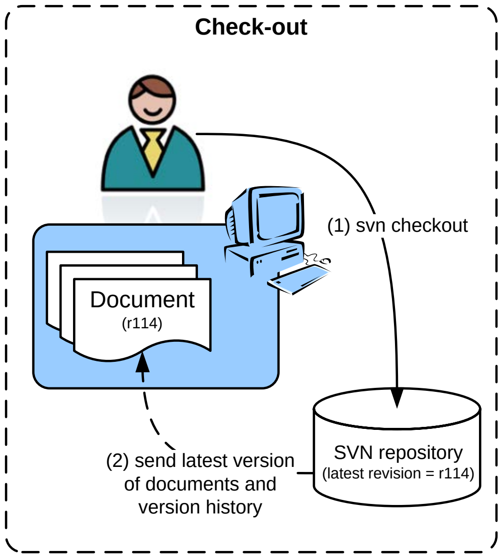


Figure 3.13: Creating a copy of the repository content on your local machine using svn checkout.

Once you have created a local copy, you can modify files and synchronize them with the repository using the `svn update` and `svn commit` commands (Figure 3.14). With `svn update` you make sure that your local copy is up-to-date. As a response to this command, all files that have been added, modified or deleted by somebody else in the meantime will be updated in your local copy. It is important to update before editing a file because working on an out-of-date copy can lead to conflicts when committing a file (see below). In the example shown in Figure 3.14, the local working copy is updated to revision r123.

[8] A slightly more user-friendly interface is available at https://inspire-twg.jrc.ec.europa.eu/voilaSVN/ (select DS_TWG as a repository).

With `svn commit` you send modified files back to the repository. You can apply a commit to just one file or several files or folders at once. Through this command your local changes (new, modified or deleted files) will be propagated to the repository, and a new revision (r124 in the example) will be created.

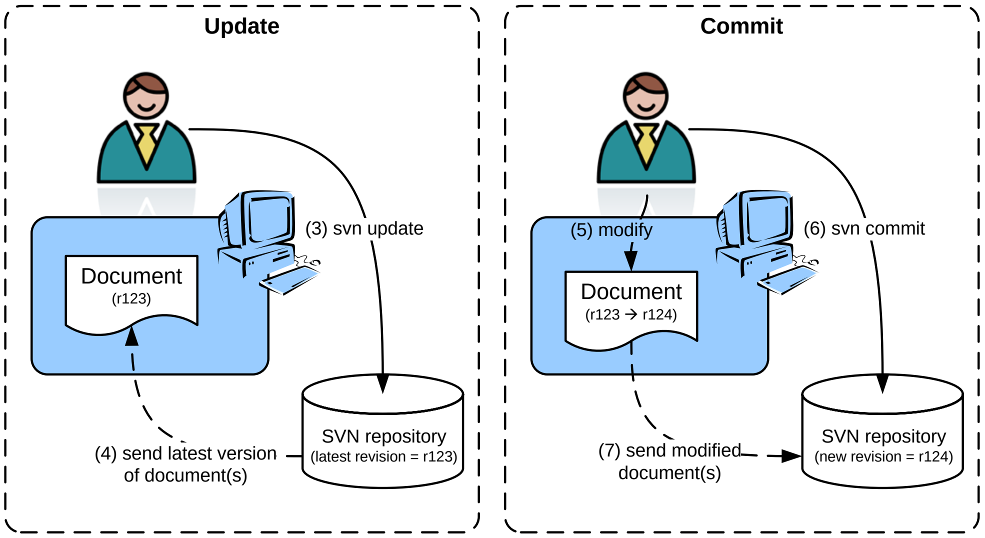


Figure 3.14: Creating a copy of the repository content on your local machine using svn checkout.

A.2 Concurrent editing and dealing with conflicts

Concurrent editing () can lead to conflicts. In the example shown in Figures 3 and 4, two users are working on the same revision (r123) of a document (Figure 3.15). User 1 submits his edits first (steps 5-7 in Figure 3.16), and a new revision (r124) is created. User 2's document is now out-of-date with respect to the repository, and when he tries to commit his changes, a conflict occurs[9] and the commit fails.

