# Where is the Proof? - A Review of Experiences from Applying MDE in Industry

Parastoo Mohagheghi[1], Vegard Dehlen[1]

[1]SINTEF, P.O. Box 124- Blindern
N-0314 Oslo, Norway
{Parastoo.Mohagheghi, Vegard.Dehlen}@sintef.no

**Abstract.** Model-Driven Engineering (MDE) has been promoted as a solution to handle the complexity of software development by raising the abstraction level and automating labor-intensive and error-prone tasks. However, few efforts have been made at collecting evidence to evaluate its benefits and limitations, which is the subject of this review. We searched several publication channels in the period 2000 to June 2007 for empirical studies on applying MDE in industry, which produced 25 papers for the review. Our findings include industry motivations for investigating MDE and the different domains it has been applied to. In most cases the maturity of third-party tool environments is still perceived as unsatisfactory for large-scale industrial adoption. We found reports of improvements in software quality and of both productivity gains and losses, but these reports were mainly from small-scale studies. There are a few reports on advantages of applying MDE in larger projects, however, more empirical studies and detailed data are needed to strengthen the evidence. We conclude that there is too little evidence to allow generalization of the results at this stage.

**Keywords:** Model-driven engineering, quality, productivity, evidence.

## 1 Introduction

The model-driven approach has received considerable attention this decade. The OMG's Model-Driven Architecture (MDA) initiative, Model-Driven Development (MDD) or Model-Driven Engineering (MDE)[1] has been hailed as the solution to handle the key problem facing the software development industry; increasing complexity, by (1) providing better abstraction techniques and (2) facilitating automation. By switching to a MDE approach, businesses are promised to reap benefits through increased productivity and software quality [26].

The motivation behind this paper is that even though many promises are made, these are in most cases poorly, if at all, supported by evidence. During recent years we have witnessed the surfacing of attempts to evaluate practices and benefits of

[1] In the remainder of the paper we use MDE to refer to a model-driven software development approach, also where MDD is used in the papers.

MDE through empirical studies; including experiments and industry experience reports. This paper, the result of an extensive literature review, contributes to the state of evidence in MDE by gathering the individual evaluations and providing a detailed overview of industry's experiences with MDE.

The remainder of the paper is organized as follows. Section 2 presents the review framework and the three research questions leading the review, the strategy used for literature search, the publication channels, and an overview of the reviewed papers. Section 3 through 5 reports our findings, before Section 6 summarizes and concludes the paper.

## 2 The Review Process and an Overview of Papers

### 2.1 The Review Framework and Research Questions

We follow the review framework presented in [19], adopted to this review and depicted in Figure 1. The formulation of the review questions follows recommendations by Dybå et al. for collecting evidence as answer to questions. Questions should be well-partitioned into *intervention*, *context* and *effect* [9]. In this review, the intervention is "MDE" (vs. non-MDE approaches), the context is "industrial settings" and the effects are "changes in productivity and quality, or cost savings".

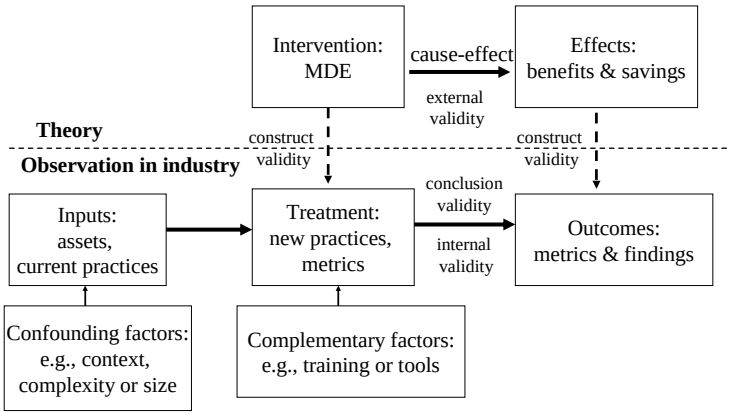


**Fig. 1.** The review process.

To understand the intervention and context, we ask the following Research Questions (RQs):

- *RQ1. Where and why is MDE applied?*
- *RQ2. What is the state of maturity of MDE?*

And to evaluate the effects, we ask:

- *RQ3. What evidence do we have on the impact of MDE on productivity and software quality?*

### 2.2 An Overview of the Reviewed Papers

We searched the following publication channels for industrial studies related to MDE:

- The Software and Systems Modeling (SoSyM) journal from 2002 (the first issue).
- The Empirical Software Engineering journal since 2000.
- Proceedings of the UML conference from 2000 to 2004, succeeded by the MODELS conference to 2006.
- Proceedings of The European Conference on MDA- Foundations and Applications (ECMDA-FA) started in 2005, and to 2007.
- Proceedings of the DSM workshops at OOPSLA since start in 2002.

