Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem?

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

[1] School of Computing and Communications, Lancaster University, Lancaster, UK
[2] Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden

Abstract. An oft-cited reason for lack of adoption of model-driven engineering (MDE) is poor tool support. However, studies have shown that adoption problems are as much to do with social and organizational factors as with tooling issues. This paper discusses the impact of tools on MDE adoption and places tooling within a broader organizational context. The paper revisits previous data on MDE adoption (19 in-depth interviews with MDE practitioners) and re-analyzes the data through the specific lens of MDE tools. In addition, the paper presents new data (20 new interviews in two specific companies) and analyzes it through the same lens. The key contribution of the paper is a taxonomy of tool-related considerations, based on 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 [1] highlights "insufficient tools" as one of the eight reasons why MDE may fail. Kuhn et al. [2] identify five points of friction in MDE that introduce complexity; all relate to MDE tools. Staron [3] found that "technology maturity [may] not provide enough 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 [4].

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. [5, 6], 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.

The key contribution of this paper is a 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., [7–9]) 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 [10]. 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. 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. Finally, the paper discusses how the taxonomy can be used to advance research and development of MDE tools (Section 6).

2 Context and Related Work

Tools have long been of interest to those considering the use of technology in industrial settings. There is general agreement that, despite considerable financial investment, their use often proves far less successful than anticipated, and that much of this lack of success can be attributed to organizational rather than technical factors.

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) [11] [12, p. 109]. In contrast, another approach, exemplified by Grudin [13] and Brown [14], 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 [14] 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 Hilkka et al. [15] 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 [16] emphasized the importance of organizational rather than technical factors.

However, when considering MDE tools there is little in the way of systematic or extensive evaluation. Cabot and Teniente [9] acknowledge MDE tools but suggest that they have several limitations regarding code generation. Selic [17] 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. [18] have claimed that MDE does not make sense without tool support. Two studies [19, 2] 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.

There have been two recent, and very different, studies about the experience of developing and deploying MDE tools. Paige and Varró [10] 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 [20] 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 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 [5, 6]. However, prior publications gave a holistic view of the findings and did not include data on tools. The additional contribution of this paper, therefore, is the tools taxonomy, which comes from a completely new analysis of the interview transcripts. The procedure for selecting and carrying out the interviews has been described elsewhere [6]. 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.

The second set consists of 10 interviews at Ericsson AB and 10 interviews 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 electrical propulsion. The interviews cover more than 20 hours of recorded conversation and were conducted in the same semi-structured fashion as the first set.

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 [21], 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 and amalgamated 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.

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.

Tables 1-4 form the taxonomy. Each category is briefly defined in the tables. 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 checklist 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 research community.

The reader should avoid the temptation to make comparisons between factors based on the table. For example, there are more technical categories than any other; this does not necessarily mean that technical issues are more important.

4.1 Technical Factors

Table 1 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 (scaleability), 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

Table 1. Technical Categories.

Category	**Sub-Category**
Tool Features *Specific functionalities offered in tools*	- Modeling Behavior - Action Languages - Support Domain-Specific Languages - Support for Architecture - Code Generation Templates - UML Profiles - Scoped Code Generation
Practical Applicability *Challenges of applying tools in practice*	- Tool Scaleability - Tool Versioning - Chaining Tools Together - Industrial Quality of Generated Code - Flexibility of Tools - Maturity of Tools
Complexity *Challenges brought on by excessive complexity in tools*	- Tool Complexity - Language Complexity - Accidental Complexity Introduced by Tools
Human Factors *Consideration of tool users*	- Whether Tools Match Human Abstractions - Usability
Theory *Theory underpinning tools*	- Theoretical Foundations of Tools - Formal Semantics
Impact on Development *Impact of tools on technical success criteria*	- Impact on Quality - Impact on Productivity - Impact on Maintainability

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.*"

Universally, 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 from tool vendors 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 [22]), 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. Whether a particular MDE tool falls down on the side of essential or accidental

Table 2. Internal Organizational Categories.

Category	**Sub-Category**
Processes *Adapting tools to processes or vice-versa*	- Tailoring to Existing Processes - Sustainability of Tools - Appropriation - Integration Issues - Migration Issues - Offsetting Gains - Maintenance at Code versus Model Level
Organizational Culture *Impact of cultural attitudes on tools and their application*	- Tailoring to Existing Culture - Inertia - Over Ambition - Low Hanging Fruit
Skills *Skills needed for tools to succeed*	- Training - Availability of Skills

complexity is down not just to the specifics of the tool but also the social and organizational context in which it is used.

4.2 Internal Organizational Factors

Category Descriptions Table 2 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 MDE 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.*" One conclusion here is that it is not easy to identify which applications are appropriate for MDE and which are not. Apart from obvious industries where MDE has been applied more widely than others (cf. the automotive industry), we do not have a fine-grained way of knowing whether MDE will succeed or not in a chosen application area.

A curious paradox of MDE is that it was developed as a way to improve portability [23]. 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

Category Descriptions External organizational factors (Table 3) are those which are outside the direct control of organizations. External Influences in-

Table 3. External Organizational Categories.

Category	**Sub-Category**
External Influences *Factors which an organization has no direct control over*	- Marketing Issues - Government and Industry Standards
Commercial Aspects *Business considerations impacting on tool use and application*	- Business Models - Cost of Tools - Selection of Tools

cludes 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 includes 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.*"

Table 4. Social Categories.

