An Approach for Empirical Evaluation of Model-Driven Engineering in Multiple Dimensions

Parastoo Mohagheghi

SINTEF, Forskningsveien 1, Oslo, Norway
parastoo.mohagheghi@sintef.no

Abstract. Evaluating methodologies and tools for software development has been subject of research for a while and various quantitative techniques have emerged. However, a common problem in any evaluation method is to select appropriate evaluation criteria. This paper describes the evaluation method and some results of work on empirical evaluation of model-driven engineering and its infusion in industry in the context of industrial participants in the EU research project MODELPLEX. The evaluation method combines qualitative and quantitative assessment, and observations with perceptions. It takes advantage of a combination of a) context-dependent research questions based on the goals of each industrial partner in adopting model-driven engineering and how it is applied, b) the Technology Acceptance Model to summarize the perceptions regarding model-driven engineering, and c) a questionnaire to evaluate opinions regarding specific tools and techniques. The method has been useful in identifying criteria for evaluation and has produced interesting results.

Keywords: model-driven engineering, empirical evaluation, technology acceptance model, measurement, usefulness

1 Introduction

There have so far been few empirical studies on Model-Driven Engineering (MDE)[1]. A systematic review performed by us in 2007 managed only to detect 25 papers on applying MDE in industry published between years 2000 and 2007 and most of these were related to small-scale studies [11]. The review also revealed that there is no systematic way in selecting evaluation criteria and the studies take advantage of only a few commonly-applied metrics such as productivity.

This paper reports results of work on evaluating MDE in the European IST project MODELPLEX (MODelling solution for comPLEX software systems) [9], running in 2006-2010. The project's main objective has been developing a coherent infrastructure for the application of MDE to development and management of complex software systems within a variety of industrial domains. The industrial domains here were enterprise business systems, telecommunication, aerospace crisis management systems and data intensive geological systems, all with complex cases to

[1] MDE here generally refers to the development methods that are model-centric.

explore and develop. MODELPLEX aimed to be user-driven; i.e., the industrial partners defined a set of requirements that reflected their business goals, objectives and needs regarding MDE methods and tools. These requirements were basis for developing solutions by tool vendors and for evaluating the success of MDE. In order to evaluate perceptions regarding MDE and its infusion and future usage intensions, we also took advantage of the Technology Acceptance Model (TAM). This paper presents the evaluation method applied in the project, the identified evaluation criteria and how these are related to how MDE is applied. We consider the method useful and applicable in other contexts.

The remainder of the paper is organized as follows. Section 2 describes the motivation and requirements for the evaluation method. Section 3 presents related work while Section 4 describes the evaluation method developed in MODELPLEX. Section 5 is conclusions and future work.

2 Motivation and Requirements

We have had several requirements in defining the evaluation method in the project:

1. Although there are some core concepts in MDE such as extensive use of modelling in different stages of software development, MDE is a generic approach that is applied in multiple ways in organizations. Thus an organization's basic characteristics- current practices and tools, strengths, weaknesses, expertise and culture- can significantly affect why and how MDE is applied. The evaluation method should therefore be context-dependent.
2. Industrial partners may have problems in revealing the details of their development processes and tools while the evaluation method should allow collecting some general experiences and disseminating them to public.
3. As a project funded by EU, MODELPLEX needed to evaluate the exploitation of project results in industry and intentions for future usage as a measure of project success.

The above points led to the definition of an "evaluation plan" in the project that had three distinct parts:

a) *Research questions* that were defined by each industrial partner and reflect their goals with applying MDE. For each research question, relevant scenarios for evaluation and success criteria were defined. Refer to the first requirement above.
b) Evaluating tools by assigning scores by users in order to collect some general feedback and measure the success of individual tools in meeting users' requirements. Refer to the second requirement above.
c) An extended version of the *Technology Acceptance Model* (TAM) that is used for collecting the developers' perceptions regarding MDE and tools, and intensions for future usage. Refer to the third requirement above.

However, these parts are integrated together and the results of each part help interpreting the other results as discussed in Section 4.

3 Related Work

In this section we discuss related work on evaluating methodologies in general, experiences with using TAM in software engineering research and related work on evaluating MDE.

3.1 Evaluation Methods

In [12], related work on evaluating methodologies is discussed in detail. Here we briefly present an overview and discuss the shortcomings.

