Adopting Open Source development tools in a commercial production environment – are we locked-in?

Anna Persson[1], Henrik Gustavsson[1], Brian Lings[1], Björn Lundell[1], Anders Mattsson[2], Ulf Ärlig[2]

[1] University of Skövde, School of Humanities and Informatics, P.O. Box 408, SE-541 28 Skövde, Sweden
{anna.persson, henrik.gustavsson, brian.lings, bjorn.lundell}@his.se
http://www.his.se
[2] Combitech Systems AB, P.O. Box 1017, SE-551 11 Jönköping
{anders.mattsson, ulf.arlig}@combitechsystems.com

Abstract. Many companies are using model-based techniques to offer a competitive advantage in an increasingly globalised systems development industry. Central to model-based development is the concept of models as the basis from which systems are generated, tested and maintained. The availability of high-quality tools, and the ability to adopt and adapt them to the company practice, are important qualities. Model interchange between tools becomes a major issue. Without it, there is significantly reduced flexibility, and a danger of tool lock-in. We explore the use of a standardised interchange format (XMI) for increasing flexibility in a company environment. We report on a case study in which a systems development company has explored the possibility of complementing their current proprietary tools with open source products for supporting their model-based development activities. We found that problems still exist with interchange, and that the technology needs to mature before industrial-strength model interchange becomes a reality.

1 Introduction

The nature of the information systems development industry is changing under the pressures brought about by increased globalisation. There is competition to offer cheaper but higher quality products faster. To stay competitive, many companies are using model-based techniques to offer rapid prototyping, fast response to requirements change and improved systems quality. Central to model-based development is the concept of models as the major investment artefacts; these are then used as the basis for automatic system generation and test. Tools for the development, maintenance and transformation of models are therefore at the heart of the tool infrastructure for environments which support model-based development practice.

One potential danger for companies is tool lock-in. Tool lock-in exists if the models developed within a tool are accessible only through that tool. It has long been recognised that the investment inherent in design artefacts must be protected against tool lock-in, not least for maintenance of a long-lived application. Such lock-in effects are recognised as a risk, which can have severe consequences for an individual company [1]. The tool market is dynamic, and there is no guarantee that a tool, or tool version used to develop a product will remain usable for the lifetime of the product [2] [3]. In order to protect against such problems, models must be stored together with the version of the tool with which they were created. Even this is not guaranteed to succeed: hardware changes may mean that old versions of tools can no longer be run – unless hardware is also maintained with the tool. Such lock-ins are therefore undesirable for tool users. This may not be the case for some tool vendors, who may view lock-in as a tactic to ensure future business by keeping customers tied to their products [1].

The availability of high-quality modelling tools, and the ability to adopt and adapt them to a company context, are also important qualities. A variety of different development tools can be applied during a systems development project, including tools for the design of UML diagrams, tools for storing models for persistence, and tools for code generation [4]. The ability to seamlessly use and combine the various tools used within a project is highly desirable [4]. The reality for many designers is therefore an environment in which a mix of tools is used, and for flexibility many companies are considering a mix of proprietary and open source tools to cover their needs. The interchange of design artefacts between tools becomes critical in such environments. One special case of this is geographically distributed development where partners in different locations are working in different environments, using different tool sets.

Model interchange functionality can therefore significantly increase flexibility and reduce exposure to lock-in effects. There are two accepted ways in which model interchange can be undertaken: via software bridges, and via an open interchange standard. For example, the i-Logix Rhapsody® tool offers a VB software bridge to allow the import of models from the IBM Rose® tool, utilising its proprietary API. Such ad hoc provision is neither guaranteed nor universal, and can inhibit the development and organisational adoption of new tools. Neither does a bridge lessen the burden of having to save a tool with the models produced by it – a bridge requires the tool to be running in order to allow access to the model.

The more flexible (and scalable) approach is to support interchange through an open interchange standard. In an ITEA report, open source software is seen as one way of avoiding dependence on a single vendor [5]. Adherence to open standards has always been viewed as central to the open source movement, and key to achieving interoperability [6, p. 83]. An implied message to the open source community is that adoption of open source tools will depend heavily on their ability to interchange models with other tools using an open data standard.

