Empirical Evaluation of CASE Tools Usage at Nokia

ALESSANDRO MACCARI alessandro.maccari@nokia.com
Software Technology Laboratory, Nokia Research Center, Helsinki, P.O. Box 407, FIN- 00045, Finland

CLAUDIO RIVA claudio.riva@nokia.com
Software Technology Laboratory, Nokia Research Center, Helsinki, P.O. Box 407, FIN- 00045, Finland

Abstract: We present the results of a research work targeted to understanding the domains and consequences of CASE tools usage in Nokia. We aim to evaluate the importance of the various CASE tools features, as rated by our developers, and how well such features are implemented in currently available CASE tools. A structured questionnaire was sent to our most experienced developers and CASE users. From this survey, it emerged that CASE tools support is reputed most useful for the following functions: graphical drawing, automatic documentation generation and storage of diagrams. The results hint to a mismatch between the features required by the developers and those offered by CASE products. Further research is needed before more definite conclusions can be drawn.

Keywords: CASE tools, empirical evaluation, software engineering, UML

1. Introduction

CASE (Computer Aided Software Engineering) tools are applications that automate certain parts of the software development cycle. Some features of CASE tools are listed in Appendix 1.

In the recent years, a number of software houses has undertaken the development of CASE tools, with excellent results at least in sales figures (Rational Software Corporation is the school example). This boom constitutes an obvious response to the assumption that automation of the software lifecycle has positive outcome on the final product.

However, such an assumption is largely unjustified (as shown by [Fenton94]). Research work on CASE tools usage is insufficient (with a few exceptions such as [Orlikowski93], [Post98], [Iivari98]), especially in particular domains, such as telecom.

The advertised benefits of CASE tool usage have never been univocally proved true. The research activities present contrasting results, which suggest caution in making definite interpretations (see [Glass99] for a review of some of the existing literature).

Moreover, the shortage of previous similar evaluative experiments makes for the lack of guidelines on how to perform them (an exception is [Hendrickson99]).

We present the initial results of a research focused on the domain of software development for mobile handsets.

Although not complete, it poses the basis for a family of experiments, as advocated by [Basili99], aimed to characterise the phenomenon under study. See Section 3.4 for a discussion on the limitations of this work.

2. Research background

2.1. Motivations

The problems that originate the need for this research are the following:

- the lack of rigorous experimental data on the usage of CASE tools in the telecom software industry.
- the uncertainty regarding the requirements of an ideal CASE tool in Nokia
- the absence of an objective comparison of the features of the various tools currently marketed.
- the lack of studies showing how CASE tool usage affects the software development organisations.

Although we will limit out scope to Nokia, these problems could be restated for other organisations in or outside the telecom domain.

2.2. Objectives

The objectives of the research work are:

- identifying the features that are most desired by the developers in a hypothetical, ideal CASE tool;
- understanding the main differences between such features and those actually offered by the commercially available CASE tools used at Nokia;
- identifying the influence of CASE tools usage on software development organisations and on the quality of the software systems they produce;
- on a broader scale: contributing to the global knowledge on the benefits of CASE tools usage.

2.3. Research philosophy and approach

Orlikowski and Baroudi provided an excellent classification of philosophies and approaches for information technology research [Orlikowski91].

Following their reasoning, we may list the following facts as characterising our research work.

- ❑ Ontologically, we do not make any assumption on the behaviour of the CASE tool users, nor on the reflections of CASE usage on the organisation. We try to deduce it from appropriate analysis of collected data.
- ❑ Socially, we do not assume any predefined regularity to rule the social reality where CASE tool users work (and thus live).
- ❑ Epistemologically, we believe that the phenomena of interest (mainly: the individual and organisational requirements for CASE products; indirectly, also the consequences of their usage on software development) can be understood by in-depth enquiries made with the development teams.

From such facts, our research work can be classified as interpretive.

However, we believe that developers possess a degree of knowledge about CASE tools features and their way of usage. We also assume that designers are individually aware of what benefits such features may bring to their work.

Under this aspect, our research work can be classified as positivist.

The two mentioned aspects may seem to collide, but in our opinion they do not. The main difference concerns the organisational versus individual point of view.

The following sections outline the empirical study design, and present our initial answer to the research problems stated in section 2.1. .

