# EVALUATION OF CASE TOOL METHODS AND PROCESSES
## *An Analysis of Eight Open-source CASE Tools*

Stefan Biffl, Christoph Ferstl, Christian Höllwieser and Thomas Moser
*Institute of Software Technology and Interactive Systems*
*Vienna University of Technology, Favoritenstrasse 9-11/188, Vienna, Austria*

Keywords: CASE Tools, Evaluation, Open source.

Abstract: There are many approaches for Computer-aided software engineering (CASE), often accomplished by expensive tools of market-leading companies. However, to minimize cost, system architects and software designers look for less expensive, if not open-source, CASE tools. As there is often no common understanding on functionality and application area, a general inspection of the open-source CASE tool market is needed. The idea of this paper is to define a "status quo" of the functionality and the procedure models of open-source CASE tools by evaluating these tools using a criteria catalogue for the areas: technology, modelling, code generation, procedure model, and administration. Based on this criteria catalogue, 8 open-source CASE tools were evaluated with 5 predefined scenarios. Major result is: there was no comprehensive open-source CASE tool which assists and fits well to a broad set of developer tasks, especially since a small set of the evaluated tools lack a solid implementation in several of the criteria evaluated. Some of the evaluated tools show either just basic support of all evaluation criteria or high capabilities in a specific area, particularly in code generation.

## 1 INTRODUCTION

Computer-aided software engineering (CASE) has become an important element of software development since the 1980s (Church and Matthews, 1995). Many approaches for supporting the full life-cycle of a software product rose up: modelling, database and application definition, and code generation and verification. In industry most of the development processes were accomplished by often expensive tools of market-leading companies, which were sometimes too complex to handle for Subject Matter Experts (SMEs) (Prather, 1993). Therefore the need surfaced to look for less expensive tools on the market, like open-source tools.

The target audience of this paper consists of system architects, software designers and programmers. These engineers need to decide which tool to use to support their projects in an effective and efficient way of software development. Beneath the assessment of common software development environments, supporting tools like CASE tools, which combine advantages of modelling and code generation in a single package, need to be considered.

Due to the fact that there is no common understanding on functionality and application in the software engineering area, the selection and application of a certain open-source CASE tool may miss user expectations. To cope with this problem, a general inspection of the open-source CASE tool market is performed.

The idea of this work is to define a *"status quo"* baseline of the functionality and the procedure models of these tools and to derive a *"quo vadis"* to give for example system architects or software engineers a baseline for decision making in questions concerning the usage of a CASE tool. The expected results of this work are "Best Practices" concerning functionality and procedure models of currently available and future CASE tools.

To get an overview of the current market situation a criteria catalogue for the evaluation of CASE tools has been created. This criteria catalogue consists of the following evaluation areas: Technical criteria (Usability, Integration with IDEs, Multi-User Support, Import/Export, Multi-Language Support, Interfaces to other modelling tools,); Modelling criteria (Modelling language, Data validation, Func-



Biffl S., Ferstl C., Höllwieser C. and Moser T. (2009).
EVALUATION OF CASE TOOL METHODS AND PROCESSES - An Analysis of Eight Open-source CASE Tools.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 41-48
DOI: 10.5220/0001865700410048

tionality for Analysis or Simulation of modelled data); Code generation criteria (database, application or presentation layers, Target language); Procedure model criteria (Adaptability, End-to-End approach); and Administration criteria (User management, Model management).

Based on this evaluation criteria catalogue, 8 open-source CASE tools were evaluated with 5 predefined scenarios. The results of this study were analyzed and best practices derived.

According to this research approach, the paper is structured as follows: section 2 identifies and describes related work about CASE and previous tool evaluations. Section 3 pictures the evaluation method. Section 4 defines the evaluation criteria and scenarios, while Section 5 presents the results of the evaluation. Section 6 discusses the evaluation results and Section 7 concludes the paper.

## 2 RELATED WORK

The requirements for software engineering steadily increased over the last decades. Thus the need for tools, which assist engineers in the complex software development process, soon became evident coining the term “computer-aided software engineering” (CASE). A good definition can be found in (Fuggetta, 1993): “A CASE tool is a software component supporting a specific task in the software-production process”. Those tasks can be merged into different classes like editing, programming, verification and validation, configuration management, metrics and measurement, project management, and miscellaneous tools. Fuggetta further makes a distinction between tools, workbenches, and environments which classifies the support of only one, a few, or many tasks in the development process.