2 Model Interchange Using an Open Standard

Over the years, many standardised interchange technologies have been proposed. Current interest centres on OMG's XML Metadata Interchange format (XMI) [7] [8] [9] [10]. In theory, any model within a tool can be exported in XMI format and imported into a different tool also supporting XMI.

In principle, XMI allows for the interchange of models between modelling tools in distributed and heterogeneous environments, and eases the problem of tool interoperability [7] [8] [9] [10] [11, p. 72] [12]. As most major UML modelling tools currently offer model interchange using XMI [13] [14], tool lock-in should not be a problem. This could offer the prospect of an invigorated tool market, with niche suppliers offering specialised functionality knowing that lock-in is not a factor in potential purchasing.

Although XMI can be used for the interchange of models in any modelling notation, according to OMG [7] one of the main purposes of XMI is to serve as an interchange format for UML models. The interchange of XMI-based UML models between tools is realized by the export and import of XMI documents. An XMI document consists of two parts: an XML document of tagged elements, and a Document Type Declaration (DTD) – or schema in XMI version 2.0 – specifying the legal tags and defining structure.

Exporting a model into an XMI document is done by traversing the model and building an XML tree according to a DTD. The XML tree is then written to a document. Other tools can recreate the model by parsing the resulting XMI document. An overview of how an XMI document for an UML model is generated is shown in figure 1.

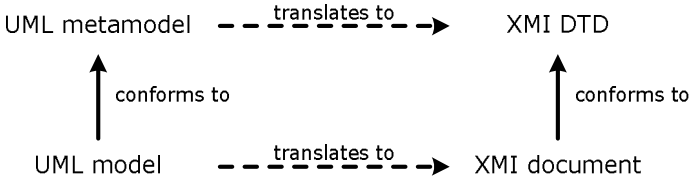

Fig. 1. Generation of XMI document for a UML model (from [14])

OMG state that "In principle, a tool needs only to be able to save and load the data it uses in XMI format in order to inter-operate with other XMI capable tools" [7]. From this description tool integration using XMI-based model interchange may seem to be simple. However, a number of reports have suggested that in practice having a tool with XMI support is no guarantee for a working interchange, something we wished to explore in a case study.

For example, Damm et al. [15] encountered some problems with XMI-based model interchange when integrating their UML modelling tool Knight with two proprietary UML modelling tools. One problem was incompatibility between tools that support different versions of XMI. Today, there are four versions of XMI recognised by OMG: versions 1.0, 1.1, 1.2 and 2.0 [7] [8] [9] [10], and different tool pro-

ducers have adopted different versions of XMI. What should be a straightforward export/import situation instead requires extra transformations, between versions of XMI. The authors state that "The IBM Toolkit and the Rose plug-in produce XMI files that are compatible, but neither of them is compatible with ArgoUML which uses an earlier version of the XMI specification" [15].

XMI-based model interchange may also be troublesome between tools supporting the same version of XMI, as discussed by Süß et al. [16] and Stevens [14]. According to Süß et al. [16]: "Most modelling tools support an XMI dialect that more or less complies with the XMI specification". According to Stevens [14]: "Some incompatibilities between XMI written by different tools still exist" since two tools using the same version of XMI and UML do not necessary generate the same XMI representation of a certain model [14].

In this paper we consider the use of XMI in UML modelling tools for model interchange. We report on a case study in which a systems development company has explored the possibility of addressing tool lock-in and complementing their current proprietary tools with open source tools for supporting their model-based development activities. The use of open source software is appealing to many organisations, given reports of "very significant cost savings" [17, p. 325]. The study concentrated on UML models, and specifically class diagrams – among the most widely used UML diagramming techniques, and with "the greatest range of modeling concepts" [18, p. 35].

In the case study, we consider class diagrams taken from commercial development projects in order to investigate whether XMI-based model interchange is a current option for the company. One aspect of the study was to explore whether it would be possible to use open source modelling tools to complement their current (proprietary) tool usage within the company context.

