# A Taxonomy of Tool-Related Issues Affecting the Adoption of Model-Driven Engineering[1]

Jon Whittle[1], John Hutchinson[1], Mark Rouncefield[1], Håkan Burden[2], and Rogardt Heldal[2].

[1] School of Computing and Communications, InfoLab21, Lancaster University, Lancaster, UK

[2] Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden

Abstract.

Although poor tool support is often blamed for the low uptake of model-driven engineering (MDE), recent studies have shown that adoption problems are as likely to be down to social and organizational factors as with tooling issues. This article discusses the impact of tools on MDE adoption and practice and does so whilst placing tooling within a broader organizational context. The article revisits previous data on MDE use in industry (19 in-depth interviews with MDE practitioners) and re-analyzes that data through the specific lens of MDE tools in an attempt to identify and categorize the issues that users had with the tools they adopted. In addition, the article presents new data: 20 new interviews in two specific companies – and analyzes it through the same lens. A key contribution of the paper is a loose taxonomy of tool-related considerations, based on empirical industry data, which can be used to reflect on the tooling landscape as well as inform future research on MDE tools.

Keywords: model-driven engineering, modeling tools, organizational change

## 1 Introduction

When describing barriers to adoption of model-driven engineering (MDE), many authors point to inadequate MDE tools. Den Haan [10] highlights “insufficient tools” as one of the eight reasons why MDE may fail. Kuhn et al. [19] identify five points of friction in MDE that introduce complexity; all relate to MDE tools. Staron [28] found that “technology maturity [may] not provide enough

[1] An earlier version of this article appeared as “Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem?” in the 2013 International Conference on Model Driven Engineering Languages and Systems (MODELS). The major additions in this version are: (i) contextual research from the CSCW (computer supported cooperative work) community is discussed, which provides important background knowledge for interpreting and generalizing from the findings; (ii) an appendix is included, which describes the taxonomy in more detail; (iii) additional information about study design and validity is presented.

support for cost efficient adoption of MDE." Tomassetti et al.'s survey reveals that 30% of respondents see MDE tools as a barrier to adoption [30].

Clearly, then, MDE tools play a major part in the adoption (or not) of MDE. On the other hand, as shown by Hutchinson et al. [16, 17], adoption barriers are as likely to be social or organizational rather than purely technical or tool-related. The question remains, then, to what extent poor tools hold back adoption of MDE and, in particular, what aspects – both organizational and technical – should be considered in the next generation of MDE tools.

A key contribution of this article is a loose taxonomy of factors which capture how MDE tools impact MDE adoption. The focus is on relating tools and their technical features to the broader social and organizational context in which they are used. The taxonomy was developed by analyzing data from two separate studies of industrial MDE use. In the first, we interviewed 19 MDE practitioners from different companies. In the second, we interviewed a further 20 MDE practitioners in two different companies (10 per company). The two studies complement each other: the first is a broad but shallow study of MDE adoption across a wide range of industries; the second is a narrower but deeper study within two specific companies with different experiences of applying MDE. Neither study was limited to tooling issues; rather, they were both designed to capture a broad range of experiences related to MDE use and adoption and, in both, we used qualitative methods to allow key themes to emerge from the data. We focus in this paper only on emergent themes related to MDE tools.

The literature has relatively little to say about non-technical factors of MDE tooling. There have been a number of surveys of MDE tools (e.g., [5, 9, 24]) but they focus on classifying tools based on what technical functionalities they provide. More recently, Paige and Varró report on lessons learned from developing two significant (academic) MDE tools [23]. Again, however, very little is said about understanding users' needs and the users' organizational context: the authors simply state "Try to have real end-users; they keep you honest" and "Rapid response to feedback can help you keep your users." Indeed, there is a distinct lack of knowledge about how MDE tools are actually adopted in industry and what social and organizational, as well as technical, considerations need to be in place for a tool to succeed. This paper makes a first attempt to redress the balance. It uses a database derived from a large, systematic, qualitative empirical investigation [17] to examine the extent to which MDE adopters refer to 'tools' in their accounts of the success or failure of MDE. What emerges in the course of interrogating the database and building the taxonomy, is a far more subtle and nuanced picture, in which tool use is linked to a range of social and organizational issues that may be far more relevant to an understanding of success and failure. It is this subtle and nuanced understanding – expressed and evidenced in the words of MDE users – that provides a series of related answers to some of the issues raised by tool use in MDE.

The paper is structured as follows. Section 2 discusses existing literature on tools, with a focus on understanding users' needs and organizational context. Section 3 describes the methodological details of our studies. Section 4 presents our taxonomy, based on emerging themes from our first study of MDE adoption. Section 5 discusses our second study and relates its findings to the taxonomy. Section 6 then presents a consideration of the social and organizational context – in particular, showing how viewing tool selection and adoption as an act of introducing new software into an organization make it amenable to analysis from the perspective of previous work on "computer supported cooperative work" (CSCW). Finally, the paper discusses how the taxonomy can be used to advance research and development of MDE tools (Section 7).

## 2 Context and Related Work

Tools have long been of interest to those considering the use of technology in industrial settings. In research on computer supported cooperative work (CSCW), there have been two distinctive approaches. On the one hand there are those interested in how individuals use tools and, in particular, how to design tools that are intuitive and seamless to use. This reflects a Heideggerian difference between tools that are 'ready to hand' (they fade into the background) and 'present at hand' (focus is on the tool to the detriment of the 'real' issue). In contrast, another approach, exemplified by Grudin [14] and Brown [3], considers how organizations use tools and argues that failure can be attributed to: a disparity of benefit between tool users and those who are required to do unrecognized additional work to support tools; lack of management understanding; and a failure by designers and managers to recognize their limits. In a comment that might cause some reflection for MDE tool developers, Brown [3] suggests that (groupware) tools are generally useful in supporting existing everyday organizational processes, rather than radical organizational change.

The issue of how software development should be organized and supported has long been discussed and remedies have often, though not always, included particular tools, techniques, and practices. For example, whilst Merisalo-Rantanen et al. [22] found that tools facilitated fast delivery and easy modification of prototypes, amongst the core values of the 'agile manifesto' was a focus on "individuals and interactions over processes and tools" and a number of studies [25] emphasized the importance of organizational rather than technical factors. However, when considering MDE tools there is little in the way of systematic evaluation. Cabot and Teniente [5] acknowledge MDE tools but suggest that they have several limitations regarding code generation. Selic [26] talks about the important characteristics of tools for the success of MDE, suggesting that some MDE tools "have now reached a degree of maturity where this is practical even in large-scale industrial applications". Recently, Stahl et al. [27] have claimed that MDE does not make sense without tool support. Two studies [1, 19] identify the impact of tools on processes and organizations, and vice versa, but the main focus is on introducing MDE in large-scale software development. Hutchinson et al. [17] describe a case study where developers so subverted their use of an off-the-shelf MDE tool that they likened its use to that of a compiler.

A survey in 2005 [21] investigated the use of, and attitude towards, UML and UML tools across Europe and included 500 participants. Although the main focus of the survey was UML itself, it revealed a number of findings about the tools that participants used and their relative importance. For example: “The majority said that UML tools are not considered as an important part of their development process.” The survey also notes that, for some, tool cost is a factor with some claiming that the cost of appropriate tools meant that they were unable to use them. One finding was that the most popular tools used by participants belonged to the IBM Rational suite but the list also included Microsoft Visio, which itself calls into question the appropriateness of the tools used in some cases.

There have been two recent, and very different, studies about the experience of developing and deploying MDE tools. Paige and Varró [23] conclude that: “using MDD tools – in anger, on real projects, with reported real results, is now both feasible and necessary.” However, it is significant that this study is about academic MDE tools. In contrast, Clark and Muller [7] use their own commercial experiences to identify lessons learned about tool development, in cases that might be considered technical successes but were ultimately business or organizational failures: “The last decade has seen a number of high profile commercial MDD tools fail ...these tools were expensive to produce and maintain ...there are a number of open-source successes but it is not clear that these systems can support a business model”. In terms of specific lessons with regard to tools, this one stands out: “ObjeXion and Xactium made comparable mistakes. They were developing elegant tools for researchers, not pragmatic tools for engineers”.

## 3 Study Method

The key contribution of the paper is a taxonomy of MDE tool-related issues. The taxonomy has been developed based on two sets of interviews: a set of 19 interviews from 18 different companies carried out between Nov 2009 and Jul 2010, and a set of 20 interviews carried out in two companies between Jan and Feb 2013. Our method was to use the first set to develop the taxonomy; the second to validate the taxonomy. The two sets are complementary: the first provides broad, shallow coverage of 10 different industrial sectors; the second provides narrow, deep coverage of two companies.

