
MAENAD

**Grant Agreement 224442**

# Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles

## Report type Deliverable D5.1.1

## Report name MAENAD Modeling Workbench

| | |
|---|---|
| **Dissemination level** | **PU** |
| **Status** | **Final** |
| **Version number** | **3.0** |
| **Date of preparation** | **2014-02-14** |

## Authors


| Editor | E-mail |
|---|---|
| Sara Tucci-Piergiovanni | Sara.Tucci@cea.fr |

| Authors | E-mail |
|---|---|
| Sara Tucci-Piergiovanni | Sara.Tucci@cea.fr |
| David Servat | David.Servat@cea.fr |
| Chokri Mraidha | Chokri.Mraidha@cea.fr |

| Reviewers | E-mail |
|---|---|
| Henrik Lönn | henrik.lonn@volvo.com |
| Frank Hagl | frank.hagl@continental-corporation.com |


## The Consortium

Volvo Technology Corporation (S) Centro Ricerche Fiat (I)

Continental Automotive (D) Delphi/Mecel (S) 4S Group (I)

ArcCore AB (S) MetaCase (Fi) Systemite (SE) CEA LIST (F)

Kungliga Tekniska Högskolan (S) Technische Universität Berlin (D) University of Hull (GB)

## Revision chart and history log

| Version | Date | Reason |
|---|---|---|
| 0.1 | 2010-12-06 | Outline |
| 1.0 | 2011-09-05 | Intermediate release |
| 1.1 | 2012-08-30 | Update to EAST-ADL 2.1.10 |
| 2.0 | 2013-09-08 | Update to EAST-ADL 2.1.11, instanceRef support and export to EATOP |
| 3.0 prel | 2014-02-14 | Update to EAST-ADL 2.1.12 for review |
| 3.0 | 2014-02-18 | Final version following review |

| Approval | Date |
|---|---|
| Henrik Lönn | 2014-02-20 |

## Table of contents

## List of figures



# 1 Introduction

This deliverable describes the UML-based modeling environment developed within the MAENAD project. This includes the UML modeler Papyrus and the UML profile for EAST-ADL developed for that tool [3]. This profile is currently compliant with M2.1.12 [4].

There are also extra plugins developed, which are described in D5.2.1 – MAENAD analysis workbench, see [1].

The workbench of EAST-ADL consists of a customized version of the Papyrus UML modelling tool, which is developed by CEA in the context of the Eclipse MDT project – see [2]. This is done by a dedicated EAST-ADL palette, which allows for direct creation of EAST-ADL stereotyped elements in the model.

The Papyrus tool provides a UML2 implementation that fully conforms to the OMG standards:

1. Papyrus conforms to the XMI format for saving models;
2. Papyrus conforms to the UML standard semantically and graphically;
3. Papyrus conforms to the Diagram Interchange (Di) standard to handle models graphical interoperability between tools.

To facilitate its extensibility, Papyrus is an Eclipse plug-in that uses other plug-ins such as UML2, EMF and GMF, ANTLR.

Papyrus 0.10 implements the following diagrams of UML2 standard:

1. Activity diagram
2. Class diagram
3. Composite diagram
4. Use Case diagram
5. Deployment diagram
6. State Machine diagram

Papyrus offers advanced functionalities for UML2 profiles support. Some of them are:

1. Profile diagrams
2. Hierarchical profiles
3. Complex typing of stereotype properties
4. Icons and shapes for stereotypes
5. Palette customization for applied profiles
6. Adding a popup menu to connect with an external tool

To create EAST-ADL entities, the user uses the dedicated EAST-ADL palette to create UML entities with the correct stereotypes applied to them. This feature is brought by a specific API which provides profile implementations (known as static profiles). This enables users to add specific functionalities associated to stereotypes, such as filters and automatic derivation of stereotype attribute values. Moreover, a model creation wizard is provided to help users have a well-configured model right from the start.

## 2 Installation

The Modeling workbench is provided as a complete bundle ready to be used, including some examples, via the link:

www.maenad.eu/public_pw/Tooling/papyrus0.10-eastadl2.1.12.zip

The installation of the tool and the profile is explained, and an empty EAST-ADL is created. There is also a more complete tutorial of the language in the project presentation material, which could be used as the next step, if the reader wants to develop a more complete EAST-ADL model.

Papyrus is based on Eclipse, and it can be used as a stand-alone RCP (Rich Client Platform), or on top of an existing Eclipse installation. This deliverable is based on the standalone version. There might be some small differences in the user interface compared with the plugin version.

Once unzipped the archive provides an executable: eclipse.exe, which launches an Eclipse application with a full Papyrus+EAST-ADL installation. The user is prompted with the choice of a workspace. You can select the workspace included in the bundle:

After unzipping the bundled archive, you get the following directory in [4].

