A Comprehensive Survey of UML Compliance in Current Modelling Tools

Holger Eichelberger, Yilmaz Eldogan, Klaus Schmid

Software Systems Engineering
University of Hildesheim
Marienburger Platz 22
31134 Hildesheim
{eichelberger, eldogan, schmid}@sse.uni-hildesheim.de

Abstract: The Unified Modeling Language (UML) specification is widely adopted in software engineering. When tools do not fully implement the UML specification, the user might be locked-in to a modeling tool, e.g. when exported models are not compatible among tools or tools implement different subsets of the UML. These compatibility problems also have significant impact on the effectiveness of model-driven development approaches. Compliance, as defined by the UML standard, is intended to characterize tools and to highlight such problems. In this paper we describe an approach to asses the UML compliance levels of modeling tools. Using UML definition of compliance, we could only identify 4 out of 68 tools as being acceptable.

1 Introduction

The OMG specifications on the Unified Modeling Language (UML) are widely used in software engineering, e.g. for the design and the documentation of software systems [DP07]. Various tools support these activities by implementing the OMG specifications. In several situations, problems may occur when the tool being used does not sufficiently adopt the specification, e.g. when required model elements are not supported or when incompatible formats prevent migration or further processing. In order to ensure interoperability among tools, the OMG defined a compliance schema consisting of several compliance levels.

Recently, we conducted a detailed and encompassing study [EES08] analyzing the functionality of current UML modeling tools. This study was developed mainly in order to provide detailed information to decision makers in industry and research, when facing the situation of a tool selection. Here, we will use the subset of results of this analysis that directly relates to UML compliance as defined by the UML. Most tool comparisons published so far provide only generic high-level information, e.g. the price of a tool, the supported diagram types or the version of XMI implemented by a tool, thus we decided to focus in this paper on the aspect of compliance as defined in the UML specification.

Therefore, we describe the evaluation approach of our study and discuss the results, i.e. the current status of professional modeling tools in this regard.

In contrast to existing UML tool evaluations, we describe in this paper a feature-based evaluation approach driven by the OMG specifications, and in particular the definition of UML compliance as given by the UML. This detailed evaluation was executed on a sample set of 68 UML tools, covering – in particular – all major professional tools that are widespread in practice. The result of our approach provides detailed tool profiles including the individual UML compliance level per tool. Thus, another important aspect of our approach is the consistent application of an evaluation approach based on a specification to a large set of tools. In fact, the data obtained by the underlying study can be used to support a systematic decision process to guide the selection and adoption of UML modeling tools. As we focus only on UML compliance in this paper, we will not discuss aspects of tool selection and adoption in general here.

The remainder of this paper is structured as follows: In the next section, we review related work on tool comparisons and research approaches for tool evaluations. Then, in Section 3, we describe our evaluation methodology and the evaluation process. In Section 4, we present an overview on the aggregated results of the study. In the final section, we draw some conclusions and discuss future work.

2 Related Work

In this section, we discuss relevant related work. The OMG defined the compliance to the UML specification in [OMG07b, OMG07a]. To our knowledge, no other evaluation directly relates to UML compliance. Even in [BFSC08] the authors describe the generation of compliance test suites for UML taking the meta model and the wellformedness constraints into account but explicitly characterizing the results in terms of the UML compliance definition, e.g. the compliance levels.

We first cover existing tool lists and tool comparisons. Then we review the research approaches on rating, testing and evaluating UML tools from different perspectives. Finally, we discuss work on cross-tool XMI compatibility as an example for work on specific compliance issues.

Several lists enumerating tool and vendor data can be found in the literature and on the internet, e.g. the OMG tool vendor listing [OMG08] or the Wikipedia page on UML tools [www08]. For a decision maker in particular tool comparison matrices like [Jec04, OOS08] summarize for up to 100 different tools high-level features like the supported types of UML diagrams, formats for import and export, target languages for code generation etc. Also, a commercial study covering several aspects on tool support for software development [BFO+05] reviews twelve selected UML tools by comparing the implemented types of UML diagrams, prices and the supported versions of the XMI format. The major drawback of these comparisons and listings is that the provided results are rather generic, i.e. the reader cannot conclude from the presence of a certain UML diagram that required modeling facilities are actually implemented. As a result, an in-depth discussion of the compliance with the UML specification is also in the case of the commercial study out of scope and left to the reader. Furthermore, there is a significant difference in the quality of the information provided, i.e. whether data is collected from vendor statements or by directly analyzing a tool implementation as done in our study.