The trend nowadays towards open-source CASE technology aims definitely at CASE workbenches and environments. To gain high-quality software products as a result of CASE tool output it is important that these tools also provide high-quality techniques. Evaluation therefore is challenging and practices have already been developed in the past.

A principal approach for selection and evaluation is given by Le Blanc and Korn (Le Blanc and Korn, 1992). They suggest a 3-stage method with: 1) screening of prospective candidates and development of a short list of CASE software packages; 2) selecting a CASE tool, if any, which best suits the systems development requirements; 3) matching user requirements to the features of the selected CASE tool and describing how these requirements will be satisfied. At each stage a comparison is made against predefined criteria, whereas the focus lies on functional requirements. The granularity of criteria at each stage increases, so the result of every step is a more precise list of final tools. For evaluation a weighting and scoring model is proposed.

Church and Matthews provide a similar work (Church and Matthews, 1995). Their evaluation focus lies on the following four topics: code generation, ease of use, consistency checking, and document generation. The assessment is done through ordinal scales of ordered attributes.

As mentioned above, CASE tool evaluation is not a simple process. To assist in avoiding potential failures in CASE tool evaluations and therefore poor quality products as result, Prather (Prather, 1993) gives some recommendations in the process itself, necessary prerequisites, knowledge about the organization, technical factors, and the management of unrealistic/unfulfilled expectations. He clearly recommends having in mind the scope of application, because rarely one tool only can fulfil all requirements.

## 3 EVALUATION METHOD

The evaluation was performed in four steps. At the beginning of our work we conducted expert talks with software engineers and an internet research for appropriate candidates of CASE tools in the open-source sector. Looking on the described feature set and a first general examination of those tools we did a further selection which. Based on expertises of software engineers as well as on their total number of downloads from the internet, eight open-source CASE tools were selected for evaluation. Table 1 gives an overview on the selected tools including their versions and the release dates.

The next step was the definition of a criteria catalogue, based on the expertises of software engineers regarding basic functionality of CASE tools. This basic definition was followed by installation and first test of the tools to get a general overview of the different functionalities provided.

After these first impressions, the basic criteria catalogue for evaluation was extended by more details based on the first impression of the CASE tools to be evaluated. This criteria catalogue focuses on different scenarios supported by the overall functionalities provided by the tools.

Those evaluation criteria define what we expect from CASE tools and are reflected through our five dimensions: Modelling in general, Definition of as-

pects on the database layer, Definition of aspects on the application layer, Integration possibilities and Usability. Based on those dimensions, we composed a template of about 100 questions which represent these criteria. The final evaluation was done through a weighting and scoring process.

Table 1: Overview of selected tools for evaluation.

| Tool name | Version | Release date |
|---|---|---|
| ArgoUML | 0.26 | September, 30th 2008 |
| BOUML | 4.8.3 | November, 16th 2008 |
| d'zine | 1.0 | July, 4th 2006 |
| EclipseUML | 4.3 | May, 9th 2008 |
| Fujaba | 4.3.2 | August, 24th 2006 |
| HIDS CASE UD | 0.9 | November, 4th 2004 |
| StarUML | 5.0 | December, 30th 2005 |
| Topcased | 2.2.0 | November, 6th 2008 |

To provide adequate means for analysis we developed a Kiviat diagram (see Fig. 1). This Kiviat diagram shows the evaluation of five scenarios, each represented as an axis. The scenarios focus on different functionality: "Modelling in General", "Definition of aspects on the database layer", "Definition of aspects on the application layer", and "Integration possibilities" and "Usability". The evaluated score on each individual axis is based on a calculation schema for each scenario.

Questions addressing the capability of the scenarios have been used. Each of these questions has been weighted and capability levels from 0 (not applicable) to 5 (high capability) have been defined. The product of weighting and capability level results in a score for a question. The sum of the single scores is the calculated axis value. If a scenario was too broad, sub scenarios were defined. In this case the axis value represents the arithmetic mean of the calculated scores of the sub scenarios. For each scenario/sub scenario a score of up to 5 was reachable.

The following section gives a short introduction of the five scenarios, a detailed listing of evaluation questions including weighting and capability definitions can be found in the appendix of (Biffl et al., 2009).

