Science of Computer Programming 89 (2014) 144–161

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure

John Hutchinson *, Jon Whittle, Mark Rouncefield

School of Computing and Communications, Lancaster University, UK

HIGHLIGHTS

- We present extensive results from a survey of MDE practices in industry.
- We present case studies of the adoption of model driven engineering (MDE) by four companies.
- We identify important factors that can affect the success or failure of MDE use from both the survey and case studies.
- MDE provides genuine benefits to those companies who use its appropriate contexts.
- Success/failure appears to be more dependent on organizational factors than technical.

ARTICLE INFO

Article history:
Received 7 March 2012
Received in revised form 26 March 2013
Accepted 27 March 2013
Available online 17 April 2013

Keywords:
Model driven engineering
Empirical software engineering
Industry practice

ABSTRACT

In this article, we attempt to address the relative absence of empirical studies of model driven engineering (MDE) in two different but complementary ways. First, we present an analysis of a large online survey of MDE deployment and experience that provides some rough quantitative measures of MDE practices in industry. Second, we supplement these figures with qualitative data obtained from some semi-structured, in-depth interviews with MDE practitioners, and, in particular, through describing the practices of four commercial organizations as they adopted a model driven engineering approach to their software development practices. Using in-depth semi-structured interviewing, we invited practitioners to reflect on their experiences and selected four to use as exemplars or case studies. In documenting some details of their attempts to deploy model driven practices, we identify a number of factors, in particular the importance of complex organizational, managerial and social factors – as opposed to simple technical factors – that appear to influence the relative success, or failure, of the endeavor. Three of the case study companies describe genuine success in their use of model driven development, but explain that as examples of organizational change management, the successful deployment of model driven engineering appears to require: a progressive and iterative approach; transparent organizational commitment and motivation; integration with existing organizational processes and a clear business focus.

1. Introduction

The complexity and pervasiveness of software in society is growing exponentially [13]. It is generally agreed that the only realistic way to manage this complexity, and to continue to provide software benefits to the public at large, is to develop

* Corresponding author. Tel.: +44 1524 510311; fax: +44 1524 510492.
E-mail addresses: johnhutchinson.uk@gmail.com, j.hutchinson@lancaster.ac.uk (J. Hutchinson), j.n.whittle@lancaster.ac.uk (J. Whittle), m.rouncefield@lancaster.ac.uk (M. Rouncefield).

0167-6423/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.scico.2013.03.017

software using appropriate methods of abstraction [20]. Today, the state-of-the-art in software abstraction is model-driven engineering (MDE) – that is, the systematic use of models as primary artifacts during a software engineering process[1]. MDE has recently become popular in both academia and industry as a way to handle the increasing complexity of modern software and, by many, is seen as the next step in increasing the level of abstraction at which we build, maintain and reason about software. MDE includes various model-driven approaches to software development, including model-driven architecture, domain-specific modeling and model-integrated computing. This article is concerned with describing and understanding the industrial experience of MDE and identifying any best practice or lessons learned.

Whether or not the current brand of MDE tools succeeds, the notion of abstract models is crucial to the future of software. But empirical evaluations are needed to ensure that future software tools will match the way that software developers think. Although MDE claims many potential benefits – chiefly, gains in productivity, portability, maintainability and interoperability – it has been developed largely without the support of empirical data to test or support these claims [4]. As a result, decisions whether or not to use MDE are based mainly on expert opinion rather than hard empirical data; and these opinions often diverge [30,31] as companies tend to adopt MDE based not on an analysis of how it will affect their business but on perception or the advice of evangelists. The lack of empirical results on MDE is a problem since industry invests millions in the development and application of MDE tools [12]. Without empirical evidence of the efficacy of these tools, there is a danger that resources are being wasted.

The benefits of MDE are frequently cited and are often considered to be obvious. However, there are also reasons why MDE may in fact have a detrimental effect on system development. Firstly, it is by no means guaranteed that higher abstraction levels lead to better software. In fact, results from psychology generally [19,5] and psychology of programming specifically [29] show that abstraction can have a negative effect because thinking in abstract terms is hard, with a tendency for individuals to prefer concrete instantiations (e.g., exemplars, simulations) over abstract conceptualizations.

Secondly, MDE involves dependent activities that have both a positive and a negative effect. For example, code generation in MDE appears, at first glance, to have a positive effect on productivity. But the extra effort required to develop the models that make code generation possible, along with the possible need to make manual modifications, would appear to have a negative effect on productivity. How the balance between these two effects is related to context, and what might lead to one outweighing the other, is simply not known. Thirdly, there are many different flavors of MDE and so companies have a difficult choice to select the right variant for their business. The benefits and drawbacks of MDE are not obvious and, in fact, may be highly dependent on context. As a result of these factors, there is as yet no clear decision-making framework that can tell whether MDE will succeed or not in a given context. Our research is intended to provide a better foundation for MDE adoption by trying to understand empirically which factors lead to successful adoption of MDE and cataloging exactly what works in MDE projects [17,18].

2. Previous work

There has not yet been a systematic and multidisciplinary programme of work to study the effectiveness of MDE in broad terms. For example, there are currently no widespread, systematic, surveys of industrial use of MDE. These can be important because they often reveal common misperceptions [13]. A 2005 survey looked at the penetration of UML and UML tools into the marketplace [10] but focused on UML not MDE. Forward and Lethbridge [10] conducted an online survey of software practitioners' opinions and attitudes towards modeling. Afonso et al. [1] documented a case study to migrate from code-centric to model-centric practices. But there has been very little research that examines the social and organizational factors related to MDE adoption and use since most empirical studies have concentrated on technical aspects of MDE. The Middleware Company conducted two studies, commissioned by Compuware in 2003 and 2004, that measured development and maintenance times of an online pet store using both MDA and a state-of-the art IDE [24,25]. The studies compared only one MDE tool (Compuware's OptimalJ) but showed a 35% increase in productivity and a 37% increase in maintainability. Anda et al. [2] reported anecdotal advantages of modeling such as improved traceability but also pointed to potential negatives, such as increased time to integrate legacy code with models and organizational changes needed to accommodate modeling.

There has been more research on evaluating the language UML [3,6,21,26]; studies attempting to empirically measure the comprehensibility of UML models [7–10,14,22–24,28]; and a range of experiments that assess software engineering techniques incorporating UML. But MDE is more than just UML. UML is just one example of a modeling language but MDE additionally incorporates the notion of multiple modeling languages (each for a different domain), the idea of automating model transformations between modeling languages, and the principle of maintaining multiple perspectives of a project (e.g., platform-specific vs platform-independent). Whilst most existing work has concentrated on evaluating the appropriateness of the UML language, almost nothing has been done to examine the wider issues of MDE: such as whether code generation works in practice; what are the trade-offs between domain-specific languages and general purpose languages; is MDE appropriately aligned with existing organizational structures?

[1] This description of MDE as "the systematic use of models as primary artifacts during a software engineering process" is essentially the definition of MDE that was used for the work described here. Although others may attempt a more formal definition, and then use that to determine if the practices observed constituted MDE, the goal here was to be as inclusive as possible in our attempt to discover what practices were actually occurring in industry. Thus our notion of MDE arises from what is being practiced by those who took part in our study.

More recent studies have also lacked the kind of empirical, industrial experience, detailing MDE in use. France and Rumpe [11] provide an overview of research in MDE, pointing to some of the 'wicked problems' involved, rather than its industrial application. Mohagheghi and Dehlen [25] provide a meta-analysis of the literature on MDE outlining how MDE techniques have been applied in a range of companies across different domains. In terms of software quality there were some anecdotal accounts of defect reductions, reduced need for code inspections, and maintenance gains. However, few of the papers in the meta-analysis provided strong empirical evidence of the impact on productivity and Mohagheghi and Dehlen suggest that there is a need for more empirical studies evaluating MDE before sufficient data will be available to prove the benefits of its adoption.

To summarize, there has been very limited empirical research assessing the benefits of MDE. In particular, there appear to be three key gaps in current understanding: a lack of knowledge on how MDE is used in industry; a lack of understanding of how social factors affect MDE use; and a failure to assess aspects of MDE beyond UML, such as the benefits of code generation, domain-specific abstractions and model transformations. The initial motivation behind our research was then to build a growing corpus of evidence, both quantitative and qualitative, concerning the lived experience of model-driven development practices in industry, through which we might further investigate some of these important and interesting gaps in our knowledge.

3. Method

Our general tactic was to tackle these key aspects of MDE using a multidisciplinary method. We attempted to link the methodological and analytic approaches of the Social Sciences (that is, we used established Social Science and Management Science research techniques) with an ontological and epistemological understanding drawn from computer science (that is, our basic understanding of MDE, coding, architecture etc. is drawn from Computer Science). We approached this interdisciplinary challenge of the empirical analysis of MDE in industry by using an eclectic mixture of research methods — in particular, an online questionnaire survey and case studies constructed through semi-structured in-depth interviewing.

3.1. The survey

Survey methods are a well-established social science technique for obtaining some broad characterization of a particular issue. Our survey consisted of 35 mostly closed questions and was designed to take about 15 minutes to complete. Closed questions were chosen because alternative replies were known, were limited in number and were clear-cut and were used to maximize the number of respondents completing the questionnaire whilst more nuanced issues were discussed in the interviews. The 35 questions were selected based on a consultation process with well-known MDE experts. The survey was deliberately targeted towards practicing MDE users. Because of the sampling method, respondents generally (although not exclusively) have had at least some success with MDE — those who have tried MDE but failed were probably not accessible within the sampled network. To account for this, a simple starter question asked "Do you consider MDE to be a good thing?" 84% answered "yes" to this question, indicating that our sample is generally made up of MDE proponents. 12% answered "neutral" or "do not know".

