Empirical Assessment of MDE in Industry

John Hutchinson, Jon Whittle, Mark Rouncefield
School of Computing and Communications
Lancaster University, UK
+44 1524 510492
{j.hutchinson, j.n.whittle, m.rouncefield}@lancaster.ac.uk

Steinar Kristoffersen
Østfold University College and Møreforskning Molde AS
NO-1757 Halden
Norway
+47 6921 5000
steinar.kristoffersen@hiof.no

ABSTRACT

This paper presents some initial results from a twelve-month empirical research study of model driven engineering (MDE). Using largely qualitative questionnaire and interview methods we investigate and document a range of technical, organizational and social factors that apparently influence organizational responses to MDE: specifically, its perception as a successful or unsuccessful organizational intervention. We then outline a range of lessons learned. Whilst, as with all qualitative research, these lessons should be interpreted with care, they should also be seen as providing a greater understanding of MDE practice in industry, as well as shedding light on the varied, and occasionally surprising, social, technical and organizational factors that affect success and failure. We conclude by suggesting how the next phase of the research will attempt to investigate some of these issues from a different angle and in greater depth.

Categories and Subject Descriptors

K.6.3 [**Software Management**]: Software Process; D.2.2 [**Design Tools and Techniques**].

General Terms

Management, Design.

Keywords

Model Driven Engineering, Empirical Software Engineering.

1. INTRODUCTION

Model-driven engineering (MDE) is a term used to describe the systematic use of software abstractions – or models – as primary artifacts during a software engineering process [15]. 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 [4]. There are many examples of success stories with MDE, but there are also many cases of failure. As a result, companies deciding whether or not to adopt MDE are often faced with confusion: the success stories tend to paint things in their best light, whereas the failure cases may not have applied MDE properly. In short, there are a lack of guidelines for deciding whether and how to adopt MDE

This paper presents results from a twelve-month empirical study with the long-term goal of providing these guidelines based on industry evidence. The study investigated factors – technical, organizational and social – that affect how companies fare with MDE. The methodology was to apply qualitative research methods to understand when, how and why companies do or do not succeed with MDE. A three pronged approach was followed: (i) a questionnaire widely disseminated to MDE practitioners, which received over 250 responses; (ii) in-depth interviews with 22 MDE professionals from 17 different companies; (iii) on-site observational studies with companies practicing MDE.

The initial aims of the study were twofold: (1) to understand and document how MDE is currently being applied in industry; and (2) to identify the most important factors affecting MDE success/failure, with a particular emphasis on uncovering social, organizational and technical factors. The latter point is particularly important, as it is often the case that a new technique's uptake depends as much on social as technical considerations.

This paper presents results from the two methods mentioned above, in the form of a survey summary and a number of lessons learned, spanning a range of issues from education and training to process and tool support. As with all qualitative research, these lessons should be interpreted with care: they should be seen as providing a greater understanding of MDE practice and as offering hypotheses for future study.

The study takes a deliberately broad interpretation of MDE, as it is intended to be exploratory. Therefore, all variants of MDE are covered, including both domain-specific modeling languages (DSMLs) and UML-based methods. The scope of the study is defined by the responses of the participants. The only hard criterion for excluding/including data was that the company must have been using models as a primary development artifact. Usually, but not exclusively, this meant code generation from models.

It is also worth pointing out that the study is not an attempt to survey penetration of MDE in industry; most of the interviewees were experienced modelers, although not necessarily proponents of MDE. The study is also deliberately agnostic with respect to the usefulness of MDE; that is, it is as much interested in MDE failure as it is in success.

The paper is structured as follows. Section 2 presents additional motivation and related work on empirically assessing MDE. Section 3 describes our methodology. Section 4 presents results from the questionnaire. Section 5 describes lessons learned from the in-depth interviews. Section 6 discusses the conclusions of the study and areas for future work.

Table 1: Illustrative influences of MDE.

Impact Factor	Illustrative Examples of MDE Influences	
	Positive Influences	**Negative Influences**
Productivity		
• *Time to develop code*	*Reduced* by: automatic code generation.	*Increased* by: time to develop computer-readable models; implement model transformations, etc..
• *Time to test code*	*Reduced* by: fewer silly mistakes in generated code; model-based testing methods, etc.	*Increased* by: effort needed to test model transformations and validate models, etc.
• *ROI on modeling effort*	*Positive* influences of modeling*:* more creative solutions; developers see the "bigger picture".	*Negative* influences of modeling*:* "model paralysis"; distracting influence of models.
Portability		
• *Time to migrate to a new platform*	*Reduced* by: simply applying a new set of transformations.	*Increased* by: effort required to develop new transformations or customize existing ones.
Maintenance		
• *Time for stakeholders to understand each other*	*Reduced* since: easier for new staff to understand existing systems; code is "self-documenting".	*Increased* since: generated code may be difficult to understand.
• *Time needed to maintain software*	*Reduced* since: maintenance done at the modeling level; traceability links automatically generated.	*Increased* since: need to keep models/code in sync, etc.

2. BACKGROUND AND RELATED WORK

2.1 MDE Influences

The primary technical advantages claimed by MDE proponents are improvements in productivity, portability, maintainability and interoperability [15]. These are complemented by social and organizational benefits of abstraction such as the facility to see the "bigger picture" and positive influences on training since organizational knowledge is formalized within a code generator. These benefits have reportedly led to productivity increases ranging anywhere from 20 to 800% [18, 5]. On the other hand, there is anecdotal evidence that cites reasons for decreased productivity – quantified at 27% in [18] – such as poor tool support, a tendency to over-model ("model paralysis"), increased complexity, and the need to keep models/code in sync. The key conclusion from these contrasting experiences is that it is difficult to provide absolute measures of the benefits of MDE. In particular, MDE involves dependent activities, some of which have a positive effect and some a negative effect. For example, code generation in MDE appears, at first glance, to have a positive effect on productivity. But the need to integrate generated code with existing systems may lead to maintenance problems. How the balance between these two effects relates to context, and what might lead one to outweigh the other, is simply not known. Table 1 illustrates this point by showing aspects that may benefit from MDE but noting that, for any of them, there are positive and negative influences that compete against each other. A major goal, therefore, of an empirical assessment of MDE ought to be to unpick these dependencies and to identify contexts where the benefits outweigh the disadvantages (or vice versa).