Our first set of interviews is the same set used in earlier publications [16, 17]. However, prior publications gave a holistic view of the findings and did not include data on tools. All interviewees came from industry and had significant experience of applying MDE in practice. The interviews were semi-structured, taking around 60 minutes each, and all began with general questions about the participant’s background and experience with MDE. All interviews were recorded and transcribed. In total, we collected around 20 hours of conversation, amounting to over 150,000 words of transcribed data.

For this first set of interviews, we identified participants through personal contacts and through responses to an online survey, which was promoted on leading software engineering and MDE mailing lists. The criteria for selecting a participant was that s/he had real industrial experience of applying MDE on significant projects. We also aimed for a broad coverage of MDE application scenarios and domains. The interviewees covered 9 different industrial sectors and application domains, including aerospace, automotive, web applications, industrial control systems, and data-heavy applications. Similarly, we ensured that our interviewees had different levels of experience: in terms of number of years applying MDE and the role which they took in an MDE adoption effort. Collectively, our interviewees had over 360 years of software development experience and represent a range of different roles, including developer, product manager, software architect, consultant, CEO, and board member. Hence, the interviewees provide broad coverage of where MDE has been applied in industry.

The second set of interviews included 10 participants at Ericsson AB and 10 participants at Volvo Cars Corporation. The interviewees at Ericsson came from the Radio Base Station unit, which has been involved in MDE since the late 1980s while the interviewees at Volvo represent a new unit that has just started to use MDE for in-house software development for electric propulsion. The interviews cover more than 20 hours of recorded conversation and were conducted in the same semi-structured fashion as the first set.

These participants were chosen to provide a complementary set of data when compared with the first study: whereas the first study provided broad coverage across a range of sectors and domains, the second study focused on more in-depth coverage of two particular companies. We chose two companies with experience in applying MDE, but where one company had many years of experience (and hence was influenced by earlier thinking around MDE such as OMG standardization) and one company was relatively new to MDE (and hence was influenced more by current development methods such as agile). Within these companies, we identified interviewees by talking with managers and other influential stakeholders within the companies. Through these discussions, we identified projects within the (very large) companies where MDE was being applied, and we then chose participants to ensure a diverse set of experiences, background and perspectives.

In both studies, our notion of MDE tool was very broad – by tool, we mean any software application that was used by our interviewees during the MDE development process to create, manipulate or analyze models.

Analysis of the interview transcripts was slightly different in each case. The first set was used to develop the taxonomy. Each transcript was coded by two researchers. The initial task was to simply go through the transcripts looking for where the respondents said anything about tools; these fragments were then coded by reference to particular ideas or phrases mentioned in the text – such as 'cost' or 'processes'. The average reference to tool issues per transcript was 11 with 3 being the lowest and 18 being the highest. Inter-coder reliability was computed using Holsti's formula [15], dividing the number of agreements by the number of text fragments. For this research, the average inter-coder agreement was 0.86 (161/187). The researchers then grouped the initial coding into broad themes relating to "technical", "organizational" and "social" issues.

The second set was used to validate the taxonomy. Researchers read the transcripts looking for tool-related issues and then mapped those to the proposed taxonomy. Any deviations from the taxonomy were noted.

This is a descriptive, qualitative interview study, rather than an experimental study in which a wide range of validity constructs – internal, external, construct and conclusion – are conventionally deployed. For those unfamiliar with our approach we should quickly state that this does not mean that our study lacks these kinds of validity – just that they are not framed in this particular (experimental) way. So, for example, internal validity comes from our analysis of inter-rater reliability and the fact that a second study was used to validate the results from the first. External validity and our ability to generalize from our findings is approached rather differently in this kind of qualitative study, and accounts for the extensive use of quotes, included in this paper, from the individuals involved in our interviews. This is because we want to be able to point to the data to support our conclusions about MDE and tool use – each of our analytic points is supported by reference to what people actually said and the way that they said it. As we argue elsewhere [16], these qualitative interview studies provide sufficiently rich and authentic detail that the generalization problem – 'how can this information be relevant to other MDE projects in other organizations'? – becomes instead an issue for our readers – 'in what respect are the details reported here sufficiently familiar and similar to those in your own organization?'. In other words, the findings must be interpreted within the context in which they were found (illustrated by the quotes) and, when it comes to generalization, this context should be compared to the new context. External validity also comes from the fact that we place the findings in the context of known results from the CSCW (computer supported cooperative work) field (see Section 6), which has for decades studied the impact of new tools and technologies introduced into organizations. Finally, we note that this is an exploratory study, so the findings should be interpreted as potential hypotheses, which warrant further, perhaps more experimental, research.

## 4 A Taxonomy of MDE Tool Considerations

This section presents the taxonomy, developed from the first set of interviews. Our analysis process resulted in four broad themes, each broken into categories at two levels of detail: (i) Technical Factors – where interviewees discussed specific technical aspects of MDE tools, such as a missing feature or technical considerations of applying tools in practice; (ii) Internal Organizational Factors – the relationship between tools and the way a company organizes itself; (iii) External Organizational Factors – influences from outside the company which may affect tool use and application; (iv) Social Factors – issues related to the way people perceive MDE tools or tool stakeholders. In all cases, it should be assumed that each issue may impact in a range of ways on an MDE process. For example, if a tool feature isn't available, that may impede the development process whereas if it is present, then that may aid the development process.

Tables 1-5 form the taxonomy. Table 1 gives the top level of the taxonomy and then each category is briefly defined in the remaining tables, and an example of each sub-category is given. Numbers in brackets are the number of interviewees who commented on a particular sub-category (max. 19). Care should be taken when interpreting these numbers – they merely reflect what proportion of our participants happened to talk about a particular issue. They do not necessarily indicate relative importance of sub-categories because one interviewee may have talked in depth about a sub-category whereas another may have mentioned it only briefly. Moreover, a common problem may well be one that is reasonably easy to overcome whereas a less common one may be the determiner of success of failure for a particular user.  A deeper analysis would be required to produce sub-category "importance" or "severity" weightings.

The following subsections present highlights from each theme: we have picked out particularly insightful or relevant experiences from the interview transcripts. We quote from the transcripts frequently; these are given italicized and in quotation marks. Quotes are taken from the transcripts verbatim. Square brackets are used to include contextual information.

The taxonomy is a data-driven, evidence-based description of issues that industrial MDE practitioners have encountered in practice when applying or developing MDE tools. We make no claim that the taxonomy covers all possible tool-related issues; clearly, further evidence from other practitioners may lead to an extension of the taxonomy. We also do not claim that the sub-categories are orthogonal. As will be seen later, some examples of tool use can be classified into multiple sub-categories. Finally, we do not claim that this is the "perfect"' taxonomy. It is simply one way of structuring the emerging themes from our data, and the reader is welcome to re-structure the themes into an alternative taxonomy which better fits his/her purposes.

The taxonomy can be used in a variety of ways. It can be used as a check-list of issues to consider when developing tools. It can be used as a framework to evaluate existing tools. Principally, however, we hope that it simply points to a range of technical, social and organizational factors that may be under-represented in the MDE tool research community.

All of the branches, categories and sub-categories are listed in the appendix along with a brief explanation of what is referred to at each level.

Table 1. Taxonomy of MDE Tool Considerations

| | |
|---|---|
| MDE Tool Considerations | Technical Factors |
| | Internal Organizational Factors |
| | External Organizational Factors |
| | Social Factors |

## 4.1 Technical Factors

Table 2. Technical Categories.

| Category | Sub-Category |
|---|---|
| **Tool Features**<br>*Specific functionalities offered in tools* | - Modeling Behavior (1)<br>- Action Languages (1)<br>- Support for Domain-Specific Languages (6)<br>- Support for Architecture (3)<br>- Code Generation Templates (6)<br>- UML Profiles (1)<br>- Scoped Code Generation (2)<br>- Model Analysis (5)<br>- Reverse Engineering Models (3)<br>- Sketching Models (1)<br>- Refactoring Models (1) |
| **Practical Applicability**<br>*Challenges of applying tools in practice* | - Tool Scalability (1)<br>- Tool Versioning (1)<br>- Chaining Tools Together (2)<br>- Industrial Quality of Generated Code (8)<br>- Flexibility of Tools (3)<br>- Maturity of Tools (1)<br>- Dealing with Legacy (2) |
| **Complexity**<br>*Challenges brought on by excessive complexity in tools* | - Tool Complexity (4)<br>- Language Complexity (5)<br>- Accidental Complexity Introduced by Tools (1) |
| **Human Factors**<br>*Consideration of tool users* | - Whether Tools Match Human Abstractions (4)<br>- Usability (4) |
| **Theory**<br>*Theory underpinning tools* | - Theoretical Foundations of Tools (1)<br>- Formal Semantics (2) |
| **Impact on Development**<br>*Impact of tools on technical success criteria* | - Impact on Quality (2)<br>- Impact on Productivity (4)<br>- Impact on Maintainability (3) |

Table 2 presents the set of categories and sub-categories that relate to technical challenges and opportunities when applying MDE tools. There are six categories.

