Model-Driven Engineering Practices in Industry

John Hutchinson
School of Computing and Communications
Lancaster University, UK
+44 1524 510492
{j.hutchinson@lancaster.ac.uk}

Mark Rouncefield
School of Computing and Communications
Lancaster University, UK
+44 1524 510492
{m.rouncefield@lancaster.ac.uk}

Jon Whittle
School of Computing and Communications
Lancaster University, UK
+44 1524 510492
{j.n.whittle@lancaster.ac.uk}

ABSTRACT
In this paper, we attempt to address the relative absence of empirical studies of model driven engineering through describing the practices of three commercial organizations as they adopted a model driven engineering approach to their software development. Using in-depth semi-structured interviewing we invited practitioners to reflect on their experiences and selected three to use as exemplars or case studies. In documenting some details of attempts to deploy model driven practices, we identify some 'lessons learned', in particular the importance of complex organizational, managerial and social factors – as opposed to simple technical factors – in the relative success, or failure, of the endeavour. As an example 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.

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 Evaluation, Software Engineering.

1. INTRODUCTION
The complexity and pervasiveness of software in society is growing exponentially [1]. 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 software using appropriate methods of abstraction [2]. 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. 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 paper 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 [3]. 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 [4, 5] 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 [6]. 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 [7, 8] and psychology of programming specifically [9] 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICSE'11, May 21–28, 2011, Waikiki, Honolulu, HI, USA
Copyright 2011 ACM 978-1-4503-0445-0/11/05 ...$10.00

flavours 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. The right choice may be crucial to a project's success. 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 cataloguing exactly what works in MDE projects.

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 [12]. A 2005 survey study looked at the penetration of UML and UML tools into the marketplace [10] but focused on UML not MDE. Forward and Lethbridge [11] 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 [13, 14]. 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, 5, 18 21]; studies attempting to empirically measure the comprehensibility of UML models [8, 9, 19, 20, 22, 7, 25]; 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?

More recent studies have also lacked the kind of empirical, industrial experience, detail of MDE in use. France and Rumpe [10] provide an overview of research in MDE, pointing to some of the 'wicked problems' involved, rather than its industrial application. Mohagheghi and Dehlen [23] 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.

3. METHOD

Our approach is to tackle these key aspects of MDE using a multidisciplinary method. We approach the interdisciplinary challenge of the empirical analysis of MDE in industry by using an eclectic mixture of research methods – questionnaire surveys, observational methods and, as reported here, case studies constructed through semi-structured in-depth interviewing. 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 want 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'. In using semi-structured in-depth interviewing we ride a fine line between the individual, qualitatively rich, 'thick descriptions' that we hope to obtain through our ethnographic, observational studies of MDE projects at work, and the far greater quantity of data we have already obtained through our on-line questionnaire survey, a quantity that lends itself to various forms of statistical manipulation and inference. 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 our problem to one more easily addressed by our audience. After all, they should be able to identify the points of similarity and difference between their 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.

Respondents with experience of industrial MDE projects were identified from personal contacts and from an online survey promoted on leading software engineering and MDE mail lists and through a link on the Object Management Group's website. To date we have carried out more than 20 interviews with a range of MDE practitioners. The data collected amounted to over 20 hours of recorded interviews and over 140,000 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.

4. 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 case 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 and electronic engineering disciplines. This places constraints on the software development cycle and in the case of The Printer Company, software production was initially considered a bottleneck in the production process. This provided the initial motivation to explore different software development approaches.

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.

The Printer Company's Experience of MDE

The following extracts from The Printer Company case study illustrate key aspects of their experience:

> Q: When you started looking … were you just trying to look around for ways to improve the job … the way you thought the job could be done?
>
> *A: It was mainly a department assignment so we had a department for embedded software engineering and there are sixty engineers within that department and within the project where I was located … we started a pilot with model driven design … In 1998 it was a pilot*
>
> Q: And what sort of pilot was it? Was it a small sub project or was it something in parallel with your product development?
>
> *A: No 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 didn't 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 that 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 is also a factor in successful process change. 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.