Although MDE has been promoted as an approach for developing software for a significant time, there have been only limited attempts to assess its efficacy in industry in a systematic way. Below, we briefly report on these. In the absence of a systematic investigation, we are mostly limited to case studies, industry reports and similar published work. Mohagheghi and Dehlen [20] undertook an extensive review of such work, which we draw upon and extend here.

2.2 Mohagheghi and Dehlen's Review

Carried out as part of the European MODELPLEX IST project, Mohagheghi and Dehlen's study was primarily interested in the impact of MDE on productivity and software quality. Their methodology was a meta-analysis of the literature, using a set of inclusion criteria to select 25 papers published in quality conferences and venues between 2000 and 2007. 21 of these papers were experience reports from single projects; four describe comparative studies. Most of the papers present results anecdotally: two include some quantitative data, three papers used qualitative methods, and one attempted an experiment. As with this paper, Mohagheghi and Dehlen limit their study to approaches that generate "models, code and other artifacts from models": i.e., they focus on MDE rather than modeling more generally.

MDE was found to be applied in a broad range of companies in a number of different domains including telecommunications, business/finance applications, defence/aerospace and web applications. The experienced maturity of MDE was assessed on automation. Mohagheghi and Dehlen note varying reports from companies generating code where the degree of code generation ranged from 65% to 100%, or possibly some unspecified lower proportion. In some cases, this relied on developing DSMLs and/or code generators. Other papers described the generation of XML schemas and automation of testing, the latter resulting in a reduction of "box test cycle time" of almost a third. Executable models were mentioned in some papers, usually with reference to accompanying difficulties.

Software processes were recognized as being of integral importance in successfully applying MDE, yet mostly in the context of describing the inadequacy of existing processes when used for MDE. They note that none of the papers used existing model-based methodologies (e.g. KobrA [16] and COMET [7]).

The importance of suitable tools was reported as of crucial importance. There are two aspects to this: the first is that the techniques necessary to apply MDE correctly depend on tool support; the second is that within many of the significant software projects in which MDE was being applied, or for which it was being evaluated, there already existed a comprehensive tool chain and integration of the necessary tools into this chain was essential.

Few of the papers surveyed provided strong empirical evidence of the impact on productivity. Of the comparative studies, results vary between a 35% gain to a 27% loss. Two of the studies showed no net impact. Further claims were made about productivity gains which were not supported by evidence from comparative studies. Mohagheghi and Dehlen suggest there is a more subjective quality to these reports and that they often emerge from environments lacking a "common baseline". Importantly, though, some of the productivity gains described are far in access of those discussed earlier: 2x, 5x and even 8x productivity improvements. The study also looked for evidence that MDE improves software quality. There were descriptions of defect reductions, reduced need for code inspections, and maintenance gains, but the evidence was anecdotal.

In conclusion, 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. They also hint that immaturity is still an issue. For example, they cite improved portability as a potential benefit of MDE but that its realization is hindered by tool platform support. They do note, though, that most papers consider models helpful in improving understandability and communication between stakeholders.

2.3 Other Related Work

Other work empirically assessing the benefits of MDE is limited. In particular, there are 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.

Regarding the lack of knowledge on how MDE is used in industry, there have been only a very limited number of attempts to survey existing MDE practice. A 2005 study looked at the penetration of UML and UML tools into the marketplace by surveying 500 developers [19] but focused on UML not MDE. Forward and Lethbridge [9] did survey practitioners' opinions and attitudes towards MDE. Surveys like these are important because they often discover that commonly-held perceptions may not hold true in practice [8].

Regarding the lack of understanding of how social factors affect MDE use, most empirical studies have concentrated on technical aspects of MDE. 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. Afonso et al [1] documented a case study to migrate from code-centric to model-centric practices.

There has been quite a lot of research on evaluating the language UML (e.g., [3, 17]) but this is not relevant to this paper, which focuses on MDE.

3. METHODOLOGY

We employed an eclectic range of research techniques to gather information for our study in an attempt to capture different kinds of data and different ranges of experience with MDE. In a scoping study such as this, the use of a variety of data collection techniques – both quantitative and qualitative methods – ensured a widespread coverage and capture of those factors that might impact on the success or failure of MDE approaches as well as a diversity of data types and, more importantly, data sources. This approach allowed us to ask a series of questions about aspects of MDE that were already apparent in the existing literature as well as surfacing and documenting other aspects of experience with MDE that were hidden. Quantitative techniques (questionnaire surveys) were designed to enable us to develop some overall sense of the number, frequency and generality of specific experiences, responses and informed conclusions about MDE. Qualitative approaches (in our case semi-structured in-depth interviewing and ethnographic case studies) encouraged our respondents to reflect on their experience and expertise and enabled us to explore in greater detail and depth specific aspects of our respondents experience of MDE as well as attempting to capture some 'sensitivities' [6] concerning the everyday practical realities of the deployment of MDE.[1]

3.1 Interpretation of Results

We want to be particularly careful about how we interpret the results of our work and the conclusions we try to draw from our admittedly limited, though thorough, study. Accordingly, at this very early stage of our work, we argue simply that our results are interesting and suggestive rather than conclusive; that they provide some sensitivities to the MDE experience rather than indications of causal links between variables, sensitivities that will be unpacked and developed as the research progresses. Analysis of our questionnaire data, at this early stage, has involved some basic enumeration and simple statistical calculations to get some overall sense of MDE deployments. In contrast, our interview data has been analyzed using a 'grounded theory' approach [12, 23] looking for patterns, themes, and categories of analysis to emerge out of the data (the transcribed interviews), rather than being imposed prior to analysis. In this way, we attempt to capture the meaning or experience of MDE from the varied situations and contexts of our respondents. As many of our respondents admit, quantification of the benefits and failures of MDE is complex and difficult. Whilst we acknowledge the argument for the need for more quantification and metrics, the evidence for our understanding of MDE at this stage derives from the quality and perceptiveness of our descriptions and our analysis rather than any simple mathematization.