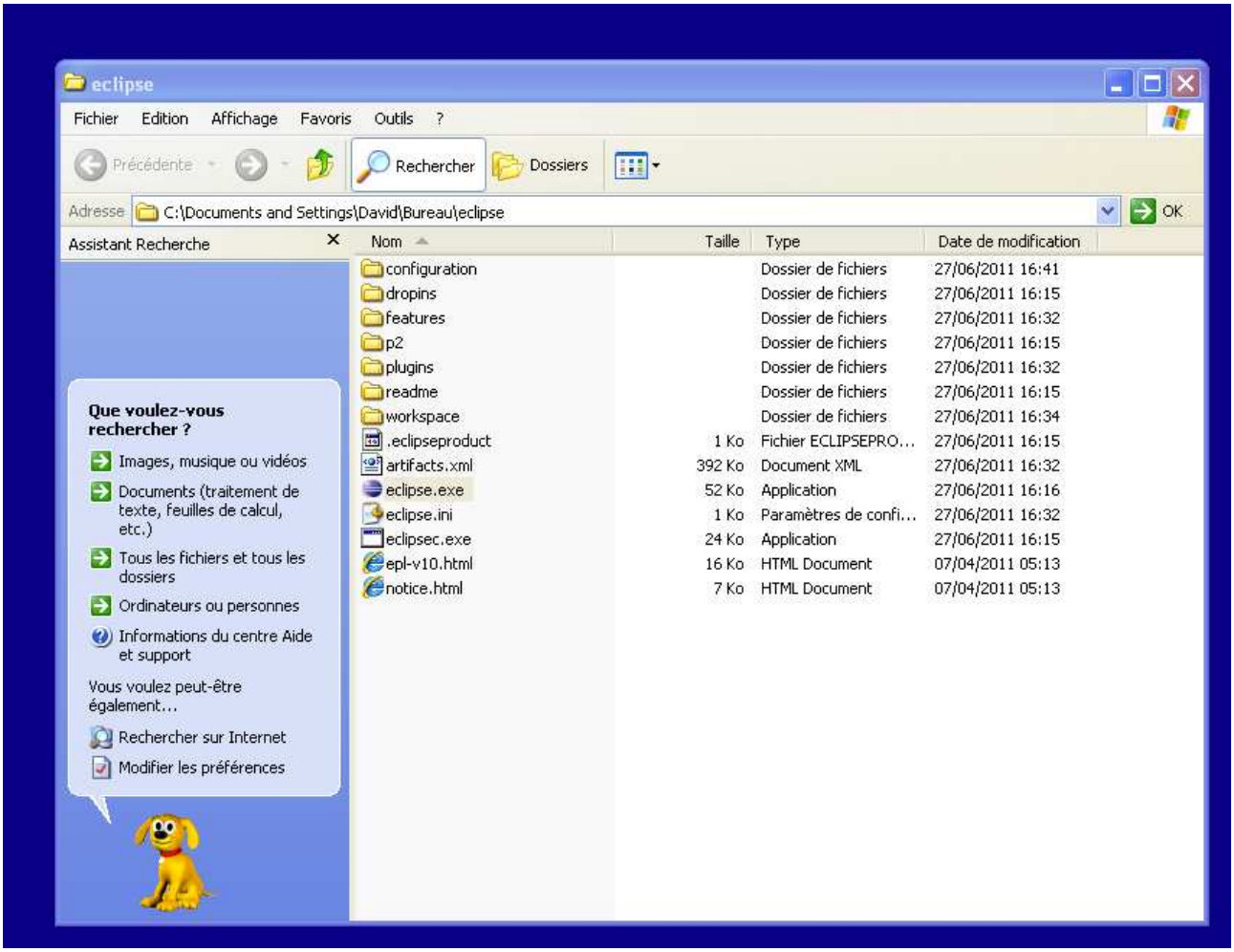



**Figure 1: The bundled archive**

Double-click on eclipse.exe, you are prompted with a workspace selection, i.e. a directory where projects are stored on the hard drive.