3. Empirical study

The study we have carried on so far tries to achieve the first two research objectives listed in section 2.2. . It was performed in the context of various mobile phone software development units inside Nokia, both in Europe and in the US.

3.1. Research study design

The *objectives of the study* presented in this paper are a subset of the objectives of the research (listed in 2.2.):

1) evaluating the usefulness of the currently available CASE tools features: this part of the research aims to investigate some facts (usefulness) in a domain that is at least partially known (CASE tools features) but has not been sufficiently examined in past research;
2) Rating the available tools with respect to their capability to fulfil the users needs: this part of the research aims to evaluate how the features offered by existing tools are rated by the developers.

3.2. Study goals

The goals of the study are listed following the goal/ question/ metric (GQM) goal template. This template was first presented in [Basili88]. Since we aim to conduct a family of experiments, we apply the GQM goal template following the guidelines described in [Basili99].

3.2.1. Evaluate the usefulness of the features present in currently available CASE tools

Analyse: the usefulness that the developers assign to features that are present in currently available CASE tools
In order to: understand the areas of CASE application
With respect to the software lifecycle phases
From perspective of the software developer
In the context of the developing team and organisation

3.2.2. Rate the commercially available CASE tools

Analyse: the user satisfaction with the features currently offered by commercial CASE tools
In order to: understand the strong and weak points of each tool
With respect to productivity, quality, support for software lifecycle automation
From perspective of the software developer
In the context of the future firm organisation

3.3. Data collection

The data presented in the following section has been collected by means of a structured questionnaire. It was sent to 48 design engineers and software developers by electronic mail during August 1999. The answers were collected and analysed during fall 1999.

CASE tools have been extensively adopted in Nokia only recently. We estimate the number of software developers using CASE products to range in the order of a hundred. However, only a fraction of them are reputed to possess a sufficiently high experience to feed our study with significant data. Trying to cover this last subset, we have intentionally kept the number of recipients low.

We received 14 responses, for a response rate of almost 30%. According to [Edwards72], as quoted in [Wood99], a response rate of 20% to 30% for mailed questionnaires is considered acceptable. Our respondents are physically located in Finland (6), Europe (7) and US (1).

The respondents have between 6 months and 25 years of software development experience, with an average of 8.25 years of experience. Their experience in software development inside Nokia ranges between 6 months and 7 years, with 2.71 years on average. Their experience in mobile phones software development ranges between the same value of 6 months and 7 years, but with an average of 2.14 years.

3.4. Validity threats and limitations of the study

In the following paragraph, we list the most relevant threats to the validity of this study.

3.4.1. Developers may be biased

Especially if very experienced, software developers tend to judge negatively any attempt of automation brought to their work. This is a naturally comprehensible behaviour, as automation of any human activity is thought to diminish the creativity of the author.

In our opinion, this factor is implicit in the research problem, and should not be considered a risk.

3.4.2. The results may be valid only for our company (or domain)

Since all subjects of the research belong to the same company, there may be some company-specific biasing that invalidates the generalisation of results.

This threat cannot be eliminated but by carrying on a number of similar surveys within different organisations.

Attention should be also paid to such factors as biasing due to the type of industry and physical location of the organisation premises. In our case, all respondents except for one are located in northern Europe. A similar survey should be carried on in other areas of the world, in order to see if the results hold.

3.4.3. The respondents set is too reduced

This is perhaps the biggest validation threat to our experiment. The number of experienced CASE tools users that we could reach with our survey was low, since CASE tools usage has spread only recently in our company.

In addition, the questionnaire was sent by electronic mail, a means which is traditionally proved to give low response rates.

3.4.4. The respondents are mainly software designers

Although intentional, this may bring some biasing as concerns evaluation of the features related to the later phases of software lifecycle (such as implementation, testing, maintenance, configuration).

3.4.5. GUI developers are not represented in the respondents set

This may have brought down the evaluation of code generation usefulness. Usually, GUI developers tend to make extensive use of code generation. The next part of the research should compensate for this incompleteness.

4. Data analysis

We have analysed the collected data using two simple descriptive statistics: median and standard deviation. The use of the median value was preferred to the mode, although the latter is theoretically the best consensus indicator. The choice is justified by the loss of information caused by modal analysis and by the absence of consensus in case of a non-unimodal distribution.