To limit the set of respondents, clear instructions were given that the questionnaire should be completed only by those with practical MDE experience. Even so, around 23% of respondents described their role as "researcher". Responses to "boilerplate" questions tell us that the vast majority of participants had significant software engineering experience, were employed in a range of different roles, and worked in a good spread of size of company with respect to the number of people actively engaged in software development. The responses also tell us that the participants were generally experienced in using MDE.

To recruit participants, we used a combination of snowball sampling and an open call. Snowball sampling [27] is a non-probability sampling technique where existing subjects help to recruit additional subjects from their own network. The questionnaire was initially sent to a list of known MDE professionals who were asked to help recruit additional volunteers. Snowball sampling is subject to numerous biases because it only attracts subjects within a certain network. As a result, it could bias the sample towards (e.g.) those who used MDE in a particular way or in a particular industry. However, this is acceptable in this case as the focus of interest was experienced MDE users who can be quite difficult to reach using random sampling. A number of questions were included to profile the participants so that any results would not be generalized beyond this profile. In terms of the open call, the questionnaire was advertised in a prominent place on the OMG's home page between the end of 2009 and the middle of 2010 (after that it was moved to the MDA part of the site where it is still available) and was also advertised on the project website and via a number of related blogs.

This "community" of respondents had over 1000 years' total experience in software development. Almost 70% had been involved in software development for 5 years or more and 44% had been so for 10 years or more. The roles filled by respondents are diverse, including Developer (17%), Modeler (21%), Team Leader (19%), Project Manager (16%), Domain Expert (4%), Researcher (23%), Architect (7%) as well as a small number of consultants or chief technology officers. Company size fell roughly into thirds: a third had 1–100 employees, a third had 100–1000 and a third had over 1000 employees; of these about half have 1–100 people involved with software development and around 40% have more. This highlights the fact that many companies focus upon some other aspect of engineering (e.g. mechanical, electrical, electronic or a combination

of these) and carry out software development as a supporting activity. Finally, there was variation in respondents' MDE experience: 10% carrying out "initial exploration", about 10% doing "prototyping", about 18% working on their "first major project" using MDE and a third describing their organization's maturity with MDE as "Extensive Experience of MDE on Many Projects and/or over Many Years".

In terms of internal validity, we did not carry out probabilistic sampling for the selection of respondents. This has likely resulted in a bias towards those who have successfully applied MDE. It would, of course, be interesting to conduct a related survey with those who have tried MDE but given up; such respondents would be difficult to reach, however. We accounted for bias in the group of respondents by asking our first question ("Do you think MDE is a good thing?"), which allows us to quantify the bias to a certain extent. Another threat to internal validity is that our survey relies on personal experiences and, to a lesser extent, attitudes. Hence, there could potentially be problems with accuracy of information. This was accounted for by stressing in all questions that we were interested in experience-based information rather than simply personal opinion. It is also possible that the bias of individual respondents could be called into question. There may be participants, for example, who have an agenda to push, such as a tool vendor. We have made every effort to deter such individuals from completing the survey by including an introductory instruction not to continue if the individual was not an industry practitioner with hands-on experience of applying MDE. Out of 449 responses, about 6% of respondents claim to be involved in consultancy, whilst only 1% mention involvement with selling tools. In terms of external validity, the results are limited to those who have experience in applying MDE in practice. We make no claims that the results generalize to software development or even modeling more broadly.

3.2. The case study interviews

The mass of quantitative data obtained from the online survey was supplemented and complemented by more qualitative data obtained through semi-structured interviewing of some of our participants that was then shaped into what we believe are instructive 'case studies' of MDE deployment. The essence of the case study method is to conduct empirical inquiry within its real-life context and, thereby, provide detailed, qualitatively rich, contextual description and analysis of a complex real-life phenomenon, that is, we wanted to gather some detailed information of exactly what it is like, what it actually feels like, to be involved in an industrial MDE project and gain some personal reflections on its organization, successes and failures. Case study research excels at producing this kind of detailed understanding of complex issues. Of course, critics of the method suggest that the detailed study of a small number of cases may produce various kinds of bias and can offer few grounds for establishing the reliability or generality of findings. Briefly, we suggest that in this particular case, where we are interested in what 'general' lessons might have been learned from the MDE experience there is a confusion of the social science use of the term generalization, where it is associated with prediction, statistics, causality and theory development and ordinary language use of generalization, where the concern is perhaps more with issues of general expectation, sensitivity or 'typicality'. We suggest that our interview studies provide sufficient detail that the generalization problem, briefly – 'how can this information be relevant to other MDE projects in other organizations'? – is transformed from being simply the researcher's problem, to one more easily addressed by the audience (or the reader); enabling them to identify the points of similarity and difference between their own organizations and the ones reported here. In deciding the relevance of the materials we present, the generalization question then becomes, for all practical purposes, – 'in what respect are the details reported here sufficiently similar to those in your own project team or organization'?

The data was gathered during a semi-structured in-depth telephone interview that lasted approximately an hour. The interviews were informal and, whilst organized around a number of themes, were designed to allow the conversation to follow the respondents' interests. The themes we were interested in, and which tended to arise fairly naturally in the conversation covered such issues as the respondents background and experience, their current role, what model-driven engineering meant in their setting, the motivation for using model-driven engineering within the organization, their experience of using model-driven engineering, its benefits and problems, the critical decisions that made MDE a success or failure, the attitudes of people at all levels to the adoption and use of MDE, any lessons learned and any significant "war stories" – examples of significant events of their adoption of MDE. These general themes formed the basis for our conversations with practitioners and we followed up their answers both for clarification and to obtain more detail and insight. Some of their answers are presented in relatively 'raw' form here largely for authenticity, this is how people reason about MDE practices, and because they also give some idea about the inherent 'messiness' of everyday organizational life within which MDE deployments have to be made to work (or not).

For our interview studies, respondents with experience of industrial MDE projects were identified from personal contacts and from our online survey promoted on leading software engineering and MDE mailing lists and through a link on the Object Management Group's website. The results reported here are based on 22 interviews we carried out with a range of MDE practitioners. The data collected amounts to around 20 hours of recorded interviews and over 150,000 words of transcribed data. Collectively our interviewees have over 360 years of software development experience between them and represent a range of different organizational roles in a number of different domains. The data was analyzed using a broadly 'grounded' approach [15] looking for and identifying common emerging themes and issues and the various ways in which these were expressed.

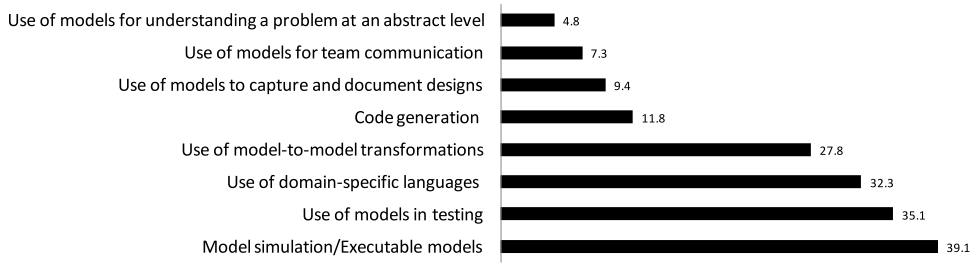

Fig. 1. "Do not use" percentages for MDE activities.

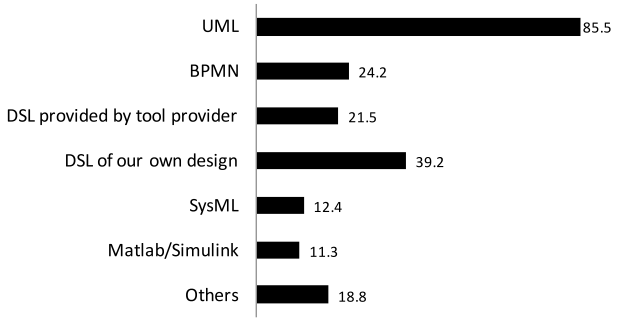

Fig. 2. Percentage of respondents using each modeling language.

4. Survey results

4.1. Survey results—general

1 What are models used for?

MDE encompasses a variety of activities, from code generation, to modeling language creation, to model-based testing, etc., so respondents were asked which of the activities in Fig. 1 they actively used. In fact, respondents were asked to note the contribution of each activity towards improving productivity and maintenance. However, a "do not use" category was included for participants who did not use a particular activity. As it turns out, it is this "do not use" category that sheds light on the level of formality of MDE use within industry — since the activities in Fig. 1 broadly require a greater degree of effort/formality further down the *y*-axis. Not surprisingly, a large majority of respondents (95%) use models for problem understanding. The use of models in documentation is also widespread (91%). This contrasts with the use of executable models or models for simulation (62%). The percentages for two of these activities are somewhat unexpected. Firstly, over 70% of respondents appear to use model-to-model transformations. Secondly, given the relative maturity of model-based testing, it is surprising to find that 35% of respondents do not use models in testing.

4.2. Which modeling languages are used?

Many people associate MDE with the use of UML. In reality, the choices of language are far more varied. Fig. 2 shows the responses to a question asking which modeling languages are used. (Note that respondents were free to select multiple.)