3.2 Forms of Investigation

3.2.1 *Questionnaire*

The questionnaire was implemented online using Survey Monkey [24] and comprises mostly closed questions, using both multiple choices and Lickert scales for answers. In the questionnaire's preamble, it was stressed that our target community was industrial practitioners with experience of using modeling in industry. Based

[1] Details of the questionnaire and interview questions are available on the project website: http://www.comp.lancs.ac.uk/~eamde

on experience of similar types of surveys, we judged that the questionnaire should take no longer than approximately 15 minutes to complete and it was designed accordingly.

Although we wished to employ the questionnaire to elicit information about modeling practices and related matters, we were primarily interested in exploring the balance between the types of positive and negative consequences of MDE use that were illustrated in Table 1. For that reason, a significant part of the questionnaire comprised a series of "paired" questions designed to explore these issues. Examples are given in Table 2.

Table 2. "Paired" questions.

MDE Aspect	First Question	Second Question
Training	Does using MDE allow you to employ developers with less software engineering experience (e.g. new graduates)?	Does using MDE require you to carry out significant extra training in modeling?
Code generation	Is your use of code generation an important aspect of your MDE productivity gains?	Is integrating generated code into your existing projects a significant problem?

Note that the questions are not contradictory – nor are they "trick" questions. Our aim was to understand something about the balance between positive and negative consequences of MDE.

3.2.2 Interviews
We conducted a series of semi-structured in-depth interviews by telephone with a number of respondents from a variety of backgrounds and with a range of opinions about MDE. We were particularly interested in attempting to balance those who had positive experiences of MDE with others who clearly had less enthusiastic accounts. The interviews lasted approximately 45-60 minutes and began with a question about the current position and history of the respondent; there then followed a series of questions that probed particular aspects of MDE experience. As with all semi-structured interviews we had a list of general topics we wanted to cover that included their particular approach to MDE, their motivation for deploying MDE, their ideas about benefit, success and failure, lessons learned and so on, but we particularly encouraged our respondents to provide more detail or supporting information about any response that they, and we, found interesting. Our respondents were allowed to "follow their interests" in order to obtain the rich detail that is the chief benefit of this approach. The interviews were recorded and transcribed for analysis with the permission of our respondents.

3.2.3 Case Studies
A final aspect of our research approach, that does not form part of this paper, is the use of ethnographic, observational methods [22] to explore the lived reality of MDE; combined with forms of cultural or informational probes [11, 13] to uncover more personal details of the everyday, mundane practice of working within an MDE project.

4. QUESTIONNAIRE
We publicized the online questionnaire in a number of different ways. A number of personal emails were sent to industrial contacts drawing attention to the questionnaire's presence. Announcements were made on leading MDE and software engineering email lists (PlanetMDE and SEWORLD respectively). Finally, a notification and link were hosted on the home page of the Object Management Group's (OMG) website. (The OMG is the body that standardizes UML and promotes Model Driven Architecture (MDA) – a specific MDE approach.)

The responses to questions about personal experience, roles, company size and MDE maturity tell us that the vast majority of respondents have significant software engineering experience (e.g. 44%>10 years), that they are employed in a range of different roles (e.g. 36% developers/modelers and 37% team leaders/project managers), that there was a good spread of size of company with respect to the number of people involved in software development (e.g. 52%<100 and 19%>1000) and that they were experienced in using MDE (60% having completed multiple MDE projects).

83% of respondents think that MDE is a good thing and 5% that it isn't. The remainder were neutral or unwilling to commit. This suggests that our respondents represent an experienced and generally successful community of MDE users and provides an important context for the answers to other questions.

The majority of respondents considered the use of MDE on their projects to be beneficial in terms of personal and team productivity, maintainability and portability (58-66%). However a significant number disagreed (17-22%). When asked if MDE was a success, 58% agreed and 20% disagreed. This latter figure is significantly higher than those who question whether MDE is a good thing and suggests that a good number of people have yet to implement it successfully.

4.1 Modeling Languages and Their Use
MDE users employ multiple modeling languages. Almost 85% of respondents make use of UML and almost 40% use a DSL of their own design. A quarter of respondents report using BPMN; a similar number use a DSL provided by a tool vendor and about 10% claim to use each of SysML and MATLAB/Simulink.

The types of modeling carried out by the survey's respondents are shown in Figure 1, which has an inclusion threshold of >2. It is clear that *Class*, *Activity*, *Use Case*, *Sequence* and *State Machine* diagrams are the most popular used. However, what Figure 1 does not show is the wide range of diagrams used by smaller numbers of people, which in our survey numbered over 35 and included *Abstract Syntax Graphs*, *Arrow Diagrams*, *UsiXML Stylistics* and *ICONIX Robustness Diagrams*.

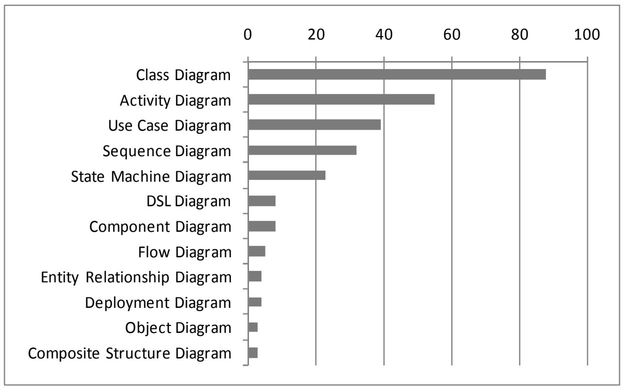


Figure 1. The most popular diagrams used.

