Interoperability in Meta-Environments: an XMI-based Approach

Roberto Riggio[1], Domenico Ursino[1], Harald Kühn[2, *], and Dimitris Karagiannis[3]

[1]DIMET, Università "Mediterranea" di Reggio Calabria, Via Graziella,
Località Feo di Vito, 89060 Reggio Calabria, Italy
roberto.riggio@gmail.com, ursino@unirc.it
[2]BOC Information Systems GmbH, Rabensteig 2, A-1010 Vienna, Austria
harald.kuehn@boc-eu.com
[3]Institute for Knowledge and Business Engineering, University of Vienna,
Brünnerstrasse 72, A-1210 Vienna, Austria
dk@dke.univie.ac.at

Abstract. In this paper we propose an approach conceived to handle the interoperability in meta-environments. The paper first illustrates the relevance of model interoperability in the present software engineering applications; then, it presents the proposed approach, with a particular emphasis to the relevant role MOF and XMI play in it. Finally, it illustrates a prototype we have realized for verifying the applicability of the proposed approach in a real case, namely the Business Process Management domain.

1 Introduction

One of the most important trends that are presently characterizing the software engineering community is the larger and larger exploitation of the model engineering paradigm. Its adoption is leading to a revolution analogous to that characterizing the 80's of last century, when procedural programming has been substituted by the object-oriented paradigm.

In such a context models will play the key role; they will be exploited not only for documentation but also for software development; they will benefit of software automatic generation techniques.

In this scenario, the Object Management Group (OMG) [9] has proposed to shift from the classic Object Management Architecture (OMA) [3], characterized by an interpretative approach based on the development of complex middleware platforms such as CORBA [8], to the Model Driven Architecture (MDA) [4] characterized by a generative approach based on model transformation.

One of the key issues characterizing this revolution is the necessity to move model-relevant information from one development environment to another one in a transpar-

[*] This work is partially supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract N° 508011, <http://www.interop-noe.org>).

ent and efficient way. This is even more important in case of round-trip engineering since, in this context, it is necessary to migrate models among modelling platforms in a bi-directional way.

Model reuse in environments different from those they have been realized in, has several motivations. Some of the most important ones are the following:

– a single modelling tool typically cannot be used during the whole life cycle of an information system under development, i.e. from its strategic planning to its maintenance;
– even in integrated modelling environments, the various components might be not able to show the best performance for each phase of the system life cycle;
– in the development of highly heterogeneous systems, an organization might decide to use different methods or development processes; as a consequence, a single development tool might be not capable to satisfy all requirements;
– the life time of some projects might be of several decades; it can be easily foreseen that no presently available modelling tool will be available, or at least be retro-compatible, for such a long time;
– in large projects, spread over different companies, there is only a little chance that all participants will use the same set of development tools.

All the examples illustrated above allow us to conclude that, without a good interoperability among different modelling environments, users will be forced to exploit a small set of development tools or, alternatively, to totally renounce to their modelling activity.

The MDA allows the definition of various approaches for handling model interoperability; in this paper we propose a solution based on meta-model transformation. The architecture underlying our solution is shown in figure 1 and is based on the ideas developed in [12].

The core of our proposal consists of the exploitation of a common meta-meta-model and a meta-data exchange facility. In our approach, this has been identified in the Meta Object Facility (MOF) [5, 26] and the XML Metadata Interchange (XMI) [7, 15]. As it will be clear in the following, the exploitation of these two standards allows uniformity among involved models to be easily gained.

We shall illustrate all details of our approach in the next section. Here, we consider extremely relevant to point out that its feasibility has been extensively verified in a real application case, in particular in the Business Process Management domain. In this field, various process modelling languages exist, some of them focusing on business aspects such as ADONIS[1] BPMS language [17], Event-driven Process Chains [19] or UML Activity Diagrams [6], others focus on execution aspects such as Business Process Modelling Language (BPML) [2], XML Process Definition Language (XPDL) [11] or Business Process Execution Language for Web Services (BPEL4WS or BPEL) [1]. These languages are characterized by significantly heterogeneous paradigms; therefore, their interoperability is difficult to be gained. In this context our approach can play a key role; indeed, the capability to define mappings between the meta-models corresponding to the various languages and the MOF meta-model would

[1]ADONIS is a registered trademark of BOC GmbH. All other brands are property of the particular companies or organisations.

automatically imply the possibility to exploit MOF as a common language for defining meta-models and XMI as a common language for meta-data exchange.