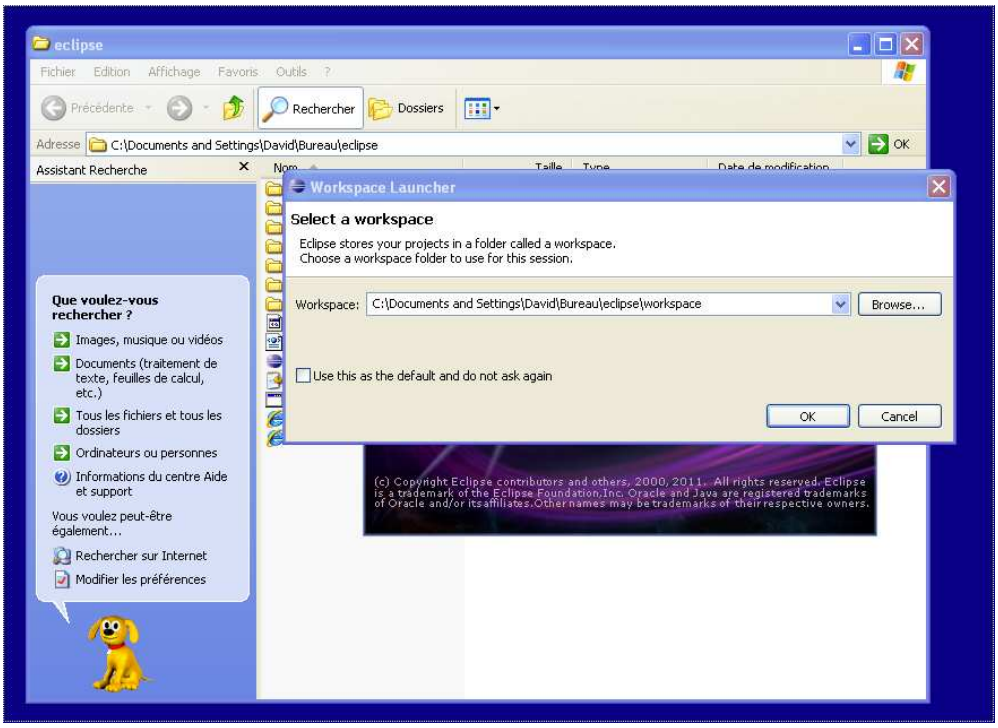


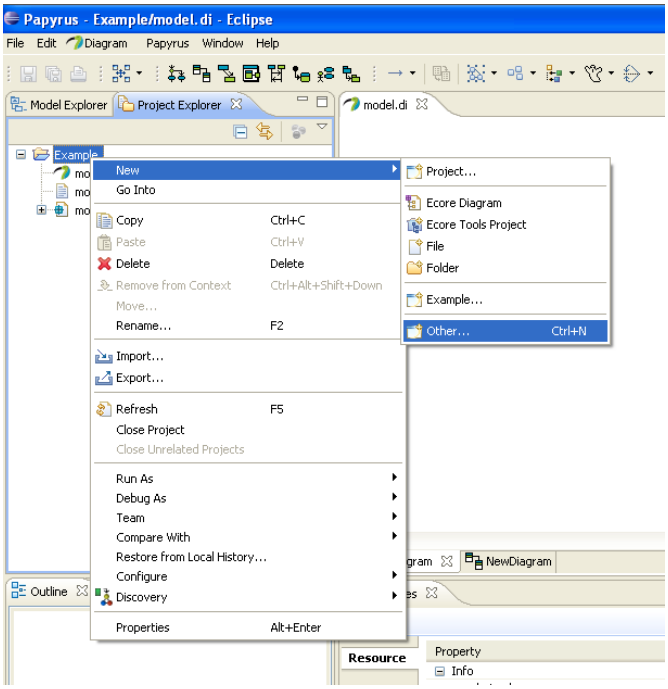

**Figure 2: The workspace launcher**

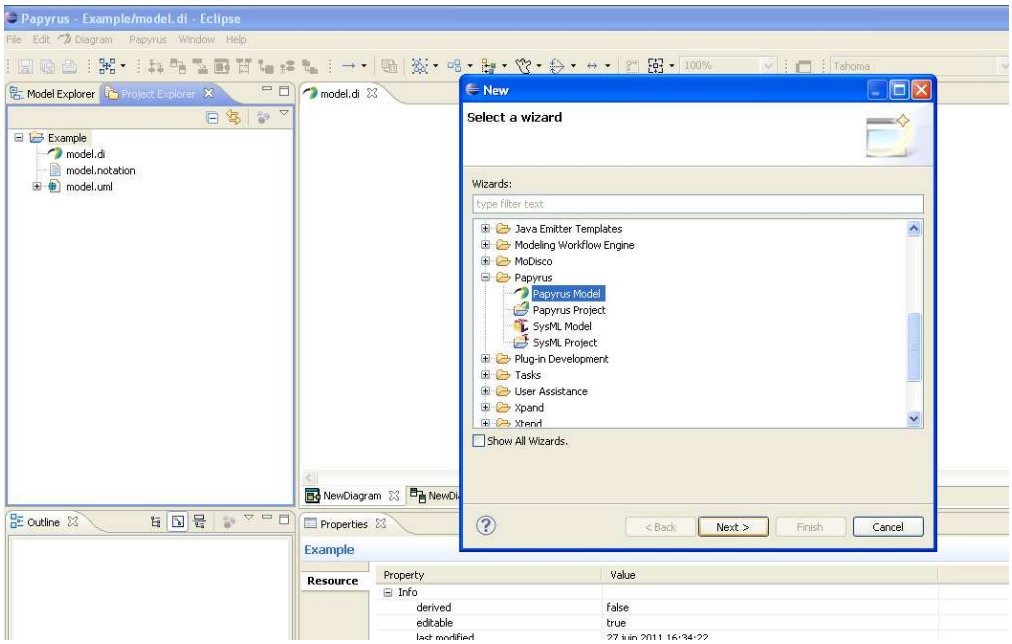
In the following section it is explained how to create an initial model with the EAST-ADL profile applied.

Let us remark that updates of this platform can be obtained at any time after this installation, thanks to Eclipse software updates, simply select Help>Check for updates. You'll be prompted with a list of potential updates of plug-ins installed in your configuration, a check for availability and restrictions will be performed. After a restart, you will have an updated platform.

# 3 Creation of a new model

You can create a completely new model, by using the Papyrus creation wizard. In the navigator view, where files are shown, right click on the project and select New>Other, then Papyrus>Papyrus model. Choose a name, select UML file, then choose the diagrams you would like to be created and choose the EAST-ADL template for the model. See the following set of subfigures of Figure 3.



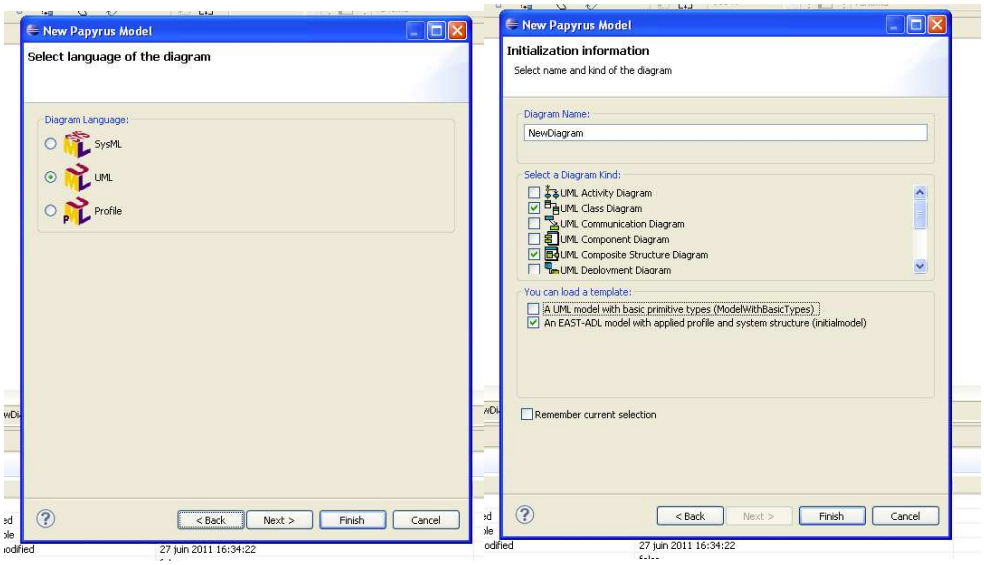



**Figure 3: Steps in creating a model from the wizard**

Now in the Project Explorer view, three files have been created, the .di file, the .uml file and the .notation file. The .di file contains the graphical information and the .uml file is where all model data is stored. At creation time the model is open, but since all diagrams are closed the diagram view is empty and the palette absent. In order to let show up the palette you need to open a diagram. To do this, scroll model elements from the model explorer and choose a diagram (see Figure 4).