4.2 MDE Activities and Their Impact on Productivity and Maintainability

Recognizing that MDE potentially means a different collection of activities for different MDE users, we were keen to understand the value of the different activities when it came to their impact on productivity and maintainability. The activities were selected to represent different interpretations of what MDE is, and different levels of commitment to MDE. Although respondents were able to indicate that they didn't use a particular activity, they were otherwise limited to saying that an activity increased or decreased productivity/maintainability or made no difference. Table 3 shows the proportions of respondents who judged each activity to *increase* productivity and maintainability, along with those who do not use that particular activity. This summary shows very clearly the levels of uptake of the different MDE activities. The proportion of respondents judging each activity to have a positive effect on productivity/maintainability was the majority response in *all* cases. (NB "Not used" figures vary slightly because slight differences occur between the groups of people answering each question.)

Table 3. The impact of MDE activities on productivity and maintainability.

Activity	Productivity		Maintainability	
	Increased	Not Used	Increased	Not Used
Use of models for team communication	73.7%	7.0%	66.7%	6.7%
Use of models for understanding a problem at an abstract level	73.4%	4.8%	72.2%	6.1%
Use of models to capture and document designs	65.0%	9.3%	59.9%	10.7%
Use of domain-specific languages (DSLs)	47.5%	32.6%	44.0%	33.7%
Use of model-to-model transformations	50.8%	24.6%	42.6%	28.4%
Use of models in testing	37.8%	33.9%	35.2%	32.4%
Code generation	67.8%	12.0%	56.9%	12.6%
Model simulation/ Executable models	41.7%	38.3%	39.4%	35.9%

4.3 MDE "In the Balance"

The paired questions, as described above, were intended to explore the potential positive and negative consequences of using MDE.

4.3.1 Training

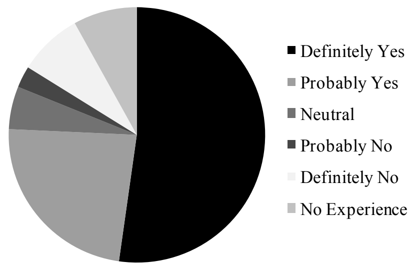
Over 47% of respondents think that using MDE allows them to employ less experienced software engineers whilst almost 35% disagree. In contrast, an overwhelming 74% think that using MDE requires them to carry out significant extra training (<9% disagree). Of course, these are not contradictory; MDE represents a need for new skills for most software engineers and regardless of who those engineers are, they are likely to require extra training. However, it seems to suggest that MDE adopters should recognize the need for this training even if they are able to make modelers out of less experienced software engineers.

4.3.2 Responding to Requirements Changes

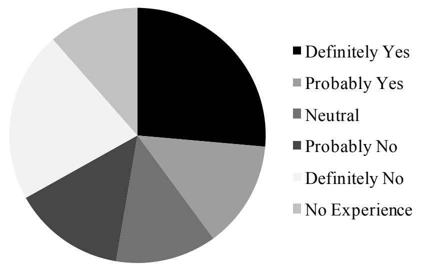
Almost three quarters of respondents agreed that using MDE made them faster at implementing new requirements. 11% disagreed. When asked if MDE *prevented* respondents from responding to business opportunities, the responses were far more ambivalent, with 32% agreeing and 38% disagreeing. This suggests that within a project, MDE users consider modeling helps to make them flexible to requirements changes, but that they do not think that this flexibility has wider benefits in responding to new opportunities.

4.3.3 Code Generation

As we saw in Table 1, it is easy to imagine positive and negative consequences of code generation. Figure 2 illustrates how respondents answered the questions about code generation in our paired questions. What is immediately obvious from Figure 2 is that there are more respondents for whom code generation has a positive impact on their productivity than there are those for whom the integration of generated code is a problem.



(a)



(b)

Figure 2 (a) "Is your use of code generation an important aspect of your MDE productivity gains?" (b) "Is integrating generated code into your existing projects a significant problem?"

The variation in the difficulty experienced by MDE users integrating generated code is almost certainly significant in the experiences of different MDE users. The *explanation* is far less obvious. It may be differences in the process used, or differences in the requirements of those projects on which is it used.

4.3.4 UML

There exists significant ambivalence about the balance between UML's complexity and its power. 43% of respondents think that UML is too complex compared to 32% who disagree. However, 23% are neutral on the matter which suggests an implicit understanding of the compromises that any general modeling language must embody. 50% of respondents think that UML is powerful enough for their needs compared to 32% who disagree.

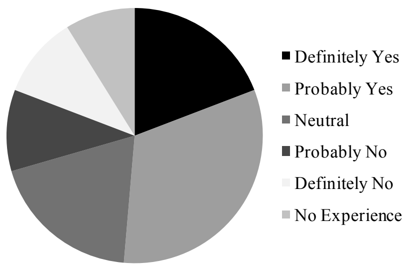
Understanding the needs of MDE users who believe UML is not powerful enough is clearly a question worthy of further research.

4.3.5 Round-Trip Engineering

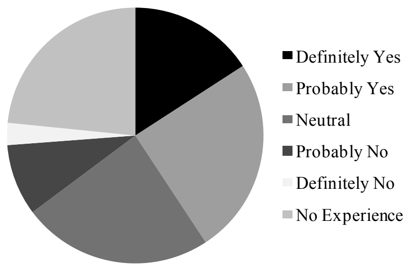
The attitude to, and the role of, round-trip engineering within a company's use of MDE, speaks about that company's interpretation of what MDE is. Over 70% of respondents say they mainly make updates on their models. Just under 40% say they spend a lot of time synchronizing their models and code whereas just over 40% say they don't. This is another subtly process-oriented question which leaves interpretation difficult, not least because there is a similarity between the distributions of answers for these questions and those illustrated in Figure 2. The majority of MDE users do try to leave generated code alone, but keeping code and models synchronized is clearly an issue and how that issue is addressed within an organization's development process may be critical to their successful use of MDE.