UML is the most used modeling language but other results are less obvious. Firstly, there is a higher than expected preponderance of DSL use — either custom DSLs designed in-house or off-the-shelf DSLs. There is a long standing debate within the MDE community on the relative strengths of UML versus DSLs (cf. [16]). Fig. 2 suggests that DSLs have achieved a significant degree of penetration.

UML and DSLs should not necessarily be considered as mutually exclusive of course since DSLs can be defined as UML profiles (e.g., [17]). 62% of respondents who use custom DSLs also use UML and many others reported using multiple modeling languages.

4.3. Which diagrams are used?

Participants were asked which diagrams they used for modeling and how often[2]. Fig. 3 shows the most popular diagrams used regularly by respondents. The threshold for inclusion in Fig. 3 is two responses. In addition, a large number of other diagrams were mentioned just once (20+).

2 Modeling may, of course, be done textually and need not involve diagrams at all.

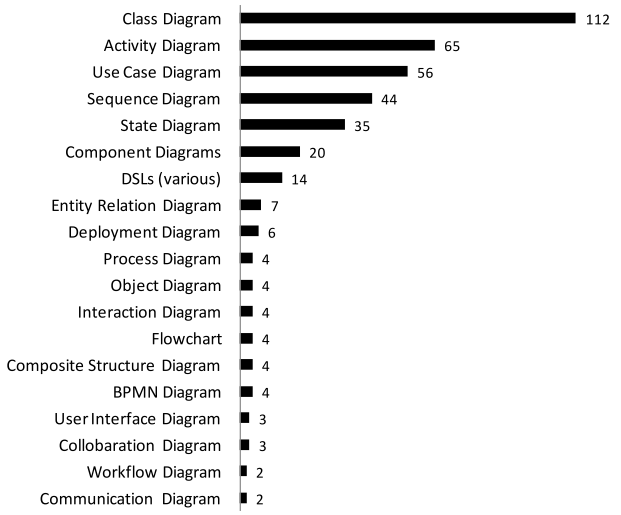

Fig. 3. Diagrams used regularly for modeling (# respondents).

The category title "DSLs (various)" groups together any mention of domain specific modeling diagrams, but does not include those domain specific approaches that were specifically named, such as AUTOSAR, WebML, etc.

4.4. Survey results—paired questions

A series of pairs of questions were asked about potentially positive and negative aspects of MDE use in order both to ascertain what reliance we might place on individual answers and to investigate how difficult respondents found it to balance the advantages and drawbacks of its use. For example, one pair of questions asked about the potential of code generation. Respondents were asked both "Is your use of code generation an important aspect of your MDE productivity gains?" and "Is integrating generated code into your existing projects a significant problem?" These questions were designed to tease out the trade-off between assumed advantages gained from code generation (improved productivity) and assumed disadvantages (legacy code integration). By comparing responses to these questions, it is possible to get a sense for the relative importance of benefits versus drawbacks.

4.4.1. Survey results—effect of MDE on training costs

A previous study has suggested that MDE may reduce training costs associated with new hires because much of the organizational knowledge can be encoded in code generation tools [22]. How does this compare with the novel training needs required to implement MDE? The pair of questions asked was:

Q: Does MDE allow you to employ developers with less software engineering experience (e.g. new graduates)?

Q: Does MDE require you to carry out significant extra training in modeling?

Around 46% of respondents thought that using MDE allows them to employ less experienced software engineers whilst around 34% disagreed. The remainder were evenly split between being neutral and having no experience.

74% thought that using MDE requires them to carry out significant extra training ($<$9% disagreed). In this instance, those who were neutral represented almost 15% of respondents, whilst 2.6% said they had no experience.

Although MDE practitioners are able to hire less experienced employees, they also incur additional training costs. The results highlight just how carefully a balance must be achieved. A company adopting MDE and assuming that they will require the same level of experience as before and expecting to invest more in training will be far better prepared than one looking to save costs by hiring inexperienced new graduates.

4.4.2. Survey results—benefits of code generation

The paired questions in this case were designed to unpick the trade-offs between increased productivity from generating code versus potential productivity losses due to increased legacy code integration costs:

Q: Is your code generation an important aspect of your MDE productivity gains?

Q: Is integrating generated code into your existing projects a significant problem?

For a majority of respondents, the ability to automatically generate code is considered an important part of the productivity gains they achieve ($>$75%). Around 10% report that code generation is probably or definitely *not* an important aspect of their productivity gains. The answers about integrating generated code suggest much more ambivalence. About 36% of respondents say it is not a significant problem, and a slightly higher 40% say that it is a significant problem.

4.4.3. Survey results—making changes to the model or to the code

Best practice guidelines usually advise modelers to make changes only on the modeling level – and then re-generate the code – rather than modifying the generated code. Although much can be achieved by the round-trip facilities in modern tools (c.f., [23] for a discussion), it is generally not possible to ensure that models and code stay perfectly synchronized without manual intervention. The pair of questions asked about these issues were:

Q: Do you mainly make updates on the model rather than the code?

Q: Do you spend a lot of time synchronizing the model and the code?

Almost 70% of respondents say they probably or definitely mainly make updates on their models whilst around 15% say they do not. Approximately 35% of respondents say they definitely or probably spend a lot of time synchronizing their models and code whereas about 45% say they do not.

These responses lend credence to the assumption that modifying the models rather than the code is the preferred approach in practice. The data suggest that, even so, there can still be issues in synchronizing models and code. It is impossible to know, from the questionnaire data, why this is the case, but possible reasons could be that not all functionality can be captured as a model or that there is other, manually-written code to be integrated.

4.4.4. Survey results—is UML too complex?

Since it was first introduced by the OMG in 1997, UML has quickly become the *de-facto* modeling language standard. Despite this, it has been widely criticized for its lack of a rigorous semantics[3] [26], the process by which it has been "designed by committee", and its focus on graphical models (cf. [25]). Nevertheless, UML is widely used (as seen from the results in Section 2) and widely taught. The next set of paired questions attempted to understand the relationship between two common criticisms of UML: firstly, that it is too complex, and secondly, that it is not powerful enough: *Q: Is UML too complex?*

Q: Is UML powerful enough for your needs?

There is some ambivalence about the balance between UML's complexity and its power. 44% of respondents thought that UML is definitely or probably too complex, compared to 32% who thought that it definitely or probably is not. A similar pattern emerges when considering UML's power: 52% thought that UML is (definitely/probably) powerful enough for their needs, whereas 31% thought that it (definitely/probably) is not powerful enough.

After fourteen years of development of the UML language, almost half of our respondents still believe that UML is too complex and almost a third believe that, despite this complexity, it is not yet powerful enough. This apparent lack of total enthusiasm for UML can perhaps go some way to explaining the prevalence of DSLs as described in Section X.X.

4.4.5. Survey results—does MDE promote understanding?

In this instance, the pair of questions asked was:

Q: Does your use of MDE lead to better understanding between stakeholders?

Q: Does your use of MDE result in unexpected confusion and/or misunderstandings between stakeholders?

Around two thirds (66%) of respondents thought that their use of MDE helps understanding between stakeholders, whilst 25% believed that their use of MDE results in confusion. Only 16% of respondents did not think that their use of MDE led to better understanding, whilst almost half (49%) did not think that it results in unexpected confusion or misunderstandings. It is generally clear then that the questionnaire respondents see more benefits in terms of understanding than drawbacks. Although it is difficult to interpret this data, it may be due to a number of reasons. Firstly, it is possible that the majority of practitioners actually use only a relatively small proportion of a modeling language. Hence, more complex features of a language like UML that typically cause confusion (such as the need to distinguish between composition and aggregation in associations) may simply not be used. Secondly, it may be that practitioners and stakeholders are able to quickly make their own disambiguations for features of a modeling language that have unclear semantics, and, that once this has been communicated within the team, everyone is "on the same page" and understands each other easily. If the latter is true, the role of (say) UML as a standard is diminished, but is nevertheless consistent with the intention of semantic variation points in UML [28].

4.4.6. Survey results—is tooling a barrier to the use of MDE?

Although certain parts of an MDE process may not require tool support, almost all practical MDE developments will require some sort of tooling. This pair of questions was designed to look at the balance between tool "appropriateness" and "cost":

Q: Are MDE tools too expensive?

Q: Do organizations attempt to use MDE with inappropriate and/or cheap tools?

[3] Although this issue is being addressed by a new standard on a semantics for a foundational subset of executable UML [21].

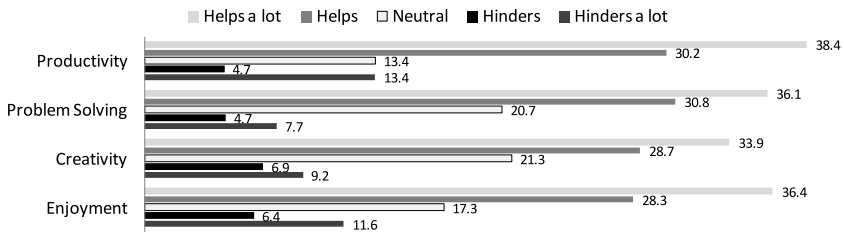

Fig. 4. How MDE affects personal experience (percentages).

According to our questionnaire, 19% of respondents thought that MDE tools are definitely too expensive and another 26% think they are probably too expensive. Those who thought that tools are probably or definitely not too expensive represent 24% of respondents.

