Journal of Systems Architecture 91 (2018) 62–82

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

A survey on modeling and model-driven engineering practices in the embedded software industry

Deniz Akdur[a], Vahid Garousi[b], Onur Demirörs[c,d,*]

[a] ASELSAN Inc., Ankara, Turkey
[b] Information Technology Group, Wageningen University, The Netherlands
[c] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
[d] Department of Computer Engineering, İzmir Institute of Technology, İzmir, Turkey

ARTICLE INFO

Keywords:
Embedded systems
Embedded software
Modeling
Model-based
Model-driven engineering (MDE)
Practitioner survey

ABSTRACT

Software-intensive embedded systems have become an essential aspect of our lives. To cope with its growing complexity, modeling and model-driven engineering (MDE) are widely used for analysis, design, implementation, and testing of these systems. Since a large variety of software modeling practices is used in the domain of embedded software, it is important to understand and characterize the-state-of-the-practices and also the benefits, challenges and consequences of using software modeling approaches in this domain. The goal of this study is to investigate those practices in the embedded software engineering projects by identifying to what degree, why and how software modeling and MDE are used. To achieve this objective, we designed and conducted an online survey. Opinions of 627 practicing embedded software engineers from 27 different countries are included in the survey. The survey results reveal important and interesting findings about the state of software modeling and MDE practices in the worldwide embedded software industry. Among the results: (1) Different modeling approaches (from informal sketches to formalized models) are widely used in the embedded software industry with different needs and all of the usages could be effective depending on the various modeling characteristics; (2) The majority of participants use UML; and the second most frequently selected response is "Sketch/No formal modeling language", which shows the wide-spread informal usage of modeling; (3) In model-driven approaches, it is not so important to have a graphical syntax to represent the model (as in UML) and depending on the type of target embedded industrial sector, modeling stakeholders prefer models, which can be represented in a format that is readable by a machine (as in DSL); (4) Sequence diagrams and state-machines are the two most popular diagram types; (5) Top motivations for adopting MDE are: cost savings, achieving shorter development time, reusability and quality improvement. The survey results will shed light on the state of software modeling and MDE practices and provide practical benefits to embedded software professionals (e.g., practitioners, researchers and also educators).

1. Introduction

It is difficult to imagine day-to-day life without embedded software systems [1]. They can be found in many devices such as cars, TVs, smart phones and also systems such as avionics or defense [2–5]. The growth rate in software-intensive embedded systems is more than 14% per annum and it is forecasted there will be over 40 billion devices world-wide by 2020 [6].

Analysis, design, implementation and testing of software-intensive embedded systems are not trivial due to multiple constraints across different dimensions of performance and quality [7–9]. Moreover, the increasing amount of components in these systems and having distinct functionalities incorporated into a single system, which require seamless integration of many hardware and software systems, make the embedded systems development more challenging [10,11].

In order to manage the complexity of these systems, software modeling helps engineers to work at higher levels of abstraction and facilitates communication [12–16]. However, the modeling approaches in embedded software industry usually vary since the characteristics of modeling such as purposes, motivations and challenges differ among systems as well as among sectors, e.g., consumer electronics, defense or automotive [17]. At one extreme, some modeling stakeholders (e.g., some project managers or systems engineers) use software modeling at a very informal level, where diagrams are sketched on a white board in order to help communicate ideas with colleagues. In such cases, the emphasis is

* Corresponding author at: Department of Computer Engineering, İzmir Institute of Technology, İzmir, Turkey.
E-mail addresses: denizakdur@aselsan.com.tr (D. Akdur), vahid.garousi@wur.nl (V. Garousi), onurdemirors@iyte.edu.tr (O. Demirörs).

https://doi.org/10.1016/j.sysarc.2018.09.007
Received 13 June 2018; Received in revised form 27 September 2018; Accepted 29 September 2018
Available online 2 October 2018

on selective communication and these diagrams might be either soon discarded or quickly become inaccurate since they are not kept updated along with the source code [12]. At the other extreme, for some other stakeholders (e.g., software developers), modeling turns into programming with automated generation of code and these diagrams have long lifespans and demand for archivability. Moreover, even in the same software project, different units within the same company can use different modeling approaches for different purposes [18]. Since a large variety of modeling practices is used in embedded software industry, it is important to identify different modeling approaches, in relation with challenges and benefits they provide.

There have been a few prior surveys related to modeling in the embedded software industry (e.g., [19–21]). They have either focused on only one aspect of modeling, (e.g., the use of Unified Modeling Language (UML) or the use of formal models), or modeling in regional contexts (e.g., UML and model-driven approaches in Brazil or in Greece). There are also some surveys, whose participants were involved with model-based/driven techniques on a single sub-domain of embedded systems (e.g., automotive [19]).

The goal of the practitioner survey reported in this paper is to understand the state-of-the-practice in modeling and model-driven engineering (MDE) practices in the embedded software industry by providing a view on the latest software modeling approaches, languages, tools used, and also the relevant challenges faced by practitioners. To achieve this goal, we designed and conducted a survey that is responded by 627 engineers from 27 countries working in different industrial sectors related to embedded software projects. The survey takes a holistic scope on the subject and covers a wide range of modeling aspects in embedded software industry. We focused on the modeling practices of the embedded software industry for two reasons: (1) given the specific characteristics of embedded software, modeling practices are usually tailored for these systems, e.g., there are specific UML extensions (profiles) such as MARTE [22] and various Domain-Specific Languages (DSLs) for this sector; (2) in the context of an ongoing industry-academia collaborative project of a major embedded software firm in Turkey, the need has raised to critically assess the global state of the modeling in the embedded software industry so that proper decisions can be made with respect to adopting the right modeling practices and modeling approaches.

We believe that the results will benefit both embedded systems professionals as well as researchers, by creating an awareness for the trends, successes and challenges of practitioners. We also believe that the survey results would provide practical benefits to all stakeholders by influencing not only the aspects related to software-intensive embedded systems development, but also the system-level design and methods for hardware/software co-design.

The remainder of this paper is structured as follows. Section 2 discusses background and the related work. Section 3 presents the research methodology used to perform the survey. Section 4 presents the results. Section 5 summarizes the results and implications, and reviews the potential validity threats. Finally, Section 6 concludes this study and discusses the future work directions.

2. Background and related work

In this section, we first present a brief overview of the concepts of Model-Based Engineering (MBE), MDE and Model-Driven Development (MDD). Related work in relation with the surveys on modeling for embedded software is reviewed next.

2.1. MBE versus MDE and MDD

In the literature, there are different terminologies in the context of software modeling. While designing the survey, we followed the terminology offered by Brambilla et al. [23] for describing and differentiating between "model-based" and "model-driven" approaches. According to Brambilla et al. [23], MDD treats models as *"the primary artifact of the development process"*. Usually, in MDD, there is an automatic code generation from the models. In addition to just development, MDE encompasses all the other tasks of the software engineering (SE) process such as testing and maintenance, and thus, MDE is considered a superset of MDD. On the other hand, MBE is a process, in which diagrams (either formal models or informal sketches) still play an important role although they are *"not necessarily the key artifacts of the development"*. As in the case of our industrial experience, we agree with and followed the idea that MBE does not "*drive*" the process as in MDE. For example, software designers specify the diagrams (e.g., on paper or by using modeling tools), but then these diagrams are directly handed out to the software developers to manually write the code (i.e., no automated code generation). Therefore, all model-driven processes are model-based but not the other way round. The Venn diagram shown in Fig. 1 (adopted from [24]) visually depicts these concepts.