Research work in this area focuses often specifically on the evaluation approach itself. In [KF04] an evaluation framework for UML tools including 29 UML criteria, economic issues and usability issues is applied for the evaluation of 7 UML tools. A hierarchical, feature-oriented approach characterizing high-level tool functionality is given in [FDSP05]. In fact, the work in [FDSP05] is most closely related to ours, but the data aggregation is more complex, does not consider unclear feature implementations and the criteria list consists only of high-level tool functionality. Compared with the tool lists discussed above, some of the evaluations in research approaches consider more detailed but also incomplete feature lists [KF04, FDSP05]. Furthermore, all research driven evaluations rely on rather small sample sets. This is not only a drawback of these studies from a practical perspective, but also from a research perspective, as it leads to an insufficient validation of the evaluation frameworks themselves.

While all related work mentioned above is intended to provide an overview on several features of the considered tools, there is also some work on individual aspects, e.g. the ability to exchange models given in XMI format. For example, in [LLPM06] the authors describe a cross-tool compatibility test for Eclipse-based UML modeling tools. The same authors published also further evaluations on that topic for open source or commercial tools. As a result, only few tools are able to pass a backward compatibility test and also few tool combinations are able to share XMI models among each other. The final conclusion of these evaluations was that the user is locked-in to the selected modeling tool. A better model format compliance would also stabilize the technological foundation for automated test-suites e.g. as described in [BFSC08]. A drawback of the work on analyzing XMI compliance of only modeling tools is that further tools being used in practice like model transformation or code generation frameworks providing tool-specific XMI filters are not considered at all. Thus, a detailed analysis of the modeling facilities provides relevant information beyond a pure analysis of XMI compliance.

As discussed in this section, the work published so far has two major shortcomings: Most tool comparisons offer rather generic information, e.g. the provided types of UML diagrams, or rely on a small tool sample set. Regarding the level of detail, we characterize the abilities of individual tools by collecting information on modeling features defined by the UML specification. By aggregating that data, we obtain detailed tool profiles on the high-level facilities, e.g. on the fraction of features implemented for a certain diagram. This leads to a more realistic description of the tools. Regarding the sample set, we initially scheduled 200 UML modeling tools for evaluation instead of selecting an arbitrary subset as done in several studies. Due to the fact that many of these tools are not available anymore, we carried out an exhaustive feature-oriented evaluation on the subset of 68 available tools.

3 Methodology

In this section, we describe the methodology used in our evaluation. The basis of our evaluation is UML compliance as defined by the OMG, in particular the defined compliance levels. In Section 3.1, we introduce the UML compliance schema. In Section 3.2, we discuss the systematic construction of the evaluation criteria based on the UML compliance schema and we outline the evaluation procedure. In Section 3.3, we sketch the overall course of the evaluation.

3.1 Compliance to the UML Specification

The UML specification documents explicitly define the conformance to the specification in the UML Infrastructure [OMG07b], the UML superstructure[OMG07c], the XMI specification [OMG07a] and the diagram interchange specification [OMG06]. Our evaluation relies on UML specification version 2.1.1 which was the published version when preparing the evaluation. Our results are also valid for the current version 2.1.2 of UML, because the minimal changes do not interfere with our work.

The UML defines a two-dimensional compliance schema in the UML infrastructure and the UML superstructure. We outline both dimensions below.

The first dimension introduces the so called **compliance levels** in terms of parts of the UML language. *Level 0* requires the implementation of the simple class-based modelling language as specified in the UML infrastructure. *Level LM* adds profiles and more detailed class modelling to Level 0. *Level 1* extends the class diagrams from Level 0 to meet the UML superstructure specification and furthermore requires use case, sequence, timing, component, composite structure and activity diagrams. For compliance with *Level 2* also deployment diagrams and state machines must be implemented. Finally, *Level 3* represents the entire UML superstructure specification, also including information flows, templates and model management. Except for the metamodel level LM, which is defined in the UML infrastructure, the UML superstructure subsumes the UML infrastructure in this regard.