*Category Descriptions* The first, Tool Features, details specific tool functionalities which interviewees felt impacted on project success. These include support for modeling system behavior, architectures, domain-specific modeling, and flexibility in code generation. Code Generation Templates, for example, refers to the ability to define one's own code generation rules, whereas Scoped Code Generation refers to an incremental form of code generation where only model changes are re-generated. The second category, Practical Applicability, contains issues related to how tools can be made to work in practice. The issues range from tool support for very large models (scalability), to the impact of using multiple tools or multiple versions of tools together, to the general maturity level of tools and how flexibly they can be adapted into existing tool chains. The third category concerns Complexity, which includes Accidental Complexity, where the tools introduce complexity unnecessarily. The fourth category is Human Factors and includes both classical usability issues but also bigger issues such as whether the way tools are designed (and, in particular, the kinds of abstractions they use) match the way that people think. The final two categories concern the way that the lack of formal foundations leads to sub-optimal tools and the reported perceptions about how tools impact quality, productivity and maintainability.

*Observations* One very clear finding that comes out of our analysis is that MDE can be very effective, but it takes effort to make it work. The majority of our interviewees were very successful with MDE but all of them either built their own modeling tools, made heavy adaptations of off-the-shelf tools, or spent a lot of time finding ways to work around tools. The only accounts of easy-to-use, intuitive tools came from those who had developed tools themselves for bespoke purposes. Indeed, this suggests that current tools are a barrier to success rather than an enabler and *"the fact that people are struggling with the tools...and succeed nonetheless requires a certain level of enthusiasm and competence."*

Our interviewees emphasized tool immaturity, complexity and lack of usability as major barriers. Usability issues can be blamed, at least in part, on an over-emphasis on graphical interfaces: *"...I did an analysis of one of the IBM tools and I counted 250 menu items."* More generally, tools are often very powerful, but it is too difficult for users to access that power; or, in some cases, they do not really need that power and require something much simpler: *"I was really impressed with the power of it and on the other hand I saw windows popping up everywhere...at the end I thought I still really have no idea how to use this tool and I have only seen a glimpse of the power that it has."*

These examples hint at a more fundamental problem, which appears to be true of textual modeling tools as well: a lack of consideration for how people work and think: *"basically it's still the mindset that the human adapts to the computer, not vice-versa."* In addition, current tools have focused on automating solutions once a problem has been solved. In contrast, scant attention has been paid to supporting the problem solving process itself: *"so once the analyst has figured out what maps to what it's relatively easy...However, what the tools don't do is help the analyst figure out what maps to what."*

Complexity problems are typically associated with off-the-shelf tools. Of particular note is accidental complexity – which can be introduced due to poor consideration of other categories, such as lack of flexibility to adapt the tools to a company's own context. One interviewee described how the company's processes had to be significantly changed to allow them to use the tool: a lack of control over the code generation templates led to the need to modify the generated code directly, which in turn led to a process to control these manual edits. Complexity also arises when fitting an MDE tool into an existing tool chain: *"And the integration with all of the other products that you have in your environment..."* Despite significant investment in providing suites of tools that can work together, this is clearly an area where it is easy to introduce accidental complexity.

It is ironic that MDE was introduced to help deal with the essential complexity of systems, but in many cases, adds accidental complexity. Although this should not be surprising (cf. Brooks [2]), it is interesting to describe this phenomenon in the context of MDE. For the technical categories, in almost every case, interviewees gave examples where the category helped to tackle essential complexity, but also other examples where the category led to the introduction of accidental complexity. So, interviewees talked about the benefits of code generation, but, at the same time, lamented the fact that *“we have some problems with the complexity of the code generated...we are permanently optimizing this tool.”* Interviewees discussed how domain-specific languages (DSLs) should be targeted at complex parts of the system, such as where multiple disciplines intersect (*“if you have multiple disciplines like mechanical electronics and software, you can really use those techniques”*) whilst, at the same time realizing that the use of DSLs introduces new complexities when maintaining a standard DSL across a whole industry: *“their own kind of textual DSL [for pension rules]...And they went to a second company and the second company said no our pension rules are totally different.”* Clearly, as well known from Brooks, there is no silver bullet.

## 4.2 Internal Organizational Factors

Table 3. Internal Organizational Categories.

| Category | Sub-Category |
|---|---|
| **Processes**<br>*Adapting tools to processes or vice-versa* | - Tailoring to a Company’s Existing Processes (5)<br>- Sustainability of Tools over the Long Term (3)<br>- Appropriating Tools for Purposes They Were Not Designed For (3)<br>- Issues of Integrating Multiple Tools (6)<br>- Migrating to different tool versions (3)<br>- Offsetting Gains: Tools bring gains in one aspect but losses in another (2)<br>- Whether Maintenance is carried out at the Code or Model Level (3) |
| **Organizational Culture**<br>*Impact of cultural attitudes on tool application* | - Tailoring to a Company’s Culture (4)<br>- Inertia: Reluctance to Try New Things (1)<br>- Over-Ambition: Asking Too Much of Tools (1)<br>- Low Hanging Fruit: Using Tools on Easy Problems First (6) |
| **Skills**<br>*Skills needed to apply tools* | - Training Workforce (11)<br>- Availability of MDE Skills in Workforce (4) |

*Category Descriptions* Table 3 gives the set of internal organizational categories. The first, Processes, relates to how tools must be adapted to fit into existing processes or how existing processes must be adapted in order to use tools. Tailoring to Existing Processes concerns the former of these; the remaining sub-categories the latter. Sustainability of tools concerns processes for ensuring long term effectiveness of tools, taking into account changes needed to the tools as their use grows within the organization. Appropriation is about how tool use changes over time, often in a way not originally intended. Integration Issues are where new processes are needed to integrate MDE tools with existing tools. Migration Issues are about migrating from one tool to another or from one tool version to another. Offsetting Gains is where a tool brings benefits in one part of the organization but disadvantages in another part of the organization. Maintenance Level is about processes that either mandate model-level changes only, or allow code-level changes under certain constraints. The Organizational Culture category relates to the culture of an institution: to what extent tools need to be adapted to fit culture (Tailoring to Existing Culture), cultural resistance to use new tools (Inertia), a lack of realistic expectations about tool capabilities (Over Ambition), and attitudes that look for quick wins for new tools to prove themselves (Low Hanging Fruit). The third category concerns Skills — both training needs (Training) and how existing skills affect adoption (Availability of Skills).

*Observations* Our interviews point to a strong need for tailoring of some sort: either tailor the tool to the process, tailor the process to the tool, or build your own tool that naturally fits your own process. Based on our data, it seems that, on balance, it is currently much easier to do the latter. Some tool vendors actively prohibit tailoring to the process, but rather a process is imposed by the tool for business reasons: *"...the transformation engines are used as services...we don't want to give our customers the source code of the transformation engines and have them change them freely. That's a business question."*

When introducing MDE tools, one should think carefully where to introduce them. One company reported, *"We needed to find a way to let them incrementally adopt the technology."* The solution was to first introduce reverse engineering of code into models, as the first part of a process of change management. Another company introduced MDE tools by first using them only in testing. The 'perfect' MDE tool may not always be necessary. For example, one company used MDE where the user interface was not so critical: *"cases which are internal applications ...where the user interface is not such an issue ...that's where you get the maximum productivity from a tool like ours."*

There is a danger, though, in believing that one "killer application" of an MDE tool leads to another: *"prior to that they had used the technology successfully in a different project and it worked and they were very happy, so they thought, ok, this could be applied to virtually any kind of application."* It is not easy to identify which applications are appropriate for MDE tools and which are not. Apart from obvious industries where MDE has been applied more widely than others (e.g., the automotive industry), we do not have a fine-grained way of knowing which MDE tools are appropriate for which jobs.

A curious paradox of MDE is that it was developed as a way to improve portability [18]. However, time and again issues of migration and versioning came up in our interviews: *"[XX] have burned a lot of money to build their own tool which they stopped doing because they lost their models when the [YY] version changed."*

This migration challenge manifests itself slightly differently as ‘sustainability’ when considering strategies for long-term tool effectiveness. It was often remarked by our interviewees that an MDE effort started small, and was well supported by tools, but that processes and tools broke down when trying to roll out MDE across a wider part of the organization: *“the complexity of these little [DSL] languages started to grow and grow and grow...we were trying to share the [code generation] templates across teams and versioning and releasing of these templates was not under any kind of control at all.”* One of our interviewees makes this point more generally: *“One of the things people forget about domain specific languages is that you may be able to develop a language that really is very well suited to you; however, the cost of sustaining just grows and it becomes eventually unacceptable because a language requires maintenance, it requires tooling, it requires education.”*

## 4.3 External Organizational Factors

Table 4. External Organizational Categories.

| Category | Sub-Category |
|---|---|
| **External Influences**<br>*Factors which an organization has no direct control over* | - Impact of Marketing Issues (1)<br>- Impact of Government/Industry Standards (4) |
| **Commercial Aspects**<br>*Business considerations impacting on tool use and application* | - Business Models for Applying MDE (3)<br>- Cost of Tools (5)<br>- How to Select Tools (2) |

*Category Descriptions* External organizational factors (Table 4) are those which are outside the direct control of organizations. External Influences include the impact of government or industry-wide standards on the way tools are developed or applied, as well as ways in which marketing strategies of the organization or tool vendors impact on the use and application of tools. Commercial Aspects include how the cost of tools affects tool uptake, how selection of tools can be made based on commercial rather than technical priorities, and how the use of tools relates to a company’s business model.