# 4 EVALUATION CRITERIA AND SCENARIOS

For evaluation criteria based on the functionalities discovered in the first tests of the tools have been enriched with assumptions and experiences of the authors. The following basic descriptions of the five scenarios depict an aggregation of the selected evaluation criteria.

## 4.1 Modelling in General

Due to the fact, that modelling is part of the scenarios "Definition of aspects on the database layer" and "Definition of aspects on the application layer" the more abstract topic "Modelling in general" is taken into consideration. All criteria regarding modelling are defined within this topic.

In comprehensive development projects a major challenge is concurrent collaboration. It is important, that not only a single person is able to design the content in a serial manner. Hence the tool must offer multi-user and parallel modelling support. It has to be possible to install the tools in a client-server environment and a proper user and rights administration needs to be in place. The most important mechanisms for this aspect are locking and synchronisation of modelled content. Based on this collaboration support, comprehensive mechanisms for change and release management produce a great advantage and must be in place.

Table 2: Evaluation Criteria for General Aspects.

| Evaluation Criterion | Weighting |
|---|---|
| Does the tool support a client server installation? | **0,25** |
| Is it possible to define different access rights on models and model groups for users and user groups? | **0,1** |
| Is it possible to exchange data between different installations? | **0,1** |
| Can models be locked? | **0,15** |
| Does the tool support synchronisation mechanisms? | **0,15** |
| Does the tool support mechanisms for change and release management? | **0,1** |
| Is it possible to assign different status attributes to models? | **0,15** |
| Is it possible to adapt status attributes for personal needs? | **0,15** |
| Is it possible to define versions? | **0,3** |
| Is it possible to send notifications? | **0,15** |
| Does the tool offer multi-language support? | **0,2** |
| Are languages for documentation predefined and can they be extended or reduced? | **0,1** |
| Does the tool support the reuse of modelling objects? | **0,35** |

Further reusability is evaluated for two characteristics: multi-language support and the reusability of modelled content (i.e., the presence of "repository-concepts"). Modelling always has the intention of documenting artefacts in order to support the work by giving an overview. In comprehensive development projects also different languages are used. Therefore it is essential to have the advantage of documenting the same content in different languages to ensure consistent understanding of all participants. An object repository, which stores all global information of the objects, would make sense to ensure that the diagrams are kept consistent. Of course, it must be possible to add specific information in context of a diagram to the modelled objects.

Table 2 presents some of the evaluation criteria used for the evaluation of the general modelling aspects of the analyzed CASE tools.

## 4.2 Definition of Aspects on the Database Layer

Together with "Definition of aspects on the application layer" this scenario defines the professional criteria for CASE tools.

Table 3: Evaluation Criteria for Database Layer Aspects.

| Evaluation Criteria | Weighting |
|---|---|
| Is it possible to transform UML diagrams (e.g. class diagrams) to EER diagrams? | **0,05** |
| Is the meta model compliant to the (E)ER notation in general? | **0,15** |
| Does the tool offer analysis mechanism checking the basic (E)ER notation? | **0,3** |
| Is it possible to evaluate a model concerning the 1st, 2nd, 3rd and Boyce-Codd normal form? | **0,3** |
| Does the tool support the generation of SQL code? | **0,3** |
| Is all modelled information transformed into code? | **0,15** |
| Is it possible to generate automatically a diagram out of a SQL dump file? | **0,15** |
| Is it possible to change generated models (from dump files) and commit them back to the original database without data loss? | **0,1** |
| Does the tool offer a procedure model for designing a database? | **0,1** |
| Is it possible to define test cases? | **0,3** |

Addressing modelling, a specific notation for the database design must be in place of course. The most famous notation in this context is the Entity Relationship model (ERM) respectively the Extended Entity Relationship model (EERM).

After the definition of the database design, it should be possible to perform several analysis mechanisms. A basic analysis mechanism for example is an automatic check whether a model is compliant to modelling guidelines defined by the (E)ER notation. A comprehensive analysis / optimization mechanism would be a check for the normalization of the defined database. Forward and reverse engineering for the database layer should ensure the generation of SQL code and import of SQL dumps at least.

The support of a database design process (also referred to as "database lifecycle") should be reflected in model types and functionalities of CASE tools including the differentiation between database views and e.g., the definition of requirements.

Table 3 lists some of the evaluation criteria used for the evaluation of the database layer aspects of the analyzed CASE tools.

