Descriptive vs Prescriptive Models in Industry

Rogardt Heldal[1,2], Patrizio Pelliccione[1], Ulf Eliasson[1], Jonn Lantz[3],
Jesper Derehag[4], Jon Whittle[5]
[1]Chalmers University of Technology | University of Gothenburg
[2]Bergen University College, [3]Volvo Cars, [4]Ericsson
[5] Lancaster University, School of Computing and Communications
heldal@chalmers.se, patrizio.pelliccione@gu.se, eliassou@chalmers.se, jonn.lantz@volvocars.com,
jesper.derehag@ericsson.com, j.n.whittle@lancaster.ac.uk

ABSTRACT

To understand the importance, characteristics, and limitations of modeling we need to consider the context where models are used. Different organizations within the same company can use models for different purposes and modelling can involve different stakeholders and tools. Recently, several papers discussing how industries use MDE have been published and they have contradictory findings.

In this paper we report lessons learned from our collaborations with three large companies. We found that it is important to distinguish between descriptive models (used for documentation) and prescriptive models (used for development) to better understand the adoption of modelling in industry. Our findings are valuable for both academia and industry. A better understanding of modeling in large companies can help academia conceiving innovative MDE solutions that can have a real impact in industry. On the other hand, industry can better understand how to properly exploit MDE and what to expect from it.

1. INTRODUCTION

The map of Model Driven Engineering (MDE) is complex and increasingly involves new aspects of the development and maintenance processes. One company may have several different MDE strategies, and these strategies are not always connected with each other.

In recent years, somewhat contradicting studies discussing the importance of modeling have been published. From [23] we learn that modeling can be suitable for embedded systems. Also, the study in [34] convinces us that modeling is ready for industry, given that one understands the consequences of such adoption. Different signals come from other studies: the study in [15] shows that there is a lack of use of modeling; similar findings can be found in [29]. None of these papers are wrong. However, we feel that some aspects still need to be clarified in order to actually understand the adoption of modeling in industry.

For small companies, since all employees are familiar with the whole development process, it might be good enough to have one or two interviews per company. However, for large companies this is not the case if the ambition is to understand how methods, processes, and techniques are used within the company; this is the case of software models. Even doubling or tripling the number of interviewees within a larger company may not provide a clear and comprehensive map of how models are used. The point is that even though the number of interviewed people is high, it might be the case that the selected sample of the population is not representative of the entire company: certain groups might be completely neglected. The number of participants is indeed important, but it is not enough: there should be a proper coverage of the variety of the heterogeneous population.

Our strategy to mitigate this intrinsic risk is to exploit pre-studies to gain additional knowledge of the considered companies. This knowledge is then used to select the population so as to represent as much as possible the different realities that have been identified within each company. The additional knowledge acquired through pre-studies is exploited first to classify the population into different groups and then to select representatives of each group to offer a coverage of the heterogeneous population.

In the context of modeling, we discovered that it is important to make a distinction between groups using models as prescriptive artifacts from groups that use models to describe subjects that already exist. Therefore, to better understand how MDE is used in practice, in this paper we selected the population to ensure the presence of representatives of both groups using descriptive models and groups using prescriptive models.

Summarizing, in this paper we answer the following research questions:

RQ1: Do companies in the embedded systems domain use prescriptive and/or descriptive models?

Main findings: The three companies use both descriptive and prescriptive models. Large and complex systems need some high-level descriptive models to obtain an overview of the system under construction. On the contrary, the use of prescriptive models is much more dependent on specific project groups and strategies. In fact, some projects do not make use of prescriptive models but go directly from descriptive models to code.

RQ2: To what purpose are prescriptive and descriptive models used within different contexts?

Main findings: The primary purpose of prescriptive modeling is development, while the primary purpose of

MODELS '16 Saint-Malo, France

DOI: http://dx.doi.org/10.1145/2976767.2976808

descriptive modeling is simplification and communication. Often there is no general agreement on who should be the consumers of descriptive models. The identification of the correct abstraction level is much more complex in the case of descriptive models. Also, there are often several levels of descriptive models within a company without any automatic transformation among them.

RQ3: What is the relationship between descriptive and prescriptive models and code?

Main findings: There is generally no need for model transformations from descriptive to prescriptive models. However, loose and flexible connections between descriptive and prescriptive models are desirable.

The paper is structured as follows: Section 2 introduces the concept of descriptive and prescriptive models and provides examples of them. Section 3 describes the companies and the specific groups that are involved in this study. Section 4 describes the research approach adopted in this paper. Sections 5, 6, and 7 summarize the results and lessons learned of RQ1, RQ2, and RQ3, respectively. Section 8 discusses models in relation to software development processes. Section 9 presents the threats to validity and Section 10 reports studies that are related to our work. Finally, Section 11 concludes with final remarks.

2. DIFFERENT TYPES OF MODELS

Models are typically used to prescribe something or to describe something [32]. The thing that the model is about is called the **subject** [19]. The model is created with the **intent** to satisfy a particular purpose [22]. The purpose is related to the **consumer** of the model and her/his/its characteristics. A consumer can be a human, but possibly also a software system. It is then clear that consumer and intent are two aspects that strongly influence the abstraction level of the model as well as the concrete syntax used to describe and present the model, e.g. graphical, textual, and/or tree-like. A model's importance might vary during its lifetime.

It is hard to see whether a model is prescriptive or descriptive by only observing it only from the consumer point of view. We need to consider other aspects, as discussed here.

2.1 Descriptive models

Descriptive models describe a subject that already exists. The content to be included in the model is derived from the observations of the system being described. This content is chosen according to the specific intent. The content of a descriptive model can be true or false depending on whether or not it correctly describes the subject [32]. Descriptive models are created from observations of the existing artifacts and are created for a specific intent. Once the intent is satisfied, the model might lose its importance. Moreover, when the subject or the intent changes the model might become obsolete and needs to be updated by observing the new version of the subject.