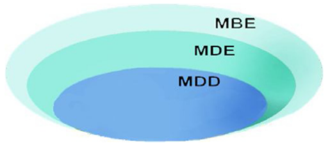


Fig. 1. Venn diagram depicting the relationship among MBE, MDE and MDD.

Note that the terminology offered by Brambilla et al. [23] focuses on "prescriptive modeling", but in the literature, there is also "descriptive modeling" terminology, in which sketching plays an important role while modeling (e.g., [18, 25, 26]). However, while designing our survey, we counted "informal sketch" as a part of MBE since these diagrams do not "drive" the development process and they have less lifespan and archivability than the ones used in MDE [26].

According to various sources, (e.g., [23,27,28]), MDE is considered as one of the most popular approaches in software abstraction. In the context of embedded domain, by abstracting out details, MDE helps software engineers manage the complexities in embedded software development [10] by automating Software Development Life Cycle (SDLC) artifacts not only in implementation [29] but also in testing and documentation. There are many books, e.g., [30–32], many conferences and a large body of knowledge in the application of MDE. Furthermore, since economic factors such as time-to-market require a reliable development process allowing quick SDLC [33], many practitioners in different domains (e.g., consumer electronics, defense and aerospace, automotive, and telecommunication) have started to adopt MDE [34–37]. More specifically, several studies point out the necessity of MDE in the embedded world to minimize the effects of platform heterogeneity and its complexity [38] besides validation and verification [39].

2.2. Related work

We were able to find three survey studies [19–21,40], which have investigated the-state-of-the-practice of model-driven techniques via opinion surveys. Some of the surveys have focused on the embedded domain, while others are generic in terms of the domain. Table 1 summarizes those three surveys, which have been conducted in this topic. The respondents of these surveys were basically from a specific embedded domain (i.e., automotive) or in regional levels (i.e., Brazil) or people who have already worked with model-based/driven techniques. Apart from these embedded-related surveys, there are also several studies, which investigate mainly UML-based modeling [27,41–49], which we also briefly review in Table 1.

The study in [19] was a 2011 world-wide survey of 67 participants which investigated the reasons of introducing model-based development in a single sub-domain of embedded systems (i.e., the automotive industry) taking into account its costs and benefits. It focused only on "development" phase (e.g., model-based development (MBD)) of the entire "engineering" (MBE) process. The main findings from this study were:

Table 1
Existing surveys explicitly on MDE.

Citation	Year	Scale/ region	Number of subjects	Goal/focus area	MBD/MBE/ MDD/MDE	Domain
[19]	2011	World-wide	67	Investigated the reasons of introducing model-based development in a single sub-domain of embedded systems (i.e., the automotive industry) with its costs and benefits. Focused on only "development" phase (MBD) of the entire "engineering" (MBE) process.	MBD	Embedded systems
[20]	2013	Brazil	209	Investigated the use of UML and model-driven approaches in the embedded software development industry	MDD	Embedded systems
[21]	2014	Europe	112	Investigated the positive & negative effects of MBE. It did not address categorization between model-based and model-driven techniques. Same authors presented another study [40] in 2018 in which they analyzed the results in more depth.	MBE (MDE)	Embedded systems
This study	2015	World-wide	627	Investigates the degree to which, why and how software modeling and its benefits, challenges, and consequences.	MDE	Embedded systems
[41]	2005	World-wide	131	Investigated the adoption and usage of UML by analyzing its perceptiveness and perceived ease of use.	MBD	In general
[42]	2006	No information given	182	Investigated how and why using UML.	MBD	In general
[43]	2006	No information given	80	Investigated UML usage and its quality in actual projects.	MDD *(but only with UML)*	In general
[44]	2006	Bulgaria	100+	Investigated the utilization of UML	MDE *(but only with UML)*	In general
[45]	2008	Europe	80	Investigated the impact of UML modeling styles.	MDD *(but only with UML)*	In general
[46]	2014	Greece	91	Investigated the role of UML.	MDD *(but only with UML)*	In general
[47]	2008	World-wide	113	Investigated software modeling experiences.	MDE	In general
[48]	2011	World-wide	250	Investigated the adoption and application of model-driven software development in industry. Same authors presented another study [27] by identifying the importance of complex organizational, managerial and social factors, as opposed to only technical factors, that appear to influence the relative success, or failure of MDD.	MDD	In general
[49]	2011	Italy	155	Investigates the modeling languages, processes and tools with MDE. Same authors presented another study [50] in 2013 in which they analyzed the results in more depth.	MDE	In general
[51]	2014	World-wide	3785	Investigates the use of software design models in software development	MDD/MDE	In general

(1) The top three motivations of model-based development are: “improvement of the product quality”, “development of functions with high complexity”, and “shorter development times”; (2) Positive experiences of MBD are "communication with other colleagues", "possibility of early simulation of the functional model", "easier maintenance if the generated code is not changed manually"; whereas the negative experiences are "high process of redesign costs" and "tool costs"; and (3) MBD can bring significant cost savings, but only with a “well-chosen” approach and an established development process with defined interfaces and role allocations. Otherwise, MBD can be much more expensive than a hand-coded manual software development.

The study in [20] investigated the use of UML and model-driven approaches in the Brazilian embedded software development industry. According to the results: (1) 45% of the participants use UML either completely or partially; (2) The subjects report increases in productivity and improvements in quality (maintainability and portability) as key advantages of model-driven techniques; (3) Models are mainly used for documentation and design with only little of code generation; (4) Class, sequence, use case, and state machine diagrams are the most popularly used diagram types. One of the interesting results is that experienced users (i.e., the ones with more than 10-year experience) can better assess the benefits of UML for the development of embedded software. On the other hand, the major problems encountered in the adoption of UML refer to the lack of modeling skills, the lack of appropriate tools, and the strict time requirements.

The study in [21] was a 2014 European survey that investigated the current state of MBE in embedded domain by analyzing its positive & negative effects and its shortcomings. Its target projects were applying model-based and model-driven approaches, where its participants had already used model-driven techniques (93%), therefore, it lacks of general embedded software professionals contribution (Note that according to their terminology, there is no model-driven but MBE includes model-driven techniques too). The results confirmed that MBE is widespread in the embedded domain. The main finding from this study was that models are clearly not only used for informative and documentation purposes; they are key artifacts of the development processes, and they are used for, e.g., simulation and code generation. Moreover, while survey respondents reported mostly positive effects of MBE, the results showed some common and major challenges (i.e., adoption, tool support and its interoperation). The same group of authors presented another study [40] in 2018 in which they analyzed the results of [21] in more depth, and offered insights into the current industrial practice.

The survey in [41] was a 2005 world-wide survey of 131 participants, which investigated the adoption and usage of UML by analyzing its perceptiveness and perceived ease of use. The results showed: (1) The majority of respondents viewed UML as accurate, consistent, and flexible enough to use on development projects; (2) Developers clearly seemed eager to use UML, which was spreading across the world; and (3) Use case, class, and sequence diagrams are the most popularly used diagrams types.