We also performed a search by keywords in the IEEE Xplore, the ACM digital library and the Internet. A few additional papers were discovered through references in the detected papers. The review identified 33 papers and reports (generally called papers). From these, we excluded 8 papers with claims on industrial application but no description of the application (a list can be provided by the authors). This left 25 papers for the review.

It was not possible to extract information on the size of projects from the majority of papers. For appraising the evidence, we asked what types of studies were performed (see [19] for a definition of study types). We concluded that:

- 20 of papers are experience reports from single projects with description of a project or development method and discussion of experiences [1-7, 8, 10, 11, 13, 20-24, 27-29 and 31]. Of these, only two include some quantitative data from the projects (both from Motorola).
- Three papers have used interviews and questionnaires in addition to observations [24-26].
- Three papers describe comparative studies (comparing projects or development of components with each other) [12, 14 and16]. From these, [12] provides no quantitative data.
- One paper describes three (quasi)experiments [16].

Only seven papers report experiences from completed projects [1, 3, 6, 13, 25, 27 and 29], while the others are from pilot studies or ongoing projects at the time of reporting, and one is from a terminated project [ABB Robotics in 26].

When it comes to publication channels, 13 papers are published in the proceedings of conferences (especially the ECMDA-FA conference), 9 papers in workshops and satellite activities of conferences, two are online reports and only one is published in a journal.

## 3 Where and why is MDE Applied (RQ1)?

A broad range of companies in various domains report their experience from investigating or applying MDE. To name some, the papers cover:

- Telecommunications domain [2, 3, 16, 21, 26, 28 and 29].
- Business applications and financial organizations [1, 7, 8, 16 and 24].
- Defense / aerodynamics / avionic systems [5 and 11].
- Web applications [6 and 14].

We found examples of safety-critical and trustworthy systems [5, 11 and 27] and embedded systems [23 and 27]. MDE approaches are also applied to software product lines as in [2, 10 and 27]. In connection with legacy systems, Bloomfield reports successful remodeling of a component [5] and Raistrick reports developing new components that were integrated with existing components [22]. On the other hand, ABB Robotics refrained from adopting MDE due to the base of legacy code [26].

Regarding motivations for evaluating or applying MDE, the papers discuss:

- *Increasing productivity and shortening development time:* as in [12, 14, 16, 25, Ericsson in 26 and 29].
- *Improving quality*: improving the quality of the generated code [25, 27 and 29], improving the quality (assurance) of system requirements [4] and managing requirement volatility [22], improving the quality of intermediate models [4], and earlier detection of bugs [12, 27 and 29].
- *Automation*: generating code and other artifacts and introducing automation into the development process [1-3, 6, 7, 8, 11-13, 16, 21, 23 and 27], and model-based simulation and testing [3].
- *Standardization and formalism*: providing a common framework for software development across the company and phases of the lifecycle [2, 24 and 25], formalize and organize software engineering knowledge at a higher level of abstraction [29], and common data exchange format [20].
- *Maintenance and evolution concerns:* maintaining the architecture intact from analysis to implementation [25], evolution of legacy systems [12], concerns over software method and tool obsolescence [5], verification of system by producing models from traces [28] and that PIMs have long lifespan [14].
- *Improved communication and information sharing:* between stakeholders [18 and 24] and within the development team [12, 26 and 27] and ease of learning [27 and 29].

Additional motivations are *traceability throughout software development artifacts* [17 and 26], *early assessment* [22 and 26], *promoting reuse* [2, 18, 24 and 29], *porting of solutions to new platforms* [12 and 13], and *the ability to estimate costs based on the models* [22 and 26].