We have visualised the data using the box-and-whisker representation, as shown in Appendix 2. The middle point of the box plot is the median of the distribution, the box height is the quartile and the whisker reflects the min-max range.

We have grouped the features according to their estimated usefulness for a hypothetical, ideal CASE tool. Subsequently, we have classified such features against the rated quality of implementation in commercial tools.

4.1. Ideal CASE tool features

Figure 1 in Appendix 2 shows the box plots for the ideal CASE tool features. According to the median value M of the responses, features are grouped as follows:

- $4 < M \leq 5$: highly useful;

- $3 < M \leq 4$: somewhat useful;
- $1 \leq M \leq 3$: not useful.

The respondents have identified the following features as *highly* useful in a hypothetical, ideal CASE tool:

Modelling features

- Support for standard UML notation
- Perform diagram analysis (e.g. consistency check)
- Support design specification
- Be intuitive and easy to use

Maintenance

- Utilise a repository

Documentation process

- Allow easy editing of text notes inside diagrams
- Allow easy editing of graphical data (diagrams)
- Automatically generated well structured documents from models

Configuration Management

- Manage versioning

The list clearly shows that the ideal CASE tool should have the following characteristics.

- The ideal CASE tool should be a *graphical, easy to use* tool that helps in drawing models of software systems. Support for UML notation is considered an important requirement.
- The ideal CASE tool should be a *documentation* tool that helps in the process of software documentation. Developers also need a support for automatic generation of well-structured design documents from models. This feature could reduce the inconsistency between design model and documentation.
- The ideal CASE tool should act as a *repository* to store the artefacts and track their changes. This is relevant for medium to large size development projects that need an efficient way to manage documents and configure product variants.

Surprisingly, a number of features that are heavily advertised by CASE tool vendors are rated only somewhat useful by the respondents. The most relevant such examples are:

- generate correct, well-structured code;
- help building prototypes
- performing simulations.

In the same way, a number of popular features were rated non useful. Examples are:

- Reverse engineering support.
- All process-management related features.
- All quality-related features.

4.2. The evaluated CASE tools

The respondents were asked to rate how well the various features are implemented in the CASE tool they had most experience with. Five different CASE tools have been listed by the respondents:

- *Rose98* by Rational Software Corporation [RationalRose],
- *ObjecTime Developer* by ObjecTime Ltd [ObjecTime],

- *Prosa/om* by Insoft [Prosa],
- *Rhapsody* by iLogix [Rhapsody]
- *QLM 2.1* by Qualiware [QLM].

Based on the collected responses, we have sketched a general outline of the features offered by existing tools, and compared it to their rated usefulness in the ideal tool.

Figure 1 in Appendix 2 shows the box plots for the ideal CASE tool features.

Features have been rated according to the median value M_u of the scores set.

- $4 < M_u \leq 5$: very well implemented;
- $3 < M_u \leq 4$: sufficiently well implemented;
- $1 \leq M_u \leq 3$: poorly or insufficiently well implemented.

No feature has been rated *very well implemented*. This may mean that, in general, commercial CASE tools fail to implement features in an optimal way.

The following list shows the features that the respondents consider *sufficiently well implemented:*

Modelling features

- Support for standard UML notation
- Be able to edit all the UML diagrams
- Perform diagram analysis (e.g. consistency check)
- Support design specification

Implementation

- Generate correct, well structured code
- Help in the debugging phase

Maintenance

- Utilise a repository

Documentation process

- Allow easy editing of text notes inside diagrams
- Allow easy editing of graphical data (diagrams)

To check how the needs of the users are supported by the existing tools, we have grouped the features in three categories:

Sufficiently well implemented and highly desired:

- Support for standard UML notation
- Perform diagram analysis (e.g. consistency check)
- Support design specification
- Utilise a repository
- Allow easy editing of text notes inside diagrams
- Allow easy editing of graphical data (diagrams)

Insufficiently well implemented but highly desired:

- Be intuitive and easy to use
- Automatically generate well-structured documents
- Manage versioning

Sufficiently well implemented but not highly desired:

- Be able to edit all the UML diagrams
- Generate correct, well structured code
- Help in the debugging phase

5. Conclusions

The results of our research allow us to make some preliminary statements on:

- what CASE tools features are rated most useful by telecom software developers;
- how well such features are implemented in commercial CASE tools.

Future research work should continue in two branches.

