PRAGMADEV
real time development tools

XMI, a -not so- standard exchange format

XMI is a file format to exchange models from a UML tool to another. Although it has been standardized, its implementations does not reach the expectations. There is still quite some work to do in order to exchange UML models properly.

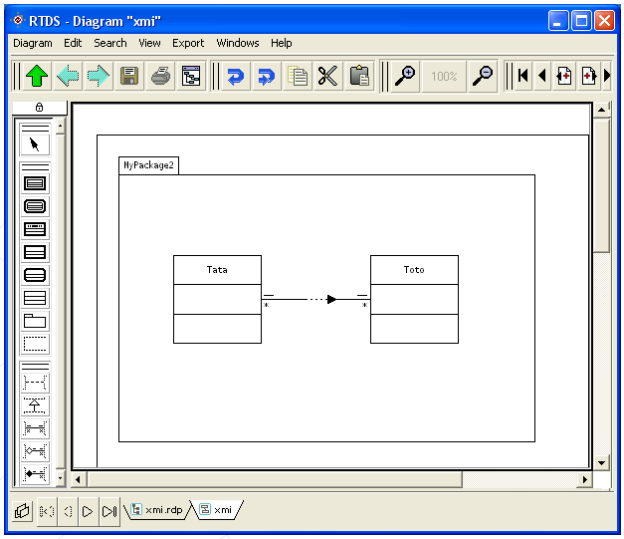
As electronic systems are getting more and more present in our everyday's life, the implied complexity of the embedded software is calling for an efficient modeling technology. Before designing a real system, modeling is about working on an abstract representation of the real implementation. Such a model can be used for documentation or to run a number of verifications very early in the development process.

UML -a merge of several object oriented modeling languages- is a possible candidate for modeling. It has a « model centric » approach meaning the diagrams are partial views of the same model, and an element of the model can be present in several diagrams. The model is the reference and the diagrams are derived from the model. This approach is to be compared with « diagram centric » languages such as SDL for example in which the model is implied by the diagrams.

The UML model representation is standardized and relies on a meta-model called the MOF (Meta Object Facility). In order to exchange models from one tool to another, UML models are to be exported and imported in XMI format (XML Metadata Interchange). XMI is a standardized textual representation of the model as described by the MOF. As we previously explained, in UML the model is the reference, that is why there was no graphical information relative to the diagrams in the first versions of XMI.

Version 1 of UML had a very generic approach making it impossible to make a detailed description of a system whatever the application domain was. Version 2 now allows to define profiles to specialize the modeling language for a specific application domain. The model is more precise and therefore tools can offer simulation, verification, or code generation capabilities. For the time being, two real time profiles have been standardized: Z.109 profile based on SDL for communicating systems, and MARTE profile for system level modeling. Version 2 of XMI includes the description of the profile associated to the model but it is important to note exchanging a model with its profile is very often not possible as tools generally support only one profile, that is very often proprietary.

As we were writing a UML import module based on XMI, it appeared the exchange format could lead to interpretation and that files exported from different tools could be conform to the standard but incompatible with each other. Let's take a very simple example: a class diagram with two classes and an association relation between them.



From the same diagram, here are two XMI files coming from different commercial tools.

```
<uml:Model xmi:type="uml:Model" name=" Model" visibility="public">
  <packagedElement xmi:type="uml:Package" xmi:id="1" name="MyPackage2" visibility="public">
    <packagedElement xmi:type="uml:Class" xmi:id="2" name="Tata" visibility="public"/>
    <packagedElement xmi:type="uml:Class" xmi:id="3" name="Toto" visibility="public"/>
    <packagedElement xmi:type="uml:Association" xmi:id="4" memberEnd="2 3" />
  </packagedElement>
</uml:Model>
```

```
<uml:Model xmi:type="uml:Model" name=" Model" visibility="public">
  <packagedElement xmi:type="uml:Package" xmi:id="1" name="MyPackage2" visibility="public">
    <packagedElement xmi:type="uml:Class" xmi:id="2" name="Tata" visibility="public"/>
    <packagedElement xmi:type="uml:Class" xmi:id="3" name="Toto" visibility="public"/>
    <packagedElement xmi:type="uml:Association" xmi:id="4" visibility="public">
      <memberEnd xmi:idref="2"/>
      <memberEnd xmi:idref="3"/>
    </packagedElement>
  </packagedElement>
```