A large proportion of MDE practitioners believe that inappropriate or cheap tools are widely used: 25% said they definitely are, with another 30% saying they probably are. The questionnaire also asked which tools practitioners used. The responses to this particular question are revealing. In particular, the sheer variety of answers provided was surprising. Indeed, almost 100 tools were listed.

How to interpret this diversity of tool usage is not obvious. It may suggest a rich array of tools to meet the different needs of users, or it may reflect an immaturity in a field where practitioners are still deciding on the best tools for the problem at hand.

4.5. Survey results—impact on personal experience

Questionnaire respondents were asked if MDE hindered or helped a variety of personal processes (Fig. 4): productivity, problem solving, creativity and enjoyment. Although productivity is an issue that has already received much attention, the other three processes are integral to the experience of an individual's work and, it might be argued, a "sense of worth" within their organization. The most obvious finding is that the majority response in each case is "helps a lot" and that, in each case, "helps a lot" > "helps" and "hinders a lot" > "hinders". This is not a trivial matter, because it shows that 8%–13% of respondents think that their use of MDE actively "hinders a lot" their productivity, enjoyment, creativity and problem solving. Bearing in mind the other findings in the questionnaire data, this is quite surprising.

5. Results: case studies

The online survey provided us with some fascinating insights and some general, quantitative, impressions about how MDE was being deployed and the general response it had evoked amongst practitioners. The four case studies that follow, constructed using a semi-structured interview process, were designed to provide us with more personal and, especially organizational, detail on the practical, everyday experience of MDE.

5.1. Case study—The Printer Company

The Printer Company is a multinational company that specializes in the manufacture of business level imaging systems, including printers, scanners, etc. and associated software and services. This particular study focuses on the case of printer development where a typical project may last in the region of 10–15 years from inception, through technology development, product development, selling and maintenance.

Within this type of environment, software engineering is one part of a complex engineering puzzle, alongside mechanical, electrical and electronic engineering disciplines. This places constraints on the software development cycle and in the case of The Printer Company, software production was considered a bottleneck in the production process at the time that they began exploring different software development approaches, and indeed this was their motivation for doing so.

This case study highlights a number of important aspects of MDE adoption:

- *Progressive* – the organization did not adopt MDE in a "gung ho" manner. Instead, they introduced it in a piecemeal way, evaluating as they went along.
- *Committed* – the introduction of a pilot project did not represent a lack of commitment on behalf of the organization. That commitment was vital as the process change necessitated by the MDE adoption was undertaken.
- *Adaptive* – the organization was willing to learn from the experience of using MDE rather than simply trying to "plug" it into their existing ways of working.
- *Business led* – MDE was used to overcome problems that could not be overcome using their existing development techniques. Ultimately, the benefits realized are expressed in business terms.

5.1.1. The Printer Company's experience of MDE

The Printer Company case study illustrates some key aspects of their experience: notably the use of a pilot and the importance of organizational commitment to MDE.[4]

> *…we started a pilot with model driven design …In 1998 it was a pilot…we put it directly in the main line of the product so we it was a pilot but we were not allowed to fail. Sounds a little bit strange but we did not have the capacity effort to have a parallel project …So we took some risks in introducing model driven design*

This description of the adoption process captures well aspects that are both progressive and committed. By carrying out a "pilot", the organization recognized that the demands of the new process should not be underestimated, and therefore it should be introduced gradually. However, by making that pilot part of the development cycle of real software – described here as a necessity, rather than choice – the organization demonstrated a commitment to the process – "*we were not allowed to fail*". This element of necessity is interesting because it highlights that serendipity, fortunate chance, may also be a factor in successful MDE adoption, and process change more generally. Had The Printer Company not needed to introduce MDE on a "live" project, the lack of necessity, and the probable deployment of less experienced software engineers, may have resulted in a far less successful experience.

Our interviews revealed some subtleties and nuances when we probed whether the outcome of using model driven design had matched the company's initial expectations. ".. in the first project I did not expect to gain in effort ..that was for the second and third project. The biggest gain I expected was in the quality of the software so the understandability, the reusability …those kinds of qualities...*indeed .. we did not gain anything in effort it even took us some more effort I think but after our first software release we were able to reduce the team fairly fast so the qualities really paid back..*"

Similar reservations and subtle understandings were outlined when discussing any productivity gains achieved: "*the benefit in the second and third project was not only because we introduced model driven design .. I think also model driven design was an enabler for implementing reuse …so we had model driven design, we had a reuse group and the third thing is that we also built a reference architecture so using the model driven toolset we started building up a reference architecture for all embedded projects and those three together really gave us the benefits*" The Printer Company approached the adoption of MDE with realistic expectations about what they could achieve in their first project. The fact that the organization implemented the pilot project whilst expecting to see no reduction in effort demonstrates its commitment to the process – it represented an investment whose pay-off would be seen further down the line.

It should be noted that this relies on a particular feature of The Printer Company as an organization; it was able to invest in a trial of MDE in one project but wait to see benefits in a later project. Organizations which structure their work on a strict project-by-project basis may find it much more difficult to successfully trial MDE in this manner.

However, the most significant aspect of this part of the interview is the description of how the company matured in its development processes as a result of using MDE. Applying MDE in this initial project led them to identify greater commonality in their systems than had previously been recognized. This initially led to the development of common components as part of an explicit reuse strategy. Ultimately, it led to the design, and adoption of a reference architecture. The process by which this came about, and the reasons for its success, is explored further in the following extract when our interviewee discussed the role and importance of modeling: *'It is a major role. I think if you did not introduce model driven design the reuse initiative and also the reference architecture would have failed because you need some very good design interfaces and also some feasibility of your components and your reference architecture. It cannot be done by purely using C++ or Java …I think that is impossible …it was really an enabler for the other two - so model driven development was an enabler for the reuse and reference architecture*"

Similarly, the introduction of modeling added a level of formality to the level of understanding rather than just capturing understandings that were already there:" *…it is better understanding of the things you already have but because you understand the things you already have you can go one step further. So reusing embedded software first starts with understanding what you have but at the moment you understand it you can make next step*" Raising the level of abstraction not only allowed the organization to realize improvements in software quality, but led to a fundamentally clearer understanding of the structure of their systems, allowing that structure to be captured in the reference architecture and the commonality to be factored out in the form of reusable components. This willingness to adapt their software development process to further exploit the potential of MDE appears to be crucial to the success of The Printer Company's adoption of the approach, allowing them to develop their embedded software significantly faster and with less effort. *"Of course the quality is on a very high level …in our products and at this moment we have also a gain in effort. I think it is very difficult to estimate that but I think we are now using 50% of the resources compared to ten years ago …because 50/60% of all software is reused …in the past ..if I look fifteen years ago or so.. embedded software was sometimes the bottleneck in the project so there was a delay in [product] introduction because the software was not ready and now embedded software is not a bottleneck in any project …mechanics, physics are the bottlenecks in products and not embedded software any more*"

Specific figures of productivity improvements and reuse must, of course, be treated with caution given that they represent subjective judgments. However, they are still revealing, not least because of the expertise of the lead software engineer

[4] Practitioner quotes are from transcripts that attempted to capture comments verbatim. They are presented here in a similar form to add to the authenticity of those comments. This means that there will be some grammatical errors, but hopefully the intent of the comments will come across strongly.

expressing them. The final comment summarizes the experience of The Printer Company wonderfully; for them, software development was a problematic part of a complex jigsaw of engineering effort. Through their adoption of MDE, they completely changed the way that they developed software so that it was delivered to market more efficiently, better and on time.

5.2. Case study: The Car Company

The Car Company is a major multinational vehicle manufacturer with a presence all round the world. During the 90s, the company was undergoing significant change — partly as a result of advances in electronics and the potential of software in road-going vehicles and partly because of the rationalization processes required to stay competitive in a global market. Both of these factors have prioritized the importance of software in vehicles: software is needed to deliver the required functionality and is more readily able to respond to changing requirements than hardware. However, the growing cost of developing software was becoming an issue. It may be the case that traditional mechanical engineering companies struggle to embrace software engineering as an equal discipline, whilst at the same time recognizing the importance of what software can deliver. It is certainly the case that some companies do not readily embrace the need to actually employ the software engineers needed to deliver the required functionality.

The Car Company case study particularly highlights a number of important factors affecting MDE adoption:

- *Iterative* – the task of developing an appropriate process in a given context is not straightforward. Trying different approaches, evaluating the results and moving forward is a costly and time consuming process, but may ultimately deliver a successful approach. Being able to let go of an approach, and associated resources, that has served its purpose may represent an important decision point for a company adopting MDE.
- *Committed* – the organization was prepared to support multiple approaches in a continuous manner, even when the investment required to implement a previous approach needed to be replicated.
- *Business led* – the company did not embark on the process of MDE adoption for "technical" reasons. It required a solution to commercial and organizational challenges that could not be met by existing means, or traditional responses.
- *Adaptive* – The Car Company proved itself able to respond to the needs (and possibilities) of integrating the new development processes in their organizational processes. This required them to be willing to evaluate objectively the advantages and disadvantages of the new approach over their existing way of doing things. Because of the iterative nature of their adoption process, they were required to do this on a number of occasions.