The second dimension details the **degree of realization**. For *abstract syntax compliance* both, UML metamodel and model persistence according to the XMI format must be implemented. The UML diagramming language as given in the UML superstructure must be realized for *concrete syntax compliance*. Realizing both, the metamodel and the diagramming language leads to *abstract and concrete syntax compliance*. When a tool also provides diagram interchange according to the UML specification, the related level is called *abstract and concrete syntax compliance with the ability of model and diagram persistence.*

In the remainder of this paper we refer to the compliance levels according to the following notation. L2-0, L2-M, L2-1, L2-2 and L2-3 denote the compliance levels of UML 2 as introduced above. To indicate the syntax compliance, we add appropriate abbreviations to the compliance levels, e.g. L2-0A as abstract syntax compliance on Level 0 or L2-1AC for abstract and concrete syntax compliance on Level 1. As some

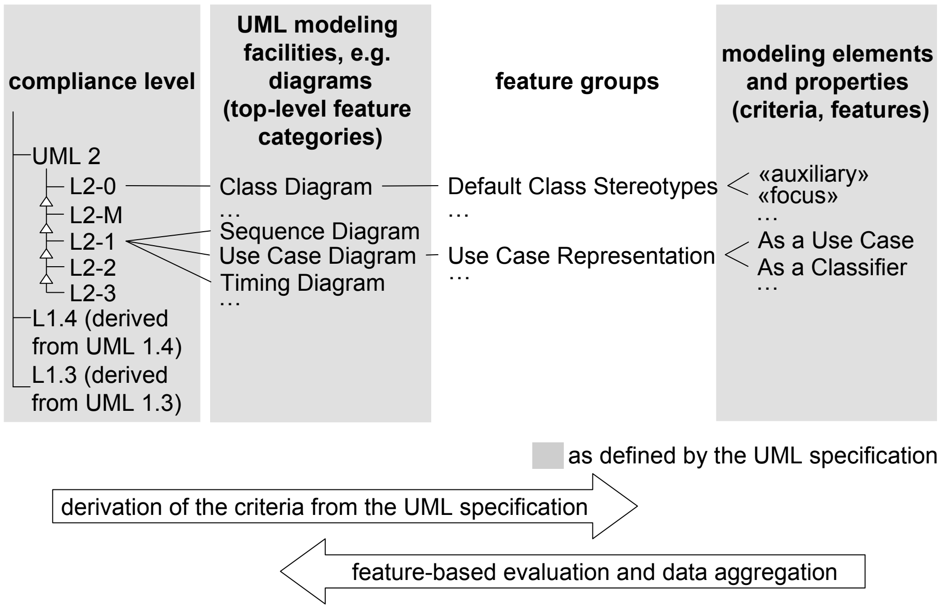
compliance level
UML modeling facilities, e.g. diagrams (top-level feature categories)
feature groups
modeling elements and properties (criteria, features)
UML 2
L2-0
L2-M
L2-1
L2-2
L2-3
L1.4 (derived from UML 1.4)
L1.3 (derived from UML 1.3)
Class Diagram
…
Sequence Diagram
Use Case Diagram
Timing Diagram
…
Default Class Stereotypes
…
Use Case Representation
…
«auxiliary»
«focus»
…
As a Use Case
As a Classifier
…
as defined by the UML specification
derivation of the criteria from the UML specification
feature-based evaluation and data aggregation

Figure 1: Derivation of features as evaluation criteria and feature-based evaluation.

tools still implement an earlier version of the UML, we use L1.3 and L1.4 to denote the compliance to all elements defined either in UML version 1.3 or 1.4, even though these versions of the UML do not define an explicit compliance mechanism. Thus, L1.3 and L1.4 denote the compliance to the entire specification of the respective version and no additional abbreviations are appended. When deriving the concrete compliance level for a tool, in some cases a tool may fulfil multiple levels simultaneously. In fact, L2-M subsumes L2-0 and L2-3 the levels L2-2, L2-1, L2-0 in the given sequence. For the analysis of our findings, we assign the maximum compliance level to a concrete tool. Therefore, we use the following ordinal scale L2-0, L2-M, L1.3, L1.4, L2-1, L2-2, L2-3 so that the subsumption of the levels as described above is still valid and tools being compliant to earlier versions of UML and only to L2-0 or L2-M are assigned to an appropriate UML 1.x level.