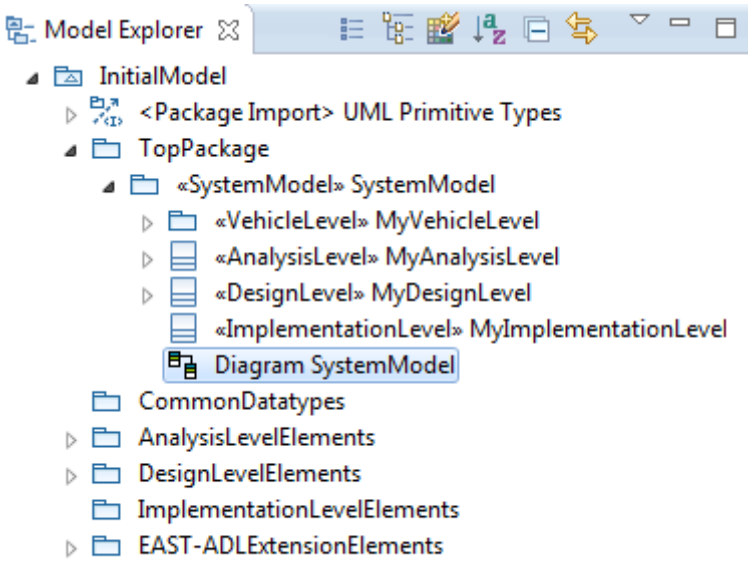



**Figure 4 Model Structure (Model Explorer View)**

Now you should obtain the layout shown in Figure 5.

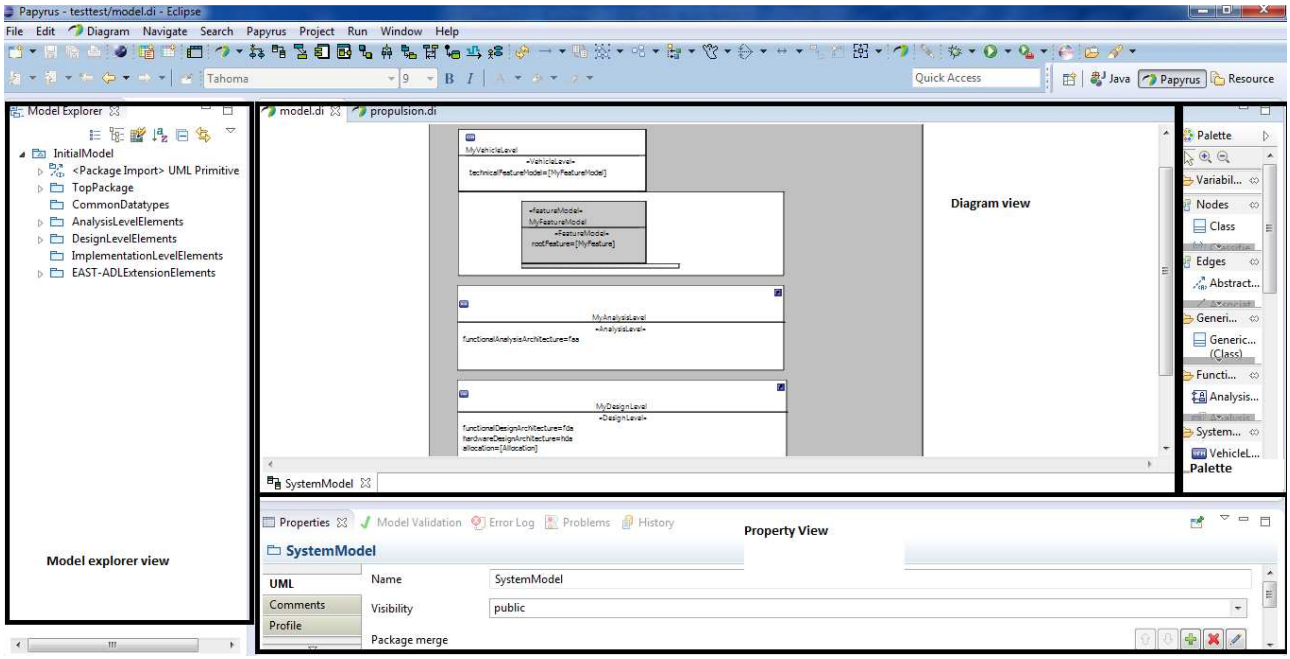



**Figure 5 Main Papyrus Views**

Note that the model is created with the correct structure (Figure 4), as suggested in the specification of the EAST-ADL language, as shown in Figure 6.

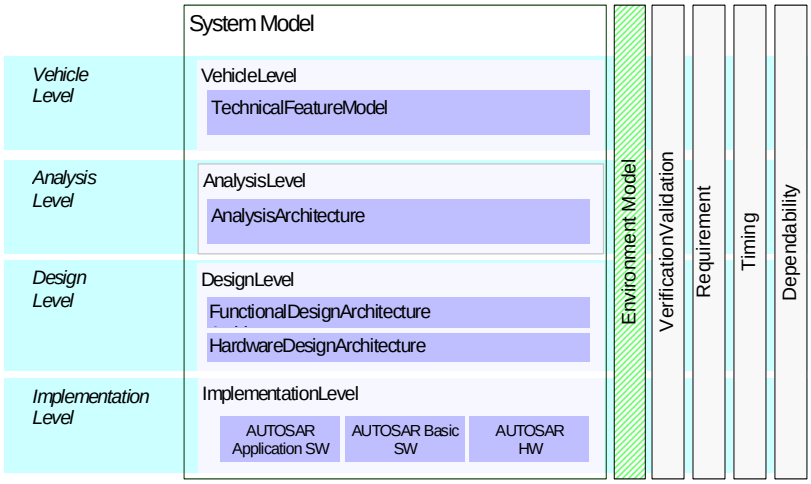



**Figure 6: EAST-ADL abstraction levels**

Let us note that only two types of diagrams are used to graphically build and represent EAST-ADL concepts, namely the Class diagram and the Composite diagram.