In a paper by Farbay and Finkelstein, they have divided evaluation methods applied on software design in two major groups [6]:

- Quantitative and comparative methods such as cost-benefit analysis, ROI and information economics.
- Qualitative and exploratory methods such as Value Analysis and Multi-Objective Multi-Criteria methods.

The above methods are usually from the management point of view. In *Empirical Software Engineering (ESE)*, on the other hand, the viewpoint of evaluation is often those of practitioners. Several books and papers are published during the recent years on types of empirical studies, how to perform them, and how to collect evidence from such studies (see [12] for references). However, selecting appropriate evaluation criteria is not discussed in the above literature.

The *Goal-Question-Metric (GQM)* approach developed by Basili and his colleagues [1] addresses the problem of identifying measures when the goals are predefined. GQM starts by expressing the overall goals of the measurement. Then questions are generated whose answers must be known to determine if the goals are met. Finally, each question is analyzed in terms of what measurements are needed to answer the question. GQM is a top-down approach that is useful when we have clear goals for introducing a methodology or product; for example when evaluating the impact of a software process improvement activity. However, GQM is difficult to apply when the improvement goals are not clear or the impact of a new method is unknown.

The *Technology Acceptance Model (TAM)*; originally developed by Davis [3] and later extended by others; is a generic model which can be applied to measure the acceptance and infusion of a technology or system. The original model explains users' intention to use a new system through two beliefs, *perceived usefulness* and *perceived ease of use*. There are several additions to TAM and those that combine TAM with other models, such as the model used by Dybå et al. [4] and depicted in Fig. 1 that includes three additional factors.

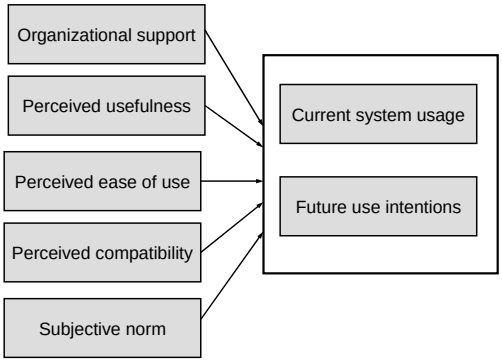

Fig. 1. The conceptual model based on TAM

The factors are defined as:

- *Organizational Support* is the degree a method is supported by an organization.
- *Perceived Usefulness* is the degree to which a person believes that using a particular method will enhance his/her job performance.
- *Perceived Ease of use* refers to the degree to which a person believes that using a particular method or tool would be free of effort.
- *Perceived Compatibility* is the degree to which a method is perceived as being consistent with existing values, principles, practices and the past experience of potential adopters.
- *Subjective Norm* is the degree to which developers think that others who are important to them think they should use a method.

Dybå et al. have applied the above model to evaluate the acceptance and utilization of Electronic Process Guides; both quantitatively using questionnaires and qualitatively using semi-structured interviews [10]. Walderhaug et al. have used TAM to evaluate the use of model-driven software development (MDSD) in the healthcare domain in the MPOWER project [13]. 16 developers from four European countries answered to a survey. The findings suggested that perceived usefulness and ease of use were the most important factors for adopting MDSD. No significant relations between future use intensions and tool performances or subjective norm were found. However, the sample is small and the value of running statistics is limited. Condori-Fernández and Pastor have also applied a modified version of TAM to analyze the acceptance of a model-based measurement procedure [2]. 20 subjects participated in their analysis. The results showed that the intensions for using the measurement method were influenced more strongly by the perceived usefulness than by perceived ease of use, and also usefulness was strongly related to the quality of results.

A popular approach in ESE for selecting measures is to start with a list of metrics from other studies or software metrics literature and select those that seem relevant for evaluating a methodology. A ranking of top three popular measures per phase by a group of experts is presented in [7] and shown in Table 1. While we may rely on expert opinion in selecting relevant measures, the rationale for selecting measures is often not well-documented.

Table 1. Top three popular metrics per phase

Requirements	Design	Implementation	Testing
Fault density	Design defect density	Code defect density	Failure rate
Requirements specification change requests	Cyclomatic complexity	Design defect density	Code defect density
Error distribution	Fault density	Cyclomatic complexity	Coverage factor

In conclusion, all of the above methods may be used to identify evaluation criteria. However, none of them help identifying criteria that is MDE-specific and relates the subject of evaluation (here MDE) to the selected criteria in a systematic way. Models such as TAM are useful in collecting perceptions but they should be combined with more specific measures in order to explain the results.