In order to verify the feasibility of our approach we have realized a prototype handling the mapping between the meta-model of ADONIS and the MOF meta-model. In our opinion, obtained results are encouraging; they are described below.

The outline of the paper is as follows: section 2 presents a general overview of our approach. Technical details are illustrated in section 3. In section 4 we describe our prototype for handling the mapping between the meta-model produced by ADONIS and the corresponding MOF meta-model. In section 5 we provide a brief overview of related work. Finally, in section 6 we draw our conclusions.

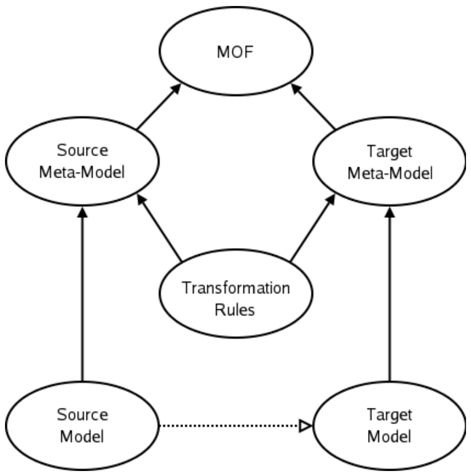

Fig. 1. A general approach for model transformation

2 Overview of the Proposed Approach

The general architecture of our approach is shown in figure 1. Both MOF and XMI play a key role in it.

Recall that XMI defines the way a generic MOF-compliant model will be represented as an XML document. For a given meta-model, each XMI-conforming implementation will produce a DTD (or an XML Schema), representing the meta-model, and an XML document, representing an instance of the given meta-model.

The specific generation rules rely on a MOF definition of the model's meta-model; therefore, a meta-model can have its models interchanged through XMI only if it is represented as an instance of the MOF meta-meta-model. It is worth pointing out that XMI works at all abstraction levels of the meta-model architecture defined by MOF. This implies that it can be used for both the object serialization and the meta-data exchange (see below).

Our architecture is a particular case of the MOF meta-data architecture. An example of the MOF architecture, tailored for the UML environment, is shown in figure 2.

– *The lowest layer*, sometimes called *original level* [18], is that originating the model and often contains run-time data. At this level XMI can be used for handling the object serialization.
– *The model layer* includes the meta-data relative to the lowest layer. Meta-data are aggregated as models. At this level XMI can be used for handling the model or the meta-data exchange among tools using the same meta-model.
– *The meta-model layer* includes the description of both the structure and the semantics of the meta-data, i.e. the meta-meta-data. The meta-meta-data are aggregated as meta-models. A meta-model is an "abstract language" for describing different kinds of data. At this level, XMI can be used for representing the model language, i.e. the meta-model.
– *The meta-meta-model layer* includes the description of both the structure and the semantics of the meta-meta-data. The use of XMI at this level allows an MOF model to be represented as an XML document.

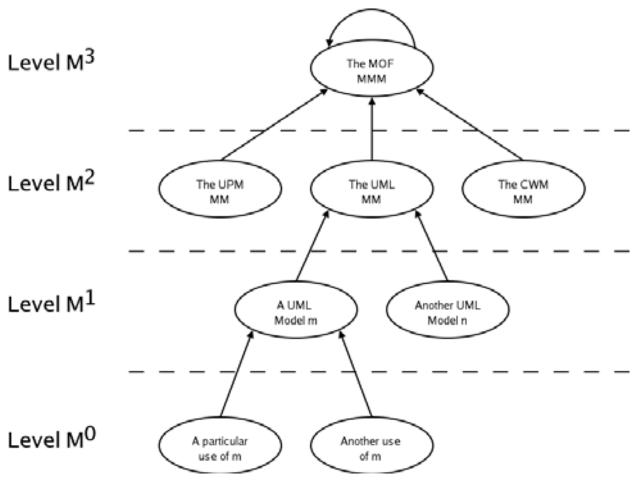

Fig. 2. The MOF four-layer architecture

In the standard OMG modelling stack, the meta-meta-model (also called MOF Model) is self-defined and allows the definition of meta-models at the third layer. The UML meta-model is one of the well-known examples of meta-models; it is possible to define also other generic languages for meta-modelling. In this paper we shall focus our attention on the exploitation of XMI at the second layer of the MOF stack.

