Tutorial

Converter: Diagram Conversion

From Papyrus 1 to Papyrus MDT

Specificities

Plugin name	**org.eclipse.papyrus.conversion.di2todi**
Papyrus version	**To be used with Papyrus MDT**
Status	**In progress** (class and composite structure diagrams are supported)
Institution	**CEA LIST**
Author	**Manel Fredj**
Required (Papyrus) plugins	• org.eclipse.papyrus.conversion.di2
Other Dependencies	• org.eclipse.m2m.qvt.oml => Install QVT • org.eclipse.ui • org.eclipse.core.runtime • org.eclipse.core.resources • org.eclipse.uml2.uml • org.eclipse.gmf.runitme.notation • org.eclipse.m2m.qvt.oml.emf.util
Extensions	• org.eclipse.m2m.qvt.oml.javaBlackboxUnits • org.eclipse.ui.popupMenus

Table of Contents

Table of Figures

Introduction

This plugin enables to convert diagrams created using papyrus 1.X version into diagram editable by Papyrus MDT. This Tutorial is twofold:

- First, it includes a user guide in order to convert your diagrams (created with Papyrus 1.X) into diagrams editable by Papyrus MDT
- Second, it includes a developer guide in order to help developers to extend the conversion to other diagram not already supported.

USER GUIDE

How to use the converter?

- Add the org.eclipse.papyrus.conversion.di2 + di2todi plugins to your eclipse configuration. These plugins are provided in the Papyrus SVN repository under extraplugins/conversion.

- Launch eclipse, including Papyrus MDT. Import your old-version model created with Papyrus 1.x, let's call it "Example". To this aim you need to import two files: "Example.di2" and "Example.uml", as shown in Figure 2.

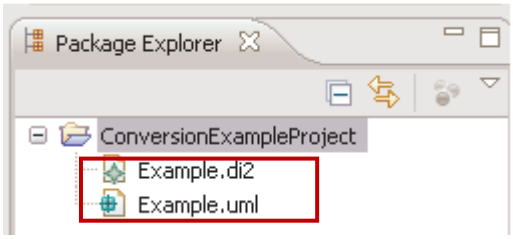


Figure 1 Import Papyrus 1.x Model

- Convert your model. To this aim:
  Right click on "Example.di2"> in the menu, select "Convert Diagram">then, "Di2 to Di Action", as shown in Figure 3.