3.2 Measures for Evaluating MDE in Related Work

In 2007, we performed a systematic review to collect industry experiences with MDE [11]. The motivations behind using MDE are numerous such as improving productivity and quality, maintenance concerns, formalism and more standardization, and improving communication in development teams and with external stakeholders. However the reported quantitative evidence focuses mainly on three aspects:

- Effort and productivity: Effort is time it takes to perform a task, for example creating or changing a model; in person-hours or person-days. Effort is typically used to estimate productivity as output divided by effort.
- Some software quality related measures: One report identified that fewer inspections were required to ensure the quality of the developed code and inspection rates are higher. Simulations were also found to be about 30% more effective in catching defects than code inspections.
- Automation degree: this metric focuses on the artefacts that are generated from models in the number or size, relative to all the artefacts.

Productivity and defect detection rate are examples of popular metrics while automation degree is specific to MDE.

In addition to analyzing quantitative data, we analyzed the reported experiences. The discussions focused mainly on the learnability of MDE and the stability and maturity of the tools since many of the techniques promoted as necessary in MDE strongly depend on proper tool support. We concluded that there is a need to identify MDE-specific measures and perform empirical studies to evaluate them.

4 The Evaluation Method in MODELPLEX

As discussed before, the evaluation method in MODELPLEX is developed with several goals in mind: It should include evaluation criteria related to how MDE is applied, be relevant for the industrial partners and meet the needs of a research project. The evaluation method developed in MODELPLEX consists therefore of

three parts: research questions for empirical evaluation of MDE in the industrial contexts, the extended TAM to evaluate perceptions and future usage intensions, and scores assigned by users to the tools and techniques developed in MODELPLEX. We explain these in the following sections

4.1 Evaluation Criteria Identified by Research Questions

In [12], we presented an approach for identifying relevant evaluation criteria that is bottom-up and starts from analyzing a methodology or tool; in contrast with GQM that is top-down and starts with defining improvement goals. The approach is called *Methodology-Practices-Promises-Metrics* (MPPM). These are defined as:

- *Methodology* is the subject of evaluation; such as MDE.
- A *Practice* of a software development methodology (or technology or tool) is a new concept or technique or an improvement to established ones that is an essential part of the methodology and differentiates it from other methodologies. We may also call it a *core practice*.
- A *Promise* is the expected improvement that is given as the main motivation for applying a practice. It is the expected benefit which often comes with a cost.

The core practices and promises of MDE are identified as:

1. *Models Everywhere*: models are primary software artefacts in all or most stages of software development. More effort will be spent in MDE on modelling and activities related to modelling such as defining modelling languages and quality verification of models than in traditional software development based on source code. The main promises related to extensive use of models are improved communication between stakeholders and improved software quality by using models for early analysis and testing. On the other hand, modelling has a cost and modelling tools must be integrated with other tools such as configuration management tools.
2. *Multiple Abstraction Levels and Separation of Concerns in Models*: Abstraction is the main technique to handle complexity of software development. The main promises (benefits) of abstraction are improved communication due to the separation of concerns, improved software quality since developers focus on one aspect of development at a time, and portability of solutions if models are defined as platform-independent. On the other hand Mellor and Balcer refer to several challenging issues that inevitably arise from the multi-view and multi-notational approach in MDE, such as keeping models consistent with one another [8].
3. *Generating Artefacts from Models:* Generation of artefacts from models is the key technology to achieve automation and reduce manual work. Generation is done through transformations; either Model-to-Model (M2M) or Model-to-Text (M2T). During transformations, output models are supplied with information not present in the input models. An example of such information is the platform concept. Generation actually supports separation of concerns and adding details later; not by manual work but by applying transformations. The main promises of generation are less manual work, consistency and traceability between artefacts and improving

the quality of models and other artefacts such as their syntactic correctness and completeness. The cost relies in developing transformations.

4. *Metamodeling*: The concepts of metadata, OMG's Meta Object Facility[2] (MOF) and the MOF-like Eclipse's metamodel[3] (Ecore) allow definition of new modelling languages or extending the existing ones; for example as Domain Specific Languages (DSLs) or UML profiles. Sharing the same language between domain and IT experts and narrowing the gap between them, and involving domain experts in all stages of design are some of the promises of developing DSLs or UML profiles. Additionally, the practice of metamodeling allows defining relations between metamodels or instances of them and exchanging models between tools; thus achieving interoperability between tools. However, defining metamodels and supporting tools requires high initial investment and needs language and tool expertise.