3 The Case Study

Combitech Systems AB (hereafter referred to as Combitech) is a medium sized, geographically distributed enterprise working with advanced systems design and software development, electronic engineering, process optimisation and staff training. It has approximately 230 employees and covers a broad spectrum of business areas such as defence, aviation, automotive, medical and telecoms.

The company has a long experience of systematic method work and model based systems development. In several development projects UML is used (e.g. [19]), but other modelling techniques are used as well. The company uses three of the major case tools supporting both UML and time-discrete modelling: Rose Realtime® (from IBM), Rhapsody® (from i-Logix) and TAU® (from Telelogic).

Combitech has an interest in exploring the potential of open source tools to complement its current tool suite, and is also sensitive to the potential problem of tool lock-in. With this in mind, a case study was set up to explore the potential of XMI-based export and import to offer a strategy for tool integration and tool-independent

storage formats. For the purposes of the case study, they chose to look at existing UML class diagrams developed using the Rhapsody tool.

Rhapsody is a proprietary development tool which supports all diagram types developed according to UML version 2.0 (for information, see [20]). Interchange of UML models is supported by export and import of XMI version 1.0 for UML version 1.1 and 1.3 [21]. Apart from UML modelling, requirements modelling, design-level debugging, forward engineering (generation of C, C++ and Ada source code) and automatic generation of test cases are also supported in the tool. The version of Rhapsody used currently by the company and in this study is 5.0.1.

Two production models developed by Combitech, hereafter referred to as "Model A" and "Model B", were used in the study. The two models, developed in different versions of Rhapsody (version 3.x and version 4.x respectively), consist of approximately 170 and 60 classes, respectively. The classes have different kinds of attributes and operations and make use of all common association types available in a UML class diagram.

Model A describes a device manager for an application platform used in an embedded system, and was developed using a "pair programming" activity. The model is one of many developed in a two-year project that in total involved about 50 system developers divided into 9 teams.

Model B is a high level architectural model of an airborne laser based bathymetry system for hydrographic surveys, and was itself developed by a single developer. The model is taken from a development project of about 4 man-years.

To explore the open source aspects, three open source UML modelling tools have been used in the study: ArgoUML v.0.16.1 (hereafter referred to as Argo; for information see [22]), Fujaba (a recent nightly build, as the most recent stable version does not support XMI; for information see [23]) and Umbrello UML Modeler v.1.3.0 (hereafter referred to as Umbrello; for information, see [24]). These tools were selected for the study since they support UML class diagrams and interchange of such diagrams using XMI. A systematic review of available open source modelling tools revealed no other tools with these properties. The tools, all supporting UML v.1.3, are presented in table 1.

Table 1. Open source UML modelling tools used in the study

	Argo http://argouml.tigris.org	Fujaba http://www.fujaba.de	Umbrello http://uml.sourceforge.net
XMI version	1.0	1.1	1.2
Storage format	Project-specific	Project-specific	XMI
UML models	All except object	Class, state, activity	All except object
Forward eng.	Java, C++, PHP	Java	Java, C++, PHP, ...
Reverse eng.	Java	Java	C++
Platform	All (Java based)	All (Java based)	Linux (with KDE)
Active developers	Approx. 25	Approx. 35	Approx. 5
License	BSD Open Source	GNU Lesser General Public	GNU General Public

It should be noted that only 5% of open source projects are developed by more than 5 developers [25, p. 68], so all of these are sizeable developments (information as published on each tool's mailing list in August 2004).

In order to explore interchange fully, a round-trip interchange scenario was devised. Each of the two models, developed at different times and by different developers within the company, were to be exported as an XMI document for import into an open source tool, and then exported by that tool for re-import into Rhapsody (see figure 2). If such a test succeeded, with no semantic loss, then we could conclude that interchange of the model was possible – and lock-in absent. Round-trip is necessary to counter the possibility that lock-in was simply extended to two tools.

In what follows our approach to model interchange is described. The procedure described applies for both models used, and also for a third (small) test model created as a control. Using this third model we were able to check the basic export/import functionality in each tool. The numbered steps relate to the numbering in figure 2.