Examples of descriptive models in the literature are:

- *Models of ideas and vision about the system to be developed* - These models are based on requirements, sketches and other textual documents. In addition, these type of models describe subjects that are implicit (what is in the mind of the modeller) in terms of explicit models. Even though the subject is implicit, it exists. The intent is often to exploit the descriptive model to perform some analysis and then to have feedback before actually implementing the system. An example is the approach described in [28], which helps software architects in conceiving the architecture of the system to be realized and in understanding, through executable models, the functioning of the system and discovering potential inconsistencies.
- *Models extracted from a running system or source code* - a model representing an abstraction of the running system is created to visualize particular aspects of the system (e.g. [35]), analyse the software behavior (e.g. [24]), or to serve other purposes, such as simulating a system upgrade [12] or helping developers to write applications that correctly use a complex service [4].

2.2 Prescriptive models

In the context of prescriptive models the subject does not yet exist, e.g. it is not yet developed or the models are used to (re-)generate a new version of the subject. Then prescriptive models are used to prescribe a subject (e.g. develop a system), not to describe it. The content of the model is derived from the information available at that time (e.g. before the development of the system), and according to the specific intent. The most common consumers of prescriptive models are model-to-model or model-to-code transformations. This implies that often it is not so important to have graphical concrete syntaxes to represent the model, but these models should be represented in a format that is readable by a machine. This also explains why, according to what we observed, UML is not so popular for prescriptive models. In fact, the semantics of UML models is not exactly defined, as explained in [2], and this would hamper the automatic translation towards other models or code.

Typically prescriptive models are defined at a low-level of abstraction since they should contain the information that is needed to implement the system. Prescriptive models might also be part of chains of model-to-model transformations that from high-level prescriptive models, through various model-to-model transformations (which at each step add information), generate low-level models and finally the actual code through a model-to-code transformation. The addition of information during transformation (if not completely automated) creates by itself a new artifact with its own lifecycle; for instance the re-execution caused e.g. by evolution of the system, needs to be carefully managed. Prescriptive models are often used for development and their importance might degrade when the system is implemented.

Examples of prescriptive models in the literature are:

- *Prescriptive models to generate code* - models tailored to a specific domain might be used to automatically generate code; examples are Matlab and Simulink models used to generate automatically C-code in critical applications [20].
- *Chain of transformations towards code* - a chain of transformations permits developers to reason at a higher level of abstraction and at each intermediate transformation new information might be added or synthesized from available information. An example might

be found in the approach described in [5, 11], where models describing the high-level mission to be realized by a team of autonomous multicopters are transformed and manipulated by a chain of transformations to instruct the robots with precise and detailed tasks.

- *Models used to generate artifacts different from code* - prescriptive models might be very useful not only for generating code but also for generating configuration files, test cases [7] and any other artifact, different from code, but necessary to develop a product.

3. DESCRIPTION OF THE COMPANIES

The three companies involved in this study are Ericsson, Volvo Cars (VC) and Volvo Group Trucks Technology (VGTT). There are many commonalities between the two automotive companies we studied but also several differences. Even though VC and VGTT used to belong to the same company, AB Volvo, they have been individual organizations since 1999 when Ford acquired VC. Also, their products are different; VC is producing cars and VGTT is producing heavy-weight vehicles such as trucks. To better understand our three studies, in the following we give a brief overview of Ericsson, VC, and VGTT.

3.1 Volvo Cars

VC has a history with several owners. Originating from the Swedish Volvo, the brand was separated from the main Volvo when the car development was sold to Ford. Ford owned Volvo for several years before it was sold to Geely. The final transition aligned with the development of a new platform, thus making the platform independent of other OEMs and removing brand-specific variability. Thus initially the complexity of the new platform was reduced; however, it has nevertheless grown - in line with the general trends of more advanced and software dependent functionality and extended functionality, such as hybridization.

Regarding in-house development of software, VC's history started with engine control software in the 90's, followed by other systems on a larger scale a decade later. Still today, the majority of the software development is outsourced while the system architecture of the car is entirely designed in-house. VC also performs the system integration and verification. Nevertheless, an increasing percentage of the software and in several cases complete ECU (Electronic Control Unit) application layers are made in-house.

Development in VC has traditionally followed a predictive process model, commonly referred to as the "V-model". The development of a new product feature or function starts with complete vehicle analysis and requirement design, often scoped as customer or user use cases. These high-level requirements represent the base for an **architecture** description. The scope of the software functionality is defined and allocated to specific ECUs.

After that, during detailed function requirement design, the concept specification is broken down into detailed requirements and a software design. This design description documents the software design of the car, its components, the mapping between components and ECUs, the interfaces, and the communication. Based on the design and detailed requirements, the software is produced either by in-house teams or sub-contractors. The sub-contractors have the responsibility for the integration on the ECUs, with the complete software, and handing it over to the OEM, which starts the tedious process of integrating and verifying the functions, first in various rigs and finally in vehicles.

3.2 The Volvo Group

Within the Volvo Group, our study focused on Trucks Technology (VGTT), which develops software for various truck brands: Volvo, Renault, Mack, UD, and Eicher. The five brands are developed on top of a common platform. The truck platform serves the same purpose and has the same overall structure as the car platform at Volvo Cars but has a longer life expectancy (around 20 years). VGTT takes overall responsibility for system design and integration, but external suppliers implement most of the software. The result is that some nodes contain a mix of in-house and third-parties software, and some other nodes contain only software provided by suppliers.

Similar to VC, development within VGTT has traditionally followed predictive process modeling. The development of a new product starts from the definition of high-level requirements and use cases. For a long time, the architecture was just a physical topology. Once the complexity of the system required a more complex architecture, they decided to create an architecture group. In the early 2000's VGTT initiated the development of a new generation of its embedded real-time Electrical architecture [25]. The new architecture was much more complex than the one of the previous generation. It was created to support several brands: Volvo Trucks, but also the Renault Truck, Mack Truck, and the North American Volvo Truck [25].

The metamodel to be used for the solution was highly inspired by EAST-ADL [8] and AUTOSAR[1]. The development teams start from a logical design specification. This specification represents the logical design and architecture and it is the central cogwheel of the software production. It is used both for requirements for the suppliers and the in-house teams. Concluding, similarly to VC, VGTT separates architecture from design.