In [42], how and why using UML were investigated. According to their results, UML may be too complex supported by phrases such as "*Not well understood by analysts*" or "*insufficient value to justify the cost*". Respondents of [42] reported that class, use case, and sequence diagrams were the most popularly used diagrams; whereas collaboration diagrams were used the least. The other interesting result was that class, sequence and state machine diagrams were considered as the most useful for capturing technical aspects; whereas use-case narratives, activity and use case diagrams were the preferred means with regard to customer involvement.

The study in [43] investigated UML usage and its quality. The results addressed UML's problems, where the main problems were synthesized as: "scattered information", "incompleteness", "disproportion" and "inconsistency". The results in [43] showed that UML practices should be improved in some areas such as modeling uniformity and standards, development of project-specific reference architectures and patterns.

The survey in [45] was a 2008 European survey of 80 participants, which investigated the impact of UML modeling styles. The findings focused only on aspects related to the improvement in software development quality and productivity. One of the results revealed that the impact of using UML on productivity was perceived mostly in the design, analysis, and implementation phases.

On the other hand, there were also some national surveys on UML. The results of survey in [44], which investigated the utilization of UML in Bulgarian companies, showed that in most cases UML was not properly used in the industry and more training was needed. A Greek survey [46] with 91 participants, which mentioned "model-driven" concept but only with UML, investigated the role of UML in all different types of applications (e.g., web, windows, or embedded). The findings indicated that UML was used successfully in the majority of software development. Among the results: (1) The most popular diagrams were class, use cases and activity, whereas the least used diagrams were package and state machines; (2) Even though UML was extensively used, its extensions (such as SysML) were not well known and a large percentage of the user group was not familiar, whereas others rarely or never use. The main conclusion was that despite the limitations and extensions needed, UML is the only general-purpose modeling language that is an industry standard for specifying software-intensive systems, that is supported by numerous tool vendors [46].

There are also surveys on MDE in general [27,47–49], which do not explicitly address embedded software industry as their target. The study in [47] was a 2008 survey with two thirds of the respondents from Canada and the United States, which investigated software modeling experiences. According to its results, UML was identified as the dominant notation. Participants reported that the biggest perceived problem of model-centric approaches is keeping the model up-to-date with the code. Moreover, another interesting result is that participants working on real-time systems are more likely to agree that their organizational culture does not like modeling.

The study in [48] was a 2011 survey of 250 participants which investigated the adoption and application of model-driven software development. According to the results: (1) MDE represented a need for new skills, including UML modeling expertise (in which significant additional training is needed); (2) Code generation was an important aspect of MDE gains, but integrating the code into existing projects could be problematic; and (3) Class, activity and use case were the most popular diagrams. The same authors presented another study [27] by identifying the importance of complex organizational, managerial and social factors, as opposed to only technical factors, that appear to influence the relative success, or failure, of MDD.

Another study [49] was a 2011 Italian survey which investigated the modeling languages, processes and tools in the Italian software industry with MDE. According to its results: (1) 68% of participants reported to always or sometimes use models, and among them, 44% reported generating codes from models; (2) The subjects who do not use models commonly stated that modeling requires too much effort and time investment (50%) or was not useful enough (46%); and (3) Models were used mainly in larger companies and that a majority of all the subjects using models (76%) apply UML although DSLs are used as well. The same authors presented another study [50] in 2013 in which they analyzed the results in more depth.

The study in [51] was a 2014 survey, which investigated the use of design models in software development. The results of this study showed that design models are not used very extensively in industry (almost ~50% of participants never use them), and where they are used, the use is informal and without tool support, and the notation is often not UML. According to results, these models are used primarily as a communication and collaboration mechanism where there is a need to solve problems and/or get a joint understanding of the overall design in a group.

Our work builds on these studies and significant extensions: our study is not limited to neither a sub-domain of the embedded systems (e.g., automotive), nor a subset of SE phase (e.g., development), nor just a specific region (e.g., Brazil). In this perspective, our survey is the first world-wide survey, which focuses on embedded software industry by investigating a wide range of modeling practices.

3. Research methodology

Survey methodology is a well-established technique for obtaining broad characterization of a particular issue by enabling collection of different information such as opinions, perceptions, attitudes and behaviors [52]. It has been applied in various fields. Surveying is a well-fitted strategy as it is suitable for collecting empirical data from large populations.

There are different surveying methods, each with different advantages and disadvantages [53]. In this study, we chose to use the online survey method since we wanted to obtain information from a relatively large number of practitioners in a quick manner so that we can easily categorize and analyze these data. The other conventional approach in the SE is to conduct interviews with subjects, which is usually more effort intensive. Compared to the latter, the former (the opinion surveys approach) may have drawbacks since there is no interviewer, ambiguous and poorly-worded questions might be problematic [52]. In order to cope with this challenge, a pilot study was applied before the execution of the survey.

Moreover, even though it is relatively easy for software engineers to fill out questionnaires, they still must do so on their own and may not find the time [52]. In that sense, the organization of survey questions is crucial and requires special considerations [54]. Accordingly, we have designed the question in order to reduce the time taken to complete the survey.

3.1. Goal and research questions

The research approach used in our survey study is the Goal, Question, Metric (GQM) [55]. By using its template [55], the goal is to understand the state-of-the-practice of modeling and MDE in the embedded software domain by identifying to what degree, why and how they are conducted with its benefits and challenges. Based on this goal, we raised the following research questions (RQs):

- **RQ1**: What is the current state of modeling in the embedded software industry?
- **RQ2**: What is the current state of MDE adoption in the embedded software industry?
- **RQ3**: What are the benefits, challenges, and consequences of using MDE in the embedded software industry?

3.2. Survey design and execution

In designing of the survey, we made sure that the questions are relevant to the embedded software industry and capture the most useful information as relevant to the goal and RQ's of the survey. In designing this survey, we utilized and benefitted from several survey guidelines (e.g., [54,56,57]), and also our previous experience in designing and executing industrial survey studies (e.g., [58]).

3.2.1. Identifying target audience

The identified target audience is anyone working in the embedded software industry, with a variety of different SE roles from requirement engineer to business analyst and from software developer to tester. This study established a sampling frame composed by a large set of embedded software professionals working in different locations around the world and in different industrial sectors.

3.2.2. Sampling method

In our study, given our limited resource constraints, it was not practically doable in the outset of our project to recruit a large pool of embedded software practitioners. As in the survey guidelines (e.g., [54,56,57]), we thus used the 'accidental non-probabilistic' sampling [54] and we targeted subjects via our industry contacts, professional social network sites such as LinkedIn, industry events, and forums. The survey was also promoted through SE and academic institutional mailing lists. Besides, we also encouraged recipients to distribute the survey to their colleagues and partners. After receiving the non-probabilistic sampled data, one could possibly perform a-posteriori probability-based (systematic) sampling, e.g., by grouping the data for various companies and then selecting the filtered data so that every member of the population has statistically seen an equal chance of being selected, in a way to mimic probability sampling. However, this was also infeasible in our setting since data in our survey were fully anonymous, since we did not want to gather company names nor any revealing information. Anonymity of data was important since revealing information could have damaged the quality of the data reported by participants since they would have hesitate to report honest opinions (such situations have been observed before, e.g., [53]).