5.2.1. The Car Company's experience of MDE

The following comments from our interview with The Car Company case study illustrate some important aspects of their experience, beginning with the role of software generally and MDE in particular in the organizational process of centralization. *"All those organizations had to converge .. bringing them altogether and focus on a single design concept … you had two different forces there: one is how do you transform an organization .. to be able to do this more centralized engineering activity ... and then we had this software organization …… software was now becoming clearly the defining point of vehicle contact – so we took a lot of the understanding of how that organization grew, how we did reuse … and took it to a different scale within the software engineering organization, how we planned it, how we evolved the growth in that organization which mainly involved taking some our past practices but also looking externally to see what practices were successful in industry in terms of software product lines from obviously the SEI … and also a lot of the activities related to the modeling world in terms of UML, profiles and model transformation – those types of techniques were the things that we founded the organization on and grew it on. And so that organization grew as a core platform team primarily to begin with in terms of defining the reusable assets, finding the process the tools that support that and then, over time, growing in size to be able to perform multiple different projects*".

Here we are given an insight into just how rapidly evolving the organization was whilst it was also trying to embrace the growing role of software in its work. Note that the significant organizational change appears to have created an opening for the adoption of some quite modern software engineering practices (e.g. software product line approaches from the Software Engineering Institute (SEI) and MDE techniques). In this particular example, knowledge of modern software engineering practices came about as a result of the educational background of a specific individual within the organization. Such skills had not explicitly been recruited by the organization in order to advance their software engineering practices. An obvious question for which there is no easy or obvious answer is this: would the choices made by The Car Company have been different if this individual's knowledge had not influenced them? (This need for what is sometimes called a "champion" to promote MDE adoption recurs throughout the interviews.) The wider implications for the adoption of MDE more broadly are interesting to consider.

Being inside an organization undergoing change is a difficult and challenging experience. The following excerpt illustrates aspects of the organization's cultural and organizational background and the consequent challenges that ensued. *"..understanding the automotive culture . in the 90s ..within most automotive companies you had mechanical engineers and electrical engineers. You had a lot more mechanical engineers than auto engineers and you could not find a computer scientist if you went on a search party … Now that is one of the reasons why we ended up also doing model based design … one of the key constraints that we were under in terms of growing that organization is we were not going to hire a whole bunch of people just to write code. We needed people who were domain experts. Domain experts in terms of automotive algorithms.. not necessarily how*

to code automotive algorithms ..so the whole model-based process was focused on what are the types of things that algorithm engineers understand in terms of control systems and in terms of state machines but not necessarily how to effectively code them and so we tried to separate those concerns … we let the computer scientist define the model and method and construct the language per se that we formalized in a … UML profile and then we turned our electrical engineers loose in terms of defining what the functionality of the vehicle was and then a few of the computer scientists … said okay what are the generation patterns relative to how to be able to transfer those models to code"

The Car Company appears to have approached software engineering as a subordinate discipline (to the better established disciplines of mechanical and electrical engineering) and therefore approached the growth of this part of the organization's capacity with few pre-conceived ideas other than of the limit on the number of people who might be employed. Most importantly, this extract roots the necessary change in business need; The Car Company was adapting to external pressures.

" The idea at that point was to be able to do reuse - planned reuse. e - there was not necessarily a formalized process on how to be able to do that and that was one of the learning activities … ad hoc reuse was not necessarily very scalable. They were able to reuse on a couple of programmes but then it started losing value and it became very difficult to be able to manage change across them which led to the next generation which was the bringing in-house of some of the code writing within The Car Company which meant that they were doing models and generating code from those models and having a planned reuse method - so a library of models which were code-generated specifically for specific parts in the vehicles and specific products"

For the Car Company, the move from ad hoc reuse through to the more structured and planned reuse was actually a kind of generational change: "*It absolutely was. .. that was part of one of the most contentious things that happened because that meant we were basically orphaning all of the investments that we had made in the models that we had previously. So all the models we had previously were in the body electronics area were Statemate models. The transition then moved from the Statemate models to UML models and that obviously necessitated a restructuring and a redefinition of a lot of the functionality on which the effort took place - so that was a very tough decision to be brought forward to the management team at the time to be able to say yes all the investment you did on models before well we got to redo some of that work"* There are a number of the important aspects of The Car Company's adoption, and development, of MDE illustrated here: they were willing to adapt to the needs of the process in a way that required them to "start again" using a different technology and there is clear evidence that the need to adapt was supported by the organization's management, demonstrating their commitment to the approach that was being developed. To understand the importance of this point in the company's history, it is interesting to ask what might have happened if the management team had refused. In refusing the need to change to a different approach, the management team would have stopped the company from adapting to the needs of their proper use of MDE. It would also have indicated a limited commitment to identifying the most effective and efficient process. Finally, by not allowing the technical team to cycle through another stage of process development, the management team may have "fixed" the software development process at an inappropriate point – which ultimately might have led to failure. Critically, it is not in the least bit difficult to imagine that management refusal could have been justified by any number of reasonable arguments – and so again we find that an element of serendipity or good fortune appears to play a part in an organization's successful adoption of MDE. How such a failure would be classified by those involved is impossible to know for sure, but in all likelihood, it would have be described as "a failure of MDE" and not one of failed commitment. When we questioned our interviewee on what would be required to convince the company to adopt MDE he replied: *"That is a very interesting question. I think it is possible. The challenge is this: that you have not only the knowledge - but also the leadership internally to be able to make that happen and across those different steps, those different generations, there is a small set but definitely a set of individuals that were driving that change. And they were driving that change both from the knowledge they had as well as the knowledge they gained from the previous generation of work. So it would have been definitely a challenge to bring that technology in all at once."* For advocates of MDE, its process developers and its tool vendors, this represents, perhaps, the "holy grail": what evidence is necessary to convince organizations that MDE can be used – in their domain, in their sector or in their company – successfully and deliver the benefits that are claimed. In the circumstances, the degree of uncertainty expressed, and the reasons for it, are interesting. Knowledge is built upon prior knowledge, which can have both positive and negative consequences for process change and the response to novel approaches. In the case of The Car Company, gains made along the way appear to have been instrumental in continuing the organization's commitment to the process development. This may reflect the importance of maturity – or the lack thereof – in different software development methods and may therefore suggest that MDE is likely to grow in acceptance as more companies migrate to model-oriented approaches. Alternatively, it may suggest that some kind of ideological opposition to raising the level of abstraction in favor of retaining code-centric approaches cannot necessarily be overcome by "a good argument". Of course, another interpretation could be that for MDE to really take off, the way that it is taught in higher education needs to be fundamentally rethought: nascent software engineers must be endowed with the skills necessary to question the received wisdom within organizations and the ability to judge properly the efficacy of alternative approaches.

As was seen above, simple knowledge of what others are doing may significantly influence what a company attempts to do itself. This look at The Car Company's case study is brought to a close by considering the results of using MDE, which was, after all, the point of changing a software development process. As our interviewer argued whilst there was a general feeling that productivity increased, measuring it in a simple fashion is difficult. *"Ah if I actually had the complete answer for that – for every time I am asked that – that would be fantastic.. each of those different .. there is also a change in responsibility in terms of more burdens being brought in house … so it is very difficult to say I we were X amount more productive on this than we were on that. I think if we did some very rough calculations that it was probably close to half in terms of if you took both supplier*

resources and The Car Company resources together in terms of how much effort it took to be able to produce a body control model. This new way versus the old way … that has a lot of different characteristics … how do you cost out this planned reuse model in terms of your investing in this core asset base? How do you calculate the communication resources between a customer and a supplier and how do you manage that relationship when you actually bring that responsibility in house. Those calculations are rather rough. What we were definitely able to get some very tangible results on though was the rework cycle in terms of the cycle time in between identifying a change that needs to take place in a model and to be able to regenerate code. And/or being able to say that, 'okay I need a change' … this is how quick it takes or being able to add your functionality. Those were the key elements that were very much automated within the process they made it very easy for people to be able to do those types of activities"

This case also highlights another difficulty for advocates of MDE — many users are employing techniques in entirely pragmatic ways with little or no interest in academic (or scientific) measurements of success and instead focus on their own organizational experience. Does this mean that their criteria for success are not "measurable" and/or should be considered untrustworthy? It is suggested not, and the focus on "known quality attributes" in the above excerpt helps to explain. For The Car Company, the whole approach to an increased reliance on software represented a pragmatic business decision. Understanding the results that have been realized by adopting MDE in the same terms appears to be not only understandable, but entirely reasonable.

5.3. The Telecom Company

The Telecom Company is actually a multinational manufacturer of electronic systems for both commercial and domestic use. This particular case study pertains to its telecommunications business and focuses upon one particular project. The project in question was a significant one involving in the order of 50 full-time equivalent engineers for a year with the aim of introducing a new switching product into the market.

Taking the positives first, the product was delivered and, as far as is known, remains in use to this day. MDE was deployed by The Telecom Company on a new product and was a "success".

Unfortunately, the positives end there. The organization's experience and, in particular, that of its software engineers, could not really be described as a success, and, perhaps, the case serves to remind us of the difficulties that can arise in understanding such bald terms as 'success' or 'failure'. These can, and perhaps should, be subtle and nuanced terms. Nevertheless, before looking at the case study in detail, it is useful to consider what factors affecting MDE adoption the study best represents:

- *Autocratic* – for a number of reasons, The Telecom Company displayed an overly autocratic attitude towards its processes and its developers. Positive elements of this are similar to the "necessity" exhibited in earlier examples. However, rather than a commercial necessity, this situation manifested itself as a rather more oppressive "succeed or be sacked" one. This was associated with "national culture" by the interviewee and thus may introduce a cultural dimension to MDE success/failure factors.
- *Wholesale* – the decision to adopt an MDE approach was not made with much understanding of the necessary process change and was implemented as an "all or nothing" approach. Given the "autocratic" nature of the organization, this equated to "all". Given that the company already had significant experience of software development using traditional techniques, this approach to process change was surprising.
- *Rigid* – both in terms of how the MDE approach was adopted and how it was "integrated" with existing processes. A new tool was introduced and it was used by all engineers. When the approach proved to be inflexible, those engineers developed "work-arounds" – primarily, inserting program logic into code fragments that would augment the generated code, to the point where these overrode the model.