> Q: And what were your expectations of using model driven design what were you hoping the outcomes would be?
>
> *A: So in the first project I didn't 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*
>
> Q: And what was your experience?
>
> *A: Indeed that we didn't 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*
>
> …
>
> Q: After that first project when you were moving on to other projects and you were benefitting from using the model driven techniques what sort of increase in productivity do you think you were achieving?
>
> *A: I have to explain something because the benefit in the second and third project was not only because we introduced model driven design but based on the experience in the first*

project we also started a re-use group … a group separate from the project and they developed components for more than one project and I think also model driven design was an enabler for implementing re-use … so we had model driven design, we had a re-use group and the third thing is that we also built a reference architecture so using the model drive toolset we started building up a reference architecture for all embedded projects and those three together really gave us the benefits we have at this moment

We see here that 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.

However, the most significant aspect of this segment of the interview is the description of how the company developed 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 re-use strategy. Ultimately, it led to the design, and adoption, of a reference architecture. The process by which this came about is explored further in the following segment:

Q: So I imagine all of those things together are inter-related - as you are able to use model driven development because you decide to move to reference architecture and because you have got a planned re-use strategy but what role do you think modeling played in actually enabling those things to come about?

A: It is a major role. I think if you didn't introduce model driven design the re-use 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

Q: Do you think that the introduction of modeling added a level of formality to your level of understanding or do you think it just allowed you to capture understanding that was already there?

A: The first … 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

Q: You can improve it once you understand what is already there?

A: Exactly

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 or 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.

A: 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 …

Q: Presumably that has a real commercial benefit that it allows you to develop your embedded software significantly faster never mind with less effort?

A: It is. Because in the past – I've been working here for twenty years - 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 expressing them. The final comment summaries 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 always on time.

5. 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 itself 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 process 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 replicated.
- *Business led* – the company did not embark on the process of MDE adoption for purely "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.

The Car Company's Experience of MDE

The following extracts from The Car Company case study illustrate important aspects of their experience:

> *A: All those organizations had to converge into a single entity and the way that they did that was bringing them altogether and focus them on a single design concept as opposed to a lot of them had their own unique design internally. And that was … a pre-cursor in terms of how the organization changed … you had two different forces there: one is how do you transform an organization that is in place to be able to do this more centralized engineering activity which is very dynamic as well as very contentious in terms of how it takes place - because you have a lot of people who did the same job that were now converging into single jobs - so how people redefined their job criteria seemed to change dramatically. And then we had this software organization … we knew that we were was growing because we recognize that … 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 re-use … 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 within terms define the reusable assets, find 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). The following segment illustrates the background of this further:

> *A: Now I mean understanding the automotive culture primarily when you are talking about 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 couldn't 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 weren't 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 in terms of how people address those … 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 is adapting to external pressures.

> *A: The ideas at that point was to be able to do re-use - planned re-use. Let me clarify that the intent there was to re-use - there was not necessarily a formalized process on how to be able to do that and that was one of the learning activities … the ad hoc re-use 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 re-use method - so a library of models which were code-generated specifically for specific parts in the vehicles and specific products*
>
> Q: So in fact that move from ad hoc re-use through to the more structured and planned re-use was actually kind of almost a generational change?
>
> *A: It absolutely was. It 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

Here we see a number of the important aspects of The Car Company's adoption, and development, of MDE: 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 we see 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 – a sort of "reductio ad absurdum" for organizational decision making. In refusing the need to change to a different approach, the management team would have stopped the company from adapting to the needs of 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 good fortune appears to play a part in an organizations successful adoption of MDE.

Q: I now want to ask you – could you imagine that if the experience that you had at The Car Company, after a number of these iterations, could have been packaged could you have gone back to the same people at the beginning of the process showing them the evidence and got them to accept it without having to go through these steps along the way?