On the above list, increasing productivity (and shortening development time) and improving quality may be regarded as the ultimate reasons for applying MDE. The other items, on the other hand, are basically means towards these two ends.

## 4 What is the Experienced Maturity of MDE (or, the-State-of-MDE) (RQ2)?

In this section we present findings related to the current state of practicing MDE. It covers automation as a key means to achieve the MDE benefits. We also discuss the state of software development processes and tools for MDE.

### 4.1 Level of Automation

By using transformations the MDE approach emphasizes generating models, code and other artifacts from models, in addition to verification and validation on the model level. In this section we analyze to what extent this is possible in the presented contexts and with the current state of tools

**Automatic generation of code.** While some papers report generating all or most of the code from the models [5 and 6], others report that only part of the code could be generated. Motorola evaluates the potential of MDE in generation to be between 65 to 96 percent depending on the type of the code (low level code is not captured in the design and is unlikely to be generated), and perceives the status of code generators as satisfactory in producing code with no introduced defects [3 and 29]. Automatic generation of code required developing Domain Specific Languages (DSLs) or UML profiles and own code generators in several cases, as in [1, 3, 7, 10, 21 and 29].

**Generating XML schemas**. In [20] a metamodel was implemented as a UML profile and the needed XML schemas were generated directly from the marked PIM models. [2]'s toolset also includes an XML schema generator, a code generator using the schemas and other outputs. In the case of [10], the developed framework included a XML schema generator, HTML documentation generator and a model browser.

**Automation of Testing**. In Motorola, by using TTCN scripts, 90% of the tests are automated which has led to a 30% reduction in box-test cycle time [29].

**Executable models**. A few papers have discussed that developing executable models is still a challenge. Deng et al. write that they used Visio as a static design tool, while a dynamic provisioning tool is desired to make the blocks executable [8]. MacDonald et al. report difficulties in specifying behavior using Telelogic Tau and that they could not develop executable models [12].

### 4.2 Software Processes

The importance of utilizing a defined process in software engineering has been known for several years. However, most "tried and tested" processes are not tailored for MDE, which does not make any assumptions on the software development process or the design methodology. Baker et al. report that many teams in Motorola encountered major obstacles in adopting MDE due to the lack of a well-defined process, lack of necessary skills and inflexibility in changing the existing culture [3]. Also, MacDonald et al. write that there is no well-defined process for developing non-trivial MDE components, especially when these are part of legacy systems [12]. Staron

means that there are two reasons for why they currently find it unrealistic to purely use a MDE process [26]:

1. Software engineering methods are not fitted to use models as main artifacts, i.e. activities such as analysis and evaluation is still largely done at the code level.
2. Software engineering environments are not mature enough.

Some have attempted to apply pre-existing software processes to MDE, such as using a modified version of the Rational Unified Process (RUP) [24], and combining agile methods and MDE [23, 27 and 30]. Others have attempted defining processes for MDE. Firstly, THALES has defined a MDE process by extending the IEEE 1471 standard [11]. Secondly, Biffl et al. propose an iterative software development lifecycle, which includes creating models with explicit stakeholder requirements, a first quality assurance (QA) step with type checking and semantic validation and transforming these into intermediate models, and a second QA step with static validation of models [4]. Thirdly, Staron et al. discuss that raising the abstraction level and employing automatic code generation moves the complexity of software development to transformations [25]. An MDE process should consequently prioritize defining transformations before defining profiles, since profiles are considered a means of making the transformations automated. The importance of developing transformations early is further supported by [21]. None of the studies report using any of the already existing – although few – model-based methodologies, e.g. KobrA[2] or COMET[3].

### 4.3 Tools

Supporting MDE with a comprehensive tool environment is crucial, as many of the techniques promoted as necessary in MDE strongly depend on proper tool support. A survey performed among industry participants (presented in [26]) showed that, when considering whether or not to adopt MDE, the availability of tools was perceived as the most influential factor. However, a tool chain has to integrate the various tools for software development (e.g., requirements management, modeling, model transformations, traceability, simulation, validation and testing [15]), support multiple platforms and domain-specific design [12] and the possibility to generate *correct* code by adding constraints and rules [1, 13, 27 and 31].

