

Adapting FUJABA for Building a Meta Modelling Framework

Carsten Amelunxen, Alexander Königs, Tobias Rötschke, Andy Schürr
 Technische Universität Darmstadt
 Institut für Datentechnik, FG Echtzeitsysteme
 Merckstraße 25
 64283 Darmstadt, Germany
 {amelunx|koenigs|rotschke|schuerr}@es.tu-darmstadt.de

ABSTRACT

The Real-Time Systems Lab performs research in the area forward engineering of automotive system software and reengineering of large industrial embedded systems in general. We need adequate CASE-tools to evaluate our approaches. These tools should be built on a shared meta modelling framework according to current standards (e.g. MOF 2.0, JMI, OCL, GXL). The FUJABA Tool Suite provides a substantial part of the required functionality (e.g. model editor, graph rewriting engine, Java code generator), but has to be adjusted to meet the standards. Currently, the necessary extensions can only be implemented by changing the FUJABA core, which is unsatisfactory. We would prefer to extract some FUJABA core functionality and distribute it over additional plug-ins. By doing so we could benefit from FUJABA for our meta modelling framework without compromising its present functionality.

1. INTRODUCTION

The mission of the Real-Time System Lab is to support meta model driven application development (MMDA) with a special emphasis on forward engineering of automotive system software and reengineering of large industrial embedded systems in general. The MMDA concept extends OMG's idea of model-driven application development (MDA) to the next higher level, where domain-specific tools and tool adaptations have to be developed as well. On this level meta tools are needed which support the specification of domain-specific modelling languages (including the adaption of general-purpose languages) and the generation of appropriate tools like editors, analyzers, and code generators.

As a consequence of this rather general perspective seven different research projects are currently under way, listed and categorized in table 1. Three projects belong directly to the automotive system software development area, whereas the

remaining projects either address reengineering and safety issues of embedded RT systems in general or belong to different application areas. The entries of the table indicate on which levels of OMG's model hierarchy (M1-M3) system development activities are addressed:

- M1** model driven application development
- M2** development, adaption, extension, and integration of (domain-specific) modelling languages and tools
- M3** development of meta modelling languages and tools for the model driven development of modelling tools

Project	Automotive			Civil Eng.	E-Learning	Med. Imaging	Security Eng.
	1	2	3				
Construction	T	M		T		t	T
Transformation			T		a	M	
Analysis	T	M	T	T	a	T	
Aspect-Weaving							M
Integration		T	M	T	T	T	
Visualization				a	M	T	
Code Generation	M			a	a		M

M – Meta tool, tool, and application development (M1-M3)
 T – Tool and application development (M1,M2)
 t – Tool development (M2)
 a – Application Development (M1)

Table 1: Topic Overview

All projects make contributions to all three modelling levels listed above, complementing each other. Due to the lack of space it is not possible to describe the specific goals of and relationships between all ongoing projects. As a consequence the following text describes only two projects in more detail which make very extensive usage of graph transformation techniques.

Evolution of complex embedded systems. Complex, software-intensive industrial systems like medical imaging systems and mobile switches evolve continuously and provide an increasing number of features. Usually, architectural models are proprietary with individual representations in design models and source code and are adjusted in isolated reengineering steps. Keeping architecture, detailed design and implementation consistent is difficult, as the architectural

consequences of small changes are not obvious to developers, and software architects can not manually check every detail. So continuous architectural analysis should be automated [18].

Data integration of CASE tools. Our industrial partner in the automotive sector uses several independent tools in different phases of the development process (e.g. DOORS for requirements engineering, Matlab for system architecture, CTE for testing), resulting in a variety of documents for the same project. Though these documents are related with each other, the tools cannot keep them consistent with each other. Trying to achieve consistency manually is time consuming and error-prone. So data integration of these tools should be automated [7].