3 Technical Details

In this section we shall illustrate the proposed architecture into detail. In order to carry out such a task we shall consider, as an example, the mapping between the ADONIS Business Process meta-model and the MOF meta-model, and the consequent translation of models produced by ADONIS into XMI-compliant documents. The considered version of the XMI specification is 1.2.

ADONIS is a business meta-modelling tool with components such as information acquisition, modelling, analysis, simulation, evaluation, process costing, documentation, staff management, and import/export [17]. Its main feature is its method independence. This means, that starting from the ADONIS meta-tool level, distinct business modelling tools with specialized meta-models can be derived. The main application area of ADONIS is Business Process Management.

Figure 3 contains a fragment of the default ADONIS Business Process meta-model. By its examination we can observe that such a meta-model consists of a composition hierarchy. This is a typical feature of most meta-models. In the hierarchy the `BusinessProcessModel` element consists of three sub-elements, namely `FlowObject`, `Variable` and `RandomGenerator`.

The composition is defined by means of the MOF's composite association form. An MOF composite association form is a (conceptually strong) binding among instances; it is characterized by the following properties:

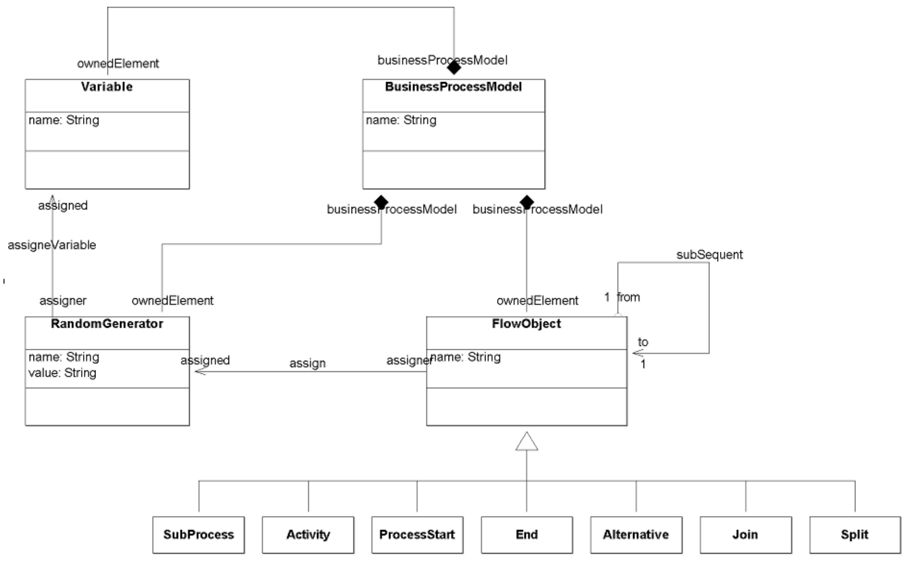

Fig. 3. The ADONIS Business Process meta-model

– it is asymmetrical, with one end denoting the "composite" and the other one representing the "components";
– an instance cannot be a component of more than one composite at a time, under any composite relationship;
– an instance cannot be a component of itself, of its components, of the components of its components, and so on, under any composite relationship;
– when a "composite" instance is deleted, all its components, all the components of its components, etc., are also deleted;
– an instance cannot be a component of an instance from a different package extent (composition closure rule).