The above view of MDE and its practices and promises allows identifying evaluation criteria related to the practices and their promises. In addition to the MDE-specific criteria, criteria such as understandability, ease of use, compatibility with other tools and processes, tool maturity and scalability for large systems are general concerns. Table 2 shows examples of what is evaluated in MODELPLEX and how.

Table 2. Identifying evaluation criteria

Subject of evaluation	**Related to MDE practice**	**Evaluation criteria in research questions**	**Evaluation method**
Modelling framework based on a metamodel for defining architectural models in different views	Metamodeling, Multiple concerns,	Does the framework support separation of concerns?	Apply on appropriate scenarios.
		Are the views expressive enough?	Check for criteria: including necessary concepts, modelling dependencies between views,
Using models for performance simulation	Models everywhere, Metamodeling	Are the performance results comparable with the actual performance of the system within +/-25%?	Exploratory case study and comparison
		Is it possible to integrate the performance modelling with testing tools?	Interoperability via XMI should be evaluated in a scenario.
		How complex is performance modelling (ease of learning)?	The concepts should be learnt in less than 4 hours in an experiment.
DSL for network modelling	Metamodeling, Generating artefacts from models	How efficient is the model transformation process?	Time taken in writing transformations in a case compared between tools.
		How readable is the generated code?	Compare with manually written code. Do the artefacts need any post-processing?

[2] http://www.omg.org/mof/

[3] http://www.eclipse.org/modeling/emf/?project=emf

The full list of the evaluation criteria identified in the project and the classification of them according to TAM factors is given in Appendix I. This classification is helpful for using the results of research questions when interpreting the TAM results.

4.2 The Applied TAM Model

While research questions are case-specific, the model depicted in Fig. 1 with some modifications is used to collect perceptions regarding MDE and some development environments such as Eclipse. The modifications to model in Fig. 1 are:

- *Organizational support* was dropped since the involved organizations are so-called "early adopters" and their participation in the research project is voluntary.
- *Perceived tool Maturity (PM)* is added which is defined as the degree to which tools are perceived as mature and suitable for the tasks in hand. A similar factor is used in [13].

We performed semi-structured interviews in the beginning of the project to map the state of the practice before project, and an on-line survey at the end of the project to collect data regarding perceptions, the state of actual usage and future intensions of use. Some examples of questions in the survey are (quite similar to [4] and [13]):

- I would like to use the MDE approach in the future for my work.
- People who are important to me think I should use the MDE approach.
- The MDE tools I use are suitable for both small and large projects.

The responses were often on a five scale rate; varying from "strongly disagree" to "strongly agree". The results showed that two companies had only used MDE on an experimental basis before project while two others had more experience with MDE. All the companies had experience with modelling for the purpose of analysis and design while practices of MDE were applied in varying degrees. For example, most companies had ad-hoc code generators or generators integrated in their development environment for generating skeletons and stubs, user interfaces and some test cases. Most MODELPLEX tools have been tried during the project with different levels of success. MDE is generally perceived as useful for solving the problems of users while it is not perceived as easy to learn. Regarding tool maturity, performance and functionality of tools were generally perceived as satisfactory, while scalability to large projects is not perceived as well enough. We also asked about the intensions for future usage which shows that all companies are interested in using MDE in their future work although not strongly agreed.

4.3 Giving Scores to the Tools

The on-line survey used at the end of the project included questions regarding each tool developed in MODELPLEX and some external tools and technologies that were widely used such as Eclipse. Here we asked participants to give scores from 1 to 5 (where 5 are best) to the tools in five dimensions: functionality, ease of use, compatibility, performance and reliability, and total impression. We also asked them

whether they intend to use the tool in their future work. These scores indicate the degree of satisfaction of users and provide feedback to tool vendors.

4.4 Discussion

The evaluation method provides several sources of data to analyse: the identified criteria from case-specific research questions as summarized in Appendix I, results of the TAM survey regarding perceptions and future usage intensions, and results for each specific tool. Thus we combine observations with perceptions and quantitative data with qualitative ones, in order to explain the results and intensions for future usage. The scores given to individual tools provide also feedback to tool providers.

The evaluation criteria identified by research questions are classified according to TAM factors to allow describing the TAM results. For example we discovered concerns with tools in research questions (such as lack of multi-user support and problems in integration with existing tools) that explain why some companies do not perceive tools as mature enough for industrial cases. Another example is that all companies express their intension for using MDE in their future work in the survey. However, the research questions show that MDE will be applied for different purposes. For example, one company is interested in using models for simulation and testing while the other is more interested in modelling different concerns and integrating them. These findings support our earlier statement that evaluation criteria should be selected related to how MDE is applied and for what purposes.