The first part should focus on investigating the reasons that lie behind some surprising results. Two evident examples are the lower-than-expected usefulness of code generation and the expressed need for configuration management and testing related features.

The second part should investigate the consequences of CASE tools usage in our software development organisation using an interpretive approach. Interesting sub-goals would be (1) establishing the connection between the rated usefulness of CASE tools features and the impact of their usage and (2) establishing to what extent the adoption of a certain CASE tool affects our organisation and its products.

We also encourage that academic and research organisations carry out similar experiments with developers and teams belonging to different telecom industries. We believe that commitment to such research activities will pay back in terms of better understanding of our software development organisations and practises.

6. References

[Basili88]: V. R. Basili, H. D. Rombach, *The TAME project: towards improvement-oriented software environments,* IEEE Transactions on Software Engineering, vol. 14 n. 6, June 1988.

[Basili99]: V. R. Basili, F. Shull, F. Lanubile, *Building knowledge through families of experiments,* IEEE Transactions on Software Engineering, vol. 25 n. 4, July/August 1999.

[Edwards72]: B. Edwards, *Statistics for business students,* 1st Edition, Collins, 1972.

[Fenton94]: N. Fenton, S. Lawrence Pfleger, R. L. Glass, *Science and substance: a challenge to software engineers,* IEEE Software, July 1994, pp. 86-95.

[Glass99]: R. L. Glass, *The realities of software technology – Payoffs,* Communications of the ACM, February 1999.

[Hendrickson99]: E. Hendrickson, *Evaluating CASE tools,* Software Testing & Quality Engineering, January/February 1999, p. 38-42.

[Iivari98]: J. Iivari, J. Maansaari, *The usage of systems development: why are we stuck to old practices?,* Information and Software Technology 40 (1998), p. 501- 510, Elsevier Science, 1998.

[ISO14102]: ISO/IEC 14102, *Information Technology, Guideline for the evaluation and selection of CASE tools,* 15 Nov 1995 (first edition).

[ObjecTime]:http://www.objectime.com/otl/products/foundation/otd_overview.html.

[Orlikowski91]: W. J. Orlikowski, J. J. Baroudi, *Studying information technology in organizations, research approaches and assumptions,* Information Systems Research 2:1, 1991.

[Orlikowski93]: W. J. Orlikowski, *CASE tools as organizational change investigating incremental and radical changes in systems development,* Management Information Systems Quarterly, vol. 17 n. 3, September 1993.

[Post98]: G. Post, A. Kagan, R. T. Keim, *A comparative evaluation of CASE tools,* The Journal of Systems and Software 44 (1998), p. 87-96, Elsevier Science, 1998.

[Prosa]: http://www.insoft.fi/prmain.html

[QLM]: http://www.qualiware.dk/doc1085.htm

[RationalRose]:http://www.rational.com/products/rose/index.jtmpl.

[Rhapsody]: http://www.ilogix.com/rhover_c.htm
[Wood99]: M. Wood, J. Daly, J. Miller, M. Roper, *Multi-method research: an empirical investigation of object-oriented technology,* The Journal of Systems and Software 48 (1999), 13-26.

Appendix 1: the questionnaire structure

The evaluation questionnaire was tailored from the international standard ISO/IEC 14102 [ISO14102]. The captions in Figure 1 and Figure 2 refer to the features listed below.

For each of the questionnaire items the respondents were asked to rate the desirability of the feature (or the quality of its implementation) on a scale from 1 to 5.

1. Modelling features
1.1 Support for standard UML notation
1.2 Be able to edit all the UML diagrams
1.3 Perform diagram analysis (e.g. consistency checks)
1.4 Support requirements specification methods
1.5 Support design specification
1.6 Help performing simulation
1.7 Help building prototypes
1.8 Be intuitive and easy to use
1.9 Allow concurrent editing of the same model
2. Implementation
2.1 Generate correct, well structured code
2.2 Help in the debugging phase
3. Maintenance
3.1 Utilise a repository
3.2 Be able to read and analyse existing code
3.3 Be able to obtain models of existing, non-modelled code
4. Documentation process
4.1 Allow easy editing of text notes inside diagrams
4.2 Allow easy editing of graphical data (diagrams)
4.3 Automatically generate well-structured documents from models
4.4 Support hypertext navigation in the model
4.5 Support free form attachments in the model
5. Configuration management
5.1 Help tracking modification within the model
5.2 Manage versioning
6. Process management
6.1 Track project deliverables in the model
6.2 Analyse and report on project status
6.3 Support process (lifecycle) management
7. Quality
7.1 Help in managing quality parameters
7.2 Provide support for risk management
8. Testing support
8.1 Automatic testing
8.2 Module testing

8.3 Regression testing
8.4 Integration testing
8.5 Runtime analysis
8.6 Analyse test coverage
8.7 Support automatic test result verification

Appendix 2: box plots for the evaluated CASE tool features.

