FAMIX and XMI [*]

Sander Tichelaar
Software Composition Group
University of Berne
Neubrückstrasse 12
CH-3012 Berne
Switzerland
tichel@iam.unibe.ch

Stéphane Ducasse
Software Composition Group
University of Berne
Neubrückstrasse 12
CH-3012 Berne
Switzerland
ducasse@iam.unibe.ch

Serge Demeyer
Department of Mathematics and
Computer Science
University of Antwerp
Universiteitsplein 1
B-2610 Wilrijk, Belgium
Serge.Demeyer@uia.ua.ac.be

Abstract

Recently exchange formats have gained lots of attention. Multiple tools need to interact and/or work on the same software system. Especially there is a need to reuse parser technology. Within the FAMOOS project we have developed a model for representing object-oriented software systems at the program entity level. The model has been designed for language independence, extensibility and information exchange. For the actual exchange of data we are currently moving to use XMI, a standard for model-based information exchange.

1 Introduction

In this position paper we discuss FAMIX, a language-independent, extensible model for modelling object-oriented software systems [DDT99, DTS99]. FAMIX has been developed and used in the FAMOOS project [DD99] to model object-oriented software systems. It supports multiple object-oriented – and since recently also procedural – languages. Software systems are modelled at the so-called *program entity level*, i.e. entities and relationships such as classes, methods, invocations and accesses are being modelled, but not complete abstract syntax trees. Additionally to the modelling of the software itself, we are currently extending the model to support *groupings* which allows for reasoning about systems on a more abstract level.

For the actual information exchange we have been using the industry-standard CDIF interchange format [Com94]. However, since CDIF is not developed anymore we had to look for something else. XMI (XML Metadata Interchange) [XMI98] is a newer standard for model driven information exchange using XML. It is based on the Meta Object Facility (MOF), a standard meta-meta-model from the OMG [Gro99]. With the appropriate tool support, the MOF and XMI provide for a generic means to generate query APIs, XML DTDs and XML documents containing actual data for models that are defined as an instance of the MOF.

The following two sections discuss FAMIX and XMI in more detail. Afterward we say a few words about our tools that support FAMIX and partly XMI and we end with a conclusion.

[*]This work has been funded by the Swiss Government under Project no. NFS-2000-46947.96 and BBW-96.0015 as well as by the European Union under the ESPRIT programme Project no. 21975 (FAMOOS).

2 FAMIX

The FAMIX model provides for a language-independent representation of object-oriented source code. It is an entity-relationship model that models object-oriented source code at the *program entity level*. Figure 1 shows the core entities and relations. The complete

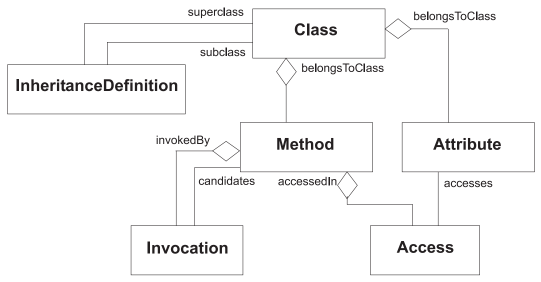

Figure 1: Core of the FAMIX model

model, which is set up as an object-oriented hierarchy, consists of more entities such as functions and formal parameters. Limited space does not allow us to discuss the complete model, but a complete specification can be found in [DTS99]. Additionally to the entities themselves, FAMIX defines for every entity a set of attributes. A Method, for instance, has attributes such as `signature` and `isAbstract`. The semantics are well defined: for every attribute it is described what its value can be and how it should be interpreted.

FAMIX has been designed with the following issues in mind:

- *Language independence.* Legacy systems exist in many different languages. We need our tools, and therefore our model, to be as independent as possible of the differences between the languages. This allows us to reason about the languages on a more abstract level and to use our tools without adaptation for the different supported languages. Initially we needed to be able to deal with C++, Java, Smalltalk and Ada. However, we have used our model and tools for IDL and Cobol as well.

- *Extensibility.* FAMIX allows for adding new abstractions and specific attributes to existing abstractions. Typically extensibility is needed for the following kinds of information:

  – Language-specific information. Certain problems need language-specific information to be solved. Additionally, typically languages have their own specific problems that are interesting in itself. An example is the analysis of include hierarchies in C++.

  – Tool-specific information. Apart from the information that is directly related to the source code, tools might want to store and exchange tool-specific information such as analysis results or layout information for graphs.

  – Whatever information people find worth modelling. The model should not hinder them in doing what they want to do.

- *Information exchange.* FAMIX has been designed with information exchange in mind. The information is stored in a flat, streaming-friendly way and it uses a unique naming scheme that is valid over multiple transfers. This makes the transferred elements self-contained rather than that they are dependent of transfer-specific information that links them to other entities. This is particularly important to be able to deal with incremental loading of information and the transfer of related information in separate transfers.