Integrating a tool suite that satisfies these requirements into a coherent environment is evidently a challenge. In the MODELWARE project, a wide range of tools were used, but all partners experienced problems with instability of the tools and their integration [15 and 17]. Also according to Motorola, third-part MDE tools do not scale well to large system development [3]. Safa writes that using third-party tools raises questions of suitability for the product, adaptability to new platforms, availability over time, and loss of differentiation factors since competitors may use the same tools [23].

[2] www.old.netobjectdays.org/pdf/02/papers/node/0308.pdf

[3] http://www.uio.no/studier/emner/matnat/ifi/INF5120/v05/undervisningsmateriale/COMET_Method_v2-4.pdf

The vendor lock-in problem persuades some users to use open source tools such as the Eclipse framework. Others combine third-party products with self-developed tools [27 and 29], or develop their own tools [2, 4 and 10]. Having to invest time and effort into the development and maintenance of an MDE tool chain raises issues of cost. France Telecom calculated that the cost for creating their tool chain in MODELWARE was approximately one person-year in terms of resources, in addition to approximately 0.4 person-year for maintenance [15].

## 5 What Evidence do we have on the Impact of MDE on Productivity and Software Quality? (RQ3)

Productivity and software quality gains are often given as main motivations for selecting new technologies, and most papers in this review include discussion of either one or both of the aspects. In this section, we present the reported data, observations and explanations on observations.

### 5.1 MDE Impact on Productivity

Three of the papers in the review report results from comparative studies on productivity (i.e., developing a product twice or comparing with company baseline data), although the studies are of small-scale.

Firstly, in a report from 2003, the Middleware Company, on behalf of Compuware, conducted a comparative case study on the productivity of MDA [14]. Two teams developed the same application, one using MDE and the other using a non-MDE approach. The result was that the MDE team developed their application 35% faster than the other team – needing 330 hours compared to 507,5. It is worth noting that the MDE team used a tool with pre-made transformation mappings, which relieved them of potential work. On the other hand, this was the developers' first experience with MDE and related tools, which would presumably hamper their productivity. Issues like application performance and maintenance were not evaluated.

Secondly, we have the results of the EU IST project MODELWARE[4] [16]. In September 2006, results from six small-scale case studies and (quasi)experiments performed by five industrial partners were disseminated. When it comes to productivity, the results are differing:

- In WM-Data (desktop business applications), two developers re-implemented a subset of requirements and the effort was compared to some baseline data. The productivity gain was on average 24% using MDE.
- WesternGeco (oil and gas exploration) performed an experiment with 24 developers who were given four tasks – two involving a traditional development process and two involving MDE. Only eight subjects finished the experiment due to problems with the MDE tooling and complexity of the tasks. The results show no difference in productivity between the two approaches.

[4] http://www.modelware-ist.org

- A team of two developers from Enabler (specialist in creation and integration of IT solutions for retailers) developed a module twice over a period of approximately 300 hours. The results show an overall loss in productivity when using MDE by 27%. When discounting the problems with the use of immature tools, the loss in productivity was 10%.
- France Telecom measured the effort needed to specify, implement and change five different functional units, normalized by the weight of their complexity, and compared to the data on effort spent in a non-MDD approach. A productivity gain of 20% was measured during design activity and 69% during coding. This observed productivity gain does not take into account the cost of the development of tool chain.

The third paper reports redevelopment of a small component of a legacy system using MDE [12]. The authors report that there is no proof that development speed is improved, especially with the workarounds required to integrate with legacy systems.

A few other papers have reported productivity gains in single projects when applying MDE, without having a clear baseline or providing detailed data. Firstly, Motorola has employed a MDE approach for more than 15 years and has shipped millions of lines of code based on MDE [3]. All in all, they have experienced a 2X–8X productivity improvement when measured in terms of equivalent lines of source code. These numbers are all approximates, as Motorola is lacking a common baseline. Also, an experience report by Trask et al. deals with the application of a combination of software product line and MDE techniques to the "software defined radio" domain [27]. The programmers reportedly experienced a 500% productivity gain, minimum, by utilizing their domain-specific modeling tool. These results are based on experiences and are not validated by data or experiments. And finally, Thales Air Traffic Management (TATM) in the MODELWARE project estimated 5 to 25% productivity gains based on the assumption that a certain type of defect (interface mismatch) cannot occur because of the MDE process.