3.2 Evaluation Criteria and Procedure

We used an empirical approach to evaluate the tools and to determine the compliance levels, because for only few tools the implemented meta-model can directly be inspected, e.g. as source code. Figure 1 depicts the derivation of the evaluation criteria from the UML as a top-down process and the evaluation as a bottom-up process. Starting with the UML compliance levels as described in Section 3.1, we systematically dissected the UML specification of the referenced diagram types to obtain the individual modeling elements, relations and their properties. These were mapped to features. We also considered the related standard stereotypes summarized in the annex of the UML superstructure as features. Then we validated the feature list against the example diagrams and the particular diagram elements summaries given in the UML specification. Afterwards, we reviewed and consolidated the initial feature list.

We assigned 476 features obtained in this step to 131 feature groups to simplify the evaluation and the final data analysis. Thus, for example all features directly related to the use case modeling element were grouped together. These feature groups do not influence the results and are used in our approach only to simplify the evaluation and the presentation of the results. Finally, we summarized the feature groups as a feature hierarchy using the modeling facilities defined by the UML, e.g. the UML diagram types, as top-level categories. Figure 2 depicts a view on the feature hierarchy, in which all top-level categories, the feature groups for use cases and some individual features are shown. In particular the category "Use Cases", the feature groups for use case diagrams and the individual features for the representation of an individual use case are displayed in Figure 2. The category "model persistence" is needed for determining the compliance level and, thus, contains the supported XMI and DI versions as well as the results of the structural validation of files produced by a tool.

The features are aggregated in a bottom-up fashion in order to gain an overview, e.g. on the degree of compliance for the entire UML or for individual diagrams. This aggregation, as depicted in Figure 1, is done according to the following schema: A feature, i.e. a certain functionality realized by a tool, is counted by 1.

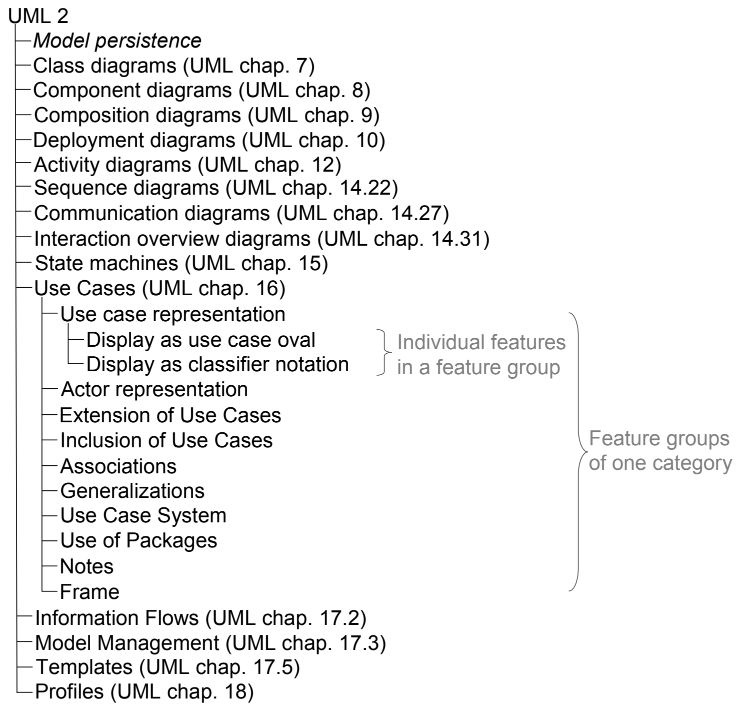
UML 2
Model persistence
Class diagrams (UML chap. 7)
Component diagrams (UML chap. 8)
Composition diagrams (UML chap. 9)
Deployment diagrams (UML chap. 10)
Activity diagrams (UML chap. 12)
Sequence diagrams (UML chap. 14.22)
Communication diagrams (UML chap. 14.27)
Interaction overview diagrams (UML chap. 14.31)
State machines (UML chap. 15)
Use Cases (UML chap. 16)
Use case representation
Display as use case oval
Display as classifier notation
Individual features in a feature group
Actor representation
Extension of Use Cases
Inclusion of Use Cases
Associations
Generalizations
Use Case System
Use of Packages
Notes
Frame
Feature groups of one category
Information Flows (UML chap. 17.2)
Model Management (UML chap. 17.3)
Templates (UML chap. 17.5)
Profiles (UML chap. 18)