A: 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 is perhaps the "holy grail" – what evidence is necessary to convince organizations that MDE can be used – in their domain, their sector, 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 therefore suggest that MDE is likely to grow in acceptance as companies migrate to model-oriented approaches. Alternatively, it may suggest that some kind of ideological opposition to raising the level of abstraction 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 veracity of received wisdom and the ability to properly judge the efficacy of alternative approaches.

We will finish this look at The Car Company's case study by considering the results of using MDE, which is, after all, the point of changing a software development process:

Q: …You required an increase in productivity that was kind of the key motivator for these generational modeling development steps. What experience did you have of achieving that?

A: Ah if I actually had the complete answer for that - for every time I am asked that - that would be fantastic. The challenge … giving you a very concrete answer to that is that as you notice that each of those different generations that I talked about 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 in there in terms of 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

Q: I was aware when I asked you that question that it was really unfair because you had already said that your experience had really resulted in package of benefits so I was just interested to know if you had any ideas that you could share because it sounds like you have got an increase in productivity on the one hand; you have got an increase in maintainability but you have also got an improved product because you are able to make later changes to it?

A: With known quality attributes which was the critical part. It is always easy to make late changes - very difficult to make late changes that are high quality or at least known quality and I think that was one of the attributes there …

This extract 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? We suggest not, and the focus on "known quality attributes" in the above segment helps to explain. For The Car Company, the whole approach to 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

6. 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 FTE 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 we know, 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 experience that the organization had, and in particular that the engineers had, was not really anything that could be described as a success. Before looking at the case study in detail, let us consider what factors affecting MDE adoption the study best represents:

- *Autocratic* – for reasons of confidentiality, we do not wish to identify the location of the organization, but for a number of reasons, it 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.
- *Wholesale* – the decision to adopt and 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 would be 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.

The Telecom Company's Experience of MDE

The extracts from The Telecom Company case study will illustrate some of the points made above.

> *A: ... this kind of huge decision to ... 50 engineers all developing by using this CASE tool, so it was quite radical at that time. I didn't quite understand it. I could understand some portion of the system - but everything developed by the tool is a really radical change.*
>
> Q: Do you have any knowledge of the processes around that decision? So for example, you've said that it was the project leader who made the decision, but if you've got a team of 50 people, presumably those people weren't experienced in using these CASE tools...
>
> *A: Yes exactly*

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. 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 segment is that existing developer skills and their attitude to change did not feature largely in the adoption process.

> Q: So this is quite critical isn't it for MDE? So how did they then maintain consistency between the generated code and the models?
>
> *A: ... they also 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 ...*
>
> Q: What sort of sized systems are we talking about? Are we talking like tens of thousands of lines of code or millions of lines of code?
>
> *A: Yes, millions, huge, huge code. So the binary is some gigabytes.*
>
> Q: Wow, ok!
>
> *A: ... that is also one of the problems ... they totally... attach to the tool and they don't know what's going on with the generated code.... it was C language they'd been using, they couldn't optimize the generated code so the way they had to ... asking the hardware guys to have more hard disc, more memory, because of the tool. So beforehand we had very small memories and we'd 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 disc. So it took, in the worst case, in took about 2 hours to upload the binary to deploy to the switching system.*

When The Telecom Company implemented MDE, it did so in a way which 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.

> Q: That was going to be my next question... so this project was completed and it worked....?

A: Yes

Q: But the company didn't then adopt this approach for their other projects.

A: I don't think so.

Q: It was almost like a relief to finish?

A: Right, right.

Q: And do you think the tool ... I suppose you can make a distinction between SDL as a technology or as a language and the implementation of that and the support for it in the tool, and then also the aspect of the tool that generates the code... … And do you think that the particular problems or the weaknesses lay in one area or all of those?

A: Yes, so, I think everyone makes mistakes. So the company, the engineers, they are too ambitious.

Q: Right - they should have maybe done a prototype?

A: Yes. 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 couldn't 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 ... weren't aware of the downside. So that was their mistake. And the second thing is the tool vendor. The tool itself is very inefficient. I ... developed a lot of CASE tools whilst I was at the university as a PhD student, but if somebody asks me how to optimize ... code from your CASE tool, then I don't know how to do that! So you already have your own design or structure to generate code and just make the consistency between ... the high modeling language and your generated code... that is the main concern ...