*Observations* External influences clearly have an impact on whether tools – any kind of tool, not just MDE – are adopted in an organization. Our interviews show that the tool market is focused only on supporting models at an abstraction level very close to code, where the mapping to code is straightforward. This is clearly somewhat removed from the MDE vision. Unfortunately, there is also a clear gap in the way that vendors market their tools and their real capabilities in terms of this low-level approach. As a result, many MDE applications fail due to expectations that have not been managed properly.

Data on the impact of the cost of tools seems to be inconclusive. Some interviewees clearly found cost of tools to be a prohibitive factor. In one case, the high cost of licenses led a company to hack the tool's license server! For the most part, however, companies do not seem to point to tool costs as a major factor: the cost of tools tends to be dwarfed by more indirect costs of training, process change, and cultural shift: *"...it takes a lot of upfront investment for someone to learn how to use the tools and the only reason I learnt how to use them was because I was on a mission."*

Government or industry standards can both positively and negatively affect whether tools are used or not. MDE tools can help with certification processes: *"they looked at the development method using the modeling tools and said, well, it's a very clear and a very comprehensive way to go and they accepted that."* In other cases, interviewees reported that MDE tools can make certification more difficult as current government certification processes are not set up to deal with auto-generated code. Sometimes, external legal demands were a main driver for the use of MDE tools in the first place: *"with the European legal demands, it's more and more important to have traceability."*

## 4.4 Social Factors

Table 5. Social Categories.

| Category | Sub-Category |
|---|---|
| **Control**<br>*Impact of tools on whether stake-holders feel in control of their project* | Ways of Interacting with Tool Vendors (2)<br>Subverting Tools: Workarounds Needed to Apply Them (1) |
| **Trust**<br>*Impact of trust on tool use and adoption* | Trust of Vendors (4)<br>Engineers' Trust of Tools (6)<br>Impact of Personal Career Needs (1) |

*Category Descriptions* When it comes to MDE tools, social factors (Table 5) revolve around issues of trust and control. Tool vendors, for example, have different business models when it comes to controlling or opening up their tools (Interacting with Tool Vendors). Subverting Tools is when a company looks for creative solutions to bring a tool under its control. The data has a lot to say about Vendor Trust, or how perceptions of vendors influence tool uptake. Engineers' Trust also affects tool success: typical examples are when programmers are reluctant to use modeling tools because they do not trust code generated. Career Needs refers to how the culture of the software industry may disadvantage MDE: an example is the ubiquitous use of consultants who are not necessarily inclined to take the kind of long term view that MDE needs.

*Observations* At a very general level, our data points to ways in which different roles in a development project react to MDE tools. One cannot generalize, of course, but roughly speaking, software architects tend to embrace MDE tools because they can encode their architectural rules and easily mandate that others follow them. Code 'gurus', or those highly expert programmers in a project, tend to avoid MDE tools as they can take away some of their control. Similarly, 'hobbyist programmers', those nine-to-fivers who nevertheless like to go home and read about new programming techniques, also tend to avoid MDE because it risks taking away their creativity. Managers respond very differently to MDE tools depending on their background and the current context. For example, one manager was presented with a good abstract model of the architecture but took this as a sign that the architects were not working hard enough!

One much-trumpeted advantage of MDE is that it allows stakeholders to better appreciate the big picture. Whilst this is undoubtedly true, there are also cases where MDE tools can cloud understanding, especially of junior developers: *“we’d been using C and we were very clear about the memory map and each engineer had a clear view...But in this case, we cannot do something with the generated code so we simply ask the hardware guys to have more hard disc.”* Similar implications can arise when companies become dependent on vendors. Vendors often spend a lot of time with clients customizing tools to a particular environment. But this can often cause delays and cost overruns and takes control away from the client: *“And suddenly the tool doesn’t do something expected and it’s a nightmare for them. So they try to contact the vendor but they do not really know what’s going on, they are mostly sales guys.”*

MDE asks for a fundamental shift in the way that people approach their work. This may not always be embraced. One example is where MDE tools support engineers in thinking more abstractly, and, in particular, tackling the harder business problems. But engineers may not feel confident enough to do this: *“when you come to work and you say, well, I could work on a technical problem or I could work on this business problem that seems not solvable to me, it’s really tempting to go work on the technical stuff.”* MDE tools require up-front investment to succeed and the return on this investment may not come until the tool has been applied to multiple projects. There is a tension here with the consultancy model which is often the norm in MDE: *“So they felt that, let me do my best in this one project. Afterwards, I am moving into some other project...[in a] consultancy organization, you measure yourself and you associate yourself with things in a limited time.”*

## 5 A Study of MDE Practice in Two Companies

This section presents insights from our second set of data: 20 additional interviews in Ericsson AB and Volvo Cars. Interviewees at Ericsson were users of Rational Software Architect RealTime Edition (RSA/RTE). At Volvo Cars, interviewees used Simulink. This set of interviews was carried out independently of the development of the taxonomy. The taxonomy was used in coding the second set of transcripts but any deviations from the taxonomy were noted. Hence, this second study can be seen as a validation of the taxonomy.

### 5.1 Modeling in Volvo Cars and Ericsson AB

Both Ericsson AB and Volvo Cars have been using modeling at several levels within their companies due to the size and complexity of their software. For example, within Volvo there are three distinct levels where different types of tools and models are used. At the top-level, the overall electrical architecture that can be used to build several types of cars is developed. This model captures the logical software architecture and patterns, etc. This is quite a creative phase, and there is less focus on the need to develop complete models.  The models are created using a number of different tools supporting editing of graphical models as well as textual descriptions. At the next level, a particular product or a particular type of car is described as a model. The architectural tool used is tailored directly to Volvo's needs, capturing in detail things such as all the ECUs (Electronic Control Units), the software components and their interfaces. One can consider the language used in this architectural tool to be a DSL for software architecture within the automotive domain. From this tool one can create code skeleton (components) to the next level down.  It is within the code skeleton that Simulink code is added. On top of this, there are several tools for version control, testing and transformations. All these tools used at different levels are quite different and it can be quite demanding if one needs to know several of them. It is extremely important therefore to have good tool chains.

The situation at Ericsson is similar to the one at Volvo in that there are several levels of modeling and different tools are used in each. The key difference is that in Ericsson, most code is built in-house. This is in contrast to the automotive industry in which a significant amount of code is produced by subcontractors.

In the following, we highlight, from our interviews, some particularly interesting observations about how these two companies use modeling tools and we relate these to the taxonomy. We do not attempt to provide examples of all sub-categories in Tables 1-5. In general, the issues highlighted by the taxonomy occur in both companies.

## 5.1 Technical Factors

When modeling at the lower levels, it was crucial for both companies that the models could be executed, so the tools need to support executable code or support the generation of skeletons within which code can easily be inserted and tested (Table 2; Code Generation Templates). It would have been next to impossible to produce the models for these companies without a good testing environment. Indeed, one of the main goals of using models within Volvo Cars was to improve software testing (Table 2; Impact on Quality). Code generation was crucial for both companies, either to produce code skeletons where C or C++ code could be inserted or to produce C code from models containing graphical elements and action code (Table 2; Action Languages/Modeling Behavior).

There are large disagreements amongst our interviewees whether modeling or coding is the best way to build a system, but a number of interviewees really liked modeling. *"And it's really fun to work in Simulink I think instead of writing code. Actually I really think it's fun. It's more graphical. You see what's happening. You get nice picture of it. So that's why really I enjoy it also. I'm not just doing coding in pure text based. So that's also why I enjoy it."* Others took a more critical view. *"It's everything from small things, small, annoying things. It wouldn't sell if it would have been sold to general consumers. Small things. You move, you change a line name and you want to undo it. It* *can't. Like why? Nobody knows."* To some extent, these differences in opinion may be due to individual differences in the way people think (Table 2; Human Abstractions) and/or individual tolerances to usability problems (Table 2; Usability).