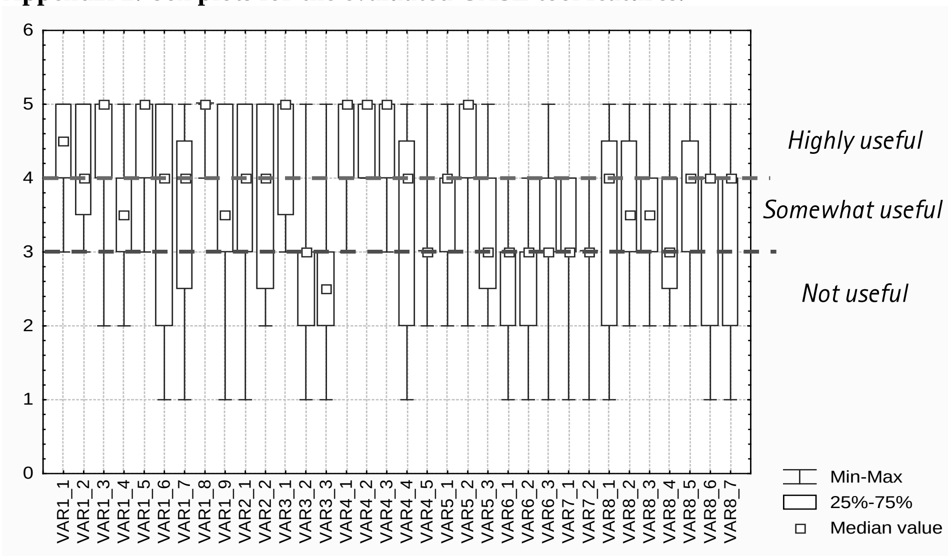

Figure 1. Mean values of the idea CASE tool features (labels as in Appendix 1).

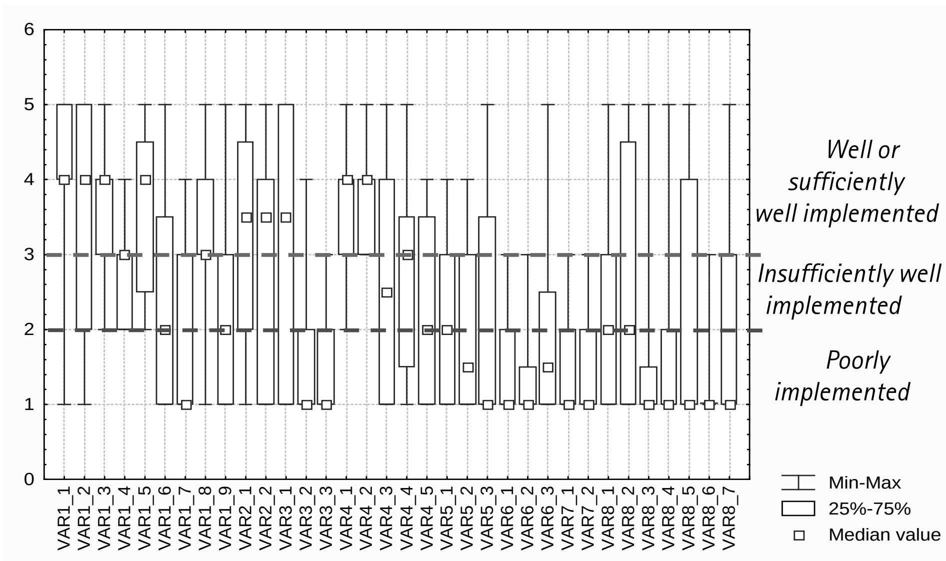

Figure 2. Mean values of the evaluated CASE tool features (labels as in Appendix 1).