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.

A: But for telecommunication system there are thousands of features, really really huge features. I have no idea [who is] really using that kind of features. We only use call forwarding or call back and typical features, but if you see the manual, really hundreds, thousands of features that're never used. But there are some customers who demand that kind of features so you have to implement all of these ideas. And it comes to the nightmare. Because you cannot use the source code, you have to do it at the SDL level so they couldn't 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's kind of difficult to make a decision. Should I change the model or should I.... there's 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're using just as a programming language. So it's like a Java compiler, C compiler - very expensive, C compiler I would say...

This short 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. Thus 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".

Q: The reason I was asking that is because if they had been very very early adopters of the approach then they could be forgiven for perhaps having to make up the way of doing it, you know, make up their own approach, whereas in the late 90s there were already people out there with experience of doing model driven development… do you think they could have benefited from bringing in some external expertise for example? Somebody who had a lot of experience of applying model driven development to guide the project?

A: Yes... not from the tool vendors. 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 don't 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 from ... didn't really know the meaning of the object orientation and the system architecture view and what is the important thing to make better of that ...

Q: So actually what you're saying is an element of domain expertise as opposed to technical expertise is valuable in that environment?

A: Yes I think so.

Here, we are reminded that the choice of 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/processes. 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.

7. CONCLUSION – SOME LESSONS LEARNED ABOUT MDE DEPLOYMENT

This paper has provided some details from three case studies of recent MDE deployments in industry. Our case studies point in particular to important social, technical and organizational factors affecting MDE success or failure. Organizationally our respondents suggest that the benefits of MDE can especially be seen in terms of improved communication and control within the organization, with how MDE has ensured wider, and perhaps better, organizational knowledge and communication. These three case studies provide some interesting empirical details of exactly how MDE is used in industry and of how social and organizational factors impact on MDE use.

The case studies particularly highlight important organizational aspects of MDE deployment:

- *Organizational adoption or deployment* – 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.
- *Organizational support – Committed* – successful MDE is dependent on organizational commitment – the kind of commitment required to support any organizational change
- *Organizational motivation* – MDE users must be motivated to use the new approach and, in accord with the comment on commitment above, the organization must be motivated to make the project a success. Problems with MDE are likely to occur where the organization adopts an autocratic attitude towards its processes and its developers.
- *Organizational integration* – 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. In 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 sets up potential clashes.
- *Organizational focus* – ultimately successful MDE has to have a business focus, where MDE is adopted as a solution to new commercial and organizational challenges.

Case studies have previously proved useful in other aspects of software development and deployment [15, 16, 27] because it seems obvious that drawing and learning from 'experience' is important. The workplace experiences detailed in our case studies highlight some real constraints in the way that MDE is developed and deployed such that, as has been discovered in other contexts of system deployment, recognizing, understanding and determining even such obvious terms as 'success' and 'failure' is far from easy [14]. There are some subtleties to consider in our empirical 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 – and this study has added yet more subtle nuances to this overall depiction.

What comes out of our analysis of the responses we received in the interviews was the clear need to consider a range of social and organizational factors (as well as technical factors) in the success or failure of MDE deployments, and consequently place our developing understanding of the deployment and use of MDE within a much wider context of organizational change and change management. Organizational decisions to adopt or deploy MDE (or any other system) are clearly products of a particular organizational interpretation of the social and organizational context of work, incorporating notions of efficiency, profit and so on. Any assessment or evaluative accounts similarly need to be placed within this social and organizational context. This includes what might be glossed as both a 'macro-organizational context' of institutional policies and politics and a 'micro-organizational context' of the practicalities of work organization – within the software development section for example. 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 and change and may even, in some instances, be in conflict.