Sometimes these companies make models that are too large for the tools to handle: *"We did an activity diagram and updated it … but the report generator couldn't take that out because it was too big."* (Table 2; Tool Scalability). Another scaling problem was: *"So we had scaling problems with tool and so on. When we tried to start to use modelling tool A instead of modelling tool B, there was some experts from Canada here trying to get the things working because no one else had this big models like we had… It was almost like we gave up."* This is a severe limitation of the tools. Even when the information is there it can be hard to find it sometimes. *"It wasn't on page 1 to 200. It was somewhere else because it was in the model somewhere. You couldn't find it. That's a bad part of models. You can actually insert lot of information that's never found."*

This second study clearly shows that MDE tools can both reduce and increase complexity. Ericsson employees found benefits of using RSA/RTE because of the complex aspects of the radio base station domain, such as synchronous/asynchronous message passing: *"It takes care of these things for you so you can focus on the behavior you want to have within a base station."* (Table 2; Impact on Productivity).

Interestingly, most of the interviewees at Ericsson have now moved to a new project where all development is done using C++ and a lot of time is spent on issues that were dealt with by the tool before. And it is a constant source of error. On the other hand, *"I don't think you gain advantage in solving all kinds of problems in modeling."* There is a danger of over-engineering the solution: *"You would try to do some smart modeling, or stuff and you would fail. After a while you would end up in a worse place than if you had done this in C++".* (Table 2; Impact on Maintainability).

Something which surprised us at Volvo was a tolerance of slow tool execution: *"And now when we have our new laptops, it actually just takes five minutes to generate a code. Before it would take up to 20 minutes, so it's quite quick now."* For some of our interviewees, it seems like the benefit or joy of modeling outweighs the disadvantage of waiting for tools to execute commands, problems with version control, problems of merging of models, or even bugs in the tools. (Table 2; Impact on Productivity).

## 5.2 Internal Organizational Factors

The proportion of in-house development has an effect on how well modeling tools are received. For example, at Ericsson, *"this system architecture tool is not really designed to facilitate in-house development and so on. It gets a lot of criticism for that."* Ericsson is working hard to resolve such problems by writing scripts to support in-house development. This is a clear example of where modeling tools have to be tailored to match a company's processes (Table 3; Tailoring to a Company's Existing Processes).

According to another employee at Ericsson, it is necessary to change the existing processes and culture in order to make the most out of MDE tools: *"I think actually that the technology for doing this [MDE] and the tools, as the enablers, they are more advanced than the organizations that can use them ...Because the organizations are not mature to do it there are few users of those tools and then the usability is poor."* (Table 3; Tailoring to a Company's Culture).

At Volvo, a substantial effort has been made in order to enable the transition from Simulink as a specification and prototype tool into a code generation tool; due to the properties of the code generator different design rules are suitable for readability versus code generation. Migrating from one tool to another also requires that old processes are updated: *"When it comes to TargetLink – a competitor to Simulink – we have the knowledge of good and bad design patterns. For Simulink, that is something we are currently obtaining, what to do and not, in Simulink models."* (Table 3; Sustainability of Tools over the Long Term; Migrating to Different Tools).

One Ericsson employee noted the importance of internal organizational support for MDE tools: *"Tool-wise I was better off five years ago than I am today...then we had tool support within the organization. And they knew everything. Today, if I get stuck there is no support to help me."* The quote comes from a system architect at Ericsson who concludes that the tools can be used effectively but it requires an effective in-house team knowledgeable about the details of the tools who can be called on to help when issues arise (Table 3; Training Workforce).

## 5.3 External Organizational Factors

Both companies illustrate how external organizational factors impact on MDE success. The functionality of Ericsson's radio base stations is accessed by Telecoms companies such as AT&T through an API. The API is developed using RSA/RTE by 7-8 software engineers. The changes to the API are managed by a forum which is responsible for ensuring that the accepted changes are consistent and that they make sense for the customers: *"We do have a process for how to change it and we review the changes very carefully. For new functions, we want it to look similar, we want to follow certain design rules and have it so it fits in with the rest."* (Table 4; Impact of Industry Standards). In fact, this example illustrates how MDE can be effectively used to manage external influences: in this case, Ericsson models the API as a UML profile and manages it through MDE.

At Volvo, the automotive standard AUTOSAR 3 has made the choice of development tool a non-issue; Simulink is the standard tool: *"...a language which makes it possible to communicate across the disciplinary borders. That the system architect, the engineer and the tester actually understand what they see."* (Table 4; Impact of Industry Standards).

## 5.4 Social Factors

Both Ericsson and Volvo Cars are large companies, with many employees involved in modeling. This influences the way that social factors manifest themselves. For example, tool vendors want to make these companies happy, so they often go a long way to support them (Table 5; Social Factors). The companies also commonly have courses to learn the basics of using the modeling tools, which affects to what extent engineers feel they can trust the tools (Table 5; Engineers' Trust of Tools).

There are situations where employees simply do not want to change the language they happen to be using. *"They don´t want to program C++, so they wouldn't do it because they were Java guys…Change is hard".* Some of the interviewees thought that modeling should not be forced on developers: *"That's the biggest mistake you can do…You should let it sort of come from the people that need it. That's the big stuff."*

The second study surfaced one additional social factor that was not highlighted in the earlier study. At Ericsson, interviewees commented that the main difference between working with RSA/RTE and code is that the latter is well-documented on the web: *"You can find examples and case studies and what not in millions."* But when searching for tool-specific help on UML tools, *"you basically come up empty-handed."* This observation prompted us to add an additional sub-category in the taxonomy, that of "Developer Forums". It appears to be quite an important point whether or not there are easily accessible, useful developer forums where developers can go to get quick answers about issues they are experiencing.  An updated taxonomy is included in the Appendix.

## 5.5 Taxonomy Validation

The study at Ericsson and Volvo is in itself revealing about MDE practice. However, for the purposes of this article, it serves primarily to validate our taxonomy. For the most part, the same issues come up in both studies. In only one case did we find that an extension to the taxonomy was necessary. This was on the role that an open community can play in supporting MDE. As discussed in Section 5.4, the lack of online support forums for MDE can lead to feelings of isolation and, in turn, lack of engagement with MDE. We therefore extend our taxonomy to reflect this – by adding a new category, Open Community, with sub-category, Developer Forums – this is shown in the Appendix.

The other issue is that it can be difficult to pick a single sub-category to which a statement applies. Often, a single statement overlaps multiple sub-categories. However, this was not unexpected. Issues of MDE adoption and tool use are complex and involve many dependencies, so it would be unrealistic to expect a taxonomy with completely orthogonal sub-categories.

## 6 Understanding the taxonomy's social and organizational context

As our interviews – and the resulting taxonomy show – MDE cannot be fully appreciated or evaluated as a software development approach without first understanding important aspects of business or organizational "context". In the social sciences the word "context" tends to do some fairly heavy duty analytic work since it is generally agreed that almost nothing can be properly understood without an awareness of the context in which it happens [6]. Hence, in the case of MDE, the context in which a model is developed, code is generated and/or some tool is used is crucial. Here, we are interested in a relatively simple notion of context by suggesting that MDE needs to be understood socially and organizationally. It has been especially notable in our studies that MDE researchers have paid relatively little attention to social and organizational issues. The focus has almost always been technical: researchers inevitably develop their own modeling language, MDE tool or methodology, without much consideration of the context in which they will be applied. Even empirical studies of MDE have largely concentrated on technical aspects of MDE. There has been little rigorous empirical research examining the social factors related to MDE adoption. Such a gap in the research seems odd, given the long history of trying to understand context in software engineering more generally (e.g., see the CSCW (computer supported cooperative work) conference series). Lehman [20] and Curtis [8] highlighted several years ago that software engineering methods are influenced by the social and organizational context in which they are developed. It would be questionable to suppose that these CSCW findings do not apply to MDE – after all, MDE involves many aspects where people and organizations are key: planning, procedures and abstraction, distributed coordination and various forms of 'awareness' of work.

In this section, we reflect on various CSCW findings that are particularly relevant to MDE research. By so doing, we hope to redress the balance to ensure that social and organizational factors are duly considered in future MDE research. Hence, we argue that MDE should be considered not merely in terms of 'tools' but as an organizational intervention, considering its impact on cooperation and collaboration in complex social and organizational settings, where cool and measured abstraction meets the messiness of the real world. In such settings, plans and tools do not simply implement themselves, but have to be implemented and used according to whatever resources are to hand and in the face of various changing and sometimes unpredictable contingencies. It is people that do the work in organizations, not idealized or abstract models and not tools. It is the everyday judgment of workers, in interpreting and improvising standard procedures, that gets work done and makes it routine. Software processes, as an example of everyday, mundane work, are clearly influenced by the social and organizational context in which they are developed.