4.3.6 Reasons to Model

Not all MDE users are convinced that organizations adopt MDE for purely technical reasons. Figure 3 shows the answers received to questions asked about this issue.



(a)



(b)

Figure 3 (a) "Do organizations adopt MDE for its technical merits?" (b) "Do organizations adopt MDE to "jump through hoops" or appear to do so?"

Figure 3 indicates that approximately 20% of respondents openly question the adoption of MDE on its technical merits and roughly 40% think there is an element of "jumping through hoops" to the use of MDE. If organizations adopt modeling or MDE techniques to "appear to do so", it suggests that some MDE users' confidence in, and commitment to, MDE is far from complete. It also suggests that some form of non-product-oriented pressure is being applied to organizations to motivate that adoption of MDE. Within the context of other responses, these are interesting issues.

4.3.7 Understandability

Two thirds of respondents think that their use of MDE helps understanding between stakeholders whilst almost a quarter believe that their use of MDE results in unexpected confusion and/or misunderstanding between stakeholders. These contrasting experiences may result from the likelihood that the progress of an MDE project will follow a different trajectory from one that is code-centric and this may cause misunderstandings. However, it is intriguing to note that more than 6% of respondents say they have no experience of improved or reduced understandability when using MDE.

4.3.8 Tool Costs

Although it may be theoretically possible to imagine an MDE process that is not reliant on tools, all practical deployments require tool support. 43% think that MDE tools are too expensive (24% disagree). Conversely, 56% of respondents think that organizations try to deploy MDE using inappropriate and/or cheap tools (12% disagree). Even allowing for a reasonable number of tool vendors in the survey sample, this indicates a significant level of disaffection with the tools used for MDE alongside a belief that the tools on offer are too expensive.

Against this background, it is interesting to note that more than 50 different tools are used by the survey's respondents, which suggests a lack of maturity in that definitive market leaders are yet to emerge.

4.3.9 Paired Question Summary

The answers to the paired questions show a fascinating balance between the positive and negative consequences of using MDE in industry. Moreover, they illustrate how easily what could otherwise be a successful attempt to adopt MDE could be fatally undermined by some process decision or incorrect assumption. For example, a company that believes that MDE will enable them to hire software engineers with less experience but doesn't prepare for the necessary training may well find itself in a difficult predicament and with a need for training time and costs that have not been planned for. Similarly, if for some legitimate reason a company finds that it must manually adapt code that is auto-generated, it will probably need to explicitly address how it should be done and what procedures are required to control the process if it is not to encounter difficulties with keeping the model and code synchronized.

The paired question answers suggest that companies which successfully adopt MDE will make numerous small and not-so-small decisions that maximize the benefits in their particular context. In contrast, decisions that negate the benefits may result in a failed MDE adoption attempt.

4.3.10 Interesting Correlations

We have looked for correlations (Pearson's product moment correlation) in some of the questionnaire data that might help to identify issues that require more investigation. Some of these reveal patterns that are quite interesting. For example, there is a significant ($p<0.01$) and medium negative correlation ($r=-0.32$) between respondents' perceptions of UML complexity and powerfulness. In other words, the tendency is that those respondents who find UML too complex also find it not powerful enough.

There is a significant ($p<0.01$) and medium ($r=0.345$) correlation between the belief that the use of MDE results in unexpected confusion and/or misunderstandings and the perception that organizations adopt MDE simply to "jump through the hoops".

This contrasts with the correlation between the belief that MDE leads to better understanding between stakeholders and the perception that organizations adopt MDE on its technical merit (r=0.49, p<0 .01). This may indicate that the understandability improvements achieved by some organizations are central to their perception of MDE use and success.

5. INTERVIEWS

We identified appropriate interviewees by focusing on their experience within industry. We asked academic contacts, and any others that we had, to identify practitioners in industry who might be willing to be interviewed for the study. This equates to an application of a type of snowball sampling, or respondent driven sampling, where the first generation contacts were disregarded because of their status as academics. Although we provided an introduction to the project, we left it to our contacts to make the request on our behalf, all of which were made by email. The following extract is from one of those requests:

> *"...I am contacting you because of your experience in applying MDE in practice and I am hoping that either you or someone you feel is equally qualified can participate in this study by answering to their call for participation (30 minutes of your time for a phone interview). The more people participate, the more accurate and the more useful the results..."*

This technique allowed us to get direct access, with introductions, to a number of extremely experienced practitioners and other recognized experts in the field. A small number of interviewees, meeting similar criteria, were identified in other ways; for example, opportune meeting of a keynote speaker at a conference, informal contact with a consultant in the area, etc.

5.1 Interviewee Profile

At the time of this analysis, we had carried out more than 20 interviews. In total, interviewees represented about a dozen roles in 17 different companies in about a dozen domains. The data totaled almost 20 hours of recorded data and >130,000 words of transcribed data. Cumulatively, our interviewees have >360 years software development experience.

5.2 Analysis

It is important to acknowledge that the data set is so large that no single analysis can capture anything other than a fraction of it. For this reason, here we attempt to present a number of "key themes" that represent a sort of "lessons learned". These themes appear in multiple interviews and refer to issues relating to technology, process and organization. The range of interpretations, activities, tools and processes is vast, so each of the following themes is essentially based upon a collection of case studies.