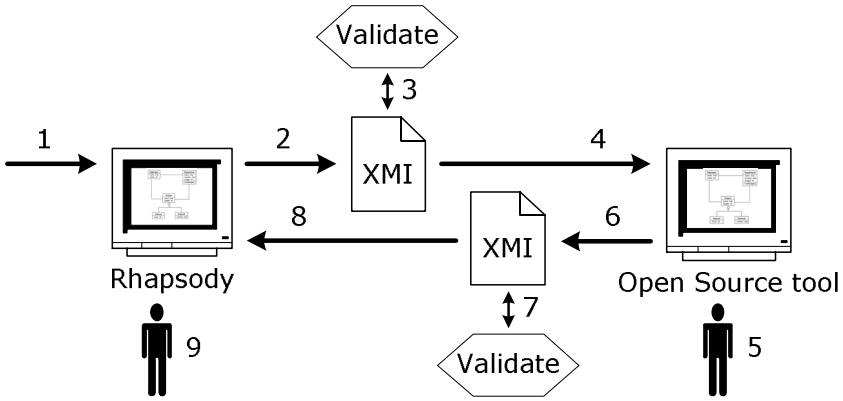

Fig. 2. Overview of model interchange

4 Results from the Case Study

4.1 Step 1: Bring up Models in Rhapsody

The models used were developed in two different and earlier versions of the Rhapsody tool used in the exploration. The first step was to bring up each model in the current version of Rhapsody (5.0.1) for visual inspection, ready for export.

4.2 Step 2: Export Models from Rhapsody

Each model was then exported from Rhapsody into an XMI 1.0 document for UML 1.3. The document representing Model A consisted of 174,445 lines of XMI code, that for Model B 36,828 lines.

4.3 Step 3: Validate XMI documents

At this stage we checked whether each document conformed to the XMI DTD (specified by OMG), by using two independent XML validation tools: XMLSPY (www.altova.com) and ExamXML (http://www.a7soft.com). Export from Model B was found to be valid, but not that from Model A. Both validation tools stated that the exported XMI document for model A had a structure which deviated from the standard specified by OMG. The problem related to non-conformance with an ordering dependency in the XMI DTD. This was repaired manually in order to allow tests to continue. Such repair is extremely difficult without specialised tool support: the file consists of 174,445 lines of XMI code, which in any case is very difficult for a human reader to comprehend.

4.4 Step 4: Import Models into Open Source Tools

An attempt was made to import each XMI document into each of the three open source tools, resulting in a model as represented in the tool's internal storage format, and available for inspection through its presentation layer.

Neither of the XMI documents exported from Rhapsody (and modified in the case of Model A) could be imported into either Fujaba or Umbrello. This was not unexpected, as Fujaba and Umbrello support only the import of later XMI versions to that used in Rhapsody, and it was evident from inspection of the documentation that backwards compatibility was not a feature of the XMI versions. This is because later versions have very different structure from XMI v.1.0. For both models, Fujaba simply hangs whilst in Umbrello nothing happens, and control is returned to the user with no feedback.

It is possible to translate between versions of XMI. At the time of writing no open source converters were available to allow further testing with these tools. However, the Poseidon tool from Gentleware (www.gentleware.com) – which is based on the code base from ArgoUML – claims to import versions 1.0, 1.1 and 1.2 of XMI and export version 1.2 (Gentleware, 2004). We therefore attempted to use Poseidon (Community Edition, version 2.6) to import Rhapsody's exported XMI 1.0 file with a view to exporting XMI v.1.2 for import into Umbrello. The XMI v.1.0 file exported from Rhapsody for model B could not initially be imported into Poseidon. However, after deleting an offending line, detected after inspection of Poseidon's log files, import was successful. Poseidon's exported XMI v.1.2 file was used for further tests with Umbrello.

Testing continued by attempting to import Rhapsody's exported XMI v.1.0 documents for Models A and B (modified in the case of A) into Argo, and import the XMI v.1.2 document exported from Poseidon into Umbrello. Success was expected with the first two tests, since the structure of the XMI documents representing the models were each confirmed as conforming to the XMI v.1.0 standard by both validation tools, and Argo and Rhapsody both claim to support this version of XMI. Successful transfer via Poseidon was considered less likely, as several transformations are involved.