3.3 Ericsson

Ericsson is the fifth largest software supplier in the world. Evolved Packet Gateway product (EPG) is a department within Ericsson, and it has been developed since the late 1990s. It has used prescriptive modeling since 2006. The EPG unit consists of approximately 700 employees with around 300 software developers. EPG delivers software for mobile communications on the GSM, 3G, 4G and multi-standard networks. The majority of the software developed at EPG is deployed on a single hardware node, so there are strict constraints on memory consumption and processing speed. EPG uses both descriptive and prescriptive models. Descriptive models are typically used on a higher level, either describing system-level architecture or requirements. Prescriptive models are used for code-generation. EPG uses IBM Rational Rhapsody[2] and a subset of UML 2.0 for prescriptive models. The reason for adopting prescriptive models and MDE techniques at EPG was primarily to handle the rising complexity of the product. At the time of the decision to introduce MDE, the ultimate goal was to transition the entire legacy code-base into MDE over time, but for practical reasons this has not yet occurred. Instead, mainly

[1] http://www.autosar.org/

[2] http://www-03.ibm.com/software/products/en/ratirhapfami

new subsystems are implemented in prescriptive MDE and legacy code tends to continue being developed in a code-centric fashion. Since Ericsson is such a large company, various ways-of-working and documentation exists so whenever in this paper we mention Ericsson it is implied to mean specifically the EPG department.

4. RESEARCH METHODOLOGY

Our methodology is based on a set of semi-structured interviews carried out at the three companies mentioned in the previous section. In addition to these formal interviews, we used informal observations of day-to-day working practices (including modeling) within the companies. These observations have been carried out mainly by the two industrial authors (i.e. Jonn Lantz and Jesper Derehag) of this paper as well as by the first and third authors (i.e. Rogardt Heldal and Ulf Eliasson, respectively) who have spent considerable time within two of these companies. Overall, the industrial authors have over 10,000 hours of experience within their respective companies and the two academic authors mentioned above have spent more than 12,000 hours in Ericsson and VC. This in-depth engagement with the companies has enabled us to get a much richer picture of modeling within the companies, which was used to inform the design of the more formal interview-based study.

4.1 Design of the interview study

To help design our interview study we performed a pre-study. The pre-study consisted of two workshops and two interviews with senior staff. In the workshops, the three academic authors and one representative from each of the three companies participated. The industrial participants were senior staff, being technical specialists and/or architects at their respective company. The objective of the workshops and these preliminary interviews was to explore the landscape of modeling in industry and to collect knowledge in order to select the population of the study, i.e. to carefully select the people to be interviewed in order to have representatives of the heterogeneity of each company. These workshops also served to focus our research questions as well as the questions to be used in the interviews.

In more detail, the focus of the first workshop was to understand how, why, and where modeling was used at the different companies. From the first workshop, a first draft of the research questions, as well as interview guides, were constructed. These were used as input to the second workshop, which had the objective of refining the topics (to be of interest for both academia and industry), going through the interview guide, and identifying potential subjects (within the companies) to be interviewed. Both workshops had a defined agenda, but discussions were encouraged to bring together and collect interests of the participants. The result of the workshops was that the use of models was quite different within groups working with architectures (which mainly involve descriptive models) and groups involved in design (which mainly involve prescriptive models). Hence, interviewees at each company were selected to ensure a coverage of both groups and uses of both prescriptive and descriptive models.

To testify that the pre-study brings value to the design of the interview study, we can mention another study that some of the authors made in the past [6]: this study basically discusses exclusively about prescriptive modeling, without really considering descriptive modeling and the relationship between them.

4.2 Interview study

According to the design of the study discussed above we identified 15 interviewees based on their role to cover the complete development process, from the architecture to the implementation. Specifically, we selected the population of the study to include people dealing with both prescriptive and descriptive models.

The interviews were semi-structured: we created a list of questions covering a set of themes such as background, development process used, different levels of modeling, personal use of modeling, the purpose of modeling, the perceived impact of models, etc. However, interviewers had the flexibility to change the order of questions and the length of time devoted to each question to pursue interesting topics. All interviews were recorded and transcribed. In addition, the interviewer took notes. Each interview lasted for about one hour. In analyzing the transcripts, we followed an inductive approach: no pre-defined hypotheses were prepared, but key findings were allowed to emerge from the data.

5. RESULTS AND FINDINGS FOR RQ1

In this section we describe the results and findings for **RQ1: Do companies in the embedded systems domain use prescriptive and/or descriptive models?**

An overview of where models are used within these three large companies is given in Section 5.1 for VC, Section 5.2 for VGTT, and Section 5.3 for Ericsson. Then, Section 5.4 presents the lessons learned for this research question.

5.1 Volvo Cars

At VC, models are used on a number of different levels: descriptive models on higher abstraction levels and prescriptive models when we get closer to the implementation.

Figure 1 shows a simplified map of the main development artifacts used during development at the electrical department of VC. The Architecture Design Components (ADC) are the architecture components. The Logical Architecture Components (LAC) allocate responsibility and functionality to the different ECUs. The Logical Components (LC) con-

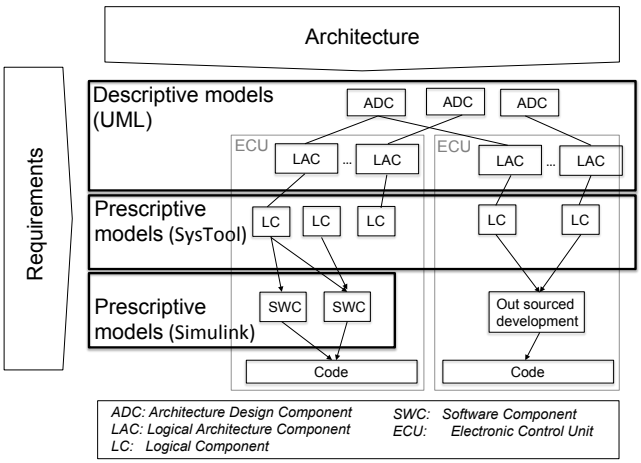

Figure 1: Main model artifacts at VC

tain requirements and interfaces which are implemented by Software Components (SWC) representing the actual AUTOSAR application software.

The architecture group is responsible for a high-level design consisting of ADCs and descriptive signaling. These descriptive models are realized in Rational Architect. Specifically, there was a need to create UML profiles to add domain- and company-specific concepts. The lowest layer of this architecture is composed of LACs.

LACs are primarily a collection of related requirements, dividing the functionality and responsibility between them. LACs are then allocated to physical ECUs, dividing the responsibility for implementing different functionality to different ECUs. One ECU can contain several LACs, but one LAC can only be allocated to one ECU. Within each ECU or LAC, the group responsible for the implementation has great freedom, including changing the signaling from the initial plan.