5.2.1 A Lot of MDE Success is Hidden

> *"...when you build an enterprise system, do you program all the low-level database stuff...the B+ trees... or do you define a data model and feed it into a database management system...are you constantly writing raw HTML or Java code or are you using tools that allow you to paint a dialog box and generate some code... for the front and back ends of our systems, we long ago abandoned low-level coding as a dominant abstraction of development..."* (Transcript: 16)

We have already seen how different activities can contribute to a process that is considered model-driven and those engaged in MDE may employ some number of these activities. An external and artificial view of MDE might suppose that MDE necessarily will carry out *all* of the activities all of the time. Furthermore, they might also suppose that MDE in practical terms equates to UML and the use of certain types of tools. However, practical MDE is often not quite like that. As the above quote suggests, the development of a typical 3-tier architecture is far from the code-centric activity that many programmers claim to be the only valid development abstraction. Much of MDE in industry is the kind of modeling and/or automation that represents pragmatism in the face of an otherwise tedious or intractable problem.

A related issue is the maturity and suitability of commercial tools and standard notations. Some users believe that had they adopted off-the-shelf tools, it would effectively have killed that adoption of MDE. This is a stark contrast when compared to tools routinely used today for non model-driven tasks.

5.2.2 Choosing the Correct Project on Which to Introduce or Trial MDE

> *"...we put it directly in the main line of the product so it was a pilot but we are not allowed to fail. It sounds a little bit strange but we didn't have the capacity effort to have a parallel project...we took some risks in introducing model driven design..."* (Transcript: 7)

When a company has made a decision to use MDE, or more likely try out MDE, the choice of project may have an enormous influence on its success or failure. This appears to be linked to notions of motivation. The MDE users must be motivated to use the new approach and the organization must be motivated to make the project a success. Unlike process changes that involve different development paradigms, languages or tools, MDE challenges embedded perceptions of what software development actually is.

Motivating Process Change

The aphorism "If it ain't broke, don't fix it" captures the notion of user motivation quite well. In many instances, if the process already employed by a company is working "sufficiently well", it will be very difficult to introduce as radical a change to that process as introducing MDE. However, if the existing process is itself a significant risk to a project, or if that process involves the developers in difficult, time-consuming and uninspiring activities, then process change in the form of MDE adoption will be more readily embraced.

Motivating Project Success

Intuitively, many potential adopters of MDE consider prototyping it on a part of a project that is isolated from important deadlines. However, many successful MDE adopters become so by using it on the critical path of a product development program. This ensures that the organization makes the necessary commitment to the project (e.g. using the best people).

5.2.3 Not Everyone Can Think Abstractly (or Wants to)

> *"So the question is, is somebody naturally inclined to think in terms of design or do they think more in terms of detail? ...I don't think that the education that people get necessarily gets them thinking towards design"* (Transcript: 2)

"There are people who you can't teach this to because they think always in examples." (Transcript: 4)

Software engineering comprises many different activities even if they are predominantly attributed to "developers". Even when not fully actualized, roles such as requirements engineer, system analyst, architect, etc. will often exist tacitly. This reflects the needs for differing abilities in the development process. Not all developers find making the transition to modeling straightforward. However, there are differing views about whether the difficulty is a result of an innate inability to think abstractly, or more a lack of experience or appetite. The situation is further confused by the fact that there are widely differing views on *who* finds it difficult; some believe that newly trained graduates who have been exposed to modeling in their education are ideal for MDE and that established "gurus" are often not ideal. Others believe that the very best programmers also make the best modelers. Some maintain that it is having the right mix in the team, from inexperienced software engineers to gurus to domain experts, that is crucial. The idea that this is a cultural rather than a technical issue is epitomized in the following quote from one of our interviewees, asked about how people responded to the apparent benefits of MDE:

"it's a mentality thing.. there are people who just don't want to change.. who see the negatives in everything..(though) sometimes the skepticism is realistic.. the most annoying (are) people who can't extrapolate from what they've seen to another environment.. you have to prove it over and over again.. then there are people who just don't get it.. why should you model in the first place?..they are people you can't teach this to. ..they always want examples, they are unable to abstract. It's the wrong kind of mindset.. then there are people who are negative and cynical and don't want to change.." (Transcript: 4)

5.2.4 Training, Education and Related Perceptions

"what we need is a community of programmers, well modelers, that really know how to abstract and that know then how to apply the tools that are available...we need a few who can abstract and many who are focused on support...Because if I was a business man I am going to say 'you want me to put my business in the hands of two or three people who might leave tomorrow are you mad? 'I'd rather have 10,000 Java programmers please'" (Transcript: 5)

MDE appears to upset the *status quo* and introduce more uncertainty where there was previously perceived to be less. Although this is a part of any change, it appears to be considered a significant risk by many in the software development world. The coverage of MDE in educational courses is considered inadequate by many – some even question whether higher education even teaches "modeling" as such and this means that trained junior modelers are far rarer than trained junior programmers. If programmers are the accepted currency of software companies, modelers are an unknown quantity. The result is a myriad of relatively small but legitimate risks embedded in significant conservatism and/or skepticism in the face of proposed change.

5.2.5 Keep Domains Tight and Narrow for DSLs

"...the broader the domain you try and cover, the less the productivity increase...the whole idea is to narrow things down...I should say, with DSM, you're not looking at building a modeling language for embedded applications...you're going narrower...you're not looking at building one for mobile applications...mobile embedded, or even mobile phones or even Brand X mobile phones but a particular product family of Brand X mobile phones..." (Transcript: 1)

What is the 'D' in a DSL? Typically, people give examples such as "insurance" or "banking". Repeatedly, we have found that successful users of DSLs create them for much narrower domains: a single product line or even a single product. A correlate of this appears to be that the creation of a DSL in an appropriate domain, along with accompanying code generators, must be accomplishable in a relatively short period of time. If creating the DSL is taking many months, or even years, it may be that the domain is not appropriate. Similarly, if an organization starts its DSL adoption process by setting out to capture in the language the entirety of their "world", enthusiasm for, and commitment to, the project will likely have faded before anything productive occurs.