Even after repair, import of model A into Argo failed. There are many comments attached to various elements in the UML model, and these were exported into XMI format. Although valid according to the XMI DTD, some of these caused problems for the Argo importer. It is unclear why only certain attachments caused problems. After significant experimentation, the XMI document was modified (with semantic loss) such that import into Argo became possible. The XMI v.1.0 document for Model B was successfully imported into Argo.

The XMI v.1.2 document exported from Poseidon was not valid, and so could not be imported into Umbrello. As a final test, a small test model developed in Poseidon was exported. Even this could not be successfully imported into Umbrello, and no further tests were made with that tool. Subsequent to the test we found that the problem lies with illegal characters generated in Poseidon's IDs, and has been noted in the vendor's forum as an issue to be resolved.

4.5 Step 5: Visual Inspection

A visual inspection was performed to compare each model as imported with its original in Rhapsody. A part of model A is shown in figure 3, firstly in Rhapsody and then in Argo. It should be noted that versions of UML earlier than 2.0 do not cater for the exchange of presentation information, so comparison will be of content only. Given the size of the models this is not a simple task, and some manipulation of the presentations was made to help in the visual checks.

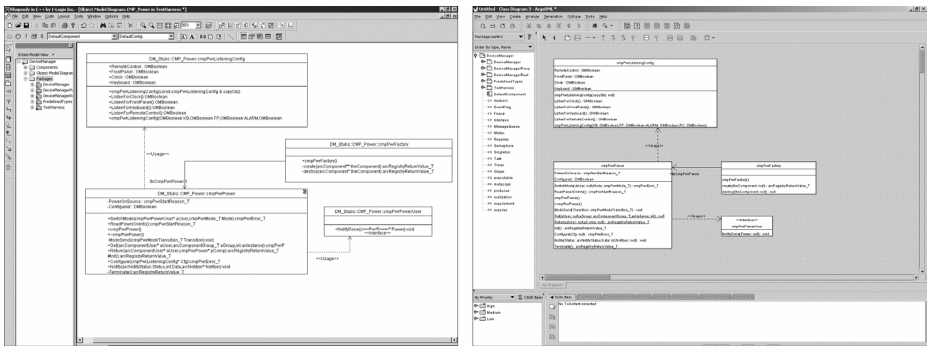

Fig. 3. Screen shots from Model A in Rhapsody(Left) and Argo(Right)

4.6 Step 6: Models exported from open source tool

Each model was exported from Argo into an XMI document in order to test its export facility. This generated a new XMI v.1.0 file for each of models A and B.

4.7 Step 7: Visual Inspection

At this stage we again checked whether the documents conformed to the XMI DTD. Neither exported XMI document was valid. This was due to a misspelling generated by Argo in the exported XMI. Once corrected (using a separate text editor) the documents became valid.

4.8 Step 8: Model Import to Rhapsody

Each model exported from Argo was imported into Rhapsody, to complete a round-trip interchange. In each case, import (of the repaired XMI) was successful.

4.9 Step 9: Visual Inspection

A visual inspection was performed to determine whether the content of each model was identical with the original version of it in Rhapsody. Once again, it is extremely difficult for models of this size to be checked for semantic loss, particularly as presentation information is not preserved with XMI versions available in the tools. However, in the visual inspection, using some manual repositioning in the Rhapsody tool to assist the process, no inconsistencies were found.

4.10 Step 10: Final Test

As a final test, each model (revised as necessary) was repeatedly put through the complete cycle. It was observed that the XMI file grew through the addition of an extra enclosing package on each export (by Rhapsody and by Argo). This makes no semantic difference to the model, but can be considered an inconvenience.

5 Summary and Implications

Like many companies, Combitech is currently committed to tools provided by more than one vendor. Although its current tool mix seems highly appropriate, Combitech's experience is that the tool market is dynamic: products come and go, and market leaders change over time. Most projects within the company involve many man years of effort and the company is very aware of the need to protect its own and

its customers' investments. It is also aware of the need to take full advantage of technology advances.