3 XMI

XMI (XML Metadata Interchange) [XMI98] is an OMG standard which combines a meta-meta-model (the OMG MOF) and a textual format (XML). It defines a set of rules to produce XML content from meta-models (such as FAMIX or UML) that are based on the MOF meta-meta-model. The MOF is quite an extensive model. It supports operations for querying purposes and a reflective interface. However, the core model is pretty simple. It is straightforward to define any entity-relationship based model. Basically you have to define what element in your model is an instance of a MOF Class (entity) and what is an instance of a MOF Association (relationship).

To deal with the complexity of the full MOF, DTD generation and XML generation tool support is necessary. However, the XML that contains the actual data (see figure 2) is straightforward to parse.

```
<Famix.Method xmi.id="_3">
 <Famix.Entity.name>print</Famix.Entity.name>
 <Famix.Entity.uniqueName>gui::Widget.print()</Famix.Entity.uniqueName>
 <Famix.BehaviouralEntity.signature>print()</Famix.BehaviouralEntity.si
 <Famix.Method.belongsToClass>gui::Widget</Famix.Method.belongsToClass>
 <Famix.Method.isAbstract xmi.value="false"/>
</Famix.Method>
```

Figure 2: A Method entity in XML

Our experiences defining FAMIX as an instance of the MOF and how the XML is generated are reported in [Fre00].

4 Tool support

Our reengineering tool environment called Moose [DLT00] is based on the FAMIX model. Moose acts as a repository for our reengineering tools. We use CDIF to import FAMIX-based information about systems written in Java, C++ and other languages. The information is produced by external parsers such as SNiFF+ [TD99]. Next to parsers we also have integrations with external environments such as the Nokia Reengineering Environment [DD99]. However, we will abandon CDIF and we are actively developing an XMI importer and exporter for our tool set.

For our production of XMI data we have a prototype called XMIforFAMIX [Fre00]. It is a Java tool that extracts FAMIX-based information from Java systems using the Java Reflective interface and puts the data in XMI-based XML files according to the FAMIX model. Its architecture is such that other parsers, models and format producers can be easily plugged in. The tool uses the XMI toolkit from IBM alphaWorks to produce the DTDs and XML.

5 Conclusion

In this paper we have described FAMIX, a program entity level model for describing object-oriented source code. FAMIX models similar information as models such as TA++ [Let98]. It does not clear advantage or disadvantage. However, the well-defined semantics and the careful consideration of issues such as extensibility and information exchange might be a valuable input for any similar model. The main limitation of FAMIX is that it models software systems up to the program entity level. Detailed abstract syntax tree information is not covered. Grouping support is limited, but we are currently working on resolving that.

XMI (and inherently the MOF) provide for a standard means to transfer information in XML. We consider XMI easy enough to use for our purposes. Although tool support is still limited, we find that the advantages of using a standard way for generating DTDs and XML files, outweights the disadvantage of the higher complexity.

References

[Com94] CDIF Technical Committee. Cdif framework for modeling and extensibility. Technical Report EIA/IS-107, Electronic Industries Association, January 1994. See http://www.cdif.org/.

[DD99] Stéphane Ducasse and Serge Demeyer, editors. *The FAMOOS Object-Oriented Reengineering Handbook*. University of Berne, October 1999. See http://www.iam.unibe.ch/˜famoos/handbook.

[DDT99] Serge Demeyer, Stéphane Ducasse, and Sander Tichelaar. Why unified is not universal. UML shortcomings for coping with round-trip engineering. In Bernhard Rumpe, editor, *Proceedings UML'99 (The Second International Conference on The Unified Modeling Language)*, LNCS 1723, Kaiserslautern, Germany, October 1999. Springer-Verlag.

[DLT00] Stéphane Ducasse, Michele Lanza, and Sander Tichelaar. Moose: an extensible language-independent environment for reengineering object-oriented systems. Proceedings of the Second International Symposium on Constructing Software Engineering Tools (CoSET 2000), June 2000.

[DTS99] Serge Demeyer, Sander Tichelaar, and Patrick Steyaert. FAMIX 2.0 - the FAMOOS information exchange model. Technical report, University of Berne, August 1999.

[Fre00] Michael Freidig. XMI for FAMIX. Technical report, University of Berne, June 2000.

[Gro99] Object Management Group. Meta Object Facility (MOF) specification. Technical Report MOF V1.3 RTF, Object Management Group, September 1999.

[Let98] Timothy C. Lethbridge. Requirements and proposal for a Software Information Exchange Format (SIEF) standard. Technical report, University of Ottawa, November 1998. http://www.site.uottawa.ca/˜tcl/papers/sief/standardProposalv1.html.

[TD99] Sander Tichelaar and Serge Demeyer. SNiFF+ talks to Rational Rose – interoperability using a common exchange model. In *SNiFF+ User's Conference*, January 1999. Also appeared in the "Proceedings of the ESEC/FSE'99 Workshop on Object-Oriented Re-engineering (WOOR'99)" – Technical Report of the Technical University of Vienna (TUV-1841-99-13).

[XMI98] XML Metadata Interchange (XMI). Technical Report OMG Document ad/98-10-05, Object Management Group, February 1998.