Another issue in our survey design, inter-related with the sampling method, is the 'unit' of interest [53]. The units of analysis in this survey might be anyone working in the embedded software domain, who individually and anonymously participated in our survey. Thus, for all the statistics and analysis, these professionals are the unit of analysis and the implications shall be tied to world-wide community under investigation and neither to companies nor projects. We also might need to emphasize that taking individual embedded professional as the unit of analysis has been considered a generally acceptable approach in previous surveys reported in the literature (e.g., [59]).

3.2.3. Designing survey questions

Surveys require special considerations [54]. In order to develop a survey that would adequately cover the latest trends on modeling, we reviewed the similar past surveys (See Section 2.2), benefitted from our professional experiences in industrial projects (for the case of all three authors), considered factors given in survey guidelines [54], and prepared a draft set of questions. We conducted a round of peer reviews with nine industrial practitioners from different industries, different software engineering roles, different experiences and five different companies, in which our personal contacts have been working. All peer reviews were conducted face to face and according to their results, we improved four questions (i.e., Q20, Q25, Q26, and Q27). The final survey questionnaire consisted of four sections: the first section gathers the profiles of the participants and their companies; the other sections correspond to each of the study RQs, as shown in Table 2 *(For each question, the type of answers are also mentioned,* e.g., *single answer from a list, or a Likert scale)*. Due to space constraints, we do not present the entire survey in this paper, but it can be found in an online source [60].

The introduction of the survey is written to attract respondents' attention. Therefore, the survey began with an informed consent, which contained the topic of the study, a confidentiality statement, the expected time to complete the survey and a thank you statement (See [60]) so that the majority of potential respondents will decide whether or not to drop out of the questionnaire based solely on the first page. By clicking through the consent statement and submitting the completed survey, individuals are indicating their willingness to participate.

It is very important to have clear definitions and easy-to-follow instructions in the survey to get high quality data [54]. The first part of the questionnaire gathered personal and organizational demographic data. The 10th question investigated how often any informal or formal software modeling (i.e., sketches and/or models) is used in SDLC by asking "*How often do you use software modeling in your software development life cycle? (informal or formal:* i.e.*, sketches or models)*". Since any informal usage of modeling was seen as "modeling usage" in this survey, the aim of this question was to understand the ratio of participants, who did not use any software modeling. After categorizing this group and made them complete the survey, the questionnaire continued with modeling approaches questions, which aimed at understanding informal usage of modeling, model-based and model-driven techniques. In other words, this second part aimed at gathering the current state of software modeling. At the beginning of 19th question, we gave the terminology, which clearly explained the difference between model-based and model-driven concepts as in Section 2.1 (See [60]) so that participants could consistently answer subsequent questions:

> *"Please read the following definitions before proceeding with the rest of the survey.*
>
> *In terms of terminology, Model Driven Development (MDD) uses models as the primary artifact of the development process. Usually, in MDD, the implementation is automatically generated from the models. Model Driven Engineering (MDE) is a superset of MDD since it encompasses other tasks of a complete software engineering process like testing and maintenance (*e.g., *documentation). On the other hand, Model Based Engineering (MBE) is a process, in which software models still play an important role although they are not necessarily the key artifacts of the development. For example, designers specify the models (*i.e.*, by using paper or modeling tool), but then these models are directly handed out to the programmers to manually write the code (no auto generation)."*

With the help of this terminology and given example, we assume that respondents, at least, can understand the concept of "*the automatic generation of an artifact*", e.g., code, or document. Then, the survey asked about the degree of model-driven techniques in SDLC. In order to prevent any misunderstanding and potential threat in this terminology, pilot study was applied. After the pilot study, instead of asking *"Do you use any model-driven techniques?"*, we modified this question into "*When you write code, document or test, to what degree do you use model driven techniques?"* by assuming that the respondent can answer whether there is an automatic generation of some artifact or not. At that point, the survey was completed for the respondents, who chose "*Never*" in the Likert scale (which means that informal usage of modeling (e.g., sketching) and/or model-based approach). Then, in the remaining parts, MDE specific questions, which were interested to know about MDE practices, benefits and, challenges, started for the respondents, whose answers were different from "*Never*" (e.g., "*Sometimes*" to "*Always*").

3.2.4. Survey piloting and execution

Performing a pilot study before distribution is an important step since it would help preventing misinterpretations in large-scale data collection of the survey. Pilot studies are carried out by using the same material and procedures but with a smaller number of participants from the target population [54]. Before the pilot study, it was necessary to decide whom to use as participants. It is recommended to select participants based on differences instead of trying to replicate similarities [61]. Therefore, the survey was firstly piloted by eight colleagues from different industries working in different SE roles, with different experiences

Table 2
List of the questions developed and used in the survey (details of the responses can be found in [60]).

RQ & Aspect		Survey questions (and metrics)	Type of answers				
			Single answer from a list	Multiple answers from a list	Free text field	Likert scale	Likert scale (Range value from never to always)
Profiles	Practitioners	**Q1.** Please choose the country that you work in.	x		x		
		Q2. What is your highest academic degree?	x				
		Q3. What is (are) your university degree(s) in?		x	x		
		Q4. What is (are) your current position(s)?		x	x		
		Q5. How many years of work experience do you have in software development?	x				
		Q11. How many years of modeling experience do you have in software development?	x				
		Q12. Where/how did you learn modeling?		x	x		
	Companies	**Q6.** What is the type of the application(s) developed in your company?		x	x		
		Q7. What is the target sector of the product(s) developed?		x	x		
		Q8. What is the number of employees working in software engineering roles?	x				
		Q9. What is the size of your typical software development team?	x				
RQ1 Current state of modeling		**Q10.** How often do you use software modeling in your software development life cycle? (informal or formal: i.e., sketches or models)					*x*
		Q13. What medium do you use to create the sketch or model?		x			*x*
		Q14. Which modeling language(s) do you use for modeling?		x	x		
		Q15. Which programming languages do you use with the above modeling language(s)?		x	x		
		Q16. Which modeling environment/tool(s) do you use, if any?		x	x		
		Q17. When modeling, which diagrams do you use?		x			*x*
		Q18. In which phase(s) of software development life cycle do you use modeling?		x			
RQ2 Current state of MDE and its adoption		**Q19.** When you write code, document or test, to what degree do you use model driven techniques?					*x*
		Q20. What do you use MDE for?		x	x		
		Q21. What is the estimated effort (in person-month) of the most representative MDE project in your company?	x				
		Q22. How would you describe your company's maturity in terms of its MDE usage?	x				
		Q23. What have been the motivations (potential benefits) that your company has considered for adopting MDE?		x		x	
RQ3 Benefits, challenges and consequences		**Q24.** Based on your experience, to what degree has each of the above motivations (potential benefits) been achieved?		x		x	
		Q25. What is (are) MDE challenge(s) in your company?		x	x		
		Q26. To what extent do the following problems apply to the MDE environment/tool(s) that you have used?		x		x	
		Q27. Based on your experience, what do you think about the following statements?		x		x	

and from different nations (four Turkish, two English, one French and one Taiwanese). This was done to ensure that the wording and terminology used in the survey is easily understandable and well-formulated to get high quality data. In order to prevent misunderstandings, which could lead to invalidity of conclusions, great importance was given to clarifying survey questions and explanations. Given their feedback and the time they needed to fill out the survey, the questionnaire was refined by modifying three questions (i.e., Q10, Q19 and Q23), the terminology given at the beginning of 19th question (See [60] for more details), and also five pre-given answers set (i.e., Q14, Q23, Q25, Q26 and Q27). The revised survey was reviewed a second time by five other colleagues and two colleagues, who were participated in the first pilot study. Therefore, the final version of this survey was reviewed by 13 professionals. After the revisions, the final version of the questionnaire consisted of 27 questions, in the form of multiple-choice (checkboxes), single-choice (radio buttons) and Likert-scale answers. Where applicable, free-text areas for additional input were provided to respondents as "Other".