Category	**Sub-Category**
Control *Impact of tools on whether stakeholders feel in control of their project*	Interaction with Tool Vendors Subverting Tools
Trust *Impact of trust on tool use and adoption*	Vendor Trust Engineers' Trust

4.4 Social Factors

Category Descriptions When it comes to MDE tools, social factors (Table 4) 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 (Interaction 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.

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 second 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. As we shall see later, the second dataset fits neatly into the taxonomy, with only one suggested extension. As with Section 4, we pick out some useful insights and illustrate with quotes.

5.1 Technical Factors

The 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."* Interestingly, this interviewee has 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 modelling."* 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++"*.

5.2 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."* This example illustrates how MDE can be effectively used to manage external influences: in this case, Ericsson model the API as a UML profile and manage 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."*

5.3 Internal Organizational Factors

One Ericsson employee notes 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 are difficult to use since they are so unintuitive. The threshold for learning how to produce and consume models can be overcome but it requires an organization where developers are not exposed to different tools between projects.

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."*

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."*

5.4 Social Factors

It seems that the effort put into tailoring the tools to the existing organization has paid off at Volvo since the domain experts trust the tools to deliver: *"I do like it. In quite a lot of ways. Especially for the kind of software we are developing. It's not like rocket science, really. It's like systems where you have a few signals*

[3] AUTomotive Open System ARchitecture; www.autosar.org/

in, you should make a few decisions, make some kind of output. It is not that difficult applications. There are no complex algorithms... And for that I think Simulink is very sufficient... I like it."

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, *"you basically come up empty-handed."*

5.5 Taxonomy Validation

The study at Ericsson and Volvo is in itself revealing about MDE practice. However, for the purposes of this paper, it serves primarily to validate our taxonomy. 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, in Table 4. 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. This, however, was not unexpected. Issues of MDE adoption and tool use are complex and involve many dependencies, so it could be unrealistic to expect a taxonomy with completely orthogonal sub-categories.

6 Discussion and Conclusions

Through 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 now discuss 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 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 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. Clearly, some industries are more successful with MDE than others, but, in general, even the research community has not clearly articulated how to decide what to model and what not to model.

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. [24]) 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 [25], for example, point out that "Current work on MDE technologies tends to focus on producing implementation. . . from detailed design models". Aranda et al. [19] 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 [3]. 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.

This paper began with the question: "Are tools really the problem?" The answer appears to be both yes and no. MDE tools could definitely be better. But good tools alone would not solve the problem. A proper consideration of people and organizations is needed in parallel. As one of our interviewees noted: "*Wait a second, the tools are really interesting, I agree, but to me it's much more about what is the process and the technique and the pattern and the practice.*"

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. Den Haan, J.: 8 reasons why model-driven approaches (will) fail. http://www.infoq.com/articles/8-reasons-why-MDE-fails (2008)
2. Kuhn, A., Murphy, G.C., Thompson, C.A.: An exploratory study of forces and frictions affecting large-scale model-driven development. [26] 352–367
3. 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.: MoDELS. Volume 4199 of Lecture Notes in Computer Science., Springer (2006) 57–72
4. 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.: EASE, IET - The Institute of Engineering and Technology / IEEE Xplore (2012) 91–100
5. Hutchinson, J., Rouncefield, M., Whittle, J.: Model-driven engineering practices in industry. [27] 633–642
6. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of mde in industry. [27] 471–480
7. Pérez-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
8. de Sousa Saraiva, J., da Silva, A.R.: Evaluation of mde tools from a metamodeling perspective. In Siau, K., Erickson, J., eds.: Principle Advancements in Database Management Technologies. IGI Global (2010) 105–131
9. 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
10. Paige, R.F., Varró, D.: Lessons learned from building model-driven development tools. Software and System Modeling **11**(4) (2012) 527–539
11. Chalmers, M.: A historical view of context. Computer Supported Cooperative Work **13**(3) (2004) 223–247
12. : Where the action is: the foundations of embodied interaction. MIT Press, Cambridge, MA, USA (2001)
13. 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
14. 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
15. 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
16. 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
17. Selic, B.: The pragmatics of model-driven development. IEEE Software **20**(5) (2003) 19–25
18. Stahl, T., Völter, M., Bettin, J., Haase, A., Helsen, S.: Model-driven software development - technology, engineering, management. Pitman (2006)

19. Aranda, J., Damian, D., Borici, A.: Transition to model-driven engineering - what is revolutionary, what remains the same? [26] 692–708
20. Clark, T., Muller, P.A.: Exploiting model driven technology: a tale of two startups. Software and System Modeling **11**(4) (2012) 481–493
21. Holsti, O.R.: Content Analysis for the Social Sciences and Humanities. Addison-Wesley Publishing Company, Reading, MA (1969)
22. Jr., F.P.B.: The mythical man-month - essays on software engineering (2. ed.). Addison-Wesley (1995)
23. 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)
24. 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., Pezzè, M., eds.: ICSE, IEEE (2012) 1471–1472
25. Briand, L.C., Wolf, A.L., eds.: International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA. In Briand, L.C., Wolf, A.L., eds.: FOSE. (2007)
26. 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. Proceedings. In France, R.B., Kazmeier, J., Breu, R., Atkinson, C., eds.: MoDELS. Volume 7590 of Lecture Notes in Computer Science., Springer (2012)
27. 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. In Taylor, R.N., Gall, H., Medvidovic, N., eds.: ICSE, ACM (2011)