Note that both views are useful during the construction of an EAST-ADL model. For instance Class diagrams are suitable to set up Requirements, type definitions in general (AnalysisTypes, DesignTypes, DataTypes, etc) or Feature models, while Composite diagrams are necessary to show prototypes as internal parts as in the case of analysis and design architectures and error models. Figure 7 and Figure 8 shows the use of respectively Class diagram and Composite

diagram for safety-related concepts representation and functional architecture representation of the Propulsion case study.

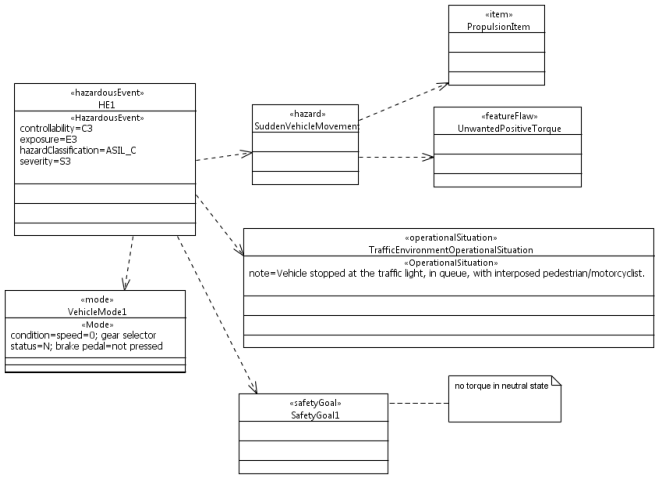



**Figure 7 Class diagram use for Safety concepts of the Propulsion case study**

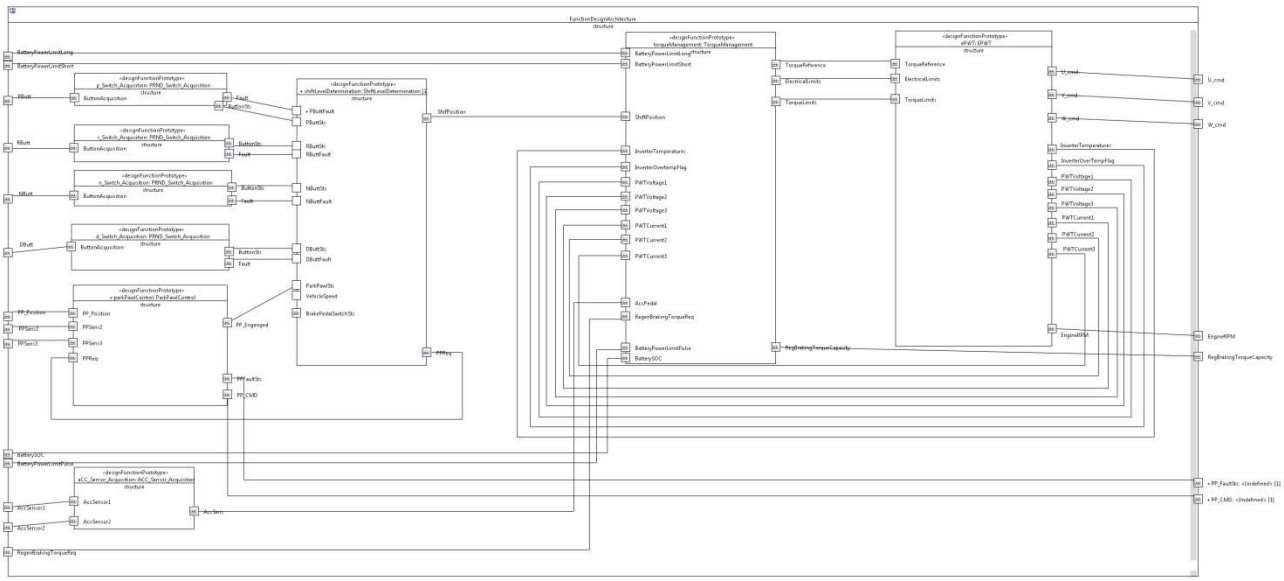


**Figure 8 Composite Diagram for Functional Architecture of the Propulsion case study**

# 4 InstanceRef support

The UML profile, up to the implementation of the EAST-ADL M2.1.10, has supported the instanceRef concept by adding a 'path' attribute to all stereotype's attributes that should represent an instanceRef. This support has been found very weak from a user point of view, but more importantly it has been found as not sufficient to correctly serialize instanceRefs in the EAXML format. Just to make an example, consider the case of the Realization concept, which has two <<instanceRef>> attributes: realized and realizedBy of multiplicity [0..*] as shown in Figure 9.