## 4.3 Definition of Aspects on the Application Layer

This scenario is very similar to the previous description of the database scenario. Thus, the requirements are quite similar. For the application layer again subtopics for specific modelling issues, analysis mechanisms, forward and reverse engineering and the procedure model are defined. In the development process the modelling focus lies on OMG's UML 2.0. Further a rudimentary support for Business Process Modelling (BPM) is desirable.

Analyzing features should be implemented to assist in designing of a proper data model that meets the specifications of the underlying design languages and determining of code quality. The emphasis regarding code generation and reverse engineering lies on the target/source languages Java and C++ and definition or administration of design patterns.

Additionally the support of a basic development process (e.g., Rational Unified Process (RUP) or the waterfall model) is useful. Therefore mechanisms for requirement management, specification, design, implementation, quality management, integration and delivery are needed.

Table 4 shows some of the evaluation criteria used for the evaluation of the application layer aspects of the analyzed CASE tools.

Table 4: Evaluation Criteria for Application Layer Aspects.

| Evaluation Criterion | Weighting |
|---|---|
| Is it possible to define business processes (e.g., BPEL)? | **0,15** |
| Which model types of the UML library are available? | **0,2** |
| Is it possible to merge between UML data models? | **0,05** |
| Are UML Profiles supported? | **0,05** |
| Are there analyzing features for achieving compliance with the specifications in UML? | **0,5** |
| Are there facilities for determination of code quality and metrics? | **0,3** |
| Is there support for forward Java Code generation? | **0,2** |
| Is there support for reverse Java Code generation? | **0,1** |
| Does the tool offer a model for designing the application layer? | **0,1** |
| Is it possible to define test cases? | **0,25** |

## 4.4 Integration Possibilities

One important feature of a complex system like a CASE tool is the communication with its environment. This allows flexible collaboration and documentation of the whole development process. In this scenario the focus is on import and export of different kinds of data. Another aspect is the assistance in implemented interfaces or the possibility for integration with integrated development environments (IDEs).

Table 5 displays some of the evaluation criteria used for the evaluation of the integration possibilities of the analyzed CASE tools.

Table 5: Evaluation Criteria for Integration possibilities.

| Evaluation Criterion | Weighting |
|---|---|
| Is it possible to import data models in XMI format? | ***0,15*** |
| Is there an Import of BPEL for Web Services (BPEL4WS) for generation of BPMN diagrams? | ***0,1*** |
| Is it possible to import Java Code from JAR-Class files and transformed into diagrams? | ***0,05*** |
| Is there support for Interface definition language (IDL) import? | ***0,15*** |
| Is it possible to export in XMI? | ***0,15*** |
| Is it possible to export the models into standard graphic formats? | ***0,05*** |

## 4.5 Usability

The last topic which is taken into consideration consists of usability aspects. A tool should always support a user in fulfilling a specific task in a comfortable way. Several of the prior defined functions should not only be available, but should also fulfil relevant usability aspects.

First of all, an easy and assisted installation is recommended for satisfying usage. Modelling as a core activity within CASE tools should be made as easy as possible for the users. The procedure of creating a model, creating an object and enriching the object with information should be self-describing and comfortable. Additionally adequate help functionality must be in place, at least a forum.

Administration tasks within the tools also need to be evaluated addressing usability. Two different administration areas are taken under consideration, administration of modelled content and administration of users, groups and access rights. The creation of models and repository objects within definable folders ensure an adequate structure of the content.

Another subtopic regarding usability deals with the ability to adapt the tool for the user's needs. In detail the adaption of the meta model and the possibilities for personalization are taken into consideration.

# 5 EVALUATION RESULTS

The general result of the evaluation is that none of the selected CASE tools is an absolute winner. In Figure 1 there is an illustration of the evaluation result. A Kiviat diagram with 5 axes represents the 5 evaluation dimensions: Modelling, Database Layer, Application Layer, Integration, and Usability. Each axis is signed with the evaluation scale from 0 to 5.

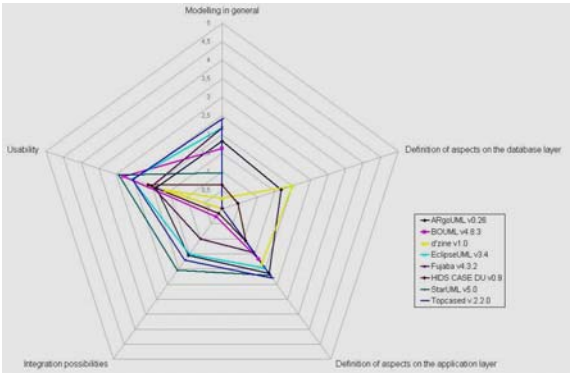


Figure 1: Open-source Case tool evaluation result.