5.3.1. The Telecom Company's experience of MDE

The extracts from The Telecom Company case study will illustrate some of the points made above. The decision to employ a modeling language and associated transformation tools (here, expressed as CASE tools) was implemented from above and with little regard for the experience of the existing developers"*…this kind of huge decision to …50 engineers all developing by using this CASE tool, so it was quite radical at that time. I did not quite understand it. I could understand some portion of the system - but everything developed by the tool is really radical change*."

They were, in fact, given training, but this appears to be concerned primarily with the major functions of the tool used. The unspoken element of this section is that existing developer skills and their attitude to change did not feature largely in the adoption process. This appeared quite critical for the MDE deployment and to maintain consistency between the generated code and the models the company implemented a rule: *"…they …developed a rule. So after generating the code, if you want to do this then you have to delete this line and that line and also you have to change this parameter like this. So there was a clear guideline …*" This applied to millions of lines of code: "Yes, millions, huge, huge code. So the binary is some gigabytes*……that is also one of the problems …they totally…attach to the tool and they do not know what is going on with the generated code…. it was C language they had been using, they could not optimize the generated code so the way they had to …asking the hardware guys to have more hard disk, more memory, because of the tool. So beforehand we had very small memories and we had been using C and we were very clear about the memory map and each engineer has a clear view on how much memory space they can use and if something happens then they can manage all these things to fit into the memory — no problem at all.*

But this case we cannot do something with the generated code so we simply ask the hardware guys to have more hard disk. So it took, in the worst case, it took about 2 hours to upload the binary to deploy to the switching system".

When The Telecom Company implemented its approach to MDE, it did so in a way which almost "defied failure". Unfortunately, this type of very rigid approach to process change does not allow for evaluation or adaptation in a meaningful sense. Instead, teams developed work-arounds that would deliver the necessary functionality and/or productivity almost regardless of the tools used. The processes developed to make this work are not really part of a considered MDE approach. So the "failure" that the company defied was the particular narrow sort associated with delivery of a product; the quality of the product, the experience (and costs) of its employees and the development of the organization itself were all dismissed as unimportant. Although the project was completed and deemed a 'success' the company did not then adopt this approach for any other projects. As our interviewee suggested: *", I think everyone makes mistakes. So the company, the engineers, they are too ambitious.... They had to make a decision which part of the system should be done by using this CASE tool … and which part of the system develop as before. So they could not really distinguish this at the beginning so without clear knowledge or vision of the pros and cons of the MDE approach, they just heard the good side – the pros – of MDE and … were not aware of the downside. So that was their mistake. And the second thing is the tool vendor. The tool itself is very inefficient. …*

This fragment of The Telecom Company interview is quite revealing. Primarily, it speaks of the unsatisfactory completion of the project – even if "delivering the product" actually happened. However, it also highlights the overly ambitious approach adopted by the organization. They did not trial the approach in any way – the culture may have made up for a number of shortcomings, but the result was that they deemed MDE to be a failure. "*… nightmare. Because you cannot use the source code, you have to do it at the SDL level so they could not really change … maybe they are not so skilful in the modeling language, but also at the same time the model itself is quite abstract, so it is kind of difficult to make a decision. Should I change the model or should I.... there is always a way you can do – some little bit of programming attached to the models – so you just use that kind of window to implement your ideas and leave it just as is, but you put in more logic by doing coding. So in the end they are using just as a programming language. So it is like a Java compiler, C compiler — very expensive, C compiler I would say..."*

This excerpt from The Telecom Company interview highlights a number of important issues. The first is that the domain is highly specialized and complex – and that the company in question already had that specialized knowledge. This was not a project being carried out in a naïve development environment. The difficulties encountered represented a failure to apply knowledge that they already had to this particular project. The second is that the modeling environment undermined the existing expertise of the developers – they were unable to extend their knowledge into the modeling required to manipulate the process and tool being employed. Finally, the lack of flexibility in the system resulted in work-arounds that ultimately defeat the point of attempting to use MDE in the first place. Developers reverted to coding the program logic using the tool's facility for inserting fragments of code. The result was that the whole MDE process was reduced to some degree of high level modeling and low level coding — hence the comment that the tool was a "*very expensive C compiler*".

The choice for organizations choosing to adopt MDE is complicated by the hype surrounding the technology. Very early adopters can actually create the methods that followers then adopt. However, they can also fall foul of the exaggerated promise of new techniques and processes, as our interviewee suggested: *". So I also heard this story about a company – they also experienced a really difficult time. So everything they asked them to identify as objects, ideas, just follow the guidelines and instructions and they have huge number of objects and they do not know what to do about this. So they did all the analysis of the requirement and then they end up with thousands and millions of objects – how we can compose this into the system? They had no idea! So their problem I think is the consultant … did not really know the meaning of the object orientation and the system architecture view and what is the important thing to make better"*

This example distinguishes between tool vendors and genuinely neutral consultants and suggests that the former are not capable of providing suitable advice to adopting companies whilst recognizing the role that the latter might play in identifying the most appropriate approach. Being receptive to this kind of advice – and actively seeking it out before embarking on an MDE adoption process – might well be the sort of organizational decision that separates success from failure. Given the fine dividing lines between the types of advice discussed, this again shows just how carefully balanced such decisions are.

5.4. Case study: The Defense Company

The Defense Company is a major multinational provider of equipment and services to the military. The company is structured as a number of distinct specialist businesses in North America and around the world. With software becoming an increasingly important part of defense equipment and systems, many of the component businesses have significant software development capabilities.

With well established software development processes already in place, The Defense Company has introduced MDE on individual projects within different businesses over the last 10 years or so. This case study, therefore, captures experiences from a number of different projects within the context of a single parent company. In particular, the positive elements of this case study illustrate the following aspects of MDE adoption:

- *Business led* – the motivations cited by The Defense Company for using MDE relate to the primary goals of the business and the sector. Any technical benefits are also understood in these terms.

- *Adaptive* – The Defense Company demonstrates that it has been able to adapt its process to the requirements of MDE even within the constraints of large military programmes.
- *Committed* – in the face of ongoing resistance to its introduction, The Defense Company has continued to champion MDE on appropriate projects, employing MDE evangelists and mentoring programmes. This demonstrates commitment on behalf of the company, but also on the individuals leading the process.

The positive aspects of The Defense Company's case study is emphasized here in part because it also contains a number of examples of more negative aspects — particularly those concerned with resistance to MDE adoption. Understanding where and why resistance, or reticence, exists is a key aspect of understanding some of the problems of introducing MDE.

5.4.1. The Defense Company's experience of MDE

The Defense Company case outlines a number of different motivations for getting involved in MDE, as our interviewee commented: *"... I know what the struggles are like to get funding for various projects [in the military], and just to make The Defense Company more competitive I wanted to find ways to get the product out the door to our customer, make them happy and do it for less. And in addition I worked on projects that struggled because perhaps somebody in the marketing department knew about some similar software and made claims about how easy it would be to get the same thing running for somebody else when it really was not that easy. In addition the code and the documentation were often out of sync so I found that working in a model was easier to communicate but only if the code was in sync with the model, and so when we do code generation it always stays in sync".*

The interviewee from The Defense Company has the role of "Model Driven Software Development Evangelist" within the business and is an expert in the area more generally across the company. It is perhaps not surprising then to hear the interviewee describe the motivations for using MDE so passionately. However, it is important to note that the motivations are all concerned with company competitiveness, delivering products to customers and business process efficiency. This focus on business led reasons for MDE adoption is an important aspect of The Defense Company's case study. That it is expressed by an evangelist within the company raises an interesting point: the role of an evangelist is not a technical one — although it no doubt calls for a high degree of technical expertise. The evangelist's role is to educate colleagues about the value of MDE, not just about how it should be done. As shown below, the need for education extends beyond engineers within The Defense Company to their customers as well: *"…. on lots of programmes I have had to go out, whether I am on the programme or not, and talk to the customer about the benefits, the risks, what we have seen in the past, that type of thing.… In fact the customers always are open to it because it is going to save them money and improve their quality .. The programme managers are always open to it, most developers — probably 80% of developers are open to it. It is the middle managers that stop it every time.…*

> *they have no responsibility for profit and they do not have any responsibility for probability of win of a contract. They only have responsibility for executing ... developing the amount of code that they were handed to develop and when they make mistakes that is really bad for them so they see any potential risk as something that they will not do ... they will not take the burden of that risk. For some reason they also see, even when the mentoring costs are rolled in to the total bottom line productivity, they see the outside mentors as a bad thing....."*

The lack of understanding that exists in general about what MDE means may represent a significant impediment to wider MDE adoption. The Defense Company attempts to counter that by having individuals who can work both within and outside the company to help provide the information required to make informed decisions. Of course, whether it is always an easy task for a vendor to "educate" a customer is a different matter (cf. the example with The Telecom Company earlier). The willingness of The Defense Company to invest in the role of evangelist is a clear indication of their commitment to the MDE adoption process. Instead of it representing a general commitment by the company to introduce MDE across the board, it appears to represent a commitment by The Defense Company to provide the necessary resources to allow their programme managers to exploit the most appropriate process even if it requires the adoption of novel techniques.