The industrial papers that reported productivity gains accredited the improvement to automatic code generation [3, 14, 27 and 29], model-based simulation and testing [3, 15 and 29], automatic test generation [3, 29], avoiding defects [27 and TATM in 16], domain-specific languages [27], and reuse of design and test between platforms or releases [29].

As discussed above, there are also reports of productivity loss. The main reasons are mentioned to be immature tools and high start up costs [Enabler in 18], and that modeling can be at least as complex as programming with a traditional third generation language [12].

### 5.2 MDE Impact on Software Quality

Among industry adopters discussing improvements in software quality due to MDE, the key experienced benefit is a drastic reduction in the number of software defects. However, there are not much quantitative data presented in the papers.

Firstly, we discuss the Motorola case. Weigert and Weil write that with MDE, there are fewer inspections required to ensure the quality of the developed code than using conventional development. In addition, inspection rates are higher and have

increased from 100 source lines per hour to in between 300 and 1000 source lines per hour [29]. Motorola data also shows that simulation is about 30% more effective in catching defects than the most rigorous inspections, and that defects are detected earlier in the software development lifecycle. They expect a 3X reduction in defects, which is backed up by an earlier Motorola study, experiencing "a 1.2X–4X overall reduction in defects and a 3X improvement in phase containment of defects". Baker et al. write that it is not unusual to see a 30X–70X reduction in the time needed to correctly fix a defect by detecting and correcting the problems at the model level [3].

That models are verified through simulation (or other techniques) and checked for completeness also improves quality significantly according to [15 and 29]. In [15], France Telecom writes that being able to validate the specification using simulation, allows them to "to eliminate uncomfortable ergonomics that would be difficult to detect otherwise".

## 6 Summary and Conclusions

This review examined experiences of applying MDE in industry published since 2000, showing the status as it is and identifying gaps for future research. Validity threats are identified to be:

- The low number of studies is the main threat to the external validity of the results (i.e., generalization to a population or theory).
- Success cases are more likely to be published than failures.
- Some companies may refrain from publishing their results to keep their competitive advantage.
- Projects with external financing, such as EU projects, may report biased results. However, in the case of the MODELWARE project, we know the details of the studies and do not consider this as a threat to the validity of the results.
- There are few results of large-scale studies and the scalability of MDE to large system development should be evaluated in more cases.
- There is a lack of baseline data in most companies, which results in subjective evaluations.
- Most studies do not include enough quantitative data or the metrics are not properly defined.

Due to the low number of experiments, we do not discuss experimentation validity threats in more details. Finally, we mainly searched journals and conferences that have a review process and are considered relevant to our subject, in addition to including two on-line reports [14 and 16]. Additional search in other publication channels may add new papers which can extend the results of this review.

We asked three research questions and the findings are summarized here:

- *RQ1-Context and motivation*. MDE is applied in a wide range of domains; including safety-critical systems and product lines. MDE is assumed to lead to higher productivity (by increased automation in the development process), increased standardization and formalism, and improved communication within

development teams and with external stakeholders, to name the most frequently given benefits. Labor-intensive and error-prone development tasks are automated and best-known solutions can be integrated in code generators, resulting in reducing defects and improving software quality.

- *RQ2-State-of-the-MDE*. The current state of MDE is far from mature. There is a varying degree of automation and it is mostly applied for code generation. Examples of using models for simulation and test generation are also given. Tools are improved during the recent years but several papers still discuss the lack of a coherent MDE environment and tool chain. Tools should scale to large-scale development and support the domain-specific approach more effectively. Software processes should also be adapted to MDE. Other challenges in adopting MDE are the complexity of modeling itself, developing PIMs that are portable to several platforms and using MDE together with legacy systems.
- *RQ3-MDE impact on productivity and software quality*. We found some quantitative evidence on productivity gains in the Motorola context [3 and 29], from a domain-specific environment [27], and three small-scale comparative studies and quasi-experiments described in [14 and 16]. The Motorola studies are the only ones providing some quantitative data on software quality improvements. Software quality benefits are discussed in several papers but are not backed up with data.