Figure 2: View on the feature hierarchy created while analyzing the UML specification.

After constructing the feature hierarchy, the next action was the actual evaluation.,The following steps were performed to evaluate a tool:

1. **Acquisition:** The evaluator obtained the concrete tool from the vendor e.g. by download. As a part of this activity, an evaluation license was requested if needed. The evaluator recorded also the reason if a tool was not available, e.g. the tool home page does not exist anymore.

2. **Installation:** The evaluator installed the tool into a virtual machine according to the individual installation instructions.

3. **UML evaluation:** The evaluator created an UML model and tested each supported diagram type. In particular, we specified reference examples for class diagrams and use case diagrams to obtain comparable results. The examples aimed at combining most of the features of the diagram in a meaningful manner. For the other diagram types the evaluator just collects whether the features are present and functional by modelling a diagram.

4. **Data collection:** The evaluator collected the data in a spreadsheet prepared according to the feature hierarchy. Thus the evaluator collects data on the realization of the concrete syntax of UML following our evaluation approach as described in Section 3.2. After collecting all requested information, he saved the set of diagrams in XMI or DI format if supported by the tool. In the case that XMI or DI was produced, he validated the resulting file structurally against the OMG specifications and recorded the results using Altova XMLSpy with the appropriate DTD or XSD file provided by the OMG. Furthermore, he produced a screenshot of the class diagram and the use case diagram. Finally, the evaluator stored all collected information in an evaluation repository.

For a given compliance level only a subset of the features in the feature hierarchy is relevant. Thus, we could derive for each compliance level in Section 3.1 a compliance profile, i.e. a projection of the relevant features in the feature hierarchy to automate the compliance level calculation. By using the compliance profiles as a feature selection mechanism, the results of an individual tool could be calculated with respect to the profile. Thus, we could obtain a feature fulfillment degree per profile and per tool. In fact, the UML requires the (complete) realization of certain diagram types per compliance level. From this strict viewpoint no tool in our evaluation would receive any compliance level at all! Thus, in this study we attested for each candidate level with 50%-75% fulfillment a partial compliance, for more than 75% fulfillment an (acceptable) full compliance.

3.3 Planning, Preparing and Conducting the Evaluation

We conducted an exhaustive evaluation including all tools being relevant from the industry as well as the research viewpoint. In order to achieve this, we first collected sales information (name, vendor, URL) of all tools published in 8 different tool listings or comparisons. This list was completed by intensive thorough internet search for UML

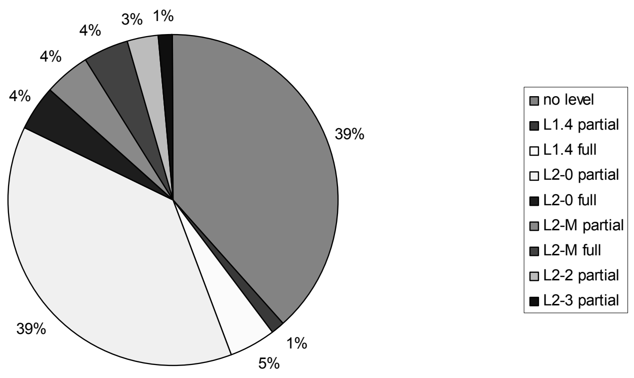

Figure 3: Summary of the maximum compliance levels for all evaluated tools.

tools, so that 200 tools were scheduled for evaluation at the beginning of our work. In parallel, we extracted the feature hierarchy from the UML specification and specified the evaluation procedure as described in Section 3.2.