To design and execute the survey, we used the Google Forms tool. The ethics approval for the survey was issued by the Human Subjects Ethics Committee of Middle East Technical University (METU) in March 2015. The survey was then executed in the period of April–May 2015. The hyperlink of the survey has been distributed to embedded software professionals via social networks as well as to our network of embedded software professionals working in all around the world.

Table 3
Plan for cross comparison of our findings with the previous surveys on modeling in embedded systems.

Study reference	Aspects to be compared in Section 4	Question # in our survey	Types of comparison
[19]	Motivations of model-based development	Q23	Quantitative
	Benefits of model-based development	Q24	Qualitative
	Positive/negative experiences of model-based development (Reasons/challenges of model-based development)	Q20-Q25	Qualitative
[20]	Modeling languages	Q14	Quantitative
	Diagram types	Q17	Quantitative
	Benefits of model-driven development	Q24	Qualitative
	Major problems in model-driven development	Q25	Qualitative
[21]	Modeling languages	Q14	Quantitative
	Modeling environments	Q16	Quantitative
	Types of notations (diagram types)	Q17	Quantitative
	Development phases where MBE is used	Q18	Quantitative
	Motivations of model-based development	Q23	Quantitative
	Benefits of model-based development	Q24	Qualitative
	Major problems in model-based development	Q25	Qualitative

3.3. Pre-analysis considerations and data validation

The last step of the survey process was to analyze the collected data. Although the title of the survey, the protocol part of the survey, the invitations and forums entries are emphasizing on “embedded”, some participants chose just "Desktop applications" or "Web applications" for Q6. The answers, which do not include any “*Embedded applications*”, were considered out of scope of this survey. Some companies develop different kinds of applications (e.g., both embedded and desktop); therefore any answer, which consisted of “*Embedded*”, was included in the sample. Aside from that requirement, there were no other criteria for inclusion or exclusion. By applying this criterion, 15 survey data were excluded. After the data validation phase, we had 627 acceptable responses from 27 different countries. To increase transparency, the raw survey data is made available online [62] for other researchers to validate and replicate. Considering that no incentive was offered to the participants, it is interesting to see that the number of participants is quite high in comparison to previous surveys (Section 2.2).

3.4. Plan for cross comparison with previous surveys

One of the important analyses that we conducted and report in Section 4 is cross comparison of our findings with previous surveys on MDE in embedded systems [19–21]. To plan the cross comparisons, we itemized the types of findings reported by each of those studies and paired them (if any) with a question in our survey. Table 3 presents an overview of our plan for the cross comparisons. For example, we will compare the benefits of model-based development as reported in [19] with results of Q24 in our survey. Based on the types of available data, some of the comparisons are quantitative or qualitative. Note that for easier traceability and understanding, we will present the results of these comparisons and the interpretations of possible reasons in the question itself (e.g., in a single section), instead of splitting their discussions in a separate sub-section (e.g., moving into discussion part).

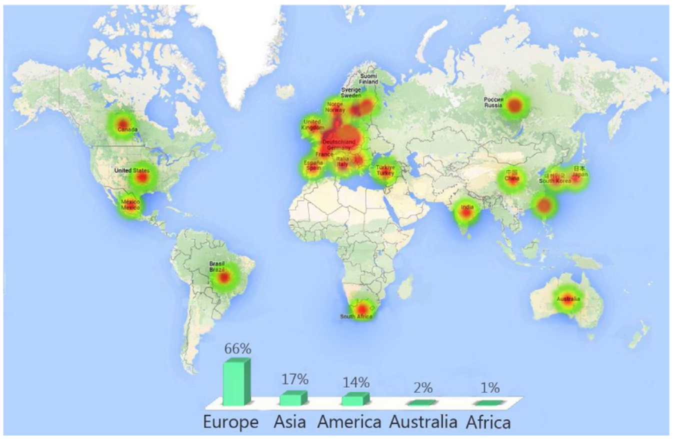


Fig. 2. Countries and geographical distribution of respondents.

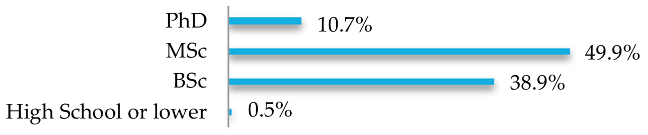


Fig. 3. Highest academic degrees.

4. Survey results

In this paper, due to space constraints, we report a subset of the survey results. All other remaining results in the survey are accessible from [63].

4.1. Demographic of participants and their companies

The first question asked respondent about their geographical location (Q1). Our goal was to reach out to as many countries as possible and to ensure that all regions where there is a presence of embedded software industry are reasonably well represented in the dataset. The final dataset had respondents from 27 different countries distributed in all the continents. Fig. 2 shows the world heat-map, and also the distribution of responses by continents, showing that most of the responses originating from Europe (66%), followed by Asia (17%) and America (14%). Of course these data do not provide any information in relation with relative sizes of the embedded software industry in different continents. Note that due to researchers’ location (i.e., Turkey), the ratio of European respondents is higher than others.

Participants were asked to provide their highest academic degrees (Q2). The result reveals that 50% and 11% of respondents have a Master’s and Ph.D.’s degree respectively. Only 3 respondents (0.5%) reporting to have High School or lower degree, denoting that the embedded software is demanding in terms of background knowledge. Fig. 3 shows that our dataset includes more Ph.D. and MSc holders than our expectation, perhaps denoting that the modeling in embedded software might be demanding more combination of academic disciplines to understand various part of the system (e.g., both hardware and software) easier (e.g., a participant, whose BSc is in Electrical/Electronics Engineering and MSc is in SE).

In order to understand the respondents’ educational skill-set, participants were then asked to provide their university degrees (Q3). The results of this multiple-response question are shown in Fig. 4. Note that the department name of computing discipline degrees might be different (e.g., depending on the university of the participant); hence it is better to analyze the underlying discipline in a single item as “Computing Disciplines” (e.g., computer engineering, computer science, software engineering, information systems) since their “software modeling” cur-

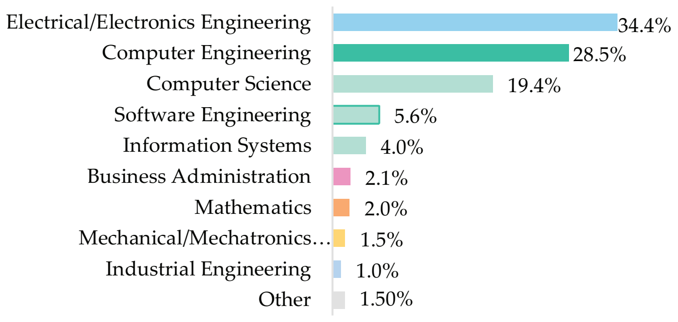


Fig. 4. University degrees.