The prescriptive design is stored in a proprietary tool, which we refer to as SysTool. This is due to the need for fine-grained control of signals, buses, and hardware as well as optimization for the distributed embedded software functions and networks typical for cars.

The design in SysTool is driven by the development groups with the architecture group in a moderating role, focusing on system-wide functionality, networks, and services. Hence, the coupling to the architecture is loose when it comes to implementation of functions and features. LCs in SysTool contain requirements and detailed signaling (ports). The lower layers of the SysTool architecture depends on whether in-house or outsourced developers are intended for ECU software development.

If outsourced, the LC design will be composed into textual requirements which are sent to the supplier together with an AUTOSAR ECU extract, generated from SysTool, defining the network interfaces. This ECU extract is generated automatically, based on the information (ports) of the LCs.

Where in-house development is used, AUTOSAR Software Components (SWC) are defined. Engineers can utilize automated creation of the ports for their SWC by allocating LCs to SWCs. The generated ECU extract is then used to create Simulink SWC models with appropriate ports; these models are finally filled with functional software. Engineers are usually working with SysTool and Simulink in parallel, updating model contents, requirements and signaling during the development. The code of the ECU is built on the code generated from each Simulink model combined with AUTOSAR code generated based on the ECU extract.

5.2 The Volvo Group Truck Technology

VGTT has similar complexity when it comes to modeling. One difference is that their architecture description is more light-weight today: they do not go as far as VC in modeling components.

For the design, VGTT uses SystemWeaver[3]. SystemWeaver is a holistic information management solution for system engineering and software development. SystemWeaver is composed of several different modules (e.g. Requirements Solutions, Test Solutions, Design Solutions, Safety Solutions, Traceability Analysis, Document Management, Code Integration, Simulink Integration, ISO 26262) that permit to realize customized solutions according to specific domains (e.g. automotive and embedded systems) and needs of specific customers.

[3]http://www.systemweaver.se

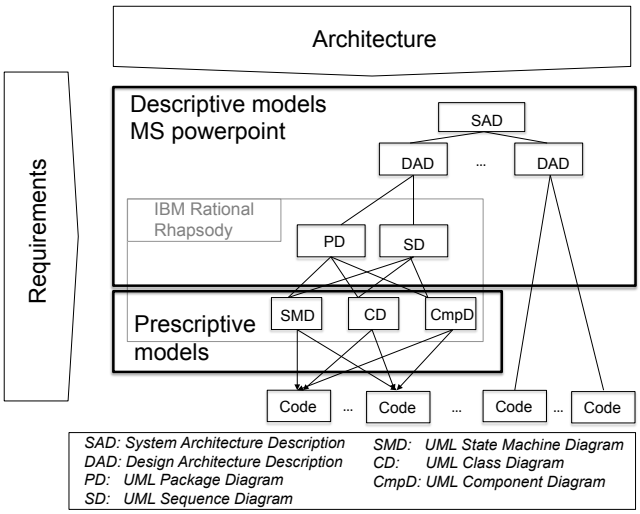

Figure 2: Main modeling artifacts within Ericsson

5.3 Ericsson

At Ericsson, there are even more levels of descriptive modeling compared to what we found at the Volvo companies. Figure 2 shows a simplified picture of modeling artifacts within Ericsson; it primarily shows architectural modeling while requirement specifications, also done partly as MDE, are excluded from the figure for readability purposes. The boxes represent prescriptive vs descriptive modeling as well as the tools used. Lines represent relationships and arrows represent a transformation from models to code.

The System Architecture Description (SAD) represents the architecture. It is produced in MS Powerpoint and contains approximately 80 pages. It uses various UML-like sketching models intertwined with text.

Each subsystem has one Design Architecture Description (DAD) that represents the architecture for that subsystem. It is also produced in MS Powerpoint and is typically around 50 pages. It contains much more details than the SAD and is balanced so that it does not contain too much information so that the overall architecture is not lost to the consumer.

The code-base at Ericsson is divided into two different domains: one written in C/C++ and the other modeled in a subset of UML using IBM Rational Rhapsody. The modeled domain holds another descriptive layer as part of the Rhapsody codebase: it consists of UML Package and Sequence diagrams. Even though such diagrams could be prescriptive, it was decided that implementing prescriptive modeling on that level would be too cumbersome. Only UML State Machine, class and component diagrams are used in a prescriptive fashion. The prescriptive models are then transformed into C++ code.

5.4 Lessons learned - RQ1

One language vs a constellation of languages - It emerges from the results a strong indication that no single modeling tool, or language, exists today, with the capacity to cover all aspects of software development. Modeling has to be considered more as a paradigm for software development which is widely accepted and integrated in the automotive domain. Modeling is then not only used for logical abstraction (of architectures) but also for abstracting systems in several other, physical and non-physical, domains. These, primarily descriptive, Domain Specific Languages (DSL) are increasingly popular. They are usually also closely connected to tools and frameworks for efficient simulation or rig utilization.

Need of a tool chain - While different and locally optimized DSLs will facilitate local development, the divergence of groups and languages will also obstruct the collaboration between teams. This problem has been observed at VC, while building up large scale (complete vehicle) integration models, involving software and plant models from several internal domains. The integration of these different types of models will enable large scale virtual integration and verification. This large scale virtual integration and verification is a rather new industrial initiative, primarily driven by the desire of having early verification and feedback thus reducing the amount of test on vehicles. Since many product variant combinations can be verified and validated using virtual integration, the total amount of test vehicles can be drastically reduced. Virtual integration is cheap and fast. It plays important roles in both the automotive companies and is expected to be used even more in the future (this topic will be further discussed in RQ2).

UML vs DSLs - At all places where descriptive architecture design models were used, general UML-tools or tools like PowerPoint, and Visio were considered good enough. However, when we come to the prescriptive models this is not the case. A much more detailed control of the model structure, behavior and transformations are then required. VC and VGTT use proprietary DSLs highly adapted to their domain and the problem at hand. Note that this also holds for the descriptive plant modeling DSLs mentioned above, while they are used for simulation, requiring mathematically correct behavior. In both VC and VGTT, more specialized and often context optimized languages are used. The tools behind the modeling languages also tend to get more advanced. On the contrary, Ericsson uses a subset of UML. The reason why specifically UML was chosen and not some other, more specialized DSL is, unfortunately, unknown.