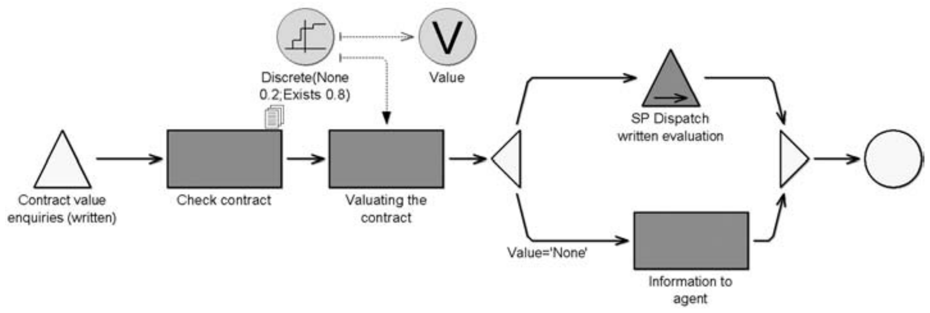

Fig. 4. A model conforming to the ADONIS Business Process meta-model

Mapping the ADONIS Business Process meta-model into the MOF meta-model implies to apply suitable production rules to obtain an XMI-compliant XML document for each ADONIS Business Process model. A simple Business Process model, conforming to the Business Process meta-model of figure 3, is shown in figure 4. The model “BP Contract value enquiries” shows an enquiry of a customer concerning the value of his insurance contract such as a life insurance contract. After contract check, the current value of the contract is calculated. The customer will be informed in written form, which is done in the sub process “SP Dispatch written evaluation”. In parallel, if no contract value was calculated (here: likelihood of 20%), the customer agent will be informed to contact the customer.

In the following we illustrate how the production rules for obtaining an XMI-compliant XML document from an ADONIS Business Process model can be applied. For this illustration we shall consider the model of figure 4 and the corresponding meta-model of figure 3.

Production rules are applied starting from the root of the model, i.e. the unique instance of the `BusinessProcessModel` element. After the root has been considered, rules are applied throughout the model hierarchy by navigating the composition links. For each object, including the root, an element start-tag is generated; to this purpose, the name of the corresponding element in the meta-model is adopted. As an example, if we consider the root in figure 3, we obtain:

```
<BusinessProcessGraph.BusinessProcessModel
 xmi.id="od.1">
```

For each attribute of the current object, a suitable XML element is generated and the attribute is enclosed in it. The name of the element is derived from the name of the attribute, as it appears in the meta-model. As an example, the attribute `name` of the root in figures 3 and 4 is translated as follows:

```
<BusinessProcessGraph.BusinessProcessModel.name>
  BP Contract value enquiries
</BusinessProcessGraph.BusinessProcessModel.name>
```

Each composite association is translated in XMI by means of the XML element containment. As an example, the composite association between the elements `BusinessProcessModel` and `FlowObject` in figure 3 is translated as:

```
<BusinessProcessGraph.BusinessProcessModel.
 ownedElement>
```

As previously pointed out, after the root has been examined, the other objects of the model are taken into account. For each of them, a suitable element is written out in the corresponding XML document; such a task is carried out by following the guidelines illustrated above. As an example, the XML start-tag for representing the object `Check contract` in figure 4 (that is an instance of the element `Activity` of figure 3) is the following:

```
<BusinessProcessGraph.Activity xmi.id="obj.2">
```

Just as before, an element is created for each attribute of the object. In our example, `Check contract` is an instance of the element `FlowObject` in figure 4 and this element has an attribute called `name`; this attribute is translated as follows:

```
<BusinessProcessGraph.Activity.name>
  Check contract
</BusinessProcessGraph.Activity.name>
```

After an element and those linked to it have been examined, the end-tag corresponding to it is generated. As an example, the end-tag associated with the element `FlowObject` and the corresponding instance `Check contract` is as follows:

```
</BusinessProcessGraph.Activity>
```

Analogously, after all the elements of a composite association have been examined, an end-tag relative to it is generated. As an example, the end-tag associated with the composite association between the elements `BusinessProcessModel` and `FlowObject` is the following:

```
</BusinessProcessGraph.BusinessProcessModel
 .ownedElement>
```