Obviously, both projects need solution for meta modelling, which is also true for the other five projects. The general requirements of our projects are:

- generating repository interfaces, marshalling and unmarshalling tools from meta class diagrams
- generating static semantics checks and model analyzers from predicate logic expressions
- generating model transformation tools from graph rewriting rules
- generating tool integrators from triple graph grammars, a graph transformation based declarative approach to define document-boundary crossing consistency relationships [19]
- aspect weaving
- code generation

So it makes sense to define a shared meta modelling framework where all pieces fit together. In section 2 we describe how such a meta modelling framework should look like. Next, we explain in section 3 how FUJABA can help us to realize the framework and which modifications are necessary. In section 4 we explain how ECLIPSE should provide our framework with a common user interface and extensibility, and why FUJABA should adopt ECLIPSE as well. Finally, we discuss alternatives in section 5.

2. WHY WE NEED A META MODELLING FRAMEWORK

To address the specific issues mentioned in section 1 we need tools that provide different, but overlapping sets of features. As summarized in table 2, an intermediate investigation of existing tools revealed, that many independent tools would be needed to provide the most important features. However, these tools do not interact with each other very well, and most commercial tools cannot be extended to match our requirements.

So we decided to select a small set of existing tools, i.e. FUJABA [2], ECLIPSE [17] and Dresden-OCL compiler [10] as a starting point for a meta modelling framework so that the tools can exchange data with each other. To increase the

	FUJABA	SFP	Together	Artisan	ECLIPSE	Rational Rose	Rational Rose/RT	ArgoUML/Poseidon	Dresden OCL-Compiler
Architecture Description Language	-	o	o	o	-	o	(+)	o	-
Integration Framework	-	o	-	(o)	(+)	(o)	-	-	-
Extensible Code Generator	o	+	o	o	-	-	-	-	-
Model Driven Architecture	+	(+)	-	(+)	-	-	+	-	-
Meta Modelling	o	o	(o)	-	(o)	-	-	(o)	-
OCL-Compiler	-	-	-	-	-	-	-	+	+
Rule Interpreter	+	-	-	-	-	-	-	-	-
Available Source Code	+	-	-	-	+	-	-	+/-	+
License costs	+	-	-	(-)	+	(-)	(-)	+/-	+

Table 2: Tools and features for meta-modelling

potential acceptance and interoperability of our framework, it should adhere to the most recent standards.

The Object Management Group (OMG) is currently the authority in the field of standardization of (meta-)modelling languages. It is about to accept the current proposal for MOF 2.0 [14] as standard meta modelling language. The meta model of the next version of the popular Unified Modelling Language, UML 2.0, which is the OMG standard modelling language, will be defined using MOF 2.0. So it is very likely, that MOF 2.0 will be *the* meta modelling language and hence widely accepted in the near future.

The OMG standard format for tool interoperability is the XML Metadata Interchange 2.0 (XMI) format [16]. Many modelling and CASE-tools support XMI already. XMI used to come in two variants, UML-XMI and MOF-XMI according to the 1.x specifications of these languages. The variants will be replaced by MOF-XMI according to MOF 2.0, and hopefully be supported by more and more tools.

Rational Rose, the most important UML modelling tool, does already support MOF-XMI according to the old 1.x specification and will most likely support the 2.0 specification soon. Many other tools however, support only UML-XMI and our framework should be able to interact with these “legacy” tools. The University of the Federal Armed Forces Munich has already developed an XSLT script which can translate UML-XMI to MOF-XMI.

Many CASE tools including FUJABA are currently written in Java. To deal with meta models in Java, Sun as owner of Java has released the Java Metadata Interface (JMI) Standard [3]. As many tools will adopt XMI to exchange metadata, they will use the JMI standard to represent it internally. So our framework should be written in Java according to the JMI standard as well. The University of the Federal Armed Forces is working on a tool to convert MOF-XMI metadata to JMI compatible Java source code.

Static constraints that cannot be expressed graphically can be written using the Object Constraint Language (OCL),

the OMG standard for constraints [15]. However, the semantics of OCL is not fully defined though effort has been spent to provide it with more precise meaning, e.g. [8].