After the preparation phase, we conducted the evaluation according to the evaluation procedure. Finally, we analyzed the data, e.g. we determined the compliance levels of all evaluated tools upon the collected data and prepared the feature compliance profiles. This provided the basis for the analysis we discuss in the sections below.

4 Results, Findings and Analysis

In this section, we present the results and findings of our evaluation. We aimed at analyzing all commercial and open-source tools being relevant to industry and research. We started with a set of 200 tools, i.e. all tools we could identify. While conducting the evaluation, 66% of the scheduled tools were not available, not maintained after 2003 when UML 2 was introduced, did not provide UML facilities at all or were not available for evaluation. Thus, we evaluated a remaining set of 68 tools, i.e. the currently available tools including all major tools used in industry and research today.

Figure 3 depicts the summary on the UML compliance levels according to the ordering given in Section 3.1. The pie slices in Figure 3 are assigned clockwise starting at the right slice with 39%, i.e. 39% of the tools are assigned to no level, 1% to L1.4 partial, etc. The results are displayed according to partial compliance and acceptable (full) compliance. We attest partial compliance to Level 2.0 for 39% of the tools and do not assign a UML compliance level to 39%. Not all compliance levels are assigned, e.g. we could neither assign full L2-2 nor full L2-3.

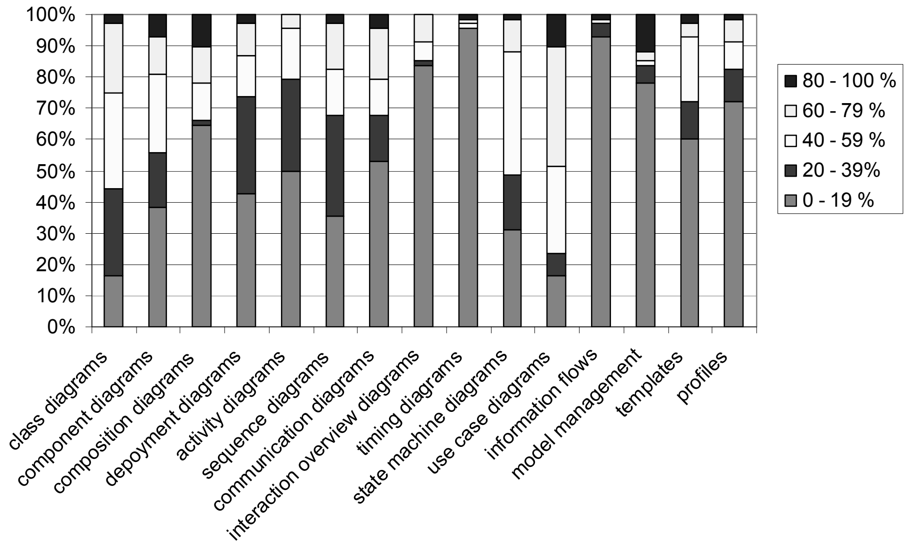

Figure 4: Summary of UML diagram realization measured by feature fulfilment structured according to the categories of the feature hierarchy (excluding model persistence). The legend denotes the degrees of feature realization, the entire chart relates to the evaluated tools.