Modeling should be easier and faster than code writing to promote MDE. Appropriate tools and processes and increased expertise on modeling are areas for improvement in most cases. Combining MDE with domain-specific approaches and in-house developed tools has played a key role in successful adoption of the approach in several cases. One of promises of MDE in increasing portability of solutions to multiple platforms has not often been feasible, mainly due to the fact that tools are bound to specific platforms. However, most papers evaluate models as useful for improving understandability and communication among stakeholders.

It is a challenge to collect convincing proof on any technology – MDE included. Future work for evaluation of MDE should focus on performing more empirical studies, improving data collection and analyzing MDE practices so that success and failure factors and appropriate contexts for MDE can be better identified. Future research should also cover evaluating Return-On-Investment (ROI) of MDE in various contexts and for different project scales. We only found an estimation of ROI in France Telecom which provided an estimation based on costs related to the training and tool chain setup and the measured productivity gain [15]. High initial investment and unsure benefits were one of the issues influencing the decision of the non-adopters [26]. In the MODELPLEX project[5], we continue the MODELWARE approach in combining research with industrial application and evaluation and will report the results of research on applying MDE in large and complex system development on the project website and in future publications.

**ACKNOWLEDGMENTS.** This research was done in the "Quality in Model-Driven Engineering" project[6] at SINTEF. We thank Dr. Arnor Solberg and Mr. Tor Neple for their comments and constructive criticism.

[5] European IST-34081, https://www.modelplex.org/

[6] http://quality-mde.org/

## References


1. Anonsen, S.: Experiences in Modeling for a Domain Specific Language. In: Satellite Activities at the Unified Modeling Language, 7th International Conference (UML 2004), LNCS, vol. 3297, pp. 187--197, Springer (2004)
2. Bahler, L., Caruso, F., Micallef, J.: Experience with a Model-Driven Approach for Enterprise-Wide Interface Specification and XML Schema Generation. In: Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03), pp. 288--295 (2003)
3. Baker, P., Loh, P.S., Weil, F.: Model-Driven Engineering in a Large Industrial Context - Motorola Case Study. In: ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML 2005), LNCS, vol. 3713, pp. 476--491, Springer (2005)
4. Biffl, S., Mordinyi, R., Schatten, A.: A Model-Driven Architecture Approach Using Explicit Stakeholder Quality Requirement Models for Building Dependable Information Systems. In: 5th International Workshop on Software Quality (WoSQ'07) at ICSE'07, IEEE, 6 p. (2007)
5. Bloomfield, T.: MDA, Meta-Modeling and Model Transformation: Introducing New Technology into the Defense Industry. In: 1st European Conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA'05), LNCS, vol. 3748, pp. 9--18, Springer (2005)
6. Brambilla, M., Ceri, S., Fraternali, P., Acerbis, R., Bongio, A.: Model-Driven Design of Service-Enabled Web Applications. In: ACM SIGMOD International Conference on Management of Data, pp. 851--856 (2005)
7. Burgstaller, B., Wuchner, E., Fiege, L., Becker, M., Fritz, T.: Using Domain Driven Development for Monitoring Distributed Systems. In: 1st European Conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA'05), LNCS, vol. 3748, pp. 19--24, Springer (2005)
8. Deng, G., Lu, T., Turkay, E., Gokhale, A., Schmidt, D., Nechypurenko, A.: Model Driven Development of Inventory Tracking System. In: 3rd OOPSLA Workshop on Domain Specific Modeling (DSM'03), 6 p. (2003)
9. Dybå, T., Kitchenham, B.A., Jørgensen, M.: Evidence-Based Software Engineering for Practitioners. IEEE Software, 22(1), 58--65 (2005)
10. Jonkers, H., Stroucken, M., Vdovjak, R.: Bootstrapping Domain-Specific Model-Driven Software Development within Philips. In: 6th OOPSLA Workshop on Domain Specific Modeling (DSM'06), 10 p. (2006)
11. Jouenne, E., Normand, V.: Tailoring IEEE 1471 for MDE Support. In: Satellite Activities at the Unified Modeling Language, 7th International Conference, LNCS, vol. 3297, pp. 163--174, Springer (2004)
12. MacDonald, A., Russell, D., Atchison, B.: Model-Driven Development within a Legacy System: an Industry Experience Report. In: Australian Software Engineering Conference (ASWEC'05), pp. 14--22, IEEE (2005)
13. Mattsson, A., Lundell, B., Lings, B., Fitzgerald, B.: Experiences from Representing Software Architecture in a Large Industrial Project using Model Driven Development. In: 2nd Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent (SHARK-ADI'07) at ICSE 2007, 6 p., IEEE (2007)
14. Middleware Company. Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach. Productivity Analysis. Report by the Middleware Company on behalf of Compuware, URL: http://www.omg.org/mda/mda_files/MDA_Comparison-TMC_final.pdf (2003)
15. MODELWARE D5.3-4 France Telecom ROI, Assessment, and Feedback. Revision 1.1, URL: http://www.modelware-ist.org (2006)