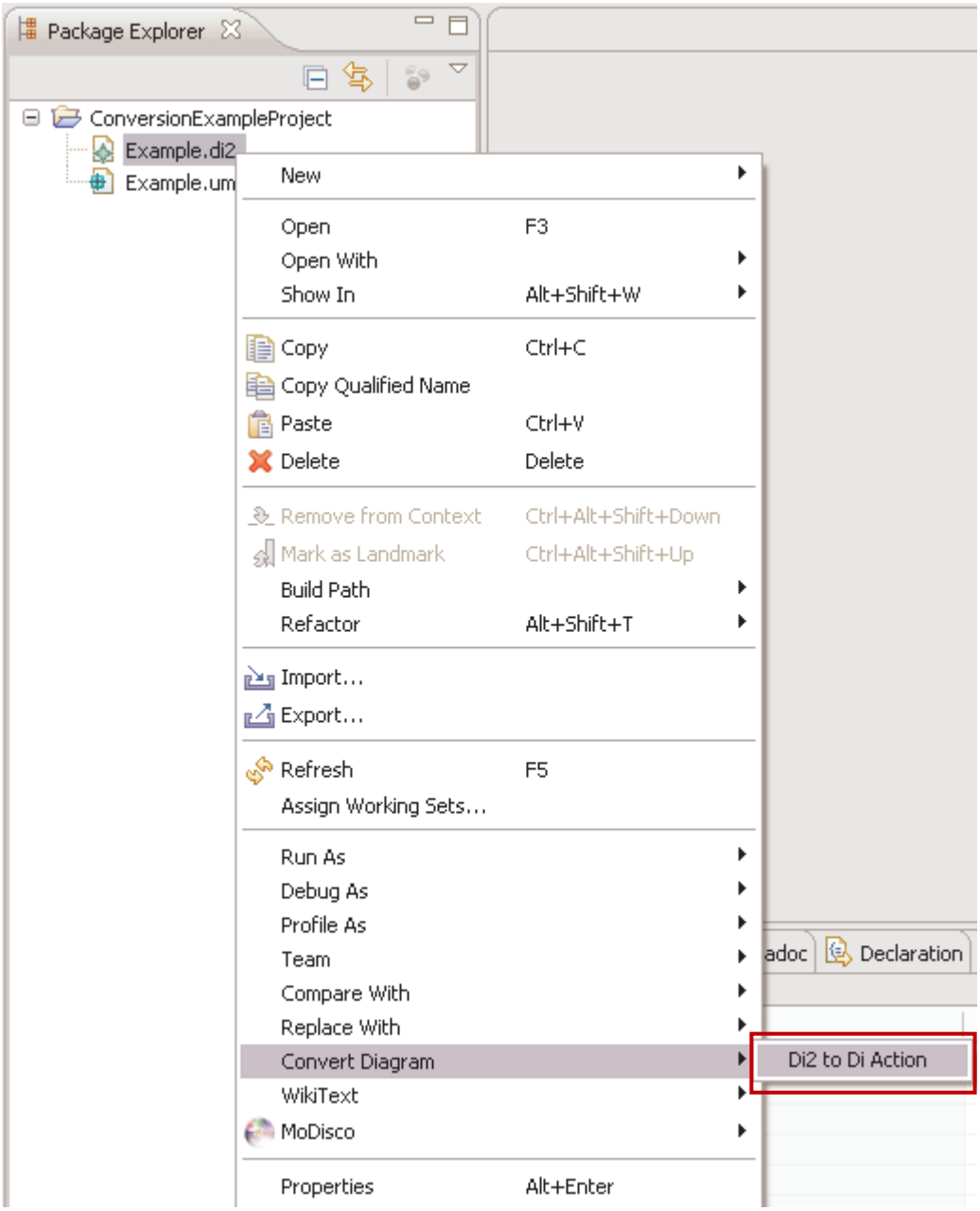


Figure 2 Convert Your Model

- After processing, a new model is created, compatible with Papyrus MDT. Indeed, the conversion creates from the di2 file (i.e., Example.di2) two new files, namely, “Example.notation” and “Example.di”. The uml file (i.e., Example.uml) is used to make reference to the uml graphical elements. At the end of the conversion a message dialog is opened to inform you of the success or the failure of your conversion.

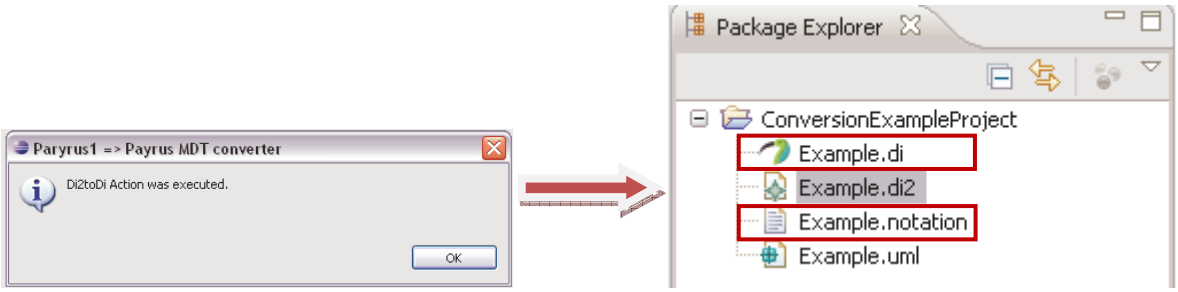


Figure 3 Conversion Result

- Open the generated di file (i.e., Example.di) to show and edit your model.

Internal Process of the conversion

This process is performed transparently to user, however, this brief description aims to provide global overview of the internal process

1st Step

As mentioned in the step2, the diagram converter requires the di2 namespace to be uniquely defined. To this aim as a first step of the internal process, the converter changed the namespace in the di2 file from "http://www.papyrusuml.org" to "http://www.papyrusuml.org/di2".

This step is meant to be transparent to the user, as the converter restores the di2 namespace to "http://www.papyrusuml.org" at the end of the conversion, in order to make your diagram editable by Papyrus 1.x.