We have not investigated why UML is not considered good enough for the design at VC or VGTT. We have only seen that both companies have chosen their own proprietary tool/language for dealing with this level. Since both companies know UML, this decision cannot be accidental.

6. RESULTS AND FINDINGS FOR RQ2

In this section, we present results and findings for **RQ2: To what purpose are prescriptive and descriptive models used within different contexts?**

Section 6.1 presents the use of models for the specification of requirements, Section 6.2 presents the use of models for architecture descriptions, Section 6.3 their use for the detailed architecture, Section 6.4 their use for simulation, and finally Section 6.5 lessons learned for RQ2.

6.1 Requirement models

Requirement engineers sometimes made sketches to understand the problem at hand better, but these types of models were not always stored for the future. And if they were included in a document that is stored, they were often considered as an addition to the text. One of the requirement engineers thought that there were too many ways to interpret a graphical model.

6.2 High-level architecture models

The purpose of the architecture description at all the companies was to show the overall structure of the product to be built. The architecture at VC went a lot further than at VGTT and Ericsson in specifying details. At VC and VGTT they discussed whether the architecture should be an abstraction of the current design or be a view of the future and what the system should look like then.

At Ericsson, they had solved this by incorporating the current but also the future architecture in the same document, as two different views. One explanation for that was that Ericsson used a single software branch for current and future releases, so one might expect that potential consumers of the architecture (i.e. new employees) would benefit from having this dual view since the software also holds the same dual view.

Furthermore, at Ericsson there was a conscious decision to reduce the level of detail in the architecture. The presumed consumer would be new employees whose primary purpose would be to get a coarse-grained understanding of the system and its architectural principles. It was not necessarily meant as reference material but more as a teaching aid. This was also the purpose for having it in Microsoft PowerPoint format so that it could easily be used as presentation material in courses for new employees. The same teaching-aid-view is identical for lower-level descriptive models as well.

6.3 Design models

At this level, i.e. the design level, it seems like the needs and constraints are unique for each of the companies. Hence, each company has a need for adapting the modeling tools, or to develop a new domain-specific modeling framework. This seems to be crucial for all of the studied companies. The design is all about how the product should be structured, and therefore, all the corrected interfaces are added. There is a strong need for tool support for the design since these models are both very large and complex. There are cases such as at Ericsson where sometimes, see Figure 2, the design is absent and emerging directly from the code. At VC, even the behavior of a component or system is done using modeling in some cases, e.g., Simulink.

6.4 Executable models - simulation

In this context, simulations via modeling is an important tool for finding early mistakes, especially in mechatronic systems [14]. Often, executable software models are tested initially in a virtual, model-in-the-loop (MIL) environment. In MIL-testing, the software models are executed within the modeling tool, on the developers PC. So-called plant models are used for simulating the connected physical system,

which can include both mechanics, environment and software in contact with the Electronic Control Units (ECUs). This enables the developer to get almost instant feedback by running and testing the models by always taking into consideration that assumptions are made in the plant and environment model or the ECU abstraction.

In the automotive industry, after the final software components are generated from models or written manually from the specification, they are transferred to the hardware and tested in hardware-in-the-loop (HIL) test rigs. HIL testing means that the code is executed on the intended ECU-hardware, communicating using real IO and network buses; however, the rest of the ECU environment is simulated. It is worth noticing that, automated code generation allows for a very short loop time also for HIL test, while having the appropriate tools and automation in place. Finally, the software will be tested and verified in physical vehicles.

The Ericsson case would seem to have a more traditional software oriented approach. Any code-base regardless of being prescriptive modeling or C/C++ had many levels of testing and verification; the same way-of-working was utilized across both domains: (i) Unit testing; (ii) Component testing; (iii) Integration testing; (iv) Functional verification; and (v) Non-functional verification.

(i)-(iii) are performed either natively or in a simulated environment, whilst (iv) and (v) are performed on target machines. For the modeled code-base the unit and component testing were realized on prescriptive models in the same tool and the language as the production code. There were also some trivial checkers of the model running as part of the tool itself, but it was not perceived as finding any crucial defects that might have been overlooked otherwise. So in principle it was mentioned that there was little difference between the modeled and the manual code with respect to testing.

6.5 Lessons learned - RQ2

Descriptive models: simplification and communication - Considering high-level descriptive models, it emerges that the primary purpose of these models is simplification and communication. In other words these models should simplify and reduce the large amount of information according to the intent and the consumer of the model. Moreover, from our study there emerges a tendency towards emphasizing flexibility in these models to meet the concerns and needs of the consumers of the models that might even evolve over time. A common concern regarding the level of detail was related to the lifetime and maintenance of the models. The main challenge is to keep the simplified high-level model up to date with the (changing) design. The number of levels of descriptive models is likely related to the size of the system. The reason why we consistently see multiple layers of descriptive models may simply be due to the fact that each node needs to be kept at a manageable size. Once again, having a manageable amount of information in each node seems to be crucial and this compartmentalization of information would seem to be consistent across all companies.

Make explicit design decisions and their rationale and communicate them - Considering the usage of descriptive architecture models during early development, some models are created by a tiny fraction of the employees but can nevertheless have a huge impact on the whole system. We have seen this to be particularly true when descriptive models are used to specify the architecture. These models are crucial in dividing the architecture into sub-domains, which can be further divided into smaller systems. With respect to the totality of the employees, only a very small subset of them is involved in architecture. Moreover, most employees know very little about these models since often they only see the effects on the design in areas where they work. Consequently, if the decisions are found to be wise by the development teams they will be used and further developed. On the other hand, if the development teams consider them inefficient or even obstructive, they will either choose not to follow the decisions or to follow the decisions with frustration and sub-optimal results arise as a consequence. We believe that the problem, in this case, is not only related to the modeling paradigms, but also and mainly due to a lack of communication and collaboration.

7. RESULTS AND FINDINGS FOR RQ3

This section presents results and findings for **RQ3: What is the relationship between different descriptive and prescriptive models and code?**

The relationship between models and code depends on whether the models are descriptive or prescriptive. Before discussing lessons learned about the use of model transformations for prescriptive and descriptive models (Section 7.3), Sections 7.1 and 7.2 describe information useful to understand the rationale about the use of model transformations. Specifically Sections 7.1 presents the main purpose of descriptive models, while Section 7.2 discusses about the importance of reducing assumptions in architectures.