Some threats to the validity of the results are:

- Construct validity is concerned with whether the selected measures reflect the intervention and effects; i.e., “right metrics”. We mean that applying the MPPM approach improves the construct validity.
- External validity of the results should be discussed to evaluate whether the results are generalizable to other contexts. Research questions are case specific and generalization to other contexts may be difficult.
- A methodology such as MDE impacts different constructs, for example improving communication while increasing the complexity of software development. If the method is not evaluated in multiple dimensions, there is a risk that conclusions are drawn based on one measure and ignoring others, which is also a threat to the construct validity of the results. The approach proposed here takes advantage of multiple criteria and several data sources and thus reduces this risk.
- Several research questions are answered by experts and based on their experience and expertise. Involving different people in the evaluation can reduce the threat of subjective judgements.
- The companies could not run comparative case studies which may be regarded as a threat to the internal validity of the results.
- The expectations of companies participating in a research project may impact the results. This threat can be reduced by discussing these expectations in the planning phase.
- In the interviews, we asked companies which MDE tools and techniques they were using or planning to use in MODELPLEX. These plans were however modified

during the project based on the availability of tools, effort required to evaluate them, and the changing priorities. For example, it showed that extracting models from legacy code and analysing them required a lot of collaboration between industry and tool developers and the companies had to open their source code to external actors. This subject was therefore abandoned and we could not evaluate the success of MDE regarding integration of legacy systems and new applications.
- The number of participants in the TAM survey was too low to extract meaningful statistical results. The results are therefore analysed and summarized qualitatively as experiences from four industrial cases, as also done in [10].

5 Conclusions and Future Work

The evaluation method described in this paper has several contributions:

- Context-dependent measures are identified while some common criteria are also applied for comparison.
- Two types of feedback are provided: The *observed values* based on research questions and the *perceived values* based on the survey, which can explain one another.
- One of the major requirements of EU projects is to measure project success by infusion of the results in industry. We have mapped the state of the practice before the project and intensions for future usage in order to evaluate the project success.
- Several validity threats are reduced by taking advantage of multiple data sources and multiple evaluation criteria.

Improving the evaluation method for MDE is considered as a contribution of the project which can be reused in other projects. While earlier studies focused on few metrics with no relation to the MDE practices, our method allows relating evaluation criteria to the applied practices.

Another advantage of the method is that it provides several sources of data to analyse from multiple methods of evaluation: case studies including qualitative (expert judgements) and quantitative data (metrics), interviews and survey. Some results of evaluation are already published (as in [5]) while we plan to publish more in future. Future work will cover applying the method in other projects and collecting more empirical results.

Acknowledgement. This work has been done in the MODELPLEX project (IST-FP6-2006 Contract No. 34081), co-funded by the European Commission as part of the 6th Framework Program.

References

1. Basili, V.R., Weiss, D.: A Methodology for Collecting Valid Software Engineering Data. IEEE Trans. Soft. Eng. 10(6), 728—738 (1984)