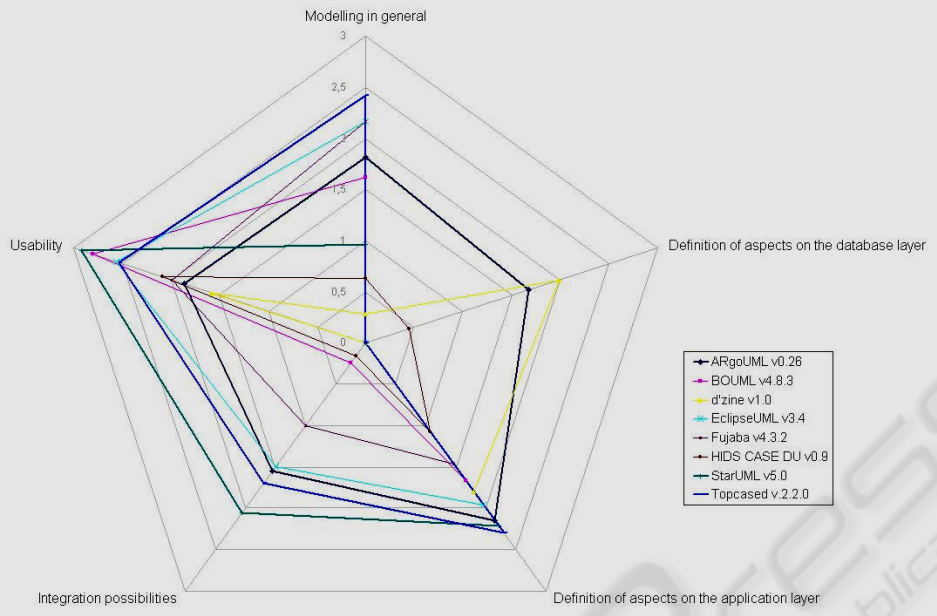


Figure 2: Open-source Case tool evaluation result “normalized”.

Table 6: Open-source Case tool evaluation result table.

| Scenario | ArgoUML | BOUML | d'zine | EclipseUML | Fujaba | HIDS CASE UD | StarUML | Topcased |
|---|---|---|---|---|---|---|---|---|
| **Modelling in general** | | | | | | | | |
| *Collaboration support* | *0,75* | *0,75* | *0,75* | *2,3* | *2.3* | *0,75* | *0,75* | *3.05* |
| *Workflow support* | *2,65* | *2,05* | *0,1* | *2,15* | *2,15* | *0,15* | *0,1* | *2,15* |
| *Reusability of modelling artefacts* | *2,05* | *2,05* | *0* | *2,05* | *2,05* | *0* | *2,05* | *2,05* |
| *arithmetic mean* | ***1,817*** | ***1,6167*** | ***0,2833*** | ***2,167*** | ***2,1667*** | ***0,6333*** | ***0,9667*** | ***2,4167*** |
| **Definition of aspects on the database layer** | | | | | | | | |
| *Specific modelling issues* | *2,3* | *0* | *2,15* | *0* | *0* | *1,4* | *0* | *0* |
| *Analysis mechanisms* | *1,9* | *0* | *2,9* | *0* | *0* | *0* | *0* | *0* |
| *Forward and reverse engineering* | *2,5* | *0* | *2,5* | *0* | *0* | *0* | *0* | *0* |
| *Procedure model* | *0* | *0* | *0,4* | *0* | *0* | *0,4* | *0* | *0* |
| *arithmetic mean* | ***1,675*** | ***0*** | ***1,9875*** | ***0*** | ***0*** | ***0,45*** | ***0*** | ***0*** |
| **Definition of aspects on the application layer** | | | | | | | | |
| *Specific modelling issues* | *3,1* | *3,1* | *2,2* | *3,35* | *2,9* | *2,5* | *3,35* | *3,75* |
| *Analysis mechanisms* | *1,8* | *0* | *1,5* | *1,5* | *0* | *0* | *1,5* | *1,5* |
| *Forward and reverse engineering* | *3,3* | *3,55* | *3,5* | *2,15* | *2,2* | *1,8* | *3,6* | *2,5* |
| *Procedure model* | *0,4* | *0* | *0* | *0,85* | *0,75* | *0* | *0,4* | *1,45* |
| *arithmetic mean* | ***2,15*** | ***1,6625*** | ***1,8*** | ***1,9625*** | ***1,4625*** | ***1,075*** | ***2,2125*** | ***2,3*** |
| **Integration possibilities** | | | | | | | | |
| | ***1,55*** | ***0,25*** | ***0*** | ***1,5*** | ***1*** | ***0,15*** | ***2.05*** | ***1,7*** |
| **Usability** | | | | | | | | |
| *Installation* | *3,2* | *4,6* | *2,9* | *4,6* | *4,4* | *4,4* | *5* | *5* |
| *Modelling pace* | *4,25* | *3,95* | *3,45* | *4,15* | *3,3* | *3,95* | *4,35* | *4,15* |
| *Administration* | *0* | *1,5* | *0* | *1,5* | *0* | *0* | *1,5* | *1* |
| *Adaptability* | *0* | *1,15* | *0* | *0* | *0,25* | *0* | *0,85* | *0* |
| *arithmetic mean* | ***1,8625*** | ***2,8*** | ***1,5875*** | ***2,5625*** | ***1,9875*** | ***2,0875*** | ***2,925*** | ***2,5375*** |