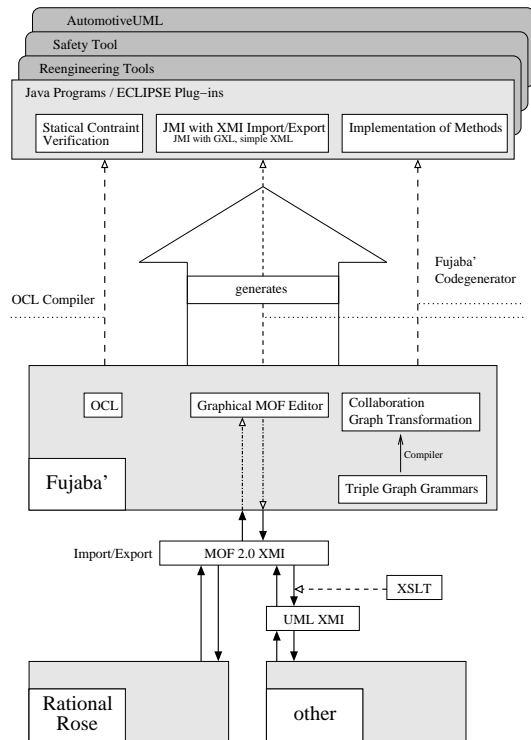


Figure 1: A vision of the meta modelling framework using FUJABA'

Figure 1 provides an overview of our framework denoted as FUJABA'. The lower part deals with the interoperability with other CASE tools using XMI. The upper part shows, how FUJABA' generates JMI compliant Java code for our meta modelling tools from MOF compliant models, graph transformation rules and OCL constraints.

3. FUJABA IN THE CORE OF THE META MODELLING FRAMEWORK

FUJABA Tools Suite 4.0 allows to create and edit UML class diagrams, activity diagrams and statecharts. Methods of classes defined in a class diagram can be implemented by specifying graph transformations on a UML collaboration diagram or just typing Java source code. The control flow is specified using UML-like activity diagrams, the graph transformations are embedded inside its activities. Because UML is widely known, it is easy to teach people how to use these so-called story diagrams.

Using this technique makes sense, as existing (meta) models are often defined in UML class diagrams. Mathematically, these can be described as graphs. As we want to reason about those existing models, graph rewriting rules provide a proper means to describe rules and transformations. Usually, graphically represented rules are easier to understand and mapped onto the existing models than textual rules. When graphically rules become too complicated to understand or are not expressive enough, the OCL still provides

a textual means to fill the gap within the standard of UML.

Consistency rules between data models should be described by means of triple graph grammars. The FUJABA team is already working to integrate them into future releases.

There is still some work to be done to adjust FUJABA to our needs. The class diagram editor has to be extended by a MOF 2.0 compatible editor. We need to propose, how MOF can be extended to cover story diagrams as standardized notation for graph rewriting rules. The code generator has to be replaced by a JMI-compliant variant. The XMI-JMI converter from the University of the Federal Armed Forces should be integrated into FUJABA. The existing path expression compiler should be replaced by integrating an OCL-Compiler, e.g. the Dresden OCL compiler. Optionally, FUJABA should be provided with UML packages, so that diagrams can be organized in a more efficient way.

Some of the extensions, like changes of the diagram editor and the code generator would require modifications of the FUJABA core. However, it would be better to realise this functionality as plug-ins, so that the user can choose between the original, initially more stable and the new functionality according to the new standards.

4. ADOPTING ECLIPSE

To provide our FUJABA-generated tools with a shared user interface and let them operate with other tools, we want to use ECLIPSE as integration framework. So all tools will be realized as ECLIPSE plug-ins. Additionally, we plan to provide ECLIPSE with a MOF 2.0 compliant meta model.

If more and more FUJABA functionality is put into plug-ins, it makes sense that FUJABA adopts the ECLIPSE plug-in concepts as well. Eventually, FUJABA should become a MOF-compliant compiler and an editor, separately usable inside the ECLIPSE framework. Figure 2 shows, how the current FUJABA core should be transformed into a smaller FUJABA' core with more plug-ins that finally can be replaced by the ECLIPSE plug-in manager. This would allow more CASE tools to use its graph transformation engine, even if third-party editors are used. However, ECLIPSE does not provide data integration, but this is a core topic of our research.