7.1 Descriptive models: understanding, creative thinking and communication

From our interviews, we found that the purpose of descriptive models is to understand the domain. One of our interviewees at Ericsson said that a lot of the creative thinking is happening when using these models. At Ericsson, there was an outspoken goal of the descriptive models to be primarily teaching aids and not reference material. It was argued that descriptive models should not contain too much information to not risk overflowing the consumer with information. It would also have the added benefit of being easier to maintain due to the fact that the information is more coarse-grained and thus less likely to be outdated. The same reasoning was used when it came to not strictly adhering to any modeling language. They argued that the end goal of teaching new employees would be better satisfied by being flexible and being able to mix different models (for instance sequence diagrams intertwined with threading models).

7.2 Avoiding unnecessary assumptions

We found that there were cases when the descriptive models got too detailed; this is for instance the case of architecture descriptions capturing information which should have been captured in design descriptions. Unnecessary assumptions are sometimes made during the creation of the architecture. This implies that those assumptions should be removed or fixed during the creation of the design. Another possible consequence is that decisions made in the architecture will be ignored in the design. This way of working has two main negative consequences: (i) it makes the job of the

architect harder, i.e. to make decisions without having all the necessary knowledge; and (ii) decisions taken do not influence the final product but produce a lot of extra work and rework. In addition, the architecture starts becoming obsolete and then it needs to be updated by hand.

7.3 Lessons learned - RQ3

No need for model transformations from high-level descriptive models - The relationship among models depends very much on whether they are descriptive or prescriptive. An interesting finding concerning VC and VGTT is that none of the interviewees working with descriptive models, such as at the architectural level, wanted to have any automatic transformation from the architecture to any other types of models or code. However, having models on this level disconnected from the lower levels also has some drawbacks. Even though the architects in some cases have written scripts to help them, a lot of time and energy is spent to keep the architecture description updated with respect to what is happening in the construction groups. The scripts help somehow, but a lot of manual work is still required.

Having a tighter connection, maybe with automatic generation or transformation between the levels would force them to be consistent. However, that would also remove some of the benefits reported by our interviewees from the architecture group, such as having the freedom to try and experiment with solutions without affecting the levels below. On the other side, engineers working closer to the implementation consider that having a connection between architecture and design would make the architecture useful for them.

For some researchers there is still a dream to formalize the requirements as models and then transform to code. This is an interesting goal, but our research shows that we are far away from this goal today for complex domains. The purpose of descriptive and prescriptive models are so different that there is a large gap between them. It is not clear from our study that it will make sense to make descriptive models more formal. Model Driven Architecture (MDA) might not make sense for the type of system we have looked at.

Need for model transformations from prescriptive models - When it comes to the prescriptive level, i.e. design level, the situation is completely different. Here it is important to have good transformations. At VC, the compositions and software components can be used by the SysTool to generate the empty Simulink-models. The tool also generates AUTOSAR-XML that is used for integration purposes. Since such artifacts are generated from the tool, it is crucial that it is always up to date, otherwise implementation and integration is impossible. The modeling on this level is prescriptive at VC; the engineers have to follow what is in the tool. For the in-house development, the teams generate C-code from Simulink-models describing the desired behavior. VGTT has a very similar work flow.

At Ericsson, there seemed to be a distinct difference in the modeled code-base and the text code-base in the amount of modeling being used in the design. The modeled code-base has additional levels of descriptive models that do not exist in the text-based domain. One engineer hypothesized that it might be due to the fact that the modeled code-base was created more recently. Thus it has its own way-of-working as compared to the legacy text-based code-base. For instance, the modeled code-base was implemented using component-based architecture and thus had the possibility to have descriptive models surrounding each component. The legacy text code-base did not have such a clear distinction and this made it harder to describe subsystems in such a natural way. It was however noted that this was from a historical perspective and is not true anymore.

8. MODELS AND SOFTWARE DEVELOPMENT PROCESSES - DISCUSSION

Since there are established techniques to obtain code from prescriptive models, there is generally no conflict between agile development and MDE [14]. Moreover, MDE can help agile development of mechatronic systems by enabling early virtual integration in the cases where integration of software with physical objects is difficult, expensive, or even impossible because the mechanical systems are not yet fully developed or delivered.

Descriptive models, on the other hand, are often associated with "waterfall" development; also, separated architecture and design might be considered signs of non-agile development. As mentioned above, organizations and not the languages or models could be blamed for this. In fact, descriptive modeling tends to fail when the produced artefacts are used as means to define strict and prescriptive rules for the design. The maintenance of descriptive models becomes in these cases simply overwhelming, as it is hard to keep them up to date when design and implementation change. No definite solution to this problem has been identified. Due to this problem, some of the organizations at Ericsson have imposed limits on the size of the architecture models and of the associated documents. Hence, they considered these documents to be crucial, but only if they are simple enough to be understood and maintained.

A trend we have seen at both the Volvo companies is the use of Simulink and other executable models, both for descriptive and prescriptive purposes. The common feature is the ability to simulate through the use of models combinations of software and real-world objects; this is valuable in business realities that are tightly connected to mechanics and to the system environment.

Models of this kind are useful both to understand functionalities and to define requirements for the product. Hence, only a fraction of the models that are used during the development of mechatronic systems can be considered as software models. The remaining are executable descriptive models representing various parts of the product or the environment. When simulating complex mechatronic systems, only languages that are designed with an abstraction that is optimized for the particular domain are and remain useful in the long run. Often models are used also to store and improve the knowledge about the system over time. This trend is similar in most physical domains, as electrical, mechanical, or thermal simulation. In the automotive domain, the idea of scaling this virtual integration to the full vehicle, as observed at VC, is a natural consequence of the methods described above. However, it poses new challenges and it has not yet been fully integrated in the organization. This is an interesting topic for further research.

9. THREATS TO VALIDITY

Regarding threats to internal validity, we followed a systematic approach to setting up the study and best prac-

tice guidelines for both data collection and analysis [30, 31, 33]. A systematic approach was followed for the interviews, and informal interactions are an established way to gather data [9, 26]. A benefit of informal interaction is that it lets the respondent be in control, which, in turn, enables the discourse to lead to new insights not anticipated by the researchers [10].