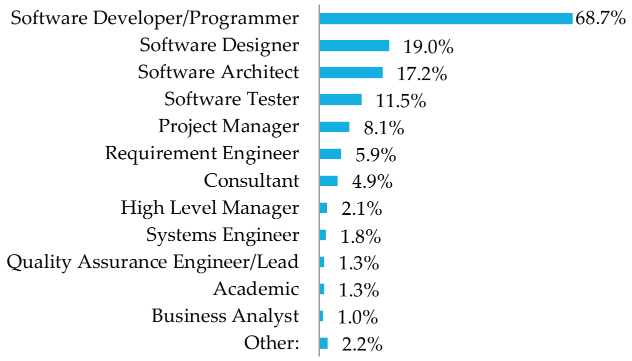


Fig. 5. Current positions.

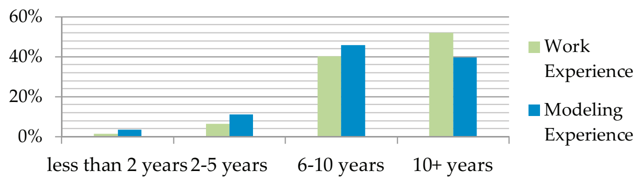


Fig. 6. Work vs. modeling experience of participants who use any software modeling.

riculum might be similar. Then we can say that Computing Disciplines and Electrical/Electronics Engineering are the top university degrees in the survey. Please refer [63] for the details of other university degrees.

The current positions of respondents (Q4) are shown in Fig. 5 (Note that multiple roles could be recorded in this question, e.g., a person can be a software developer/programmer and software designer at the same time). Most of the participants have “Software Developer/Programmer” role. “Software Designer”, “Software Architect” and “Software Tester” roles are the other majority roles in the survey.

When work experience of the participants in software development was asked (Q5), it is seen that the majority of respondents have 10+ years (52%) and 6–10 years (40%) work experience. 41 participants (6%) reported to have 2–5 years of experience; whereas only 10 participants (2%) have less than 2 years of experience. This indicates that our participants are generally experienced industry professionals in embedded systems (assuming that their work experience is on embedded systems). We also asked the participants to report their modeling experience (Q11) in software development (Fig. 6). The interesting point here is that, although the majority of survey respondents have 10+ years (52%), which is followed by 6–10 years (40%) of work experience, in this question the majority is in 6–10 years (46%), followed by 10+ years of modeling experience (40%). This might be occurred by some possible reasons. Firstly, some respondents might have learned software modeling after getting the job or employment (i.e., after graduation, during

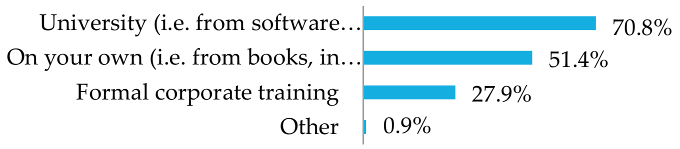


Fig. 7. Where/how software modeling was learned.

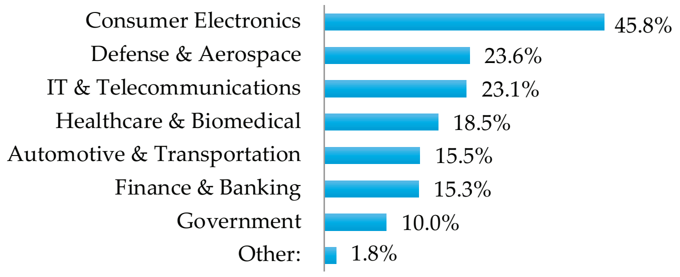


Fig. 8. Target sectors of products.

the job or with some training). Secondly, modeling in embedded domain might require some initial work experience to understand embedded requirements and systems.

Q12 was again a multiple-response question, in which we asked where/how the participant learned software modeling. (e.g., participants might learn modeling in university and from formal corporate trainings). The answers are compatible with the previous question, which investigates the modeling experience and explains why 6–10 years modeling experience is the majority. For example, some participants, who were graduated from Electrical/Electronics Engineering, have learned software modeling after getting the job (after graduation, on his/her own or with formal corporate training). Therefore, his/her work experience is longer than modeling experience since he/she did not take any software engineering or computer science courses on modeling during university. However, any computing discipline graduate’s work experience and modeling experience are most probably the same. As expected, “University” is the majority, followed by “On your own” and “Formal corporate training”. The given responses are shown in Fig. 7 with “Other” responses.

Q6, in which the type of the applications developed was asked, is the only question, which is used for inclusion or exclusion of data points gathered from the respondents. Since this was again a multiple-response question, multiple type of application could be recorded, e.g., a company can develop both embedded and desktop applications. 77% of participants reported developing “Embedded" applications and 13% of participants (13%) both “Embedded” and “Desktop” applications. Some participants used the free-text area as “Other” (10%) to explicitly indicate their type of applications developed in their company. Some responses (e.g., “Smart TV applications”) are also counted to be in the embedded domain and included in our dataset.

Q7 was about the target sectors of the products developed by the company employing the participants (Fig. 8). Seven possible choices were pre-provided in the questionnaire, which were designed in discussions with embedded software industry partners. The most popular target is “Consumer Electronics”, followed by “Defense & Aerospace” and “IT & Telecommunications*” (Please refer Section 4.5 for cross-factor analysis based on these sectors).*

To get a sense of the size of the companies, instead of asking the size of the company (in order to eliminate non-engineering roles as technicians, office workers, etc.), the number of employees in SE roles was asked (Q8). Results are shown in Fig. 9.

We should note that, as it has been established in studies on information quality (for example by Garvin [64]), people in different positions see and rate importance of different issues differently and in general

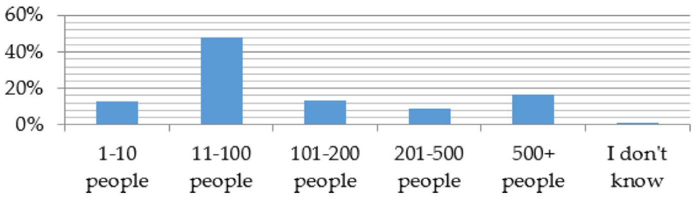


Fig. 9. Number of employees in SE roles.

have varying viewpoints on SE and related processes. As seen, there is a good mixture of respondents from various embedded software industry and different number of employees in SE positions (from developer to tester and project manager to quality assurance engineer), which would enable our analysis to cover a wider spectrum of inputs.

4.2. Current state of modeling (RQ1)

4.2.1. Degree of using software modeling in SDLC (Q10)

This question investigated how often the participants use software modeling in the SDLC by including both informal and formal usage (i.e., sketches or models) using a 5-point Likert-scale (Notice that sketching is counted as software modeling in the survey). As we can see from Fig. 10, the “often” choice is the most reported one.

4.2.2. Media used to create sketch or model (Q13)

In this multiple-response question, respondents were asked to report the media they use to create (draw) models. A 5-point Likert-scale was utilized for the answers. Results are depicted in Fig. 11. By far, using modeling software on PCs for modeling is the most used medium. Modeling using pen and paper is the next common approach.