Further, in the company's experience, different developers prefer different aspects of tools and it is quite likely that a particular developer may prefer a specific tool for a particular task. In fact, the company view is that some current open source tools have clear potential for supporting aspects of its UML modelling activities, and envision a hybrid tool mix as the most likely scenario in the future. Combitech is also increasingly finding that customers are knowledgeable about UML, and envisages a future scenario in which parts of solutions are developed at customers' sites (perhaps using specialised tools). All of this heightens the company's interest in model interchange between tools, and XMI is currently the most commonly supported open data standard. It can be noted that OMG describes XMI as a general interchange standard and does not in this respect distinguish between different XMI versions, stating that "XMI allows metadata to be interchanged as streams or files with a standard format based on XML" [7]. This raises the question of whether XMI should actually be referred to as *a* standard interchange format. If tools supporting different XMI versions cannot interchange their XMI documents then the interchange format may seem weakly standardized, and it is the different versions of XMI by themselves that are standardized, not the overall XMI format. It is also worthy of note that this distinction is not made clear by all manufacturers of products, many making interchange claims for their products which are therefore not sustainable in practice. It is important that companies are well aware of the exact position with XMI, as it can feature highly in adoption decisions – as witnessed, for example, in OMG News [27], where one company focused on adherence to standards (including XMI) when adopting the Rhapsody tool.

Although OSS tools offer support for XMI-based model interchange equal to that in commercial tools, better could be expected. It is interesting to note that no open source tool yet offers conformity with the latest version of the standard, or offers the ability to import documents formatted in more than one version of XMI. It is also interesting that a major commercial tool only offers conformance with XMI v.1.0.
Commercial tools offer proprietary bridges to other tools, particularly market leaders, and may even make efforts to improve XMI interchange by catering for product-specific interpretations of XMI. However, the OSS community can be expected to offer high conformance with any open standard, and not to resort to tool-specific bridging software. Further, it could be argued that a goal for OSS tools should be to offer reliable import and export of documents conforming to any of the XMI versions, in this way offering both openness and an important role in the construction of interchange adapters – especially useful for legacy situations. As a special case, one hopes that OSS tools will lead the way in conformance with XMI 2.0 and UML 2.0. With the advent of UML 2.0 and XMI 2.0, there is a real possibility of standard interchange both horizontally and vertically within the tool chain.

Compatibility between XMI versions is not the only requirement for successful XMI-based model interchange between tools. Tools must guarantee the export of XMI documents which conform to any normative XMI document structure specified

by OMG. As apparent from this study this is not yet guaranteed. Export of invalid XMI documents is a serious issue that tool developers need to address.

The results of the study also show that complexity of models may cause interchange problems: less complex models seem easier for tools to handle. It is important that future studies should explore interchange issues using medium to large-scale models in order to subject tools to realistic modelling constructs from real usage contexts. Architectures for model-based systems development rely heavily on model interchange. To support such development in a globally distributed environment, robust and general export/import functionality must be provided. This will require effective and continued feedback from practice on the actual and attempted use of open data standards in systems development.

Acknowledgement

This research has been financially supported by the European Commission via FP6 Co-ordinated Action Project 004337 in priority IST-2002-2.3.2.3 'Calibre' (http://www.calibre.ie).

References

1. Statskontoret: Free and Open Source Software – a feasibility study. 2003:8a. Statskontoret, Stockholm. http://www.statskontoret.se/upload/Publikationer/2003/200308A.pdf (2003)
2. Lundell, B. and Lings, B.: Changing perceptions of CASE-technology. Journal of Systems and Software, 72(2) (2004) 271-280
3. Lundell, B. and Lings, B.: Method in Action and Method in Tool: a Stakeholder Perspective. Journal of Information Technology, 19(3), (2004) 215-223
4. Boger, M., Jeckle, M., Mueller, S. and Fransson, J.: Diagram Interchange for UML. In: Proceedings of UML 2002 – Unified Modeling Language: Model Engineering, Concepts, and Tools (Jezequel, J.-M., Hussmann, H. and Cook, S. Eds.). Springer-Verlag, Berlin (2003) 398-411
5. ITEA: ITEA Report on Open Source Software. January 2004, ITEA Office Association, Available via: www.itea-office.org (2004)
6. Fuggetta, A.: Open Source Software: An Evaluation. Journal of Systems and Software, 66(1) (2003) 77-90
7. OMG: OMG-XML Metadata Interchange (XMI) Specification, version 1.0, http://www.omg.org/docs/formal/00-06-01.pdf (2000)
8. OMG: OMG-XML Metadata Interchange (XMI) Specification, version 1.1, http://www.omg.org/docs/formal/00-11-02.pdf (2000)
9. OMG: XML Metadata Interchange (XMI) Specification, version 1.2, http://www.omg.org/cgi-bin/doc?formal/2002-01-01
10. OMG: XML Metadata Interchange (XMI) Specification, version 2.0, http://www.omg.org/docs/formal/03-05-02.pdf (2003)
11. Obrenovic, Z. and Starcevic, D.: Modeling multimodal human-computer interaction. IEEE Computer, 37(9) (2004) 65-72