In addition to these formal interviews, we exploited the knowledge of three industrial authors and of our strong collaboration. We have taken great care in our study to validate emerging findings from the interviews. The results were cross-checked for validation against a set of interviewees carried out in prior studies [18, 6, 13] and by exploiting the knowledge of the authors (three of them are from industry). For example, the third author (Ulf Eliasson) is employed at VC as an industrial PhD-candidate, hence he could observe meetings and had unrestricted access to systems and artifacts. At Ericsson and VGTT, we had separate meetings, with architects and technical experts to discuss various development artifacts. Throughout the study, the companies have been involved in validating our analysis: specifically, the validation has been performed through seminars with the respective contact persons and through recurring interactions with engineers in both formal and informal settings.

Concerning external validity, all three companies are large and based in Sweden. They develop large-scale software for embedded systems and the software they produce is expected to have a long lifetime. Therefore, our findings may not apply to MDE in smaller companies, other countries or for software with a shorter life expectancy.

Moreover, our study is based on 15 semi-structured interviews carried out at the three companies. As discussed in the introduction the number of interviews is indeed an important parameter; however, there should be a proper coverage of the variety of the heterogeneous population. In order to mitigate this intrinsic risk we exploited pre-studies to gain additional knowledge of the considered companies. We then exploited this knowledge to select the population so as to represent as much as possible the different realities that have been identified within each company. There is however always the risk that we have not completely covered the modeling landscape in these large companies.

10. RELATED WORK

There are a vast number of papers describing case studies of MDE in practice, as well as a smaller number of empirical studies on the use of MDE in industry. To limit the scope, we here describe only the most prominent of these papers, and especially those related to the automotive and telecommunication domains.

We already mentioned in the introduction the works in [23, 34] that give a positive signal on the adoption of modeling in industry and the works in [15, 29] that give a more negative signal. What is the reason behind such divergence of results? The reason might be related to the population involved in the studies. For instance, the study in [15] mostly involves developers/programmers that probably do not work every day with models, while the study in [23] mostly involves professionals working with model-based engineering.

For what concerns model transformations, the study in [6] testifies that model transformations can have a huge impact. Another finding of the work in [6] is related to the use of UML. Specifically, even though all three companies (the same companies that are considered in this paper) are multinational companies making complex software for embedded systems with thousands of people involved, i.e. the ideal place for UML, the authors found a limited use of UML. Now we understand that the study in [6] was focused exclusively on prescriptive models (the study has been performed by some of the authors of this paper).

The work in [18, 36] points out that MDE is not only about code generation. There are many other reasons to use MDE and the benefits can often be found in a better understanding of the system, a better articulation of the architecture, etc.

Kuhn et al. [21] and Aranda et al. [1] report on two parallel studies of applying MDE at General Motors (GM). The former focus on individual perceptions of the adoption of MDE at GM; the latter reports on how MDE induced changes at the organizational level. At the individual level, engineers experienced both forces and frictions related to MDE tools and languages - for example, a lack of support for developing secondary software. At the organizational level, the work in [1] finds, for instance, that software developers felt "downgraded" when MDE was introduced, because they were now asked to focus on secondary software whereas domain experts focused on primary functionality. Baker et al. [3] describes a process of introducing MDE at Motorola. They also report on the lack of organizational maturity regarding which processes to use in combination with MDE, the difficulties in adapting existing skills to the new challenges of MDE and the unwillingness of the organization to change to make the most out of the transition into MDE. In a study reported in [27], the emphasis is on MDE in relation to architectural concerns (deployment, algorithms, performance, etc.) but the authors point out some organizational aspects of MDE, such as the possibility that the cost of modeling can outweigh the benefits.

11. CONCLUSION AND FUTURE WORK

In this paper, we have shown the importance of distinguishing between descriptive and prescriptive models in order to understand how models are used in practice. In fact, we have shown in this paper that there are substantial differences between these two types of models, on their use, and on their importance during the development; this might also affect their maintenance and management.

From this study, we also learned that there is no evident need for model-to-model transformations from descriptive to prescriptive models. However, loose and flexible connections between descriptive and prescriptive models are desirable.

As future work, we plan to investigate how to establish flexible connections between descriptive and prescriptive models. The consistency checks within the MEGAF framework [17, 16] might be an interesting starting point. A complementary way is to generate descriptive models from prescriptive ones, thus creating specific views according to stakeholder' concerns. However, producing models readable for humans is rather hard, especially if the goal is to mimic high-level design models.

Testable hypothesis for further research: We conclude the paper with some testable hypothesis for further research that come out from our exploratory research:

- *Hypothesis 1*: No single modelling tool or language can exist with the capability to cover all aspects of software

development. Therefore, interoperability among tools and models is a key challenge for software intensive companies.

- *Hypothesis 2*: Model-driven engineering can be exploited for early validation and it can help having early feedback even during the system architecting. In this sense the use of models can be a key enabler for scaling agility in big and complex organisations.
- *Hypothesis 3*: Descriptive and prescriptive models should be connected through loose, lightweight, and flexible mechanisms.
- *Hypothesis 4*: Model transformations have a great value when considering prescriptive models; when dealing with descriptive models they are not so important and sometimes they are also undesired.

Acknowledgements

The work acknowledges support by the ASSUME-project Vinnova, the Vinnova-projects NGEA and NGEA step 2, and the Software Center initiative (software-center.se). Also, we would like to thank Ekdahl Anders from VGTT and Staffan Ehnebom from Ericsson for their precious comments.

12. REFERENCES

[1] J. Aranda, D. Damian, and A. Borici. Transition to Model-Driven Engineering - What Is Revolutionary, What Remains the Same? In *Proceedings of MODELS'12*, pages 692–708. Springer, 2012.

[2] M. Autili, P. Inverardi, and P. Pelliccione. Graphical scenarios for specifying temporal properties: an automated approach. *Automated Software Engineering*, 14(3):293–340, 2007.

[3] P. Baker, S. Loh, and F. Weil. Model-Driven Engineering in a Large Industrial Context – Motorola Case Study. In *Proceedings of MoDELS'05*, pages 476–491, Berlin, Heidelberg, 2005. Springer-Verlag.