The purpose of the modeling and the category of software modeling (e.g., sketch, model-based or model-driven) are strongly related with the medium used [26]. It is possible that some of the respondents were referring to descriptive modeling and others to prescriptive modeling while answering this question as in [18]. If there is no auto-generation of any software artifacts (e.g., code, document or test scripts – as in the case of model-based usage, which includes “sketching” in the survey), analog media like paper or whiteboard are enough for communication or understanding a problem at an abstract level. It does not mean that model-driven users do not use paper or whiteboard; indeed, such analog mediums might be a quick solution for a better communication and faster idea sharing technique in some situations. However, the lifespan of these sketches or diagrams is less than the ones created digitally via PC or tablet/smartphone. In that sense, the digital mediums like PC or tablet/smartphone have advantageous on archiving and have longer lifespan. Therefore, by providing modeling tools and archiving diagrams (either informal sketches or formal models) easier as being digital, PC is the most used medium.

Cross-factor analysis of the above data with Q14 (Modeling languages) showed that the participants, who do not use any formal software modeling (i.e., the ones who draw some sketches), use just paper or whiteboard. On the other hand, the participants, who use any formal modeling language (e.g., the ones, who use UML), usually use modeling tools on PCs. We have a specific question to ask about the modeling tools (Q16).

4.2.3. Modeling languages (Q14)

Notice that any informal usage of modeling (as a sketch) is seen as "modeling usage" at that point and this question tried to understand the modeling language that participant use, if any. Since this was again a multiple-response question, multiple items could be recorded (e.g., participants might use both UML and DSL). The majority of participants (77%) use UML (not surprisingly), but it is interesting that the second most frequently selected response is “Sketch/No formal modeling language” (65%), which is the informal usage of modeling. “DSL”, “Any UML extensions (profiles) such as MARTE”, “Systems Modeling Language (SysML)", “MATLAB”, “Any Business Process Modeling Language such BPML” and “Service Oriented Architecture Modeling Language (SoaML)” took also some responses as shown Fig. 12.

Another interesting result is that some respondents chose both UML and also “Sketch/No formal modeling”, which show that these participants use modeling both formally and informally as in [17] depending on their purposes. Apart from the pre-given choices, many “Other” modeling languages (8,6%) were reported (e.g., AUTOSAR, Eclipse Modeling Framework (EMF), Markov Chain Markup Language, AADL or Modelica) which you can access its detail from [63]. This denoted that there exists a wide spectrum of modeling languages in this domain and engineers select the modeling languages suitable for their needs (e.g., target sector of the product or modeling purpose) in their projects (See Section 4.5).

Agner et al. [20] and Liebel et al. [21] have reported the usage share of modeling languages in their survey pool. According to Agner et al. [20], 45% of participants use UML either completely or partially. In [20], only 1% of participants reported that they use another modeling languages than UML, and the names of those other modeling languages were not explicitly reported. Thus, the results of Agner et al. [20] are different from our results and it is not easy to explain why. In [21], the majority of participants (46%) reported using UML, followed by SysML, various DSL’s, Modelica and the MARTE UML profile.

Since UML is a general-purpose modeling language, its usage is not only restricted to modeling software, but it is also used for system engineering, for business process modeling and for representing the organizational structures [46] although there are some specific modeling languages for these disciplines (e.g., SysML for system engineering, BPML for business process). Moreover, although UML is built upon object-oriented concepts such as classes and operation, non-object oriented systems may also be modeled using it. Furthermore, during university (e.g., from SE courses, if taken), mostly UML is taught as modeling language. Therefore, UML’s popularity is not a surprise [65]. On the other hand, a very recent study on the usage of UML in practice [17] shows that although UML is viewed as the ‘de facto’ standard, it is by no means universally adopted. The majority of those interviewed in [17] who do use UML tend to do so selectively and often informally. This finding also

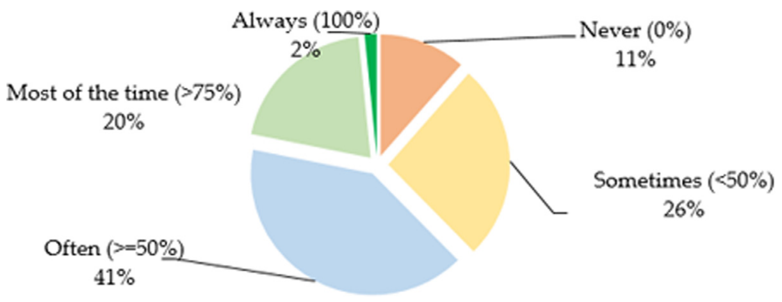


Fig. 10. Degree of software modeling usage.

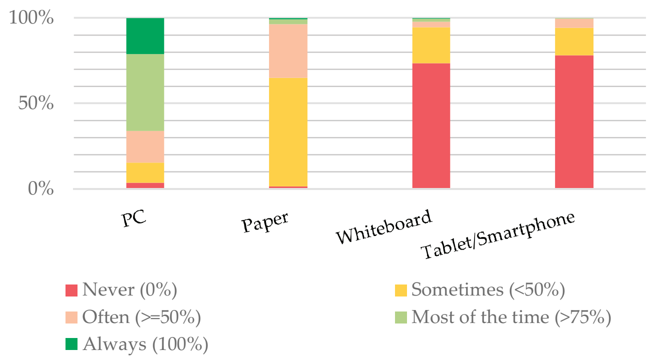


Fig. 11. Mediums to create diagrams and their usage frequency.

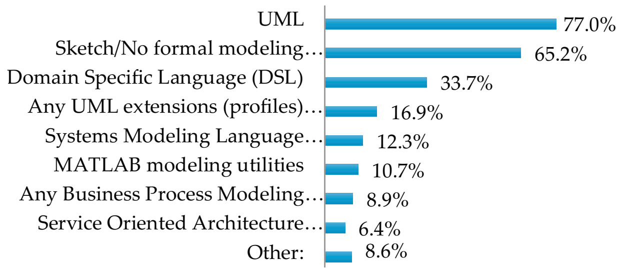


Fig. 12. Modeling languages.

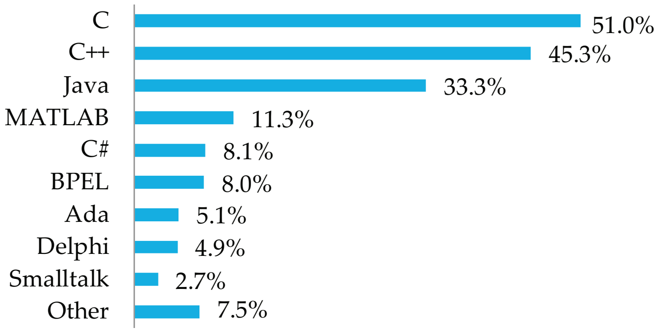


Fig. 13. Programming languages.

supports the ratio of our second most selected response as “Sketch/No formal modeling”.

As observed in [18], UML is not so popular for prescriptive modeling since its semantics is not exactly defined and this would hamper the automatic translation towards other models. We also found that in model-driven approaches, it is not so important to have a graphical syntax to represent the model (as in UML), but these models should be represented in a format that is readable by a machine (as in DSL) [66]. This also supports our findings on “DSL”s.