What can be seen at first sight is that none of the tools is getting beyond level 3 at any scenario. This reflects that some of the CASE tools have their strengths in some of the dimensions but neither of them has a consistently good result for all dimensions. In Figure 2 the axes are shortened in scale from 0 to 3 for a better view of the several CASE tools. Based on these results some of the tools can be categorized as better than others. These tools are marked bold in the diagram, namely ArgoUML, StarUML and Topcased. In these cases high values have been reached only in single dimensions. ArgoUML is a good candidate for an overall winner cause of its broad approach but only average results have been reached within the dimensions. StarUML and Topcased fulfilled the stated requirements apart from the scenario “Definition of aspects on the database layer” best.

In Table 6 an overall view of the evaluation results for all the CASE tools is presented. The columns represent the calculated score values.

# 6 DISCUSSION

Beside the above stated results of the evaluation, which represent pure figures, some findings concerning general impressions and soft facts are discussed in this section. Although evaluations reflect to some extent the evaluators' views, some grievances cannot be denied.

**Business-IT Alignment** – a famous and too often used phrase within IT Management – craves for the orientation of IT towards business needs and value creation. These constitutional ideas of a company also need to be considered within software processes. A good approach would be to integrate these value creation processes into the documentation of software and link these two "layers" by given requirements. Such a construction would help to give a glance how software collaborates in the creation of value and would also bridge the gap between the business and the technical points of view. Unfortunately, none of the evaluated tools touches with these issues.

The issue of **missing process model support** needs to be raised. While a full integration of established software engineering process models like the RUP or the V-Model is not always required, some initial approaches like the definition of tests for different artefacts and feedback mechanisms would be very helpful in many project contexts but are insufficiently supported by the evaluated tools.

Since a process model comes with the concept of roles, a minimum set of **collaboration support features** (e.g., the usage of the same data within a client server environment) is required. Adequate user and rights management should also be available, supporting a clear definition of responsibilities leading to quality improvement of the results.

Many of the evaluated tools **focus on the application layer**. By integrating a few further diagram types like EER Diagram, respectively DB Schema Diagram, and by providing export possibilities of SQL code significant additional benefit could easily be achieved. In an ideal case, it should be possible to integrate the documentation of the application and the database layer to ensure easy analysis of change and incident impacts and to support the project planning by pinpointing dependencies.

Because almost every company uses a large number of different tools within the software engineering process, integration possibilities are important that avoid work needing to be done twice. Therefore some **standard interfaces** for example via XML need to exist but also possibilities for the definition of proprietary interfaces must be implemented. These interfaces have to provide functionality for both importing and exporting data without loss of information. Also a lack within this area needs to be stated.

The last general impression, before focussing on soft facts, is missing functionality for the **adaption and administration** of the software. None of the tools provides a really good solution for structuring and administrating diagrams and objects including a prior mentioned user and rights management. Concerning adaption some functionality, e.g., the creation of additional attributes or modelling classes, basic changes of the code generation or even the customization of the graphical representation, would help users to design some kind of "personal" CASE tool, which exactly, or at least better, could fit their needs.