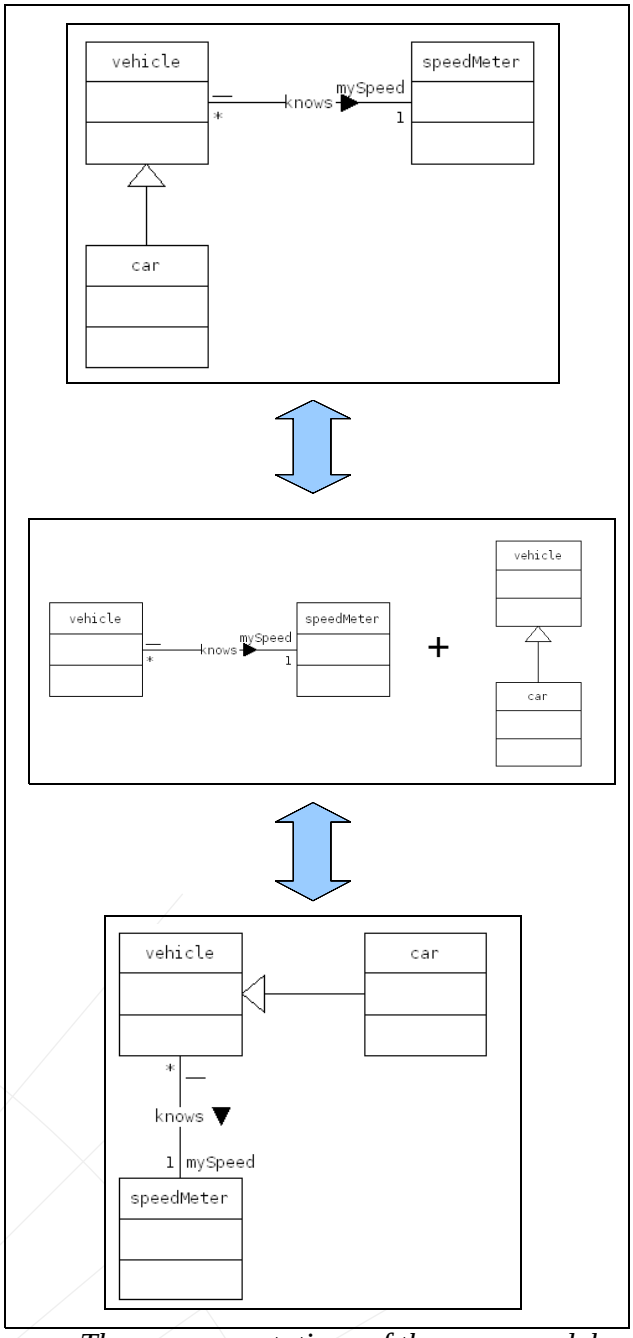
The first tag uml:Model indicates it is a UML model. The second tag packagedElement with Package attribute defines the package containing the two classes. The third and fourth tags with Class attribute describe the two classes inside the package. At last, the fifth tag with Association attribute describes the association between the two tags. But in order to designate the connection at the ends of the association, one tool will use the memberEnd attribute and list the two identifiers, while the other tool will use the memberEnd tag for each end of the association with the idref attribute to identify the classes.

Fondamental differences in the export file organization appear on that very basic example. Exchange of this model between these two tools is obviously not possible.

As explained above the graphical information regarding the symbol placement in the diagrams are not exported, only the model itself is exported. But the same model can be described with very different diagrams and the graphical layout can ease the understanding of the model. The following models have different layouts but will actually produce the same XMI file.

The lack of graphical information is obviously a real loss when exchanging even simple models, not to say models based on different profiles.

As version 1 of UML seemed to be a converging point for modeling languages, version 2 has lead to the creation of multiple proprietary profiles very often undocumented. Very recenlty new domain specific modeling languages have appeared such as, AADL in the space and avionic industry, or Autosar in the automotive industry, that do not come from UML. Portability from one tool to another and ambiguities within the XMI format make basic UML model exchange difficult and any avanced or profile based model unrealistic.



Three representations of the same model

Charles Castelli
Laboratoire COSI - ESIEE-Paris

Emmanuel Gaudin
PragmaDev

Copyright 2009

real time development tools