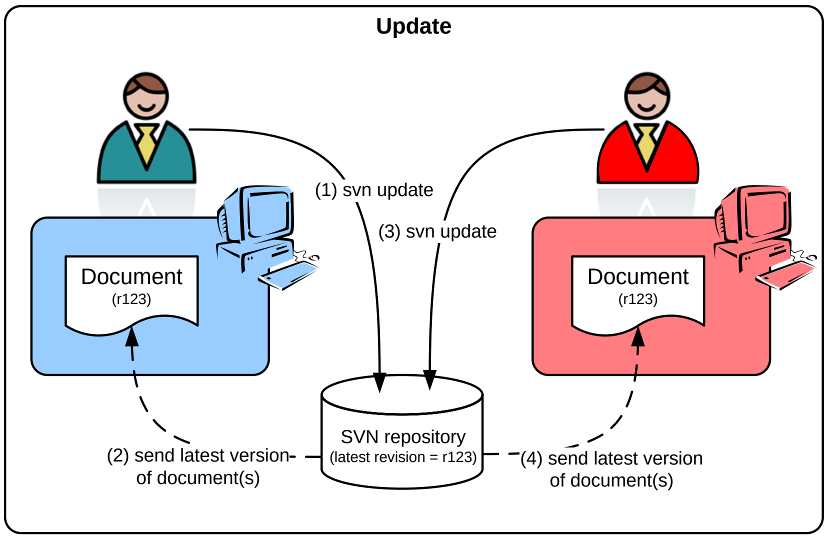


Figure 3.15: Two users are concurrently updating a document.

[9] Note that for text files, SVN is able to merge concurrent edits of the same file, and conflicts only occur when the same parts of the text file are edited concurrently. For binary files like Word documents, however, concurrent editing will always lead to a conflict.

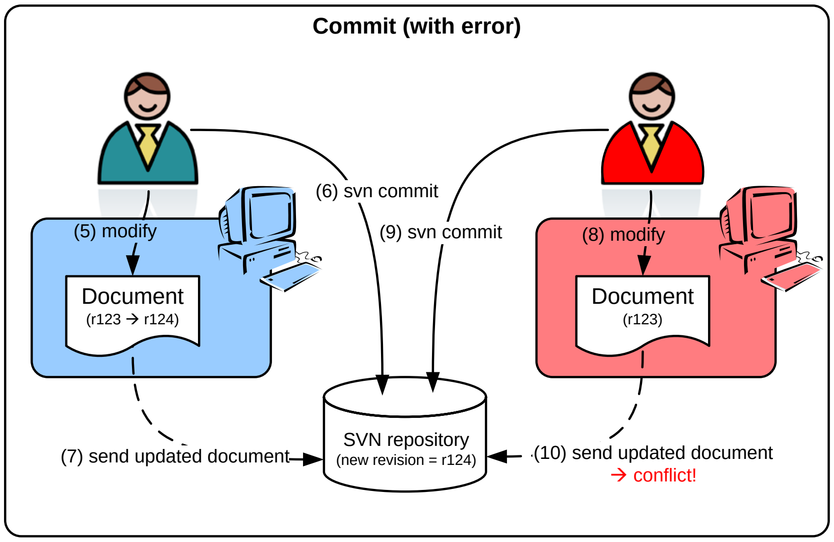


Figure 3.16: Two users are concurrently modifying a document and a conflict occurs when the second user tries to commit his modifications.

In such a case, SVN will create 3 files in the local copy of user 2:

(1) the latest revision of the document in the repository (r124 in the example)
(2) the revision of the document that was created in the local copy of user 2 by the last update (r123)
(3) the current document in the local copy of user 2 (with all local changes)

Usually, it is possible to resolve the conflict by merging (using Word's "Compare and Merge Documents" functionality) documents (1) and (3).

Once the conflict has been resolved the merged file can be committed to the repository, and a new revision (r125) is created.