Additionally, it needs to be considered that some facts have not been taken into account for the evaluation results. These soft facts like numerous crashes of the software or higher effort for installation and incorporation due to lack of documentation and presumed skill levels are of course also a crucial factor for the selection of a tool. These factors are reflected in the time it took for the evaluation. This time varied between 6 hours and 36 hours for one tool. In summary, we can state that many of the evaluated approaches lead in the right direction, but high potential for improvements exists.

# 7 CONCLUSIONS

Increasingly complex software engineering projects need the best possible support with professional CASE tools. In this paper we conducted a market evaluation of representative open-source CASE tools to get an impression what level of tool support is available for free. We set up a criteria catalogue that reflects the needs of professional software developers and evaluated these with 8 open-source CASE tools.

Major results are: a) No open-source CASE tool exceeded level 3 of a 6-level scale on any of the evaluation criteria, which seems surprisingly low. b) None of the tools is a champion in many of the evaluation dimensions. c) Some of the evaluated tools show either just basic support of all evaluation criteria or high capabilities in a specific area, particularly in code generation.

The results show that there is no comprehensive CASE tool which assists and fits well to a broad set of developer tasks. Some of the tools have a very weak implementation in some of the dimensions especially in modelling of the database layer and in integration possibilities. Therefore, it is necessary to make a distinction between a global or specific approach. A candidate for a global approach is ArgoUML which has coverage on all dimensions but with rather low values.

On the other side some CASE tools have specific specialization capability particularly in code generation. Those tools are Topcased as well as StarUML. It should be said that Topcased and EclipseUML can benefit from the integration into the Eclipse Platform and can therefore make use of the Eclipse Plug-In Technology where for example Model Transformation with QVT is already supported. Disadvantage is that Eclipse related tools are highly concentrated on Java programming language.

The results of the evaluation seem rather unsatisfactory regarding the original evaluation criteria dimensions. We can not be disappointed because the sector where we have done our research is in the open-source field. A general important remark is that it was never the intention of the authors to single out any specific tool, but provide an overview of functionality and give some situational recommendations. In the open-source sector one should always keep in mind that work is done from developers under enthusiastic circumstances often without any reward.

A real problem seems to be that there exists a wide range of commercial CASE tool software, for which less information is available about the quality of those products. Thus we wanted to achieve primarily the definition of an evaluation template and further to give an objective selection as possible of available CASE tools.

As the focus of this research was on open-source CASE tools only; future should compare open-source and commercial CASE tools, eventually with an adapted criteria catalogue. The result should be a cost-benefit analysis of the rich feature set of commercial CASE tools and the reduced features of open-source CASE tools.

## REFERENCES

ArgoUML Homepage, http://argouml.tigris.org/, November 2008.

BIFFL, S., FERSTL, C., HÖLLWIESER, C. & MOSER, T. (2009) Evaluation of Case Tools Methods and Processes: A Case Study Analysis of eight Open-source CASE Tools. Technical Report. Available online at: http://www.ifs.tuwien.ac.at/files/TR_Case_Tools_Eval.pdf.

BOUML, Homepage: http://bouml.free.fr/, November 2008.

CHURCH, T. & MATTHEWS, P. (1995) An evaluation of object-oriented CASE tools: the Newbridge experience. Seventh International Workshop on Computer-Aided Software Engineering. Toronto, Canada.d'zine Homepage: http://samparkh.com/dzine/index.html, Nov 2008.

EclipseUML, Omondo, Inc. Homepage: http://www.eclipsedownload.com/, November 2008.

FUGGETTA, A. (1993) A classification of CASE technology. Computer, 26, 25-38.

Fujaba Homepage, Fujaba Tool Suite Developer Team, http://www.fujaba.de/downloads/index.html, Nov 08.

HIDS CASE UD - UML CASE Tool Homepage: http://sourceforge.net/projects/hidscaseud, Nov 2008.

LE BLANC, L. A. & KORN, W. M. (1992) A structured approach to the evaluation and selection of CASE tools. 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's. Kansas City, Missouri, United States, ACM.

PRATHER, B. (1993) Critical failure points of CASE tool evaluation and selection. Sixth International Workshop on Computer-Aided Software Engineering (CASE '93).

StarUML Homepage: http://staruml.sourceforge.net/en/, November 2008.

Topcased Homepage: http://topcased.gforge.enseeiht.fr/, November 2008.