Finally, as far as the simple association is concerned, its links are represented in the content of a suitable element contained in the standard `XMI.content` element. As an example, consider the cyclic association `subSequent` in figure 3, recursively linking the element `FlowObject`, and the corresponding instance in figure 4, linking `Check contract` to `Valuating the contract`; the associated XML code is the following:

```
<BusinessProcessGraph.subSequent xmi.id="con.1">
  <BusinessProcessGraph.subSequent.from>
    <BusinessProcessGraph.Activity xmi.idref="obj.2" />
  </BusinessProcessGraph.subSequent.from>
  <BusinessProcessGraph.subSequent.to>
    <BusinessProcessGraph.Activity xmi.idref="obj.3" />
  </BusinessProcessGraph.subSequent.to>
</BusinessProcessGraph.subSequent>
```

In an analogous way all the other elements, attributes, composite associations and simple associations of the model can be represented within the XMI-compliant XML document.

At the end of the whole process, the end-tag of the root is generated. As far as our example is concerned, the following end-tag is written out:

```
</BusinessProcessGraph.BusinessProcessModel>
```

This closes our discussion about the translation modalities of our approach.

It is worth pointing out, that even if in our discourse we have considered the translation from ADONIS to MOF, the approach we are proposing here is general and could be applied for translating any Business Process model to MOF. For this reason, we can say that it guarantees the interoperability among different meta-models.

4 Prototype

In this section we describe the prototype we have realized for handling the mapping between the ADONIS Business Process meta-model and the MOF meta-model, and the consequent translation of models produced by ADONIS into XMI-compliant documents. Our prototype is characterized by two main features:

– Exporting a model produced by ADONIS into an XMI-compliant XML document.
– Importing an XMI-compliant XML document representing a model into ADONIS.

As a consequence, it allows the model interchange between ADONIS and any XMI-compliant CASE tool available in the market.

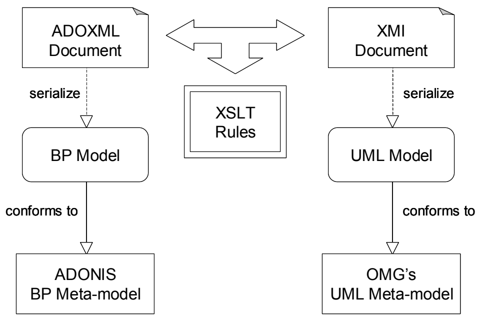

Fig. 5. The architecture of the prototype

Our prototype is strongly based on the W3C family of XML-based standards conceived for handling MOF compatible meta-data. These are very variegated and allow various transformation systems, like XSLT, to be applied to meta-data at any abstraction level. The overall framework of the prototype is shown in figure 5.

The core of the system consists of a set of XSLT templates; these are applied to the source XML document returned by ADONIS (hereafter, ADOXML document) and representing a Business Process model; they produce an XML document compliant with the XMI specifications (hereafter, XMI document). The XSLT templates can be applied also for translating an XMI document into an ADOXML one.

The structure of the XSLT templates is shown in figure 6. Three main packages can be recognized, namely: *Import*, *Export* and *Language*.