16. MODELWARE D5.3-1 Industrial ROI, Assessment, and Feedback- Master Document. Revision 2.2, URL: http://www.modelware-ist.org (2006)
17. MODELWARE D5.3-5 Western Geco ROI, Assessment, and Feedback. Revision 0.3, URL: http://www.modelware-ist.org (2006)
18. MODELWARE D5.3-2 Enabler ROI, Assessment, and Feedback. Revision 1.1, URL: http://www.modelware-ist.org (2006)
19. Mohagheghi, P., Conradi, R.: Quality, Productivity and Economic Benefits of Software Reuse: a Review of Industrial Studies. Empirical Software Engineering Journal, 12(5), 471--516 (2007)
20. Pagel, M., Brörkens, M.: Definition and Generation of Data Exchange Formats in AUTOSTAR. In: 2nd European Conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA'06), LNCS, vol. 4066, pp. 52--61, Springer (2006)
21. Presso, M.J., Belaunde, M.: Applying MDA to Voice Applications: an Experience in Building an MDA Tool Chain. In: 1st European Conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA'05), LNCS, vol. 3748, pp. 1--8, Springer (2005)
22. Raistrick, C.: Applying MDA and UML in the Development of a Healthcare System. In: Satellite Activities at the Unified Modeling Language, 7th International Conference (UML 2004), LNCS, vol. 3297, pp. 203--218, Springer (2004)
23. Safa, L.: The Practice of Deploying DSM, Report from a Japanese Appliance Maker Trenches. In: 6th OOPSLA Workshop on Domain Specific Modeling (DSM'06), 12p. (2006)
24. Shirtz, D., Kazakov, M., Shaham-Gafni, Y.: Adopting Model Driven Development in a Large Financial Organization. In: 3rd European Conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA'07), LNCS, vol. 4530, pp. 172--183, Springer (2007)
25. Staron, M., Kuzniarz, L., Wallin, L.: Case Study on a Process of Industrial MDA Realization: Determinants of Effectiveness. Nordic Journal of Computing 11(3), 254--278 (2004)
26. Staron, M.: Adopting Model Driven Software Development in Industry- a Case Study at two Companies. In: ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML 2006), LNCS, vol. 4199, pp. 57--72, Springer (2006)
27. Trask, B., Paniscotti, D., Roman, A., Bhanot, V.: Using Model-Driven Engineering to Complement Software Product Line Engineering in Developing Software Defined Radio Components and Applications. In: ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'06), pp. 846--853 (2006)
28. Ulrich, A., Petrenko, A.: Reverse Engineering Models from Traces to Validate Distributed Systems- an Industrial Case study. In: 3rd European Conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA'07), LNCS, vol. 4530, pp. 185--193, Springer (2007)
29. Weigert, T., Weil, F.: Practical Experiences in Using Model-Driven Engineering to Develop Trustworthy Computing Systems. In: IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06), pp. 208--217 (2006)
30. Wegener, H.: Agility in Model-Driven Software Development? Implications for Organization, Process, and Architecture. URL: http://www.softmetaware.com/oopsla2002/wegenerh.pdf (2002)
31. Wegener, H.: Balancing Simplicity and Expressiveness: Designing Domain-Specific Models for the Reinsurance Industry. In: 4th OOPSLA Workshop on Domain Specific Modeling (DSM'04), 12 p. (2004)