Finally, what these case studies have emphasized, in terms of understanding the evaluation of MDE deployment and use, 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. Additionally the need for a clear understanding of the necessary support for implementing and managing change can create problems involving the practical prioritization and reconciliation of long-term policy and short-term organizational contingencies. These, it seems to us, are generic problems across any commercial organization intent on relatively rapid innovations in culture and practice. Moreover, in managing these myriad concurrent changes, organizations also have to conduct their business, determine appropriate training, maintain the commitment of the staff, develop and evaluate new ways of working, and so on. It is amongst and in the light of the competing demands of these change processes that any empirical evaluation of MDE takes place. Finally, much of our work, and this paper, is about organizational factors in the successful or unsuccessful deployment of MDE, however MDE is essentially, that is, at heart, a technical development in the way software is developed – so one interesting, future avenue to explore is to identify precisely which technical features of MDE development (abstraction? code generation? etc) mesh most easily with features of organizational change and which create most problems and therefore require rather more in the way of organizational effort and support.

8. ACKNOWLEDGMENTS

Our thanks go to many MDE experts who agreed to speak to us about their experiences (although we have presented only three 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.

9. REFERENCES

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

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

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

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

[5] Briand, L., Labiche, Y., Di Penta, M., and Yan-Bondoc, H., "An Experimental Investigation of Formality in UML-based Development," IEEE Transactions on Software Engineering, vol. 31, pp. 833-849, 2005.

[6] Cruz-Lemus, J., Genero, M., Morasca, S., and Piattini, M., "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

[7] Cruz-Lemus, J., Genero, M., Manso, M., and Piattini, M., "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.

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

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

[10] France, R and Rumpe, B. (2007) Model driven development of complex software: A Research Roadmap. Future of Software Engineering, IEEE The Computer Society.

[11] Frankel, D., Model Driven Architecture: Applying MDA to Enterprise Computing: John Wiley & Sons, 2003.

[12] Ganssle, J., "A Trillion Lines of Code?," in Embedded.com (21/9/08), 2008.

[13] Genero, M., Piattini, M., and Manso, E., "Finding Early Indicators of UML Class Diagrams Understandability and Modifiability," in International Symposium on Empirical Software Engineering (ISESE), 2004, pp. 207-216.

[14] Grudin, J. (1988) 'Why CSCW applications fail: problems in the design and evaluation of organizational interfaces.' Proc. CSCW '88, 85-93. New York. ACM.

[15] Kaner, C., "Software Testing as a Social Science Problem", Canadian Undergraduate Software Engineering Conference, Montreal, Canada, 2006.

[16] Kim, J-M., A. Porter, and G. Rothermel, "An Empirical Study of Regression Test Application Frequency", Proc International Conference on Software Engineering ICSE'00, Limerick, 2000. pp.126-135.

[17] Kramer, J., "Is Abstraction the Key to Computing?," in Communications of the ACM. vol. 50, 2007, pp. 37-42.

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

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

[20] "Model Driven Development for J2EE Utilizing a Model-Driven Architecture Approach: Productivity Analysis (White Paper)," The Middleware Company, 2003.

[21] Nugroho, A., Flaton, B., and 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.

[22] "Model Driven Development for J2EE Utilizing a Model Driven Architecture Approach: Maintainability Analysis (White Paper)," The Middleware Company, 2004.

[23] 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, p. 432-443.

[24] Purchase, H., Colpoys, L., McGill, M. J., and Carrington, D. A., "UML Collaboration Diagram Syntax: Anempirical study of comprehension," in International Workshop on Visualizing Software for Understanding and Analysis, 2002.

[25] Razali, R., Snook, C. F., Poppleton, M. R., Garratt, P. W., and Walters, R. J., "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.

[26] Soley, R., Frankel, D., and Parodi, J., The MDA Journal: Model Driven Architecture Straight from the Masters: Meghan Kiffer Press, 2004.

[27] Talby, D., A. Keren, O. Hazzan, and Y. Dubinsky, "Agile Software Testing in a Large Scale Software Testing Project", IEEE Software, July/August 2006. pp.30-37.

[28] Thomas, D., "MDA: Revenge of the Modelers or UML Utopia?," in IEEE Software. vol. May/June 2004, 2004, pp. 22-24.