Because of the clear pervasiveness of software, software process changes have a significant effect at both the macro- and micro-organizational level. At the macro-organizational level, software is now so important and is so often a feature of organizational change initiatives that software failures can threaten the existence of the organization. This is typically true of efforts to adopt MDE, which are often designed to effect organizational-wide change. But project managers must assess the benefits of the change and implement that change without adversely affecting other project planning. At what might be called the micro-organizational level, the importance of organizational and cultural factors highlights the need to explore and understand the ‘lived work’ of software project management. We are especially concerned with thinking beyond MDE tools to explicating exactly how MDE projects are managed in the face of uncertain and evolving requirements, while keeping the project on ‘track’, and doing so within budget and, ideally on time. As Button and Sharrock [4] indicate, the key to a successful project is “not merely a matter of disposing of the individual’s work in hand, but of carrying it out as work which is done in orientation to the project’s needs, problems and objectives.”

Our studies provide ample evidence of how the realities of a CSCW perspective play out in practice. MDE adoption scenerios are complex and inherently collaborative – in particular, they are often collaborative between different kinds of expertise or disciplinary background:

*“....it's an interdisciplinary engineering task because they are working together with mechanical engineers, electrical engineers and our software engineers and we all together build the automation application for example for a packaging machine or something like this. And so this is one application of model driven solutions to model integration of these different disciplines and this is something not found at the moment in current model driven approaches because they are usually only targeted on software development and not on the other disciplines.”*

Another aspect of collaboration is what in CSCW is referred to as ‘awareness of work’ – which highlights the way in which work tasks are made available to others and the important role that this plays in the ‘real world, real time’ social organization of work. The different ways in which ‘awareness’ is developed, in which work is made visible to others, are essential ingredients in ‘doing the work’ as part of a socially distributed division of labour. In our interviews, respondents reflected on how the nature of modeling requires building awareness and understanding:

*“Q: Was that because once you started to model successfully you recognised that there was a lot of commonality in your systems?*

*A: Yes.*

*Q: OK so the modeling actually helped you understand your systems?*

*A: Yes, and it is much easier to discuss the features of the system together with a customer on a basis of the models than on the basis of the code.”*

These extracts from our interviews show that successful practitioners of MDE in industry highlight all sorts of subtle aspects of its use, asking for a fundamental shift in the way that people approach their work, requiring that engineers think more abstractly about the relationship between their work and the overall business and, in particular, the ‘harder’ business problems, rather than what can be the simpler, technical, issues.

## 7 Conclusions and Arising Research Challenges

There are a number of well-known studies that document how technology often fails to deliver on the radical organizational changes expected of it. As we have already suggested, Brown's 1990 study of Lotus notes [3], for example, suggests that such investments in new technology are generally more useful in supporting existing everyday organizational processes, rather than radical organizational change. Similarly, in Grudin's (1988) classic paper on 'why applications fail' [14], the organizational use of tools and technology is considered and evaluated and the argument then advanced that comparative failure can be attributed to organizational rather than technical issues: a disparity of benefit between users and those who are required to do unrecognized additional support work; lack of management understanding and the difficulties of evaluation.

What has emerged from our research on MDE in practice, perhaps surprisingly to those who are interested only in the technical facets of MDE, is an emphasis on the importance of social, managerial and organizational factors in shaping successful MDE adoption and use. The move towards MDE is an indicator and precursor of important changes. Management in a period of change is often a complex and difficult process, especially when the changes, in organization, technology and perhaps culture, are being introduced concurrently. Among these difficulties are the fact that often changes in organizational culture, structure and technology do not all originate from a single integrated managerial or business strategy and inevitably tensions arise which involve reconciling what sometimes can turn out to be incompatible goals. Our interviews suggest a need for a clearer understanding of the necessary support at lower organizational levels for implementing and managing change, especially when attempting to prioritize or reconcile long-term policy goals and short-term contingencies.

The vast majority of modeling approaches – both industrial and academic – are developed without an appreciation for exactly how people and organizations work. In contrast, what seems to be emerging from our current work is the argument, essentially the CSCW argument, that software modeling technologies should be designed and deployed to match the way that people and organizations work. What is required is more understanding of exactly how software stakeholders work with abstraction, how they maximize their capacity for abstract thinking and how they use models and tools in their everyday work. On a related note, we also need more insight into how organizations structure themselves to solve complex problems, the role that abstraction plays in this process, how they use models as part of their practices, and how we might develop innovative modeling approaches that better match both individuals' and organizations' abstraction processes as well as tools that better support the way people want to model. While the theoretical benefits of MDE are often considered obvious, in the 'real world, real time' practical (and often messy) business world, MDE adoption and deployment can impact on organizational or business success in unanticipated ways. Consequently, adopting an MDE approach is, or probably should be, a *business* decision and therefore should be judged in terms of whether it meets a number of *business* and *organizational* goals rather than mere technical goals.

To summarize, through our two separate studies of MDE practitioners, comprising a total of 39 interviews, we have developed a taxonomy of technical, social and organizational issues related to MDE tool use in practice. This taxonomy serves as a checklist for companies developing and using tools, and also points to a number of open challenges for those working on MDE tool development. We end this article by highlighting some of these challenges, which have emerged from the data.

*Match tools to people, not the other way around.* Most MDE tools are developed by those with a technical background but without in-depth experience of human-computer interaction, CSCW or business issues. This can lead to a situation where good tools force people to think in a certain way. We recommend that the MDE community pay more attention to tried-and-tested HCI and CSCW methods, which can help to produce more useful and usable tools. There is empirical work on studying MDE languages and tools, but this is rarely taken into account. Research should avoid competing with the market. The research community should focus on issues not already tackled by commercial vendors. Our study found that the majority of tools support the transition from low level design to code. However, many bigger issues of modeling – such as support for early design stages and support for creativity in modeling – are relatively unexplored.

*Finding the right problem is crucial.* Our studies suggest that finding the right place for applying MDE is a crucial success factor. However, there is very little data about which parts of projects are good for MDE and which are not. Nor is there data about which tools are right for which jobs. In general, even the research community has not clearly articulated how to decide what to model and what not to model, and what tools to use or not to use.

*More focus on processes, less on tools.* The modeling research community focuses a lot on developing new tools and much less on understanding and improving processes. A particular case is the importance of tailoring. Very little research has been carried out on how best to tailor: what kinds of tailoring go on, how tools can or cannot support this, and how to develop simpler tools that can fit into existing processes with minimal tailoring.

*Open MDE Communities.* There is a distinct lack of open MDE developer forums. Those who do take the plunge with MDE are left feeling isolated, with nowhere to go to get technical questions answered or to discuss best practice. There are few examples of 'good' models online which people can consult, and efforts towards repositories of such models (cf. [11]) have achieved limited success. There is a chicken-and-egg dilemma here: if MDE is widely adopted, developer communities will self-organize; if it is not, they will not.

The big conclusion of our studies is that MDE can work, but it is a struggle. MDE tools do not seem to support those who try. We need simpler tools and more focus on the underlying processes. MDE tools also need to be more resilient: as with any new method, MDE is highly dependent on a range of technical, social and organizational factors. Rather than assuming a perfect configuration of such factors, MDE methods and tools should be resilient to imperfections.

For the most part, our sub-categories are already known and have been noted either in the literature or anecdotally. France and Rumpe [13], for example, point out that "Current work on MDE technologies tends to focus on producing implementation...from detailed design models". Aranda et al. [1] found that tailoring of processes is critical for MDE. Similarly, Staron found that organizational context has a huge impact on the cost effectiveness of MDE [28]. Indeed, many of our observations about organizational aspects of MDE adoption are not necessarily specific to MDE but are true of technology adoption generally. However, the contribution of the taxonomy is that it brings all of the factors – both technical and non-technical – together in one place to act as a reference point.

*Acknowledgments* The authors would like to thank all those who took part in the interviews, including those who facilitated the study at Ericsson and Volvo.