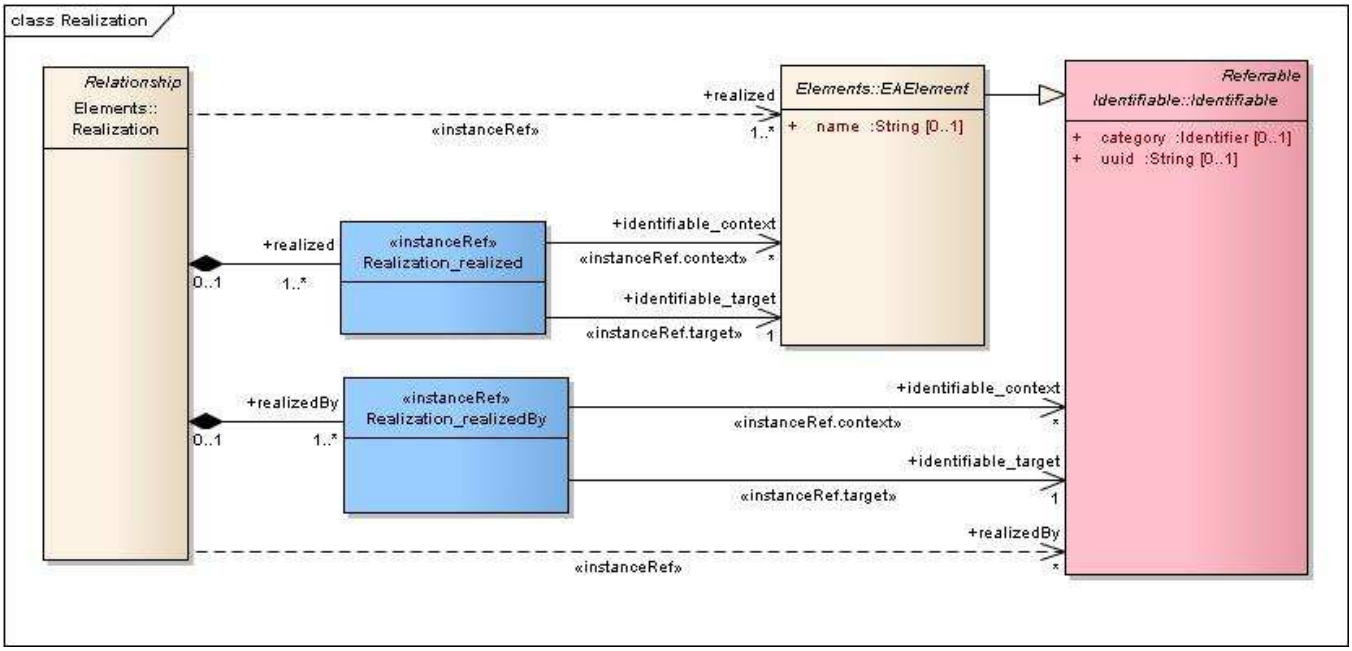



**Figure 9 Domain Model for Realization**

Up to the implementation of EAST-ADL M2.1.10, the Realization concept has been represented as <<Realization>> stereotype with a 'realized' attribute of type NamedElement to set the target of the instanceRef and a 'realized_path' attribute to set the instanceRef context. The problem arises when multiple 'realized' elements should be set, in which the context may not be set for all of them. In this case the correspondence between each target and its own context in the *_path attribute (maybe not defined) cannot be traced.

An alternative solution has been pursued in the implementation for EAST-ADL V2.1.11, now M2.1.12. The idea is to map each instanceRef to a corresponding DataType, defined in the profile. Figure 10 shows the two dataTypes corresponding to 'realized' and the 'realizedBy' instanceRefs.

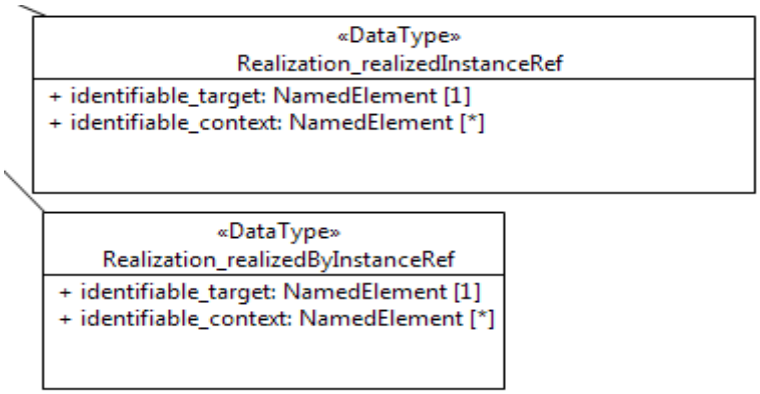



**Figure 10 DataTypes for instanceRefs of Realization attributes**

Figure 11 shows that the realized and the realizedBy attributes are explicitly typed by the previously defined instanceRefs.

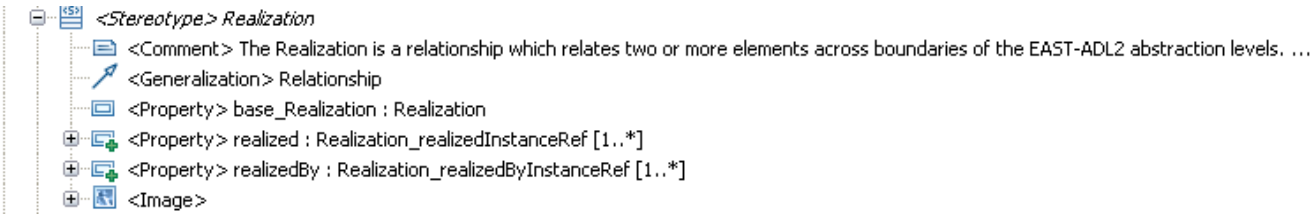



**Figure 11 UML Profile for Realization**

At the user level, data types defined in the profile will be instantiated on the fly. Figure 12 shows a sample model where the target attribute of a functionalAllocation is being set. In the properties view the Allocation Target and the Allocation Target context will appear and values for these properties can be set. Note that values must pre-exist in the model.