Factors that indicate the suitability of a domain include: a degree of stability with recognized concepts; the domain is clearly bounded; the domain evolves over time (rather than radically shifting). For these reasons, DSLs are particularly successful when developing embedded software.

5.2.6 Successful MDE Users Often Have to Lie

"A: Well we've seen cases of 4 or 5 times productivity and with DSLs we've seen 7 or 8 times productivity.
Q: And what do you tell people?
A: 1.5 to 2...you cannot associate those higher numbers with [my company]" (Transcript: 8)

The old saying "If it sounds too good to be true, it probably is" can be a problem for some more successful users and proponents of MDE – the productivity gains claimed can seem so large as to be simply unbelievable. *"We had supportive management because it was an easy sell to management to say I used to type this thousand lines of code and it took me a couple of days and now I generate it in 10 seconds"* (Transcript: 18). Some argue that the motivation to use MDE will only be significant if the benefits are large; small increases in productivity will never outweigh the risks associated with the change required. However, some users are so successful that when they try to report their results, colleagues simply refuse to believe them. Perhaps the most interesting aspect of this is that where it is encountered, the high levels of productivity gains claimed are surprisingly similar. Of course, productivity gains are not the only or even necessarily the most important benefit of MDE. As one of our respondents commented:

"So the idea of applying a sort of methodology approach to modeling and code generation was really attractive – it seemed like it was an obvious next step to take in insuring quality and consistency and again not solving the same kinds of problems over and over again. And for me it wasn't necessarily productivity enhancement although I guess that is there.... I've never really been completely convinced or even all that worried about the productivity side of model driven engineering. To me it's more about quality and consistency." (Transcript: 18)

But what these comments do point to is the need for some subtleties in organizational knowledge, and in the politics of the organization in order to foster and maintain support.

5.2.7 Companies That Don't Do Software Do MDE

"At the moment, model driven approaches is not so much intended for internal use - not for our own software development. In our own software development, we have a more, let's say, classical approach because there's a long history of bad experiences [with, for example, CASE tools] and so for traditional software development, at the moment it is very difficult to introduce new [approaches] like model driven engineering" (Transcript: 12)

Organizations that consider themselves to be in the business of software engineering appear to find process change, and particularly the adoption of MDE, a bigger challenge than those for whom software development is subsidiary to some other function. For example, companies that produce plant control systems or electronic hardware must create software to control their devices but appear to consider their main business the hardware product. These companies are more willing to embrace MDE as a way of rationalizing and improving their software production because it represents a means to an end. By contrast, software engineering organizations tend to see MDE as something for the customers to do and develop tools to support it using traditional, code-centric, approaches.

5.3 Lower-Level Analysis

In contrast to the higher level data-driven analysis presented above, we also carried out a lower-level analysis of the data where we applied a series of thematic labels, which emerged from the data, to segments of text. As expected, many of the dominant themes are related to issues described above, including processes and practices, adoption process, risks, etc. However, other themes appeared which were a little more unexpected. Some of these are described below, along with example comments.

5.3.1 Culture

"It is rarely the case that it is business driven today" (Transcript: 2)

"I think creating a software engineering organization that hasn't done formal software engineering is definitely a much easier task" (Transcript: 6)

Factors relating to the way things are perceived to be within organizations: as in all walks of life, perceptions about "the way things are" are rife within software engineering. These perceptions, true or imagined, have an influence on decisions.

5.3.2 Expertise

"I don't think the level of adoption is significant. I would say still there is not a lot of experience in the industry" (Transcript: 2)

"I never thought about how important it was that the domain expert is integrated into the construction of the DSL" (Transcript: 4)

Issues relating to the availability and application of the necessary knowledge: all process change requires expertise but MDE appears to represent more than evolutionary change to some organizations. It seems to be necessary to identify the need for expertise and to deploy it where it is needed.

5.3.3 Evangelism

"I would say that it is predominantly driven by somebody who I would call a senior architect". (Transcript: 2)

"[This guy] was fantastic ... he went over to individuals one person at a time because it can't just come down as a managerial dictate" (Transcript: 6)

The role and impact of specific people in a company's decision to adopt MDE: this is probably related to the idea of expertise. However, such is the personal investment to what being a software developer is and the mental challenge that MDE represents, that a specific individual often plays a key role in facilitating the required changes. As one respondent stated:

"Somehow in my mind the question always comes back to this – the individuals. And the right individuals being in place, and having the right leverage, if you like. And that's where I think the ones that were successful. I have not actually seen it fail ever when you have those right people in there, and again just to repeat what I mean by right people, these are opinion leaders, they're open minded, they're interested in the product and how it gets used and they are can-do types. So that's right now I think the necessary factor for the success of these things.... where it has failed is where there is no such competent, respected, can-do open-minded individual or group of individuals" (Transcript: 21)

6. CONCLUSION – EMPIRICAL ASSESSMENT OF MDE

Although MDE claims many potential benefits in terms of gains in productivity, portability, maintainability and interoperability, it has been developed largely without empirical support for these claims. This paper stands in contrast to previous studies in that we have chosen to talk directly to those involved in MDE in business. This approach is intended to address particular, key, gaps in our current understanding: a lack of knowledge of exactly how MDE is used in industry; a lack of understanding of how social and organizational factors impact on 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 consequence of this lack of knowledge is that organizational decisions whether or not to use MDE are rarely based on hard empirical data but diverging expert opinions. Without empirical evidence there is a danger that resources may be wasted and that software tools will fail to develop appropriately. Of course such empirical evaluation of MDE is hard [10], not least because the social and organizational aspects of MDE deployment and use, as opposed to mere technical factors, require an interdisciplinary methodological perspective. The approach we have adopted here marks the first step in the development of an initial evaluation framework that can be used by researchers and practitioners to measure and evaluate MDE deployment.