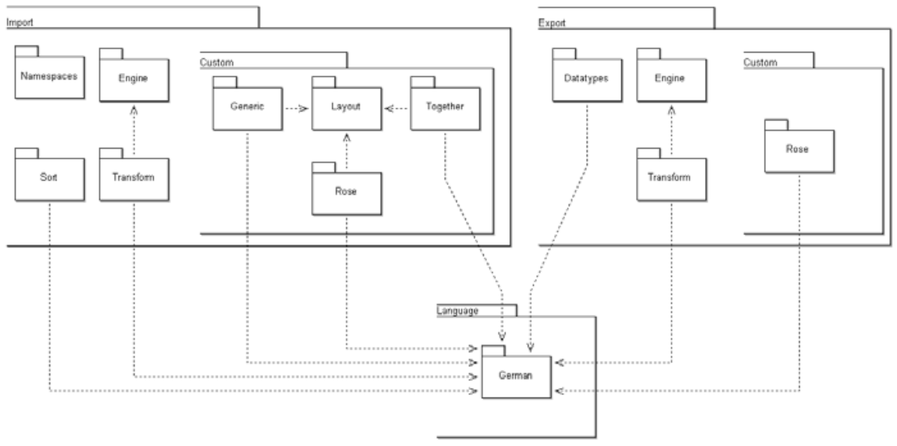

Fig. 6. The XSLT template structure of the prototype

The *Import* package defines the stylesheets used for translating an XMI document into an ADOXML one. This package is decomposed into the following sub-packages: *(i)* The *Namespace* package defines the stylesheets for fixing the namespace declarations in the source XMI document. *(ii)* The *Engine* package defines the stylesheets for translating UML diagrams into an intermediate ADOXML document. *(iii)* The *Custom* package defines the stylesheets for handling the layout information of specific CASE tools. This sheet manages also the heterogeneities regarding the representation of UML attributes and relations [16]. *(iv)* The *Sort* package contains the stylesheets for sorting the intermediate ADOXML document in order to produce a final XML document compliant with the ADONIS specifications.

The *Export* package defines the stylesheets used for translating an ADOXML document into an XMI one. This package is decomposed into the following sub-packages: *(i)* The *Datatypes* package defines the stylesheets for generating the XML document containing all the data types used in the ADOXML document. *(ii)* The *Engine* package defines the stylesheets for handling the UML diagrams defined into the source XMI document. *(iii)* The *Custom* package defines the stylesheets for tailoring the XMI document to a specific CASE tool.

The *Language* package is a support package for making our prototype available in various natural languages such as English, German etc.

The behaviour of the *Import* process is illustrated in figure 7.

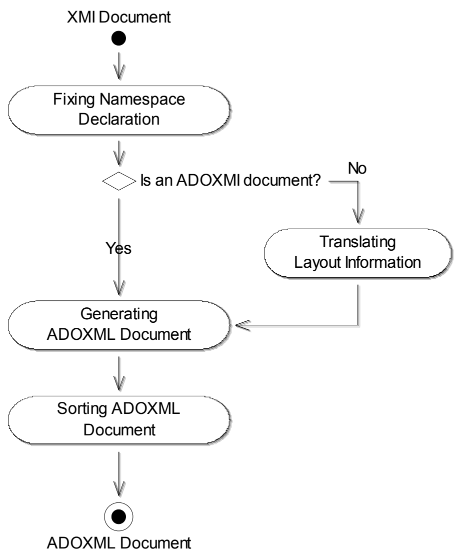
XMI Document
Fixing Namespace Declaration
Is an ADOXMI document?
No
Yes
Translating Layout Information
Generating ADOXML Document
Sorting ADOXML Document
ADOXML Document

Fig. 7. An UML Activity Diagram showing the behaviour of the *Import* process

For each activity shown in the diagram an XSLT sheet is applied to the input XML document. The various activities related to the *Import* process behave as follows:

- *Fixing Namespace Declaration*. The XSLT technology requires the explicit declaration of the namespaces used during the transformation. However, the UML namespace declaration is not consistent through different XMI implementations. Such an inconsistency precludes the stylesheet to work with a generic document. This activity aims at removing such an inconsistency.
- *Translating Layout Information*. The XMI document is examined in order to determine the exporter software. Then, a suitable XSLT sheet is applied to the document for generating an intermediate XMI document. This sheet also handles heterogeneities regarding the representation of UML attributes and relations [16].
- *Generating ADOXML Document*. An intermediate ADOXML document is generated starting from the intermediate XMI document.
- *Sorting ADOXML Document*. In this step the intermediate ADOXML document is sorted for producing the final ADOXML document compliant with the ADONIS specifications.

The behaviour of the *Export* process is illustrated in figure 8. For each activity shown in the diagram an XSLT sheet is applied to the input XML document. The various activities related to the *Export* process behave as follows:

- *Generating DataType Document*. In this step an XML document specifying data types used in the ADOXML document is generated.
- *Generating XMI Document*. During this step the output XMI document is generated starting from the input ADOXML document and the XML document containing the data type definition produced during the previous step.
- *Translating Layout Information*. Starting from the ADONIS diagram representation, a third-party CASE tool diagram representation is generated.