One can conclude from this summary that currently a lot of tools do not implement all UML diagrams and most tools offer only basic support. Only few tools provide the diagrams introduced with UML 2 as indicated by the fulfilment of L2-2 or L2-3. As a summary of the compliance level rating, 42.6% of the tools are assigned to partial or

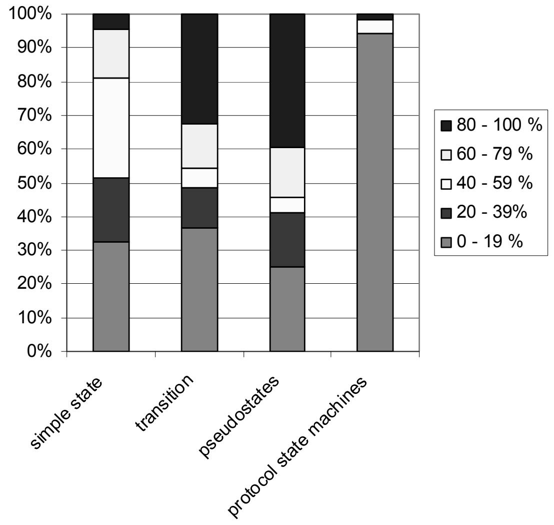

Figure 5: Realization of UML state machine diagrams in terms of feature fulfilment displayed according to selected diagram feature groups. The legend denotes the degrees of feature realization, the entire chart relates to the evaluated tools.

Vendor	Tool name	Version number	Assigned UML Compliance Level	Feature Fulfillment Level
Altova	UModel	Professional 2008	L2.2AC	64.33 %
IDS Scheer	ARIS UML designer	7.0.2.207949	L2.0C	42.78 %
No Magic	Magic Draw	14.0	L1.4 / L2.MAC	77.07 %
Telelogic	Rhapsody	7.11.0	partial L2.0AC	39.92 %
SparxSystems	Enterprise Architect	7.0.817	partial L2.3AC	71.44 %
Visual Paradigm	Visual Paradigm	6.1 sp_1	partial L2.2AC	71.87 %

Table 1: Excerpt from the tool summary table in [EES08].

complete level L2-0, but 82.8% of these tools are considered for level L2-0C, because the tools produced no appropriate XMI. Additional information including detailed characterizations for each tool can be found in [EES08].

Figure 4 displays the realization of features structured by categories, i.e. per UML diagram types and auxiliary UML features. Several of the new UML 2 facilities like composition diagrams, interaction overview diagrams, timing diagrams and information flows are supported only by few tools. We identified also that the ability to work with profiles is not supported by many tools. Despite the fact that a template mechanism was also defined in earlier versions of the UML, templates are also not supported at all by 60% of the tools. The overall support of class diagrams, use case diagrams and state machines appears to be good.

As an example for the data collection on individual diagrams, Figure 5 shows the realization of some selected diagram elements for state machines. Obviously, only few tools consider protocol state machines, but the frequently used elements including the various types of pseudo states are well supported.

Table 1 shows an excerpt from the tool summary table in [EES08]. Only the tools mentioned in this paper are listed in the table. Magic Draw, the tool with the highest feature fulfilment level of 77.07% was assigned to a basic compliance level due to two missing UML 2 diagrams (interaction overview and timing diagrams).

5 Conclusions

UML models are widely used in software engineering. If a UML modelling tool does not properly adopt the specification, this can seriously impede the usefulness of the tool and the user may get locked-in to the tool due to the lack of model exchange capabilities. In particular, this is a drawback for model-driven software engineering approaches. Thus, detailed compliance evaluations can provide an in-depth description of the tools, may support tool selection and migration and give an overview on the current level of implementation.

In this paper, we presented a feature-based evaluation approach to determine the specification compliance of UML tools. The approach relies on an encompassing feature hierarchy extracted form the UML specification and on compliance profiles for given UML versions or UML compliance levels. We evaluated 68 out of an initial set of 200 tools, covering all major UML tools used in industrial practice today. The remaining 132 tools were not available for evaluation, did not exist anymore, or were not maintained for a longer time, etc.

While four tools reach acceptable compliance levels, more than 39% of all evaluated tools do not implement the UML specification sufficiently. 6% of the tools are classified on UML 1.3 or 1.4 level, 39% on basic UML 2 level. Our results on XMI model exchange fit to related evaluations: 47% of the tools do not pass the structural XMI validity test, 3%, i.e. two tools (ARIS and Rhapsody), pass the XMI validity test while the remaining 50% offer no XMI at all.