## References

1. Aranda, J., Damian, D., Borici, A.: Transition to model-driven engineering – what is revolutionary, what remains the same? Model Driven Engineering Languages and Systems, Spronger, (2012) 692–708
2. Brooks Jr., F.P.: The mythical man-month – essays on software engineering (2nd ed.). Addison-Wesley (1995)
3. Brown, B.: The artful use of groupware: An ethnographic study of how Lotus Notes is used in practice. Behavior and Information Technology 19(4) (1990) 263–273
4. Button, G., & Sharrock, W. (1996). Project work: the organisation of collaborative design and development in software engineering. *Computer Supported Cooperative Work (CSCW)*, *5*(4), 369-386.
5. Cabot, J., Teniente, E.: Constraint support in MDA tools: A survey. In Rensink, A., Warmer, J., eds.: ECMDA-FA. Volume 4066 of Lecture Notes in Computer Science., Springer (2006) 256–267
6. Chalmers, M.: A historical view of context. Computer Supported Cooperative Work 13(3) (2004) 223–247
7. Clark, T., Muller, P.A.: Exploiting model driven technology: a tale of two startups. Software and System Modeling 11(4) (2012) 481–493
8. Curtis, B., Krasner, H. and Iscoe, N., "A Field Study of the Software Design Process for Large Systems," Communications of the ACM, 31(11), pp. 1268-87, 1988.
9. de Sousa Saraiva, J., da Silva, A.R.: Evaluation of MDE tools from a metamodeling perspective. In Siau, K., Erickson, J., eds.: Principal Advancements in Database Management Technologies. IGI Global (2010) 105–131
10. Den Haan, J.: 8 reasons why model-driven approaches (will) fail. http://www.infoq.com/articles/8-reasons-why-MDE-fails (2008)
11. France, R.B., Bieman, J.M., Mandalaparty, S.P., Cheng, B.H.C., Jensen, A.C.: Repository for model driven development (ReMoDD). In Glinz, M., Murphy, G.C., Pezz` e, M., eds.: 34th

International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland, IEEE (2012) 1471–1472

12. France, R.B., Kazmeier, J., Breu, R., Atkinson, C., eds.: Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Volume 7590 of Lecture Notes in Computer Science., Springer (2012)
13. France, R.B., Rumpe, B.: Model-driven development of complex software: A research roadmap. In Briand, L.C., Wolf, A.L., eds.: International Conference on Software Engineering, ICSE 2007, Track on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA. (2007) 37–54
14. Grudin, J.: Why CSCW applications fail: Problems in the design and evaluation of organization of organizational interfaces. In Greif, I., ed.: CSCW, ACM (1988) 65–84.
15. Holsti, O.R.: Content Analysis for the Social Sciences and Humanities. Addison-Wesley Publishing Company, Reading, MA (1969)
16. Hutchinson, J., Rouncefield, M., Whittle, J.: Model-driven engineering practices in industry. Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21-28, 2011, ACM (2011) 633–642
17. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21-28, 2011, ACM (2011) 471–480
18. Kleppe, A.G., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
19. Kuhn, A., Murphy, G.C., Thompson, C.A.: An exploratory study of forces and frictions affecting large-scale model-driven development. Model Driven Engineering Languages and Systems, Spronger, (2012) 352–367
20. Lehman, M.M. “Process Models, Process Programs, Programming Support,” in Proceedings of the 9th International Conference on Software Engineering. IEEE Computer Society Press, Los Alamitos, CA, USA . (1987) 14-16.
21. MediaDev Survey, “Wide Gap Amongst Developers -’ Perception of the Importance of UML Tools, DeveloperEye Study Reveals” http://www.prweb.com/releases/2005/04/prweb231386.htm (accessed 19/9/2014), 2005.
22. Merisalo-Rantanen, H., Tuunanen, T., Rossi, M.: Is extreme programming just old wine in new bottles: A comparison of two cases. J. Database Manag. 16(4) (2005) 41–61
23. Paige, R.F., Varró, D.: Lessons learned from building model-driven development tools. Software and System Modeling 11(4) (2012) 527–539
24. Perez-Medina, J.L., Dupuy-Chessa, S., Front, A.: A survey of model driven engineering tools for user interface design. In Winckler, M., Johnson, H., Palanque, P.A., eds.: TAMODIA. Volume 4849 of Lecture Notes in Computer Science., Springer (2007) 84–97

25. Robinson, H., Sharp, H.: The social side of technical practices. In Baumeister, H., Marchesi, M., Holcombe, M., eds.: XP. Volume 3556 of Lecture Notes in Computer Science., Springer (2005) 100–108
26. Selic, B.: The pragmatics of model-driven development. IEEE Software 20(5) (2003) 19–25
27. Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S.: Model-driven software development - technology, engineering, management. Pitman (2006)
28. Staron, M.: Adopting model driven software development in industry – a case study at two companies. In Nierstrasz, O., Whittle, J., Harel, D., Reggio, G., eds.: Model Driven Engineering Languages and Systems, 9th International Conference, MODELS 2006, Genova, Italy, October 1-6, 2006. Volume 4199 of Lecture Notes in Computer Science., Springer (2006) 57–72
29. Taylor, R.N., Gall, H., Medvidovic, N., eds.: Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21-28, 2011, ACM (2011)
30. Tomassetti, F., Torchiano, M., Tiso, A., Ricca, F., Reggio, G.: Maturity of software modelling and model driven engineering: A survey in the Italian industry. In Baldassarre, M.T., Genero, M., Mendes, E., Piattini, M., eds.: 16th International Conference on Evaluation & Assessment in Software Engineering, EASE 2012, Ciudad Real, Spain, May 14-15, 2012, IET - The Institute of Engineering and Technology (2012) 91–100

## Appendix: Extended description of sub-categories in the taxonomy

This appendix expands briefly on the meaning of the sub-categories in the taxonomy. These extended descriptions were not included in Tables 1-5 due to lack of space. The appendix can therefore be used as a reference in case the titles of sub-categories in Tables 1-5 are unclear. Each sub-category defines a domain of discourse which interviewees highlighted. Interviewees may have made negative or positive remarks in each case: the presence of a sub-category therefore merely marks that this domain is of key importance to interviewees when applying MDE tools in practice.

### 1. Technical Factors (Table 2)

This branch of the taxonomy deals with technical issues that may affect the adoption of an MDE tool. That is, the sub-categories in Table 2 concern technical limitations of tools which may affect adoption, particular features of tools, and/or the impact of technical features of tools on the overall software development process.

#### 1.1 Tool Features / Specific functionalities offered in tools

This category details the presence or otherwise of particular tool functionalities or features. The sub-categories should not be considered an exhaustive list of possible tool features nor a list of features that a tool is expected to have; the list merely covers the features most regularly discussed by our interviewees.

##### 1.1.1 Modeling Behavior (1)

Does the tool allow system behavior to be modeled? If so, how is behavior modeled and is it an effective way of modeling behavior?

##### 1.1.2 Action Languages (1)

Does the tool support the use of action languages? If so, what action language(s) is supported? Is the action language and tools support provided for it effective?

##### 1.1.3 Support for Domain-Specific Languages (6)

Does the tool support the definition and application of domain-specific languages? If so, in what way – what kinds of facilities are provided?

##### 1.1.4 Support for Architecture (3)

What facilities, if any, does the tool have for specifying and modeling architecture?

##### 1.1.5 Code Generation Templates (6)

Does the tool support the use of code generation templates as a mechanism for allowing tool users to generated customized code? Or does the tool provide only a default format/style for generated code which cannot be changed?

### 1.1.6 UML Profiles (1)

Does the tool support the definition and application of UML profiles?

### 1.1.7 Scoped Code Generation (2)

Does the tool have features that support the incremental and iterative generation of code; that is, when the model changes, does the entire code base need to be re-generated, or only the code that is affected by the model change?

### 1.1.8 Model Analysis (5)

What facilities, if any, does the tool have for analyzing models either statically or dynamically?

### 1.1.9 Reverse Engineering Models (3)

Does the tool support reverse engineering – where models are created from existing code?

### 1.1.10 Sketching Models (1)

Does the tool support the creation and use of informal "sketched" models which might be used to capture ideas at an early stage in the modeling process?

### 1.1.11 Refactoring Models (1)

Does the tool support the refactoring of existing models?

## 1.2 Practical Applicability / Challenges of applying tools in practice

This category is not concerned with the features a tool has but how the tool is used in practice and whether it is possible to adapt it according to different processes and procedures.

### 1.2.1 Tool Scalability (1)

Is the tool able to cope with large-scale models – of the sort that may be found in large industrial projects?

### 1.2.2 Tool Versioning (1)

Are there issues associated with tool versioning? For example, frequent upgrades, maintaining compatibility with previous formats, etc.

### 1.2.3 Chaining Tools Together (2)

How easy/difficult is it to use multiple tools in conjunction with each other to provide end-to-end functionalities?

1.2.4 Industrial Quality of Generated Code (8)

Is the generated code of the quality, efficiency and size that would be expected in an industrial setting?

1.2.5 Flexibility of Tools (3)

To what extent is the tool flexible enough to adapt to different processes, other tools and/or ways of working? For example, does it impose strict processes on users? Or does it require other tools to be used with it?

1.2.6 Maturity of Tools (1)

Has the tool reached a level of maturity where robustness makes it suitable for an industrial project?

1.2.7 Dealing with Legacy (2)

What facilities, if any, does the tool have to support the use of existing artefacts – e.g. existing models, existing code, etc?

1.3 Complexity / Challenges brought on by excessive complexity in tools

This category concerns issues of complexity brought on by modeling tools or languages.

1.3.1 Tool Complexity (4)

How complex or otherwise is the tool itself? Is that level of complexity considered an appropriate level of complexity or otherwise?