However, if the conversion fails before restoring the original name space, the old-version model may be no more editable by Papyrus 1.x. Hence, the user may need to restore it manually by editing the di2 file using a text editor, and removing the "/di2" from the namespace "http://www.papyrusuml.org/di2"..

2nd Step

As a second step, the converter transforms elements described in the di2 file into elements in the notation and di files according to their respective metamodels.

The di file contains references to the different diagrams in your model. The notation file contains a specific description of the elements that are represented graphically in each diagram: size, coordinates, nested elements, and so on.

To perform this step, the converter performed a set of QVT (Query/View/Transform) Operation mappings.

3rd Step

Finally, the converter stores the result of the transformation in to two files, namely, notation and di file, and restores the di2 namespace to "http://www.papyrusuml.org".

Supported Conversions

Herein, we present the different structures supported by the converter. We start first by the class diagram and then, we detail the transformations in the composite diagram.

Class Diagram

- Simple class (cf. Fig.5)
- Class with operations and attributes (cf. Fig.5)
- Comment
- Connection between classes (cf. Fig.5):
    - Dependency
    - Association
    - Link with a comment
    - Realization
    - Generalization

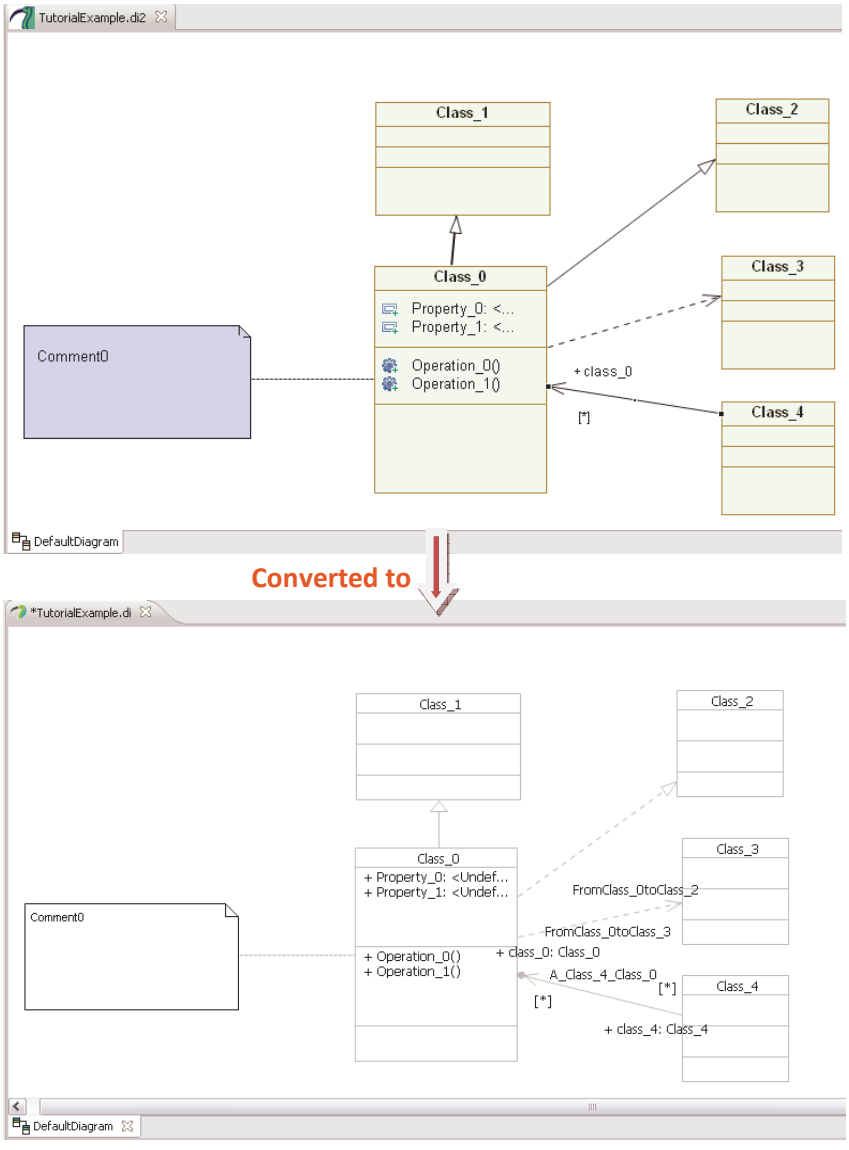
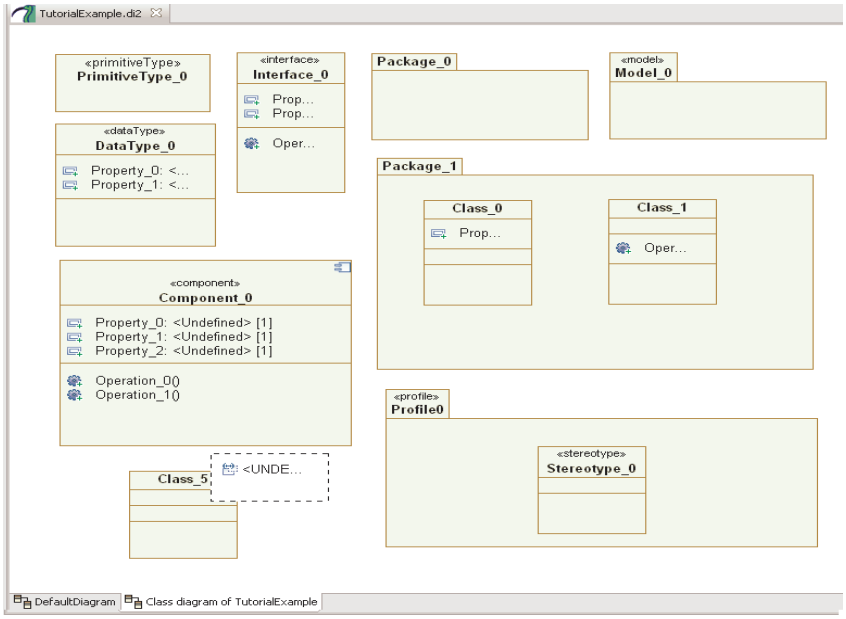