The Defense Company's case study highlights some important examples of resistance to MDE adoption. What is particularly interesting is that this case suggests that an organization's structure may unintentionally motivate a type of process change resistance. Whilst recognizing that business led motivation is important to make MDE adoption a success, it is intriguing to consider that the definition of some roles may insulate important team leaders from consequences of business oriented decisions at the same time as making them personally responsible for the delivery of lower level tasks.

These next excerpts from The Defense Company case study interview explores issues related to process and adaptability:

> *" ... I had say 100% it is critical for the model and the code to stay in sync.*
>
> *it is still just as important to be able to go in and look at what you have created whether it is in the textual DSL, the GUI or the more traditional model and that they are all in sync with the code. we rarely start programs from scratch. We are mainly re-using large chunks of software and then adding bits in so we tend to use the code generation for those bits but they absolutely have to integrate in with some legacy code ... given CMMI and our government requirements, we have to be able to do unit testing on code ... and eventually we turn it over to the system integrators. We tend to do very large projects with system integrators that had nothing to do with the software development integrating and testing the code, and they want code... At least in our foreseeable future I think it is necessary because we do a lot of embedded ... and so we run on*

some custom hardware, some commercial hardware, some custom operating systems, even some commercial operating systems, but we are always down ... you know trying to figure out problems with timing, things like that, and at least for the foreseeable future I think you need code to be able to do that.

This highlights some of the process adaptability that The Defense Company has needed to utilize MDE within the strict constraints of military contracts. Those constraints impose a necessarily code-centric end product, but because of the importance of maintaining model/code synchronization, generated code is considered an entirely derived artifact and must not be modified (other than for error evaluation purposes).

"so say I have got my code and I am testing it, I have found a bug, I think I might know what the problem is and I could make a quick fix in the code to check and see if that is what the problem is. I can I will allow that type of quick fix but it does not stay in the baseline. So a trouble report has to go back and the development team has them fix the model or the code generator, regenerate the code and move on."

Being able to settle on the correct process and ensuring that adhering to it meets its key requirements, and the key requirements of both the project and MDE, is an important aspect of MDE adoption.

Asked whether the results are worth it our interviewee responded: *"once people hear [that they have complete control over the generated code] they feel a little bit better about it. It feels a little bit like magic or too good to be true to a lot of the people that I talk to, so it is interesting I have actually quite reporting real productivity improvements and have scaled them way back in what I recommend to people [to] propose ...Q: So what is it that you would tell people if you thought they had believe you in terms of productivity?*

A: Well we have seen some cases with the PathMate UML code generation — we have seen cases of 4 or 5X productivity and with DSLs we have seen 7 or 8X productivity.

Q: And what do you tell people?

A: One and a half to two. ... you cannot associate those higher numbers with The Defense Company. So we recommend to organizations that they propose one and a half times productivity improvement and that includes.... and we always include with that the mentoring, the training, all the things that need to set them up for success."

Clearly the life of an MDE evangelist is not always an easy one! It would be common sense to believe that the potential of huge productivity increases would be an important factor in persuading people of the merits of adopting a new process. Instead, the potential benefits sound so extravagant that colleagues are prone to disbelieve them and thus they become self-defeating. Whether this represents general cultural reticence – "if it is too good to be true, it probably is" – or a specifically technical resistance is uncertain, but it is clearly a problem for somebody hoping to explain the benefits of the approach they are advocating.

6. Lessons about MDE deployment

What do we learn from these two very different approaches to document and understand industrial experience of MDE? The findings from our survey and case study interviews point to a number of interesting and perhaps counter-intuitive lessons. Firstly, DSLs are far more prevalent than expected. UML is not yet universally accepted as the modeling language of choice. A key finding in the questionnaire, and supported by the case studies, was a surprisingly high use of DSLs, suggesting that it is typical to develop small DSLs for narrow, well-understood domains. This may be related to another survey finding showing that many of those using MDE do not work in software companies, but in companies in other engineering domains, which require software development as an enabler. Secondly, and perhaps less surprisingly, our studies suggest that pragmatism, essentially practical decision-making, dominates decisions on deploying MDE in industry. Accordingly, practitioners generally do not stick to one language but rather pick and choose a combination of languages and tools according to the task under consideration. The wide diversity of languages and tools reported in our survey suggest that, more than ten years after the OMG brought out its MDA specification, there remains a lack of consensus on the best language and tool.

A third important finding is that the survey and the case studies together highlight some conflicting influences, concerning both productivity and cost, when deploying MDE. Overall, and perhaps not surprisingly, MDE adoption appears to increase training costs. The survey data suggests that productivity gains from code generation outweigh losses from integration with existing code and MDE allows for faster turnarounds on new requirements. Although the interviewees do not disagree with this general analysis, it is interesting to note that, for many the real driver for MDE was not code generation and its associated productivity gains. Indeed, our data suggests that productivity gains from code generation alone are not considered significant enough to drive an MDE adoption effort: because MDE can be associated with increased training costs and substantial organizational change that may therefore offset productivity increases. This does not mean, however, that companies do not adopt MDE. Rather, it turns out that the main advantages are in the effect that MDE has on helping to create a well-defined software architecture. Unanimously, our interviewees argue that MDE makes it easier to define explicit architectures, especially when MDE is a ground-up effort.

The studies provide some fascinating details of recent MDE deployments in industry — they capture details of MDE adoption in large companies by experienced software engineers and illustrate some of the factors that interviewees considered important. What should immediately be obvious, and in many cases what might also be quite surprising to

Table 1
"Dimensions of organizational attitude" to MDE adoption.

Helpful response	Unhelpful response	Explanation
Adaptive	Rigid	The ability or willingness of an organization to transform, or adapt, itself according to the needs and opportunities presented by adopting MDE
Business led	Technology led	Whether the motivation for adopting MDE originates from the need to overcome a limitation or problem with existing software development processes
Committed	Not Committed	Is the company willing to provide the necessary resources to make MDE adoption possible and is it willing to respond to unforeseen needs?
Iterative	Autocratic	The willingness of the organization to "try again" with an improved understanding of what is necessary to implement MDE properly, in the event of difficulties rather than stick with a particular approach because "that as what was decided".
Progressive	Wholesale	The process of "starting small" in adopting MDE and then growing and so committing more resources to adopting MDE on a wider scale

programmers, software developers, even software architects, is that many of the significant factors are not specifically technical[5]. Within a computing world where the next, best technology is the answer to everybody's problems, it is perhaps rather odd to discover that real practitioners, firmly embedded within that world, do not tend to focus upon the technical deficiencies, or the technical benefits, of some or other approach. Instead, they tend to focus upon issues that have been described as social or organizational factors.

It is now necessary to attempt to amalgamate some of the issues raised in the survey and the individual case studies into a more coherent whole. The list of issues identified is as follows: Progressive; Committed; Adaptive; Business led; Iterative; Autocratic; Wholesale; Rigid. It is then possible to map these responses onto a number of "dimensions of organizational attitude" towards, or in response to, the adoption of MDE. These are illustrated in Table 1.

In some ways, these "dimensions" may appear quite broad and not necessarily MDE-specific. However, the explanations provided in Table 1 show how they refer directly to MDE adoption.

What does mean in terms of an organization's response to the challenge of adopting MDE? It will usually mean that combinations of these "helpful" and "unhelpful" responses to MDE adoption will have a significant impact on the organization's chance of success.

- Successful MDE appears to require a progressive and iterative approach. Where MDE deployment seems to run into problems is where the decision to adopt an MDE approach is made without any real understanding of the necessary process change and instead takes on an autocratic top-down and wholesale character, implemented as an "all or nothing" approach. Where an organization adopts MDE iteratively, it allows its engineers to make choices and to correct mistakes. If a particular approach is introduced autocratically, it does not allow discussion of whether it is appropriate or not. It is important to recognize that this can occur at different levels – e.g. an organizational decision to move to MDE compared to an organizational decision to introduce MDE using notation X and platform Y. Whilst it is argued elsewhere that the individual responses of software engineers to the introduction of MDE will vary significantly, as a group, it seems reasonable to believe that professional software engineers will generally try to identify the most appropriate approach available.
- Successful MDE is dependent on organizational commitment – the kind of commitment required to support any organizational change. Commitment seems to be felt in different ways. On the one hand, introducing a significantly different process is hard – both in terms of the new skills required and the problems to be overcome. Knowing that the organization is committed to the process is important to help maintain the necessary level of motivation in the practitioners doing the work. On the other hand, the level of organizational commitment might be felt at very specific points in time: decision points require supportive organizational responses. Positive support at these points is crucial to making MDE a success.
- An organization using MDE needs to be willing to respond to the needs (and possibilities) of integrating the new development processes into the wider organization, to adopt new ways of doing business and adapt their own processes along the way. By contrast, problems are likely to appear when the organization is inflexible and unresponsive, and the MDE approach fails to be integrated with existing processes in a way that avoids potential clashes. This ability of the organization to adapt to the new requirements and potential of MDE being central to success makes a lot of sense. MDE is not a "bolt on" process – i.e., it does not appear to offer benefits if it is simply added to existing processes. Instead, what is usually required is an overhaul of attitudes to certain aspects of how software should be developed.
- Ultimately successful MDE has to have a business focus. In contrast, attempts to adopt MDE because it is "interesting" or "technically brilliant" are likely to result in wasted (if interesting) effort. Where existing techniques are simply unable

[5] It is likely that there exists a difference between academic research and industry attitudes on this issue. Within academia, it has long been recognized that the more "social" issues are a major factor — it does not appear that this message has been transferred entirely to industry.

to deliver what is required, whether in response to existing requirements or new, more challenging requirements, MDE may provide a potential solution. The main point is that the motivation comes from need rather than experimentation.