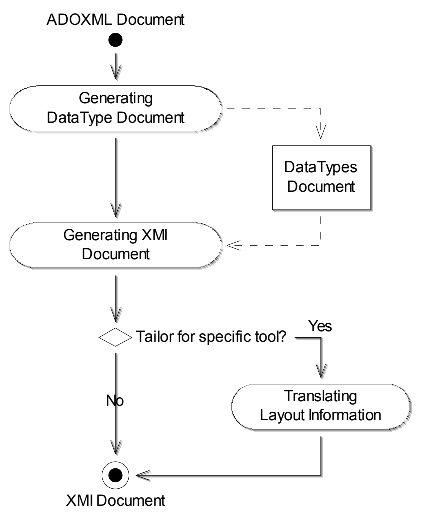
ADOXML Document
Generating DataType Document
DataTypes Document
Generating XMI Document
Tailor for specific tool?
Yes
No
Translating Layout Information
XMI Document

Fig. 8. An UML Activity Diagram showing the behaviour of the *Export* process

5 Related Work

In the following we provide a brief overview of three major categories of related work: data and systems integration, XML-based languages for Business Process Management, and model transformation approaches.

In the database management domain, issues such as schema integration [13] and data migration in federated databases [24] focus on comparable problems such as metamodel integration and model interoperability. Solutions from the systems integration domain also provide valuable input for data and meta-data integration [20, 22, 25]. Nevertheless, the richness of modelling language semantics needs additional aspects to be solved in model interoperability. One of the problems which is not covered by the aforementioned approaches is the problem of heterogeneous process flow semantics in the exchange of models in different process modelling languages.

Since the advent of XML various XML-based process description languages were published [23] such as XPDL [11], BPML [2], and BPEL [1]. Additionally, XMI [7] is a candidate to be accepted as a general model and meta-model exchange format. These languages will provide valuable support for model interoperability.

But even if XMI will serve as a general model exchange facility, the semantic interoperability of models and meta-models in heterogeneous meta-environments still needs further mechanisms such as semantic model transformations to connect different modelling domains appropriately [21]. In this area, we see a strong contribution from transformation approaches in the domain of model-driven development and model engineering. In [14], a good overview of various model-to-model and model-to-code transformation approaches can be found.

6 Conclusions

The large heterogeneities presently characterizing Business Process Management languages makes model interoperability to play a key role in the context of meta-environments management. The presented paper gives a contribution in this setting by proposing a framework for handling interoperability among different typologies of enterprise models. Such a feature is gained by exploiting the MOF and the XMI standards.

We have developed a prototype applying the underlying ideas of the proposed framework. This guarantees the interoperability between the meta business modelling tool ADONIS and any XMI-compliant CASE tool available in the market. Obtained results are particularly encouraging; a proof of this is that an extension of the prototype realized, will be made available for ADONIS customers.

Nevertheless, we still see a number of open issues, which will guide our further research. One of these issues is the semantic interoperability of models and meta-models in different meta-environments. Even if models can be exchanged e.g. using XMI, the semantic meaning of the models and meta-models in each meta-environment may be different. Ontology may serve as an appropriate tool in this context.

Other issues for further evaluation are non-functional aspects such as performance, ease of use and security in model interoperability. E.g. currently one important performance obstacle in the practical application of the presented approach is the extensive main memory usage of the XSLT processor during transformation of large model bases.