Figure 4 Class Diagram Conversion

- Components (cf. Fig.6)
- Package (cf. Fig.6)
- Package with nested classes (cf. Fig.6)
- Model (cf. Fig.6)
- Interface (cf. Fig.6)
- Profile with nested stereotypes (cf. Fig.6)
- Template (cf. Fig.6)
- Data type (cf. Fig.6)
- Enumeration of literals (cf. Fig.6)
- Primitive type (cf. Fig.6)



Converted to

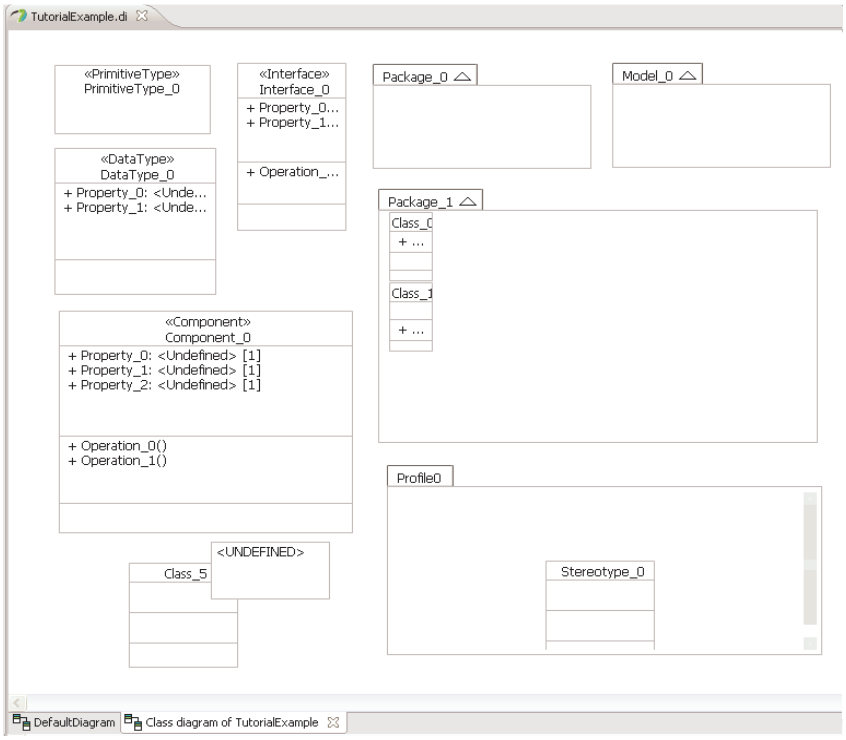
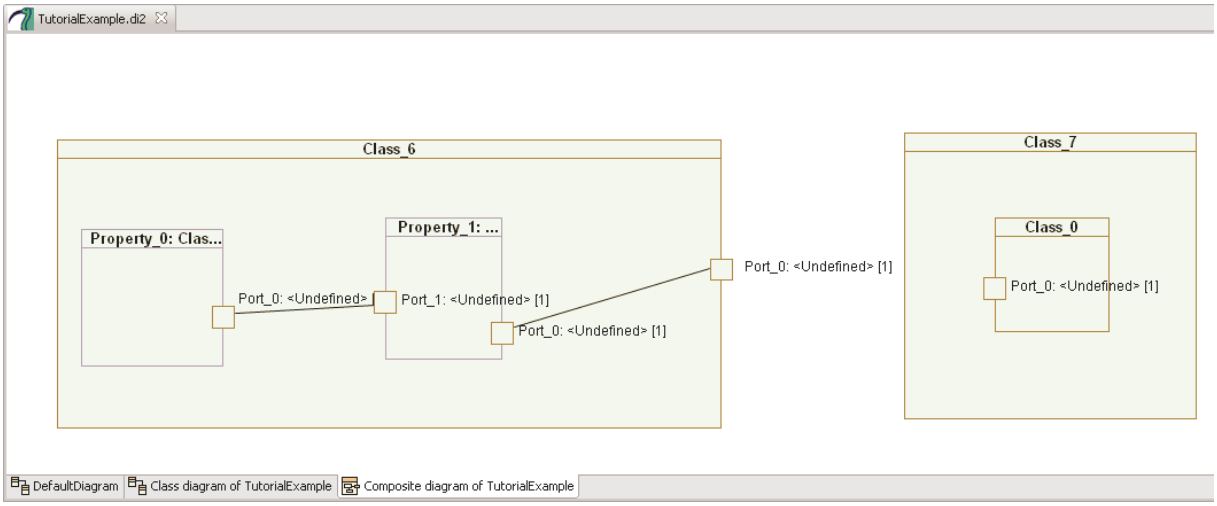


Figure 5 Class Diagram Conversion (cont'd)

Composite Diagram

- Composite classes (cf. Fig. 7)
- Properties of composite classes (cf. Fig. 7)
- Nested classes (cf. Fig. 7)
- Ports (cf. Fig. 7)
- Connection between ports: Connector (cf. Fig. 7)