1.3.2 Language Complexity (5)

If the tool supports a particular modeling language, how complex or otherwise is the language?

1.3.3 Accidental Complexity Introduced by Tools (1)

Does the tool – through its design – tend to introduce unnecessary complexity into either the modeling process or the resulting artifacts?

1.4 Human Factors / Consideration of tool users

This category is concerned with the effect the tool's design and features have on its users.

1.4.1 Whether Tools Match Human Abstractions (4)

Do the abstractions in the tool match the way that humans think about abstraction? Or does the tool force users to recast their own internal abstractions in to a form that can be captured in the tool?

1.4.2 Usability (4)

Is the tool designed with usability in mind so that users find it easy or intuitive to learn or is it designed in such a way that it actually impedes its use?

1.5 Theory / Theory underpinning tools

This category concerns theoretical underpinnings of tools, such as whether they are grounded in a formal theory or not.

1.5.1 Theoretical Foundations of Tools (1)

Does the tool actually have theoretical foundations? If so, what are they and what is their impact?

1.5.2 Formal Semantics (2)

Does the tool provide facilities to support – or impose – formal semantics in the modeling process? If so, how does this influence/impact modeling?

1.6 Impact on Development / Impact of tools on technical success criteria

This category is concerned with the effect tools have on higher-level project outcomes rather than the specific process of modeling/development itself.

1.6.1 Impact on Quality (2)

Are there features of the tool that affect the quality of the software developed using the tool – and are these positive effects or negative effects?

1.6.2 Impact on Productivity (4)

Are there features of the tool that affect the productivity of users – individually or as a team – when using the tool? Are these positive effects or negative effects?

1.6.3 Impact on Maintainability (3)

Are there features of the tool that affect the maintainability of the software produced using the tool – and are these positive effects or negative effects?

## 2. Internal Organizational Factors

This branch of the taxonomy is concerned with how tools relate to the way an organization is structured managerially, to any existing procedures or processes, and/or to any pre-existing factors such as the culture of the organization or the skill levels available.

### 2.1 Processes / Adapting tools to processes or vice-versa

This category is concerned with to what extent process change is necessitated by the introduction of a tool. For example, does the tool mean that existing organizational processes have to be changed to support the tool? Or is the tool flexible enough that it can be easily adapted to fit an existing process?

#### 2.1.1 Tailoring to a Company's Existing Processes (5)

In a sense, this is an "applied" version of the tool flexibility issue – how possible is it to adapt the tool to existing processes? How easy is it to leverage existing expertise?

#### 2.1.2 Sustainability of Tools over the Long Term (3)

This sub-category concerns the effort needed to maintain the use of a tool within the organization over the long term. For example, does it require significant ongoing maintenance efforts that require internal resources? Or can the tool be bought once and then used at no cost indefinitely? How easily can the tool be adapted over time when the nature of the organization's business evolves?

#### 2.1.3 Appropriating Tools for Purposes They Were Not Designed For (3)

It is well known that users will use any tool in ways that were unforeseen by the tool's developers – are there examples of this happening with MDE tools and what is the impact of this appropriation on a project/organization?

#### 2.1.4 Issues of Integrating Multiple Tools (6)

Again, this is an "applied" version of the "tool chaining" technical issue – what are the consequences, from an organizational perspective, of adopting a particular tool when that tool has to be integrated with a range of other tools?

#### 2.1.5 Migrating to different tool versions (3)

How does the support for migrations between tool versions affect an organization's processes? Is migration easily supported or does it require an organization to introduce lengthy and complex internal processes?

#### 2.1.6 Offsetting Gains: Tools bring gains in one aspect but losses in another (2)

Are there aspects of a tool that appear to bring gains in one part of an organization but incur losses in another? An example would be code generation, which could bring clear productivity gains to one team, but could bring losses to another if the code generated is inefficient and has to be adapted before deployment.

### 2.1.7 Whether Maintenance is carried out at the Code or Model Level (3)

What processes does an organization have in place to enforce good practice, such as ensuring changes are made at the model level rather than by modifying generated code? Or are new processes required because an organization makes changes to generated code which then have to be reflected back in the model?

## 2.2 Organizational Culture / Impact of cultural attitudes on tool application

This category concerns cultural issues related to an organization.

### 2.2.1 Tailoring to a Company's Culture (4)

However flexible a tool is, it will impose new ways of working on any organization adopting its use – does the tool offer any facilities to support its tailoring to an organization's existing culture? And if not, what are the consequences?

### 2.2.2 Inertia: Reluctance to Try New Things (1)

Is there a culture of reluctance in the organization to try new things? If so, does the tool naturally exacerbate this problem or alleviate it?

### 2.2.3 Over-Ambition: Asking Too Much of Tools (1)

What do tools promise? And are these promises realistic when they are conveyed to companies? Do companies believe that tools will deliver more than they realistically can?

### 2.2.4 Low Hanging Fruit: Using Tools on Easy Problems First (6)

How do organizations apply the tool in practice, and to which problems? Some companies, for example, have found success in applying a tool to easily solved and well understood problems. Other companies have tried to apply MDE tools to very complex problems. Which is the best stragegy?

## 2.3 Skills / Skills needed to apply tools

This category is concerned with how adoption/use of a particular tool affects notions of "skills" in an adopting company – and how they are acquired.

### 2.3.1 Training Workforce (11)

What impact does adoption of a particular tool have on the training requirements of the adopting company?

### 2.3.2 Availability of MDE Skills in Workforce (4)

Is the choice of tool affected by the availability of suitably experienced practitioners for recruitment? Or is there a type of MDE skill that transcends a particular tool? Alternatively, is tool choice a result of available expertise?

## 3 External Organizational Factors

This branch of the taxonomy deals with issues concerning external influences on the organization when they adopt MDE tools and how those influences affect application of the tools.

### 3.1 External Influences /Factors which an organization has no direct control over

If a company has collaborators, suppliers or standards to adhere to, how does the use of a tool affect those relationships? Do external partners impose conditions that affect tool use or does use of particular tool impose conditions on external partners?

### 3.1.1 Impact of Marketing Issues (1)

Does using the development method *du jour* influence the adoption of any particular tool? And what is that effect?

### 3.1.2 Impact of Government/Industry Standards (4)

Some software is subject to extreme external verification. Does the use of a particular tool (and associated methods) impact on this – in a positive or negative way?

### 3.2 Commercial Aspects / Business considerations impacting on tool use and application

This category is concerned with how tool choice/adoption impacts how businesses make their commercial decisions – whether advantages offset costs, for example.

### 3.2.1 Business Models for Applying MDE (3)

Does the tool (approach) support the use of existing business models? Or do business models have to be adapted to adopt MDE tools?

### 3.2.2 Cost of Tools (5)

Is the cost of a commercial MDE tool an impediment to the use of that tool – or otherwise? Does the cost of a tool override other considerations?

#### 3.2.3 How to Select Tools (2)

How should potential users of a tool make such a decision? Are there appropriate resources available to inform decisions? For example, are there reports available that compare capabilities of tools?

## 4 Social Factors

This branch of the taxonomy deals specifically with social issues such as trust and control regarding tool selection and use.

### 4.1 Control / Impact of tools on whether stakeholders feel in control of their project

This category is particularly concerned with the relationship that tool users have with stakeholders such as tool vendors.

#### 4.1.1 Ways of Interacting with Tool Vendors (2)

What opportunities, if any, are there for interacting with the tool vendor and how do these relate to how open the tools are? For example, will the vendor adapt the tool for a particular organization? If so, does this incur a cost? Or is the vendor adamant that tools cannot be adapted for particular situations?

#### 4.1.2 Subverting Tools: Workarounds Needed to Apply Them (1)

Is it necessary to create specific work-arounds to allow tools to match more closely the working practices of the user? For example, these can often arise because lack of flexibility of the tool vendor and can lead to a lack of trust in the tool.

### 4.2 Trust / Impact of trust on tool use and Adoption

This category is concerned with all aspects of how trust affects tool use and attitudes towards tool adoption.

#### 4.2.1 Trust of Vendors (4)

How is trust in the tool vendor established (or lost)? For example, what is the vendor's reputation when it comes to matters such as cost, support, updates, etc?

#### 4.2.2 Engineers' Trust of Tools (6)

Are there aspects of the tools that impact on the specific engineering aspects of their use? For example, is it possible to establish trust in the quality of the code that is generated or the robustness and accuracy of model checking/analysis?

### 4.2.3 Impact of Personal Career Needs (1)

Does the tool meet or hinder the expectations of individual users? For example, are developers unsure about using obscure tools that do not offer them skills that might have broader appeal – and therefore career opportunities?

## 4.3 Open Community

This category is concerned with the availability or otherwise of an open community of tools users that provide peer support in the use of tools.

### 4.3.1 Developer Forums

Do developer forums exist to support users of the tool by providing examples, advice and assistance?