5. RELATED WORK

Although not yet complete, we have considered some related work when working on our vision. Preliminary decisions for the choice of FUJABA and JMI over other alternatives are made based upon the following considerations.

5.1 IPSEN and PROGRES

IPSEN [13] is an existing integration framework, that could be a starting point for our framework. The graph rewriting system PROGRES [20], which is based on IPSEN, could be used as an alternative for FUJABA. PROGRES has a more powerful language than FUJABA and is more stable due to its higher degree of maturity. However, IPSEN and PROGRES are written in Modula-3 and do not run on Windows platforms, which are most common among our industrial partners. PROGRES uses a proprietary GUI and does

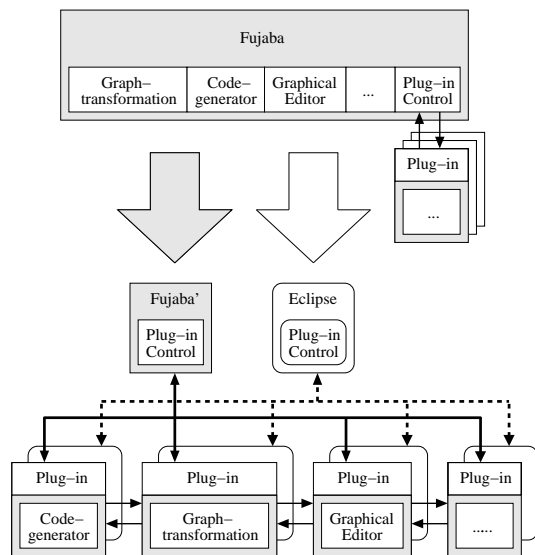


Figure 2: Evolving from FUJABA' to ECLIPSE

not comply to the newest standards, resulting in potentially less acceptance. Besides, IPSEN and PROGRES are very complex and do not have plug-in concept like FUJABA, so further extensions are more difficult to realize.

5.2 JAXB and XML Schema

The Java Metadata Interface (JMI) defines a mapping for MOF compliant models onto java technology. JMI uses XMI for the interchange of metamodel and metadata. A similar mapping could be achieved by using general data binding frameworks like JAXB [6], Zeus [4] or Castor [5] together with adequate XML schemata. The use of general data binding frameworks provides a less powerful mechanism because they are developed for common purpose applications as opposed to JMI which is dedicated to metadata. Due to the universal approach the entropy of such a mapping is higher than the entropy of a domain specific approach like JMI. JMI is a specialized interface and developed to fulfill the demands of a MOF mapping onto java technology. This is the advantage of JMI which makes JMI more suitable than a general data binding framework.

However, support for JAXB and XML Schema could be an optional feature of our framework. One possible application would be interoperability based on GXL [21] instead of XMI.

5.3 Meta CASE Tools

Available Meta CASE Tools like MetaEdit [12, 11], DOME [9], StP [1] would provide the possibility to generate CASE Tools using Meta Models. Unfortunately, they do not provide graph transformations and are usually closed source. So there is no way to extend those tools to meet our needs.

6. CONCLUSIONS

Adopting FUJABA for our meta modelling framework requires reasonable effort, but starting from scratch would be even worse. On the other hand, adjusting FUJABA towards the upcoming standards by using XMI-MOF 2.0 as exchange format, generating JMI-compliant Java code and integrating

FUJABA with ECLIPSE will increase the interoperability and acceptance of FUJABA. As far as we can see, our proposals follow the current trend of FUJABA development. So joining the existing FUJABA community would be beneficial both for us *and* the community.

Our next steps will be a more detailed analysis of alternative approaches and a feasibility study, to find out the best strategy to realize our ideas.