This paper has attempted to carefully document some of our early research into exactly how MDE is currently being applied in industry and to identify the most important social, technical and organizational factors affecting its success or failure. There are a number of points to note about the lessons we have learned but we focus on organizational and social factors since the success or failure of MDE is affected as much by cultural considerations as by purely technical ones. At this stage in our research we can point to a number of interesting conclusions, highlighting issues of implementation, communication, control and skill. Some research conclusions undoubtedly simply support existing research; some are quite surprising and even counter-intuitive and some are divided (if not, on occasion, divergent). Some findings point to specific issues with regard to DSLs or MDE tools, that, for example, they were expensive or needed to be used in specific ways. Some of the more significant results pointed to organizational or human factors/training issues in the success or failure of MDE [25]. Organizationally the benefits of MDE were seen in terms of communication and control – responding quickly to changing requirements, and increasing and communicating organizational knowledge to stakeholders. Similarly, and perhaps unsurprisingly, our research supported some existing ideas about organizational ICT change, such as the need for a 'champion', for picking the initial projects carefully and for various forms of management intuition; as Grudin puts it: 'who does the work and who benefits' [14]. Finally, some of our findings have seemed genuinely surprising and merit further investigation, such as the level of productivity gains associated with MDE and the extent to which such success has been down-played, and the issue of abstraction and the importance of training and education. These latter views are especially worthy of further examination not least because they point to organizational matters and suggest that MDE users' confidence in, and commitment to, MDE is far from complete. Exactly what this looks like in practice as well as other facets of MDE work, of what an MDE project looks like on a daily basis and how success and failure manifests itself organizationally and unfolds in real time, will be uncovered by our forthcoming ethnographic, observational research.

7. ACKNOWLEDGMENTS

Our thanks to the hundreds of people who took the time to take part in our online survey and the many MDE experts who agreed to speak to us in person. Thanks also to Øystein Sørebø for his assistance with the statistical analysis and the EPSRC who funded this research: EP/H006249/1.

8. REFERENCES

[1] Afonso, M., Vogel, R., and Teixeira, J. 2006 "From code-centric to model-centric software engineering: practical case study of MDD infusion in a systems integration company," in *Workshop on MBD/MOMPES*.

[2] Anda, B., Hansen, K., Gullesen, I., and Thorsen, H. 2006 "Experiences from Introducing UML-based Development in a Large Safety-Critical Project," Empirical Software Engineering, vol. 11, pp. 555-581.

[3] Arisholm, E., Briand, L., Hove, S. E., and Labiche, Y. 2006 "The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation," IEEE Transactions on Software Engineering, vol. 32, pp. 365-381.

[4] Arisholm, E., Briand, L. C., and Anda, B. C. D. 2008 "First Workshop on Empirical Studies of Model-Driven Engineering at MODELS," CEUR Workshop Proceedings.

[5] Baker, P., Loh, P.S. and Weil, F. 2005 "Model-Driven Engineering in a Large Industrial Context - Motorola Case Study". In: ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML 2005). LNCS, vol. 3713, pp. 476–491. Springer, Heidelberg.

[6] Blumer, H. 1954 What is wrong with social theory? American Sociological Review, 18, pp. 3-10.

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

[8] Dobing, B. and Parsons, J. 2006 "How UML is Used," in Communications of the ACM. vol. 49, pp. 109-113.

[9] Forward, A. and Lethbridge, T. 2008 "Problems and opportunities for model-centric versus code-centric software development," in Workshop on Models in Software Engineering (at ICSE), pp. 27-32.

[10] France, R. 2008 "Fair treatment of evaluations in reviews," Software and System Modeling, vol. 7, pp. 253-254.

[11] Gaver, B., Dunne, T., and Pacenti, E. 1999 Cultural probes. Interactions: New Visions of Human-Computer Interaction, 6(1), pp. 21-29.

[12] Glaser, B. G. and Strauss, A. L. 1967 The discovery of grounded theory: Strategies for qualitative research. CHI: Aldine.

[13] Graham, C., Rouncefield, M., Gibbs, M., Vetere, F., and Cheverst, K. 2007 "How probes work. " In Proceedings of the 2007 Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces (pp. 29-37). New York, NY: ACM.

[14] Grudin, J. 1989 "Why Groupware Applications Fail: Problems in Design and Evaluation." Information Technology & People. Vol. 4, no. 3, pp. 245-245.

[15] Kleppe, A. G., J. Warmer, et al. 2003 MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley Longman Publishing Co., Inc.

[16] KobrA: http://www.old.netobjectdays.org/pdf/02/papers/node/0308.pdf

[17] Lange, C. F. J. and Chaudron, M.R.V. 2006 "Effects of Defects in UML Models: An Experimental Investigation," in International Conference on Software Engineering.

[18] MODELWARE D5.3-1 Industrial ROI, Assessment, and Feedback- Master Document. Revision 2.2 (2006),

[19] MediaDev. 2005 "Wide gap amongst developers' perception of the importance of UML tools".

[20] Mohagheghi, P., Dehlen, V. 2008 " Where is the Proof? – A Review of Experiences from Applying MDE in Industry". Proc. 4th European Conference on Model Driven Architecture Foundations and Applications (ECMDA'08), LNCS 5095, pp. 432-443.

[21] Randall, D., Harper, R and Rouncefield, M. 2005 "Fieldwork And Ethnography: A Perspective From CSCW". Proceedings - Ethnographic Praxis in Industry Conference Volume 2005, Issue 1, pp. 81–99.

[22] Randall, D., Harper, R, & Rouncefield, M. 2007 Fieldwork for Design: Theory and Practice Kluwer

[23] Strauss, A., and Corbin, J. 1990 Basics of qualitative research: Grounded theory procedures and techniques. Newbury Park, CA: Sage.

[24] SurveyMonkey: http://www.surveymonkey.com

[25] Völter, M. 2009 "MD* Best Practices", JOT - Journal of Object Technology, 2009-09.