The workplace experiences detailed in the survey and these case studies highlight some real constraints in the way that MDE is developed and deployed. As has been observed in other areas of system deployment, recognizing, understanding and determining the applicability of even such obvious terms as "success" and "failure" is not necessarily easy. There are subtleties to consider in the understanding of MDE success and failure, most notably that there are interesting differences in what might be considered a "technical" success and what might be regarded as an "organizational" success — even understanding this might add to an individual's, or an organization's, understanding of their own experience of using MDE on real projects.

7. Conclusions

What comes out of the online survey and the case studies very clearly, and particularly in the comments made by the interviewees, is the need to look beyond the technical benefits of a particular approach to MDE when trying to determine if deployment or adoption is likely to work, and instead concentrate on social and organizational issues. Not surprisingly, this places MDE adoption in the much wider context of organizational change and change management. This means that many of the findings are likely to resonate with those who are familiar with wider organizational change. However, it is important to note that as with any specific technological intervention, the details can differ. Much of what follows is general to any significant organizational change, but there are specific observations that are primarily relevant to MDE adoption and use.

However, recognizing this is both necessary for organizations to make the correct decisions about their MDE adoptions policies and processes and when evaluating their outcome. It might be helpful to distinguish between what might be called the "macro-organizational context", that referring to institutional policies and politics, and the "micro-organizational context", which refers to the practicalities of work organization — within the software development team, for example. As The Telecom Company showed, these experiences may vary significantly. Developing a nuanced view of success and failure of MDE in commercial contexts (as opposed to research contexts) obviously requires that the full social and economic circumstances surrounding the introduction and use of MDE should be clearly understood; an appreciation that systems are introduced and developed in accord with a range of organizational, managerial and local priorities and policies; priorities and policies that are themselves subject to pragmatic adjustment change and may even, in some instances, be in conflict.

Finally, what this multidisciplinary and eclectic methodological approach emphasizes is the importance of a series of social and organizational issues loosely concerned with "management" and especially the management of change. Management in a period of change is a complex and difficult process, especially when the changes – organizational, technological and cultural (and there is a strong sense in which MDE involves a cultural change) – are being introduced concurrently. Any organization involved in complex changes of culture, structure and technology faces a number of difficulties such as reconciling what can sometimes turn out to be incompatible organizational or departmental goals.

A positive aspect of looking at the survey and case studies in this article is that it is possible to identify, and therefore highlight, a number of specific issues that organizations, and the people within in them, should both be aware of and should address when considering the adoption of MDE: for example "Do we have the resources available to properly support MDE adoption?" or "Can we identify a small sub-project on which to initiate the adoption process?" Issues such as these should be considered before committing to a particular process change.

A less positive aspect is that there currently exists very limited previous, or even no, understanding of how these organizational issues relate to the technical aspects of MDE and therefore which present which requirement for different organizational response. This work does help, but it is clearly an area for further research. For example, the differing abilities of existing software developers to adapt to working at a higher level of abstraction is, as shown elsewhere in this thesis, an issue that is widespread but not particularly well understood. This is the kind of issue that could well result in the unforeseen need for extra resources to carry out the extra training that is required. And that, in turn, could upset the financial calculations on which MDE adoption is based. The important point is that MDE is a technical intervention, but the impacts are far, far wider. Understanding the relationship between the organizational/personal aspects of adoption that the positive impacts of MDE use is something that requires more study.

The consequences of these sorts of arguments are potentially huge. There is currently no way of knowing who may or may not have difficulties when presented with the demands of MDE — whether those be ability or attitude-based. This lack of knowledge requires even more flexibility in terms of response. The organizational response may be crucial because the unwillingness or inability to provide those extra resources may have serious knock on effects.

Acknowledgments

Our thanks go to the many MDE experts who agreed to speak to us about their experiences (although we have presented only four case studies, they have been chosen to represent much broader experiences). We also acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) who funded this research: EP/H006249/1.

References

[1] M. Afonso, R. Vogel, J. Teixeira, From code-centric to model-centric software engineering: practical case study of MDE infusion in a systems integration company, in: Workshop on MBD/MOMPES, 2006.
[2] B. Anda, K. Hansen, I. Gullesen, H. Thorsen, Experiences from introducing UML-based development in a large safety-critical project, Emprical Software Engineering 11 (2006) 555–581.
[3] E. Arisholm, L. Briand, S.E. Hove, Y. Labiche, The impact of UML documentation on software maintenance: an experimental evaluation, IEEE Transactions on Software Engineering 32 (2006) 365–381.
[4] E. Arisholm, L.C. Briand, B.C.D. Anda, First workshop on empirical studies of model-driven engineering at MODELS 2008, in: CEUR Workshop Proceedings, 2008.
[5] I. Blanchette, K. Dunbar, Analogy use in naturalistic settings: the influence of audience, emotion and goals, Memory and Cognition 29 (2001) 730–735.
[6] L. Briand, Y. Labiche, M. Di Penta, H. Yan-Bondoc, An experimental investigation of formality in UML-based development, IEEE Transactions on Software Engineering 31 (2005) 833–849.
[7] J. Cruz-Lemus, M. Genero, S. Morasca, M. Piattini, Using practitioners for assessing the understandability of UML statechart diagrams with composite states, in: Advances in Conceptual Modeling: Foundations and Applications, 2007, pp. 213–222.
[8] J. Cruz-Lemus, M. Genero, M. Manso, M. Piattini, Evaluating the effect of composite states on the understandability of UML statechart diagrams, in: Model Driven Engineering Languages and Systems, MODELS, 2005, pp. 113–125.
[9] B. Dobing, J. Parsons, How UML is used, Communications of the ACM 49 (2006) 109–113.
[10] A. Forward, T. Lethbridge, Problems and opportunities for model-centric versus code-centric softwaredevelopment, in: Workshop on Models in Software Engineering (at ICSE), 2008, pp. 27–32.
[11] R. France, B. Rumpe, Model driven development of complex software: a research roadmap, in: Future of Software Engineering, IEEE The Computer Society, 2007.
[12] D. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, John Wiley & Sons, 2003.
[13] J. Ganssle, A trillion lines of code?, in: Embedded.com (21/9/08), 2008.
[14] M. Genero, M. Piattini, E. Manso, Finding early indicators of UML class diagrams understandability and modifiability, in: International Symposium on Empirical Software Engineering, ISESE, 2004, pp. 207–216.
[15] B.G. Glaser, A.L. Strauss (Eds.), The Discovery of Grounded Theory, Adlin, Chicago, 1967.
[16] J. Grudin, Why CSCW applications fail: problems in the design and evaluation of organizational interfaces, in: Proc. CSCW '88, ACM, New York, 1988, pp. 85–93.
[17] J. Hutchinson, J. Whittle, M. Rouncefield, S. Kristoffersen, Empirical assessment of MDE in industry, in: ICSE 2011, 2011, pp. 471–480.
[18] J. Hutchinson, M. Rouncefield, J. Whittle, Model-driven engineering practices in industry, in: ICSE 2011, 2011, pp. 633–642.
[19] P.N. Johnson-Laird, How We Reason, Oxford University Press, 2006.
[20] J. Kramer, Is abstraction the key to computing? Communications of the ACM 50 (2007) 37–42.
[21] C.F.J. Lange, M.R.V. Chaudron, Effects of defects in UML models: an experimental investigation, in: International Conference on Software Engineering, 2006.
[22] MediaDev, Wide gap amongst developers' perception of the importance of UML tools, 2005.
[23] Model driven development for J2EE utilizing a model-driven architecture approach: productivity analysis (White Paper), The Middleware Company, 2003.
[24] Model driven development for J2EE utilizing a model driven architecture approach: maintainability analysis (White Paper), The Middleware Company, 2004.
[25] P. Mohagheghi, V. Dehlen, Where is the proof? – a review of experiences from applying MDE in industry, in: Proc. 4th European Conference on Model Driven Architecture Foundations and Applications, ECMDA'08, in: LNCS, vol. 5095, 2008, pp. 432–443.
[26] A. Nugroho, B. Flaton, M.R.V. Chaudron, An empirical analysis of the relation between level of detail in UML models and defect density, in: Model Driven Engineering Languages and Systems, MODELS, 2008.
[27] P. Biernacki, D. Waldorf, Snowball sampling: problems and techniques of chain referral sampling, Journal of Sociological Methods and Research 40 (3) (2011).
[28] R. Razali, C.F. Snook, M.R. Poppleton, P.W. Garratt, R.J. Walters, Experimental comparison of the comprehensibility of a UML-based formal specification versus a textual one, in: Conference on Evaluation and Assessment in Software Engineering, EASE, 2007.
[29] J. Segal, Learning about the algebraic specification of abstract data types, in: W.D. Gray, D.A. Boehm-Davies, and J.C. Spohrer, (Eds.) Empirical Studies of Programmers 6th Workshop, Alexandria, Virginia, 1996, pp. 195–218.
[30] R. Soley, D. Frankel, J. Parodi, The MDA Journal: Model Driven Architecture Straight from the Masters, Meghan Kiffer Press, 2004.
[31] D. Thomas, MDA: revenge of the modelers or UML Utopia? IEEE Software May/June (2004) 22–24.