7. REFERENCES

- [1] Aonix. Software through Pictures (StP). <http://www.aonix.com/stp.html>.
- [2] S. Burmester, H. Giese, J. Niere, M. Tichy, J. Wadsack, R. Wagner, L. Wendehals, and A. Zündorf. Tool Integration at the Meta-Model Level within the FUJABA Tool Suite. In *Proc. Workshop on Tool-Integration in System Development (TIS 2003)*, pages 51–56, Helsinki, Finland, Sept. 2003.
- [3] R. Dirckze. *Java Metadata Interface (JMI) Specification, v1.0*. Unisys Corporation, Sun Microsystems, Inc., June 2002. <http://java.sun.com/products/jmi/>.
- [4] Enhydra.org. ZEUS 3.5: Open Source Java/XML Data Binding. <http://zeus.enhydra.org/>.
- [5] Exolab.org. Castor 0.9.5. <http://castor.exolab.org/>.
- [6] J. Fialli and S. Vajjhala, editors. *The Java Architecture for XML Binding (JAXB), v1.0*. Sun Microsystems, Inc., Jan. 2003. <http://java.sun.com/xml/jaxb/>.
- [7] R. Freude and A. Königs. Tool integration with consistency relations and their visualisation. In *Proc. Workshop on Tool-Integration in System Development (TIS 2003)*, pages 6–10, Helsinki, Finland, Sept. 2003.
- [8] R. Hennicker, H. Hussmann, and M. Bidoit. On the Precise Meaning of OCL Constraints. In T. Clark and J. Warner, editors, *Advances in Object Modelling with the OCL*, volume 2263 of *LNCS*, pages 70–85. Springer, 2001.
- [9] I. Honeywell. *DOMe Guide, v5.2.2*, 1999. <http://www.htc.honeywell.com/dome/>.
- [10] H. Hussmann, B. Demuth, and F. Finger. Modular Architecture for a Toolset Supporting OCL. In A. Evans, S. Kent, and B. Selic, editors, *UML 2000 - The Unified Modelling Language. Advancing the Standard*, volume 1939 of *LNCS*, pages 278–293, York, Oct. 2000. Springer. <http://dresden-ocl.sourceforge.net>.
- [11] S. Kelly. Improving the Integration of a Domain-Specific Modelling Tool. In *Proc. Workshop on Tool-Integration in System Development (TIS 2003)*, pages 57–60, Helsinki, Finland, Sept. 2003.
- [12] S. Kelly, K. Lyytinen, and M. Rossi. MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE Environment. In P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, editors, *CAiSE*, volume 1080 of *LNCS*, pages 1–21. Springer, May 1996.

- [13] M. Nagl, editor. *Building Tightly Integrated Software Development Environments: The IPSEN Approach*, volume 1170 of *Lecture Notes on Computer Science*. Springer, 1996.
- [14] Object Management Group, Inc. *Meta Object Facility (MOF) 2.0 Core Proposal*, Apr. 2003. <http://www.omg.org/docs/ad/03-04-07.pdf>.
- [15] Object Management Group, Inc. *Response to the UML 2.0 OCL RfP (ad/2000-09-03)*, Jan. 2003. <http://www.omg.org/docs/ad/03-01-07.pdf>.
- [16] Object Management Group, Inc. *XML Metadata Interchange (XMI) Specification, v2.0*, May 2003. <http://www.omg.org/docs/formal/03-05-02.pdf>.
- [17] Object Technology International, Inc. *Eclipse Platform Technical Overview, v2.1*, Feb. 2003. <http://www.eclipse.org/whitepapers/eclipse-overview.pdf>.
- [18] T. Röttschke and R. Krikhaar. Architecture Analysis Tools to Support Evolution of Large Industrial Systems. In *Proc. IEEE International Conference on Software Maintenance (ICSM)*, pages 182–193, Oct. 2002.
- [19] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG '94, Herrsching, Germany, June 16-18, 1994, Proceedings*, volume 903 of *Lecture Notes in Computer Science*, pages 151–163. Springer, 1995.
- [20] A. Schürr, A. J. Winter, and A. Zündorf. Developing Tools with the PROGRES Environment. In Nagl [13], pages 356–369.
- [21] A. Winter, B. Kullbach, and V. Riediger. An Overview of the GXL Graph Exchange Language. In S. Diehl, editor, *Software Visualization*, volume 2269 of *LNCS*, pages 324–336. Springer, Berlin Heidelberg, 2002.