References

1. BPEL4WS (Business Process Execution Language for Web Services) Version 1.1 May, 5 2003. http://www-106.ibm.com/developerworks/library/ws-bpel/.
2. BPMI.org: Business Processing Modelling Language - Specification 1.0. http://www.bpmi.org/bpml-spec.esp.
3. Object Management Group: Object Management Architecture Guide. http://doc.omg.org/ab/97-05-05.
4. Object Management Group: MDA Guide, Version 1.0.1, June 12 2003.
5. Object Management Group: Meta Object Facility (MOF) Specification, Version 1.4, April 2002.
6. Object Management Group: OMG Unified Modeling Language Specification, Version 1.4, September 2001.
7. Object Management Group: OMG XML Metadata Interchange (XMI) Specification, Version 1.2, January 2002.
8. Object Management Group: Common Object Request Broker Architecture. http://www.omg.org/technology/documents/formal/corba_iiop.htm.
9. OMG, Object Management Group. http://www.omg.org.
10. W3C: XSL Transformations (XSLT) Version 1.0, November 1999.
11. Workflow Management Coalition: Workflow Process Definition Interface - XML Process Definition Language. Document Number WFMC-TC-1025, Document Status-Version 1.0

Final Draft October 2002. http://www.wfmc.org/standard/docs/TC-1025_10_xpdl_102502.pdf.
12. Bézivin, J.: From Object Composition to Model Transformation with the MDA. In Proceedings of TOOLS'USA, volume IEEE TOOLS-39, Santa Barbara, California, USA, 2001.
13. Bernstein, P. A., Levy, A. Y., Pottinger, R. A.: A Vision for Management of Complex Models. Microsoft Research Technical Report MSR-TR-2000-53, Juni 2000. ftp://ftp.research.microsoft.com/pub/tr/tr-2000-53.pdf.
14. Czarnecki, K., Helsen, S.: Classification of Model Transformation Approaches. OOPSLA'03, Workshop on Generative Techniques in the Context of Model-Driven Architecture, 2003.
15. Grose, T.J., Doney, G.C., Brodsky, S.A.: Mastering XMI: Java Programming with XMI, XML, and UML. John Wiley Sons, 2002.
16. Jeckle, M.: OMG's XML Metadata Interchange Format XMI. In: [23], pp. 25-42.
17. Junginger, S., Kühn, H., Strobl, R., Karagiannis, D.: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen. WIRTSCHAFTSINFORMATIK, Vol. 42, No. 5, 2000, pp. 392-401.
18. Karagiannis, D., Kühn, H.: Metamodelling Platforms. Invited Paper. In: Bauknecht, K., Min Tjoa, A., Quirchmayer, G. (Eds.): Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-Web2002) in conjunction with DEXA2002, Aix-en-Provence, France, 2002, LNCS 2455, p. 182.
19. Keller, G., Nüttgens, M., Scheer, A.-W.: Semantische Prozessmodellierung auf der Basis "Ereignisgesteuerter Prozessketten (EPK)". Publications of Institute of Wirtschaftsinformatik, No. 89, University of Saarbrücken, 1992.
20. Kohoutková, J.: Meta-Level Transformations in Systems Integration. In: Manolopoulos, Y., Návrat, P. (Eds.): Proceedings of the Sixth East European Conference of Advances in Databases and Information Systems (ADBIS'02), Vol. 2 Research Communications, Bratislava, Slovakia, September 2002, pp. 121-130.
21. Kühn, H., Murzek, M., Bayer, F.: Horizontal Business Process Model Interoperability using Model Transformation. In: Proceedings of the Workshop on Interoperability of Enterprise Systems (INTEREST2004) held in conjunction with ECOOP 2004 conference, Oslo, Norway, June 2004.
22. Linthicum, D. S.: Enterprise Application Integration. Addison-Wesley, 2000.
23. Nüttgens, M., Mendling, J. (Eds.): Proceedings of the First Workshop on XML Interchange Formats for Business Process Management (XML4BPM2004). German Informatics Society, Marburg, Germany, March 2004.
24. Sheth, A.P., Larson, J.: Federated Database Systems for Managing Heterogeneous, Distributed and Autonomous Databases. ACM Computing Surveys, Vol. 22, No. 3, 1992.
25. Skoupý, K., Kohoutková, J., Benešovský, M., Jeffery, K.G.: HYPERMEDATA Approach: A Way to Systems Integration. In: Proceedings of Short Papers of the 3rd East European Conference on Advances in Databases and Information Systems (ADBIS'99), Maribor, Slovenia, September 1999, pp. 9-15.
26. Smith, H.: BPM and MDA: Competitors, Alternatives or Complementary. Business Process Trends, 2004.