There is an obvious discrepancy between the compliance level and the feature fulfilment level in Table 1. Compliance levels focus on larger units of the UML language while the feature fulfilment level can provide a more realistic insight for decision makers. In fact, the data aggregation schema described in this paper is designed for the derivation of compliance as defined in the UML specification. To establish a tailorable decision process considering user or company specific weightings, we will consider alternative feature aggregation schemas in future work, e.g. the logic scoring preference method [FDSP05]. As our focus in this paper was on UML compliance, additional data gathered in our evaluation like code generation was not discussed here.

According to our experience it is a complex task to obtain complete and consistent functional requirements for implementing the UML, because much information is given implicitly, as part of the metamodel specification, in descriptive text, as wellformedness-rules, in example diagrams and in partly inconsistent summaries, e.g. tables. Thus, an improved structure of the UML specification including clear requirements also for the semantics would help implementers validating their individual approaches.

Evaluating complex modeling tools on feature level is an immense effort (at least six person months, fulltime equivalent). As the implementation of most tools is continuously improved, it is impossible to always report on the latest versions of all tools. For some tools in Table 1 the version number indicates that in the meantime newer versions have been released. Thus, we will perform a continuous reevaluation and an analysis of the differences in the future.

Acknowledgements

We thank all the vendors for providing their tools according to their terms of evaluation licenses. For future evaluations we hope that the companies will support us again with download versions and evaluation licenses.

References

[BFO+05] C. Bröcker, F. Ferrandina, B. Oestereich, G. Versteegen, and T. Weilkiens. *iX-Studie "Bessere Software!" Konfigurations- und Änderungsmanagement sowie UML*. Heise Verlag, 2005. (in German).

[BFSC08] P. Bunyakiati, A. Finkelstein, J. Skene, and C. Chapman. Using JULE to generate a compliance test suite for the UML standard. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 827–830, New York, NY, USA, 2008. ACM.

[DP07] B. Dobing and J. Parsons. How UML is Used. *Open Source Security*, 5(1), February 2007.

[EES08] H. Eichelberger, Y. Eldogan, and K. Schmid. A Comprehensive Survey of UML Tool Capabilites and Compliance. Technical report, Institut für Informatik, University of Hildesheim, 2008. (internal report).

[FDSP05] A. Funes, A. Dasso, C. Salgado, and M. Peralta. UML Tool Evaluation Requirements. In *Proceedings of ASIS 2005*, 2005.

[Jec04] M. Jeckle. UML Tools (CASE & Drawing), 2004. Online available at: http://www.jeckle.de/umltools.htm.

[KF04] L. Kirchner and U. Frank. Evaluierung von UML-Modellierungswerkzeugen. *OBJEKTspektrum*, (1), 2004.

[LLPM06] B. Lundell, B. Lings, A. Persson, and A. Mattsson. UML Model Interchange in Heterogeneous Tool Environments: An Analysis of Adoptions of XMI 2. In O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, editors, *Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006, Proceedings*, number 4199 in Lecture Notes in Computer Science, pages 619—630. Springer-Verlag Inc., 2006.

[OMG06] OMG. Diagram interchange, v1.0, April 2006. Online available at: http://www.omg.org/docs/formal/06-04-04.pdf.

[OMG07a] OMG. MOF 2.0/XMI Mapping, v2.1.1, December 2007. Online available at: http://www.omg.org/spec/XMI/2.1.1/.

[OMG07b] OMG. Unified Modeling Language: Infrastructure version 2.1.1. Specification v2.11 2007-02-06, Object Management Group, February 2007. Available online at: http://www.omg.org/docs/formal/07-02-06.pdf.

[OMG07c] OMG. Unified Modeling Language: Superstructure version 2.1.1. Specification v2.11 2007-02-05, Object Management Group, February 2007. Available online at: http://www.omg.org/docs/formal/07-02-05.pdf.

[OMG08] OMG. OMG UML Vendor Directory Listing, 2008. Online available at: http://uml-directory.omg.org/vendor/list.htm.

[OOS08] OOSE. UML Tools, 2008. Online available at: http://www.oose.de/umltools.htm.

[www08] Wikipedia, list of UML tools, 2008. Online available at: http://en.wikipedia.org/wiki/List_of_UML_tools.