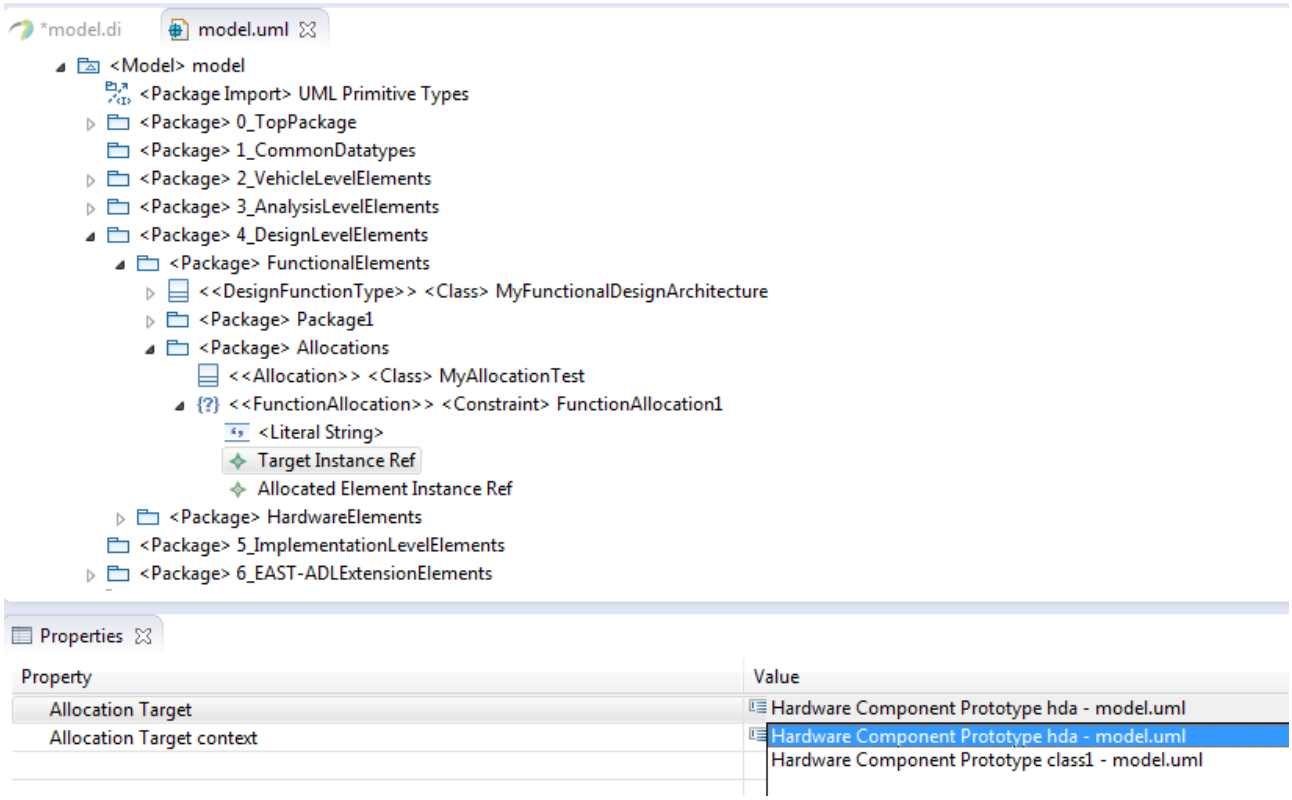



**Figure 12 UML model with TargetInstanceRef for FunctionalAllocation**

# 5 EAXML export

An EAXML export has been realized for the version of the UML profile for EAST-ADL v2.1.11 and then migrated to EAST-ADL 2.1.12. The export uses the serialization services of EATOP which means that the EAXML export will work only if EATOP is installed as well. Figure 13 shows how to run the export: the root model element must be selected in the Model Explorer, then 'Generate eaxml from EASTADL' option must be chosen.

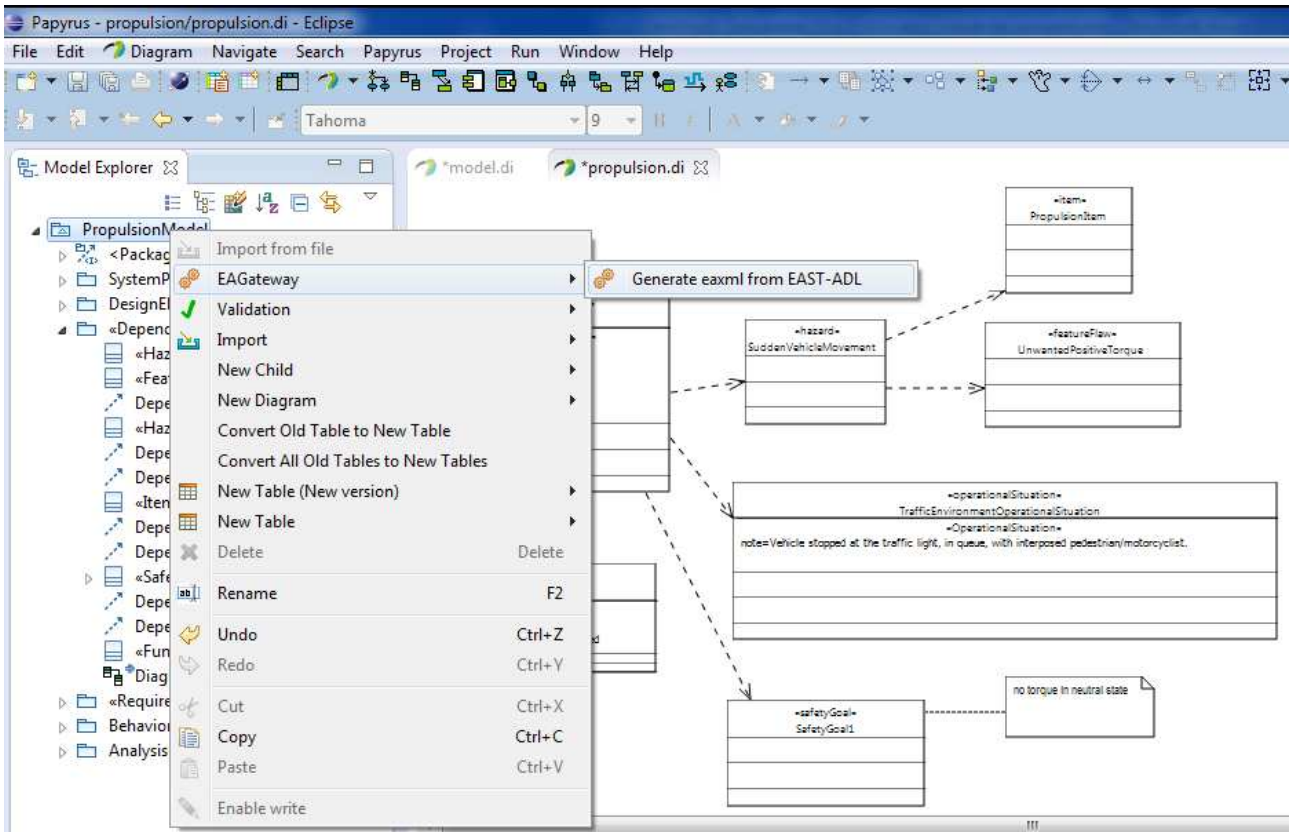



**Figure 13 EAXML export from the Papyrus model explorer**

The EAXML file will be generated and stored in the model project folder as shown in Figure 14.