Converted to

Hree part

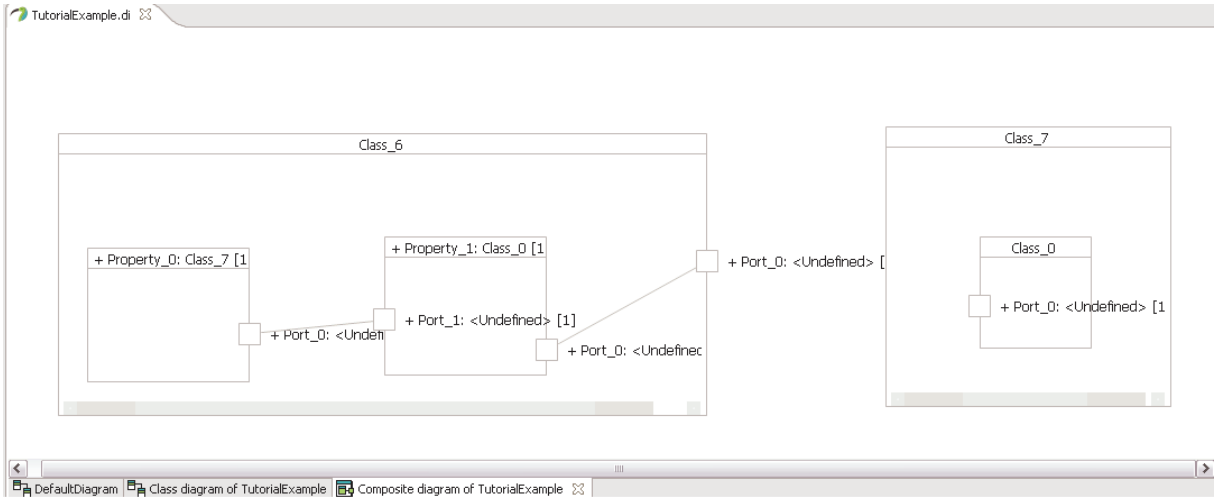


Figure 6 Composite Structure Diagram Conversion

DEVELOPER GUIDE

To convert models from Papyrus 1.x to Papyrus MDT, this plugin uses Operational mappings.

The plugin is developer in Java and its source code is divided into three parts:

- Model Transformation libraries implemented in QVTO
- Black Boxes implemented in JAVA and used by the model transformations
- Java code that is used to call the model transformations and extend eclipse right menu with the conversion action, i.e., “Convert Diagram”.

Black boxes

Java code (glue between model transformations and eclipse pluains)

Model transformations

Figure 8 provides an overview of the plugin content and structure.

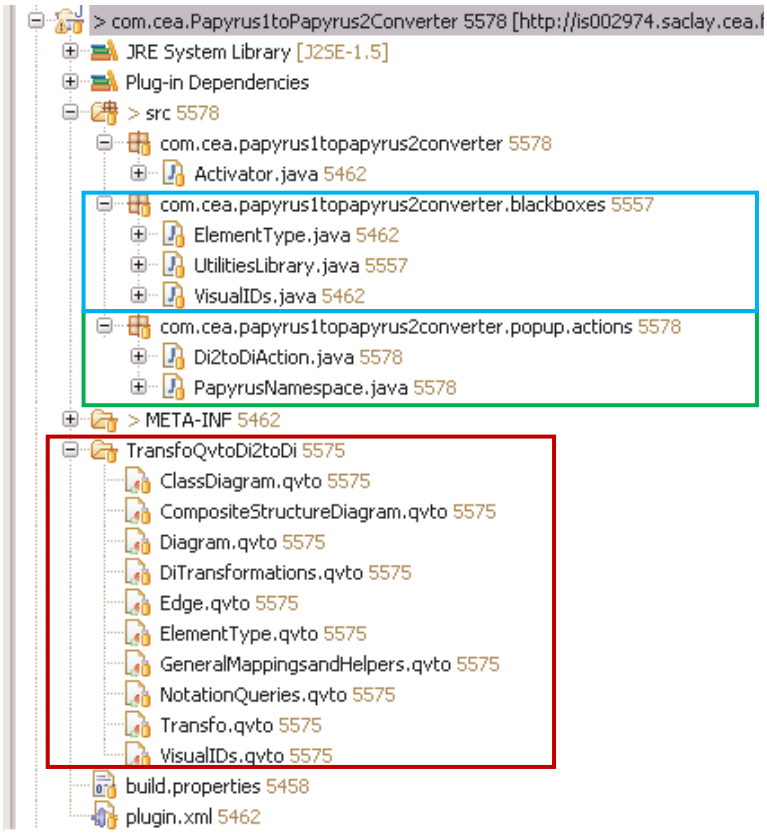


Figure 7 Plugin Content

The main part of the plugin is the model transformation, which includes

1. General libraries as:
    - VisualIDs.qvto
    - ElementType.qvto
    - GeneralMappingsansHelpers.qvto
    - NotationQueries.qvto includes all the queries that are made to the di2 model in order to be used in the notation model.
2. Specific libraries
    - Edge. qvto that converts all sorts of edges (realization, dependency, generalization, and so on).
    - Diagram that converts diagram, for example from a “composite diagram” to a “composite structure diagram”.
    - ClassDiagram.qvto converting class diagram elements.
    - CompositeStructureDiagram.qvto converting composite diagram elements.
    - DiTansformation.qvto enable to create the elements of the di model from the di2 model.
3. The main transformation
    - Transfo.qvto, which triggers the model transformation.