2. Condori-Fernández, N., Pastor, O.: Analyzing the Influence of Certain Factors on the Acceptance of a Model-based Measurement Procedure in Practice: An Empirical Study. In: Empirical Studies of Model-Driven Engineering (ESMDE'08), pp. 61—70 (2008)
3. Davis, F.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly 13(3), 318--339 (1989)
4. Dybå, T., Moe, N.B., Mikkelsen, E.M.: An Empirical Investigation on Factors Affecting Software Development Acceptance and Utilization of Electronic Process Guides. In: Proc. of the Software Metrics, 10th International Symposium, pp. 220--231 (2004)
5. Evans, A., Fernández, M.A., Mohagheghi, P.: Experiences of Developing a Network Modeling Tool Using the Eclipse Environment. Model Driven Architecture - Foundations and Applications (ECMDA-FA 2009), LNCS 5562, pp. 301–312 (2009)
6. Farbay, B., Finkelstein, A.: Evaluation in Software Engineering: ROI, but More than ROI. In: 3rd International Workshop on Economics-Driven Software Engineering Research (EDSER-3) at ICSE 2001, online at http://www.cs.ucl.ac.uk/staff/A.Finkelstein/papers/edser3eval.pdf (2001)
7. Li, M., Smidts, C.S.: A Ranking of Software Engineering Measures Based on Expert Opinion. IEEE Trans. Soft. Eng. 29(9), 811—824 (2003)
8. Mellor, S.J., Balcer, M.J.: Executable UML: A Foundation for Model-Driven Architecture. Addison-Wesley (2002)
9. MODELPLEX website; http://www.modelplex.org/
10. Moe, N.B., Dybå, T.: The Adoption of an Electronic Process Guide in a Company with Voluntary Use. In: Software Process Improvement, LNCS vol. 3281, pp. 114--125 (2004)
11. Mohagheghi, P., Dehlen, V.: Where is the Proof? - A Review of Experiences from Applying MDE in Industry. In: European Conference on Model Driven Architecture Foundations and Applications (ECMDA 2008), LNCS vol. 5095, pp. 432—443, Springer (2008)
12. Mohagheghi, P.: Evaluating Software Development Methodologies Based on their Practices and Promises. In: New Trends in Software Methodologies, Tools and Techniques - Proceedings of the Seventh SoMeT, pp. 14—35 (2008)
13. Walderhaug, S., Mikalsen, M., Benc, I., Erlend, S.: Factors Affecting Developers' Use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project. In: From Code Centric to Model Centric Software Engineering: Practices, Implications and ROI. Workshop at European Conference on Model-Driven Architecture (2008)

Appendix I

An overview of the evaluation criteria identified in research questions and classified according to TAM factors is given below. The term "solution" generally refers to the developed tools, methodologies, languages or techniques.

Table I. Evaluation criteria related to the usefulness of solutions

Criteria	Definition
Architecture model quality	Whether the architecture model satisfies criteria such as support for separation of concerns and integration of several viewpoints.
Design quality	The solution improves the quality of design by identifying poor design.
Quality of generated artefacts	The quality of code, documentation etc. that are generated from models is acceptable (understanble, compliant with coding standards etc.).
Solution suitability	The solution can solve the problem in hand.
Generation ratio	The number or size of the generated elements divided by the total number of elements or size. The question is whether the saving by generation can compensate

Criteria	Definition
	for modelling effort.
Simulation model accurateness	The percentage of inaccuracies between the predicted results of a dynamic simulation compared with the actual performance behaviour of the running system.
Simulation model correctness	The correctness of the boundaries of time behaviour (lower and upper boundaries) and critical paths detected by analytic simulation compared with the actual performance behaviour of the dynamic simulation or of the running system.
Reverse-engineered models usability	A reverse-engineered model will be usable if it can be used in the same way as any other model manually created (with minimum effort).
Model completeness	The model is complete for the purpose of generating specified artefacts. MODELWARE defines "model well-formedness and completeness".
Code readability	The automatically generated code must be clear, well formatted and adequately commented. It should be consistent with a code convention.
Test coverage	The percentage of requirements covered by generated test cases.
Effort spent on development	This can be modelling effort, coding effort or editor development effort in case of DSLs compared to non-MDE.

Table II. Evaluation criteria related to the ease of use of solutions

Criteria	Definition
Model understandability	The developed models are easy to understand for different stakeholders, measured in effort needed to understand a model.
Learnability of solution	The time a user needs to achieve a specified proficiency level with a solution. Documentation is complete and understandable. Tutorials or discussion forums are available.
Effectiveness	Effort required by users to solve a task after learning a solution.

Table III. Evaluation criteria related to the compatibility of solutions

Criteria	Definition
Cost of adoption	Effort required for setting up a tool and customizing it.
Integration with other solutions	The degree a solution can be integrated with other practices or tools.
Standards compliance	A solution must conform to selected standards.

Table IV. Evaluation criteria related to the maturity of solutions

Criteria	Definition
Scalability	The solution is scalable with respect to an increasing number of model elements. The explosion of the number of model elements must not compromise the efficiency. Synchronisation between models and traceability are also issues that need to be handled for large models.
Efficiency	Time required by a tool to perform a task (such as model composition, generating models or code during a transformation).
Transformation development effort	The effort required to write and deploy a model transformation.
Preservation of properties	Model properties are preserved during an action such as composition and transformation.
Ease of change	The effort required to do a change and generate required models or assets.
Traceability	It is possible to define trace links between models and with generated artefacts.
Multi-user support	A tool should allow simultaneous usage by multiple users that work on different artefacts.
Change management	Functionalities such as history, diff and merge are required.