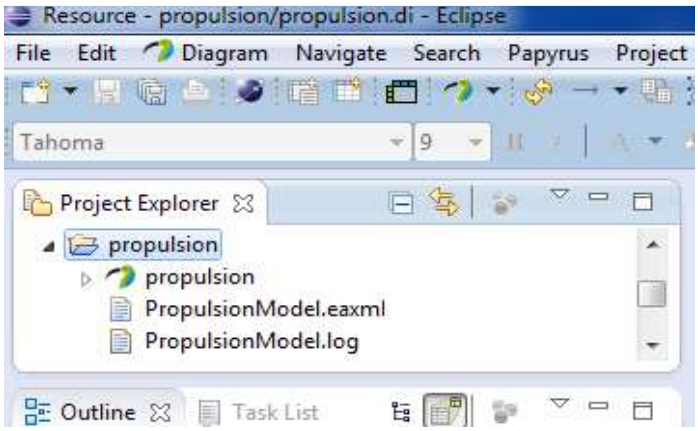



**Figure 14 generated EAXML file**

To open the generated file, you need to click-right on it and choose ‘Open with EAST-ADL Example Editor as shown in Figure 15.

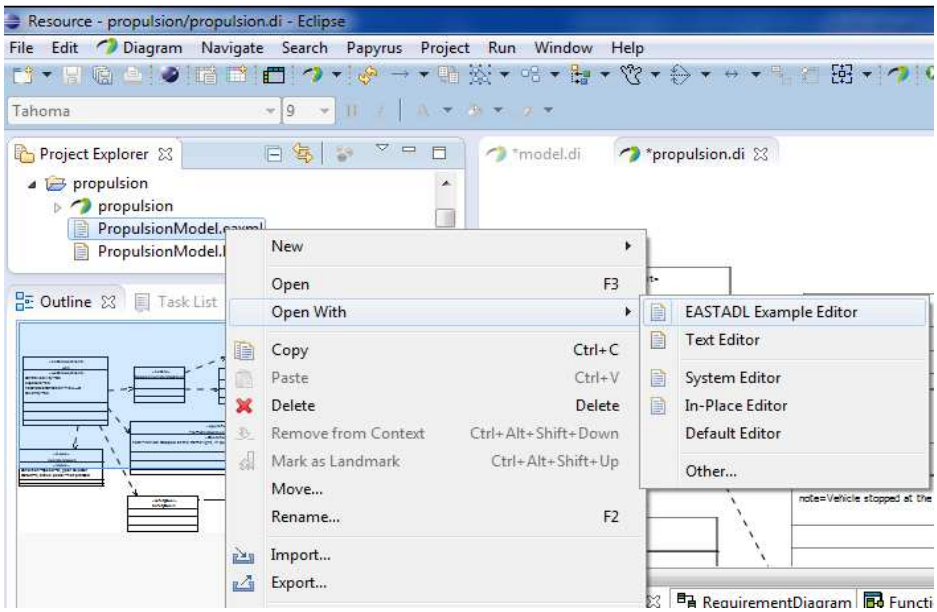



**Figure 15 Opening the EAXML file**

Figure 16 shows the EAXML file content for the Propulsion case study:

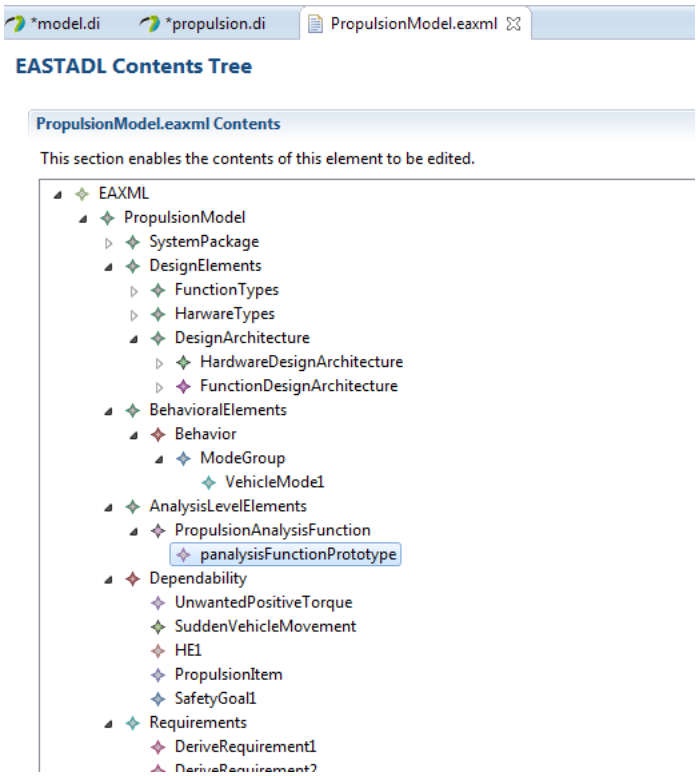



**Figure 16 EAXML model in EATOP**

# 6 References

[1] MAENAD Deliverable D5.2.1 MAENAD analysis workbench.

[2] Papyrus MDT website, http://www.eclipse.org/modeling/mdt/papyrus/

[3] EAST-ADL Profile specification for M2.1.12, http://maenad.eu/publications.html

[4] EAST-ADL Domain Model Specification M2.1.12, http://maenad.eu/publications.html