[4] A. Bertolino, P. Inverardi, P. Pelliccione, and M. Tivoli. Automatic Synthesis of Behavior Protocols for Composable Web-services. In *Proceedings of the ESEC/FSE '09*, pages 141–150. ACM, 2009.

[5] D. Bozhinoski, D. Di Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli. FlyAQ: Enabling Non-Expert Users to Specify and Generate Missions of Autonomous Multicopters. In *Proceedings of ASE'15, IEEE/ACM*, 2015.

[6] H. Burden, R. Heldal, and J. Whittle. Comparing and Contrasting Model-driven Engineering at Three Large Companies. In *Proceedings of ESEM '14*, pages 14:1–14:10, New York, NY, USA, 2014. ACM.

[7] F. Ciccozzi, A. Cicchetti, T. Siljamäki, and J. Kavadiya. Automating Test Cases Generation: From xtUML System Models to QML Test Models. In *Proceedings of MOMPES '10*. ACM, 2010.

[8] P. Cuenot, P. Frey, R. Johansson, H. Lönn, Y. Papadopoulos, M.-O. Reiser, A. Sandberg, D. Servat, R. T. Kolagari, M. Törngren, and M. Weber. The EAST-ADL architecture description language for automotive embedded software. Proceedings of MBEERTS'07, pages 297–307, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] K. Davis. Methods for Studying Informal Communication. *Journal of Communication*, 28(1):112–116, 1978.

[10] K. DeWalt and B. DeWalt. *Participant Observation: A Guide for Fieldworkers*. Anthropology / Ethnography. Rowman & Littlefield Pub Incorporated, 2002.

[11] D. Di Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli. Automatic Generation of detailed Flight Plans from High-level Mission Descriptions. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2016.

[12] D. Di Ruscio and P. Pelliccione. Simulating upgrades of complex systems: The case of Free and Open Source Software. *Information and Software Technology*, 56(4):438 – 462, 2014.

[13] U. Eliasson and H. Burden. Extending Agile Practices in Automotive MDE. In *XM 2013 - Extreme Modeling Workshop*, page 11, Oct. 2013.

[14] U. Eliasson, R. Heldal, J. Lantz, and C. Berger. Agile Model-Driven Engineering in Mechatronic Systems - An Industrial Case Study. In *MODELS 2014*, volume 8767 of *LNCS*, pages 433–449. Springer, Oct. 2014.

[15] T. Gorschek, E. Tempero, and L. Angelis. On the use of software design models in software development practice: An empirical investigation. *Journal of Systems and Software*, 95(0):176 – 193, 2014.

[16] R. Hilliard, I. Malavolta, H. Muccini, and P. Pelliccione. Realizing Architecture Frameworks Through Megamodelling Techniques. In *Proceedings of ASE '10*, pages 305–308. ACM, 2010.

[17] R. Hilliard, I. Malavolta, H. Muccini, and P. Pelliccione. On the Composition and Reuse of Viewpoints across Architecture Frameworks. In *Proceedings of WICSA'12*, pages 131–140, 2012.

[18] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen. Empirical assessment of MDE in industry. In *Proceedings of ICSE '11*, pages 471–480, New York, NY, USA, 2011. ACM.

[19] M. Jackson. Some Notes on Models and Modelling. In A. Borgida, V. Chaudhri, P. Giorgini, and E. Yu, editors, *Conceptual Modeling: Foundations and Applications*, LNCS v.5600. Springer, 2009.

[20] J. Krizan, L. Ertl, M. Bradac, M. Jasansky, and A. Andreev. Automatic code generation from Matlab/Simulink for critical applications. In *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on*, pages 1–6, May 2014.

[21] A. Kuhn, G. C. Murphy, and C. A. Thompson. An exploratory study of forces and frictions affecting large-scale model-driven development. In *Proceedings of MODELS'12*, pages 352–367. Springer-Verlag, 2012.

[22] P. Ladkin. Abstraction and Modeling. Technical Report Research Report RVS-Occ-97-04, University of Bielefeld, 1997.

[23] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson. Assessing the State-of-Practice of Model-Based Engineering in the Embedded Systems Domain. In *Proceedings of MODELS'14, 2014.*, pages 166–182, 2014.

[24] D. Lorenzoli, L. Mariani, and M. Pezzè. Automatic Generation of Software Behavioral Models. In *Proceedings of ICSE '08*, pages 501–510, New York, NY, USA, 2008. ACM.

[25] J. S. Martin Rohdin. A PLM approach for development of automotive electronic systems. In *2nd Nordic Conf. on Product Lifecycle Management*, 2009.

[26] G. Michelson and V. S. Mouly. 'You Didn't Hear it From Us But...': Towards an Understanding of Rumour and Gossip in Organisations. *Australian Journal of Management*, 27(1 suppl):57–65, 2002.

[27] L. Pareto, P. Eriksson, and S. Ehnebom. Concern coverage in base station development: an empirical investigation. *Software and Systems Modeling*, 11(3):409–429, 2012.

[28] P. Pelliccione, P. Inverardi, and H. Muccini. CHARMY: A Framework for Designing and Verifying Architectural Specifications. *Software Engineering, IEEE Transactions on*, 35(3):325–346, May 2009.

[29] M. Petre. UML in Practice. In *Proceedings of ICSE '13*, pages 722–731. IEEE Press, 2013.

[30] C. Robson. *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*. Regional Surveys of the World Series. Blackwell Publishers, 2002.

[31] P. Runeson and M. Höst. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Softw. Engg.*, 14(2):131–164, Apr. 2009.

[32] R. Salay. Using Modeler Intent in Software Engineering, PhD thesis, Graduate Department of Computer Science, University of Toronto, Canada, 2010.

[33] C. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, 1999.

[34] A. Vallecillo. On the Industrial Adoption of Model Driven Engineering. Is your company ready for MDE? *Journal of Information Systems and Software Engineering for Big Companies (IJISEBC)*, 1(1):52 – 68, 2014.

[35] R. Wettel and M. Lanza. Visualizing Software Systems as Cities. In *Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on*, pages 92–99, June 2007.

[36] J. Whittle, M. Rouncefield, and J. Hutchinson. The State of Practice in Model-Driven Engineering. *IEEE Software*, 2013.