12. Brodsky, S: XMI Opens Application Interchange. IBM, 30 March, http://www-4.ibm.com/software/ad/standards/xmiwhite0399.pdf (1999)
13. Jeckle, M.: OMG's XML Metadata Interchange Format XMI. In: Proceeding of XML Interchange Formats for Business Process Management (XML4BPM 2004): 1st Workshop of German Informatics Society e.V. (GI) in conjunction with the 7th GI Conference "Modellierung 2004", 25 March 2004 Marburg, Germany (2004)
14. Stevens, P.: Small-scale XMI programming: a revolution in UML tool use? Automated Software Engineering, 10(1) (2003) 7-21
15. Damm, C.E., Hansen, K.M., Thomsen, M. and Tyrsted, M.: Tool Integration: Experiences and Issues in Using XMI and Component Technology. In: Proceedings 33rd International Conference on Technology of Object-Oriented Languages and Systems TOOLS 33. IEEE Computer Society, Los Alamitos (2000) 94-107
16. Süß, J.G., Leicher, A., Weber, H. and Kutsche, R.-D.: Model-Centric Engineering with the Evolution and Validation Environment. In: Proceedings of UML 2003 – The Unified Modelling Language: Modelling Languages and Applications (Stevens, P., Whittle, J. and Booch, G. Eds.). Springer-Verlag, Berlin. (2003) 31-43
17. Fitzgerald, B and Kenny, T.: Open Source Software in the Trenches: Lessons from a Large-Scale OSS Implementation. In: 2003 – Twenty-Fourth International Conference on Information Systems (March, S.T., Massey, A. and DeGross, J.I. Eds.). Seattle, Washington (2003) 316-326
18. Fowler, M.: UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd edition. Addison-Wesley, Boston (2003)
19. Mattsson, A.: Modellbaserad utveckling ger stora fördelar, men kräver mycket mer än bara verktyg. On Time, April, Available via: http://www.ontime.nu (in Swedish) (2002)
20. Douglass, B.P.: Model Driven Architecture and Rhapsody, i-Logix Inc. http://www.omg.org/mda/mda_files/MDAandRhapsody.pdf (accessed 15 October 2004) (2003)
21. i-Logix (2004). XMI TOOLKIT VERSION 1.7.0 README FILE. 26 April 2004, i-Logix Inc
22. Robbins, J.E. and Redmiles, D.F.: Cognitive support, UML adherence, and XMI interchange in Argo/UML. Information and Software Technology, 42(2) (2000) 79-89
23. Nickel, U., Niere, J. and Zundorf, A.: The FUJABA environment. In: Proceedings of the 2000 International Conference on Software Engineering: ICSE 2000 the New Millennium. ACM Press, ACM, New York (2000) 742-745
24. Ortenburger, R.: Software modeling with UML and the KDE Umbrello tool: One step at a time. Linux Magazine, August (2003) 40-42
25. Zhao, L. and Elbaum, S.: Quality assurance under the open source development model. Journal of Systems and Software, 66(1) (2003) 65-75
26. Gentleware: Gentleware Product Description: Community Edition 2.6. http://www.gentleware.com/products/descriptions/ce.php4 (accessed 21 October 2004) (2004)
27. OMG News: OMG News: The Architecture for a connected world. April 2002, Available via: http://www.omg.org (2002)