4.2.4. Programming languages (Q15)

The responses given for this question is shown in Fig. 13. According to this multiple-response question, the C language is the first, followed by C++ and then Java. Notice that, although C is the most popular programming language in the embedded world, the total responses for C++ and Java combined, which are both object-oriented programming languages are much more than C. MATLAB, C#, BPEL, Ada, Delphi and Smalltalk took some responses, which were in the pre-given answer set.

Apart from these pre-given choices, Python (2,7%), Objective-C (2,7%), JavaScript (1,2%) and Scala (1%) were among the “Other” answers for this question.

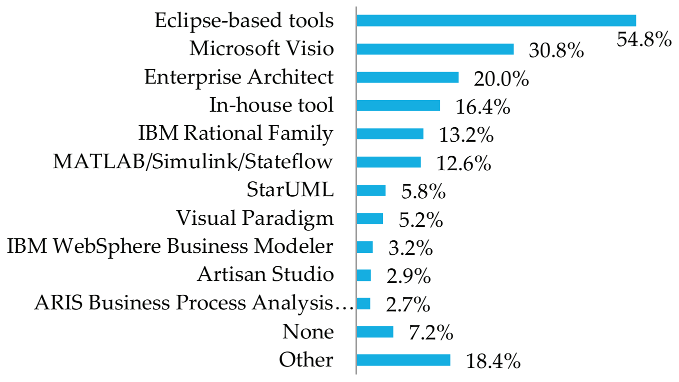


Fig. 14. Modeling tools.

We observed that the participants, whose type of application developed is related with "mobile" (the ones, who explicitly mentioned "mobile" in the "Other" free-text area in Q6) are using mostly Java and Objective-C, which also showed that mobile applications are developed with such programming languages.

Notice that this question was not intended to inquire about automated code generation (e.g., model-based users also responded this question while they might use software modeling as a communication tool). Thus, this question did not answer the target languages/encoding used by code generators (e.g., C, C++ or Java). We thus postpone such questions and inquiries to the future work.

4.2.5. Modeling environments/tools (Q16)

This question was also a multiple-response question, and thus multiple modeling tools could be recorded. As seen in Fig. 14, the majority of respondents use “Eclipse-based” tools, which is followed by “Microsoft Visio”. About 7.2% of the respondents indicated that they do not use any modeling environment or tool, which almost all came from users which reported not using PC-based tools.

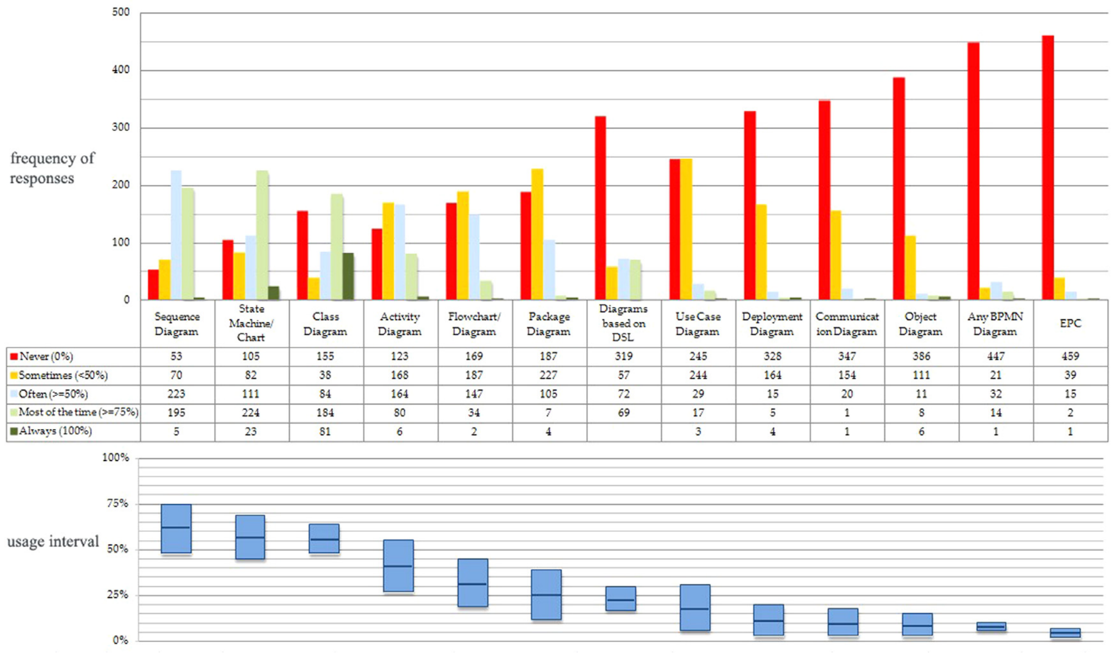
Again, among the “Other” answers, respondents mentioned modeling tools such as: Papyrus, MaTeLo, argoUML, MetaEdit+, Astah, and Artop. Notice that although Papyrus (~3%) is an eclipse-based tool, some participants wanted to explicitly mention on this tool in “Other” part. (For the details of “Other”, please see [63]).

The study [20] stated that survey studies are needed to investigate the types of UML tools used in practice. As a comparison, in the dataset of the survey reported in [21], the majority (%50) used Matlab/Simulink/Stateflow, followed by Eclipse-based tools, Enterprise Architect, in-house tools and IBM Rational Software Modeler.

4.2.6. Diagram types (Q17)

Participants were then asked about the diagram types that they use while modeling via the same 5-point Likert-scale used in previous questions. Notice that, it was not mandatory to select a frequency answer on each item, therefore, total responses for each diagram types might vary (i.e., total response for Class Diagram is 542, whereas this number is 516 for Deployment Diagram). Note that the respondents, who state that they were doing informal modeling, make the sketches, which include some essences of UML (e.g., some elements of state machine/charts, but not dependent on strict UML rules) as in [17], who do use UML tend to do so selectively and often informally. Therefore, these participants, who do informal modeling, answered this question by selecting some model (diagram) types (e.g., some participants draw a use case diagram or sequence diagram informally). All responses for each diagram types are shown in Fig. 15.

According to the responses, sequence diagrams and state -machines/-charts are the most popular diagram types in the embedded software by analyzing their usage interval values [63]. It came as a surprise that sequence diagrams were more popular than state machines/-charts,



	Sequence Diagram	State Machine/ Chart	Class Diagram	Activity Diagram	Flowchart/ Diagram	Package Diagram	Diagrams based on DSL	Use Case Diagram	Deployment Diagram	Communicat ion Diagram	Object Diagram	Any BPMN Diagram	EPC
🟥 Never (0%)	53	105	155	123	169	187	319	245	328	347	386	447	459
🟨 Sometimes (<50%)	70	82	38	168	187	227	57	244	164	154	111	21	39
🟦 Often (>=50%)	223	111	84	164	147	105	72	29	15	20	11	32	15
🟩 Most of the time (>=75%)	195	224	184	80	34	7	69	17	5	1	8	14	2
🟩 Always (100%)	5	23	81	6	2	4		3	4	1	6	1	1

Fig. 15. Usage frequency and interval of different diagram types.

since the latter are discussed more commonly in the embedded-software-focused research. By an in-depth look at the data, we found that most people use sequence diagrams informally and selectively to convey the communication among the entities in a given system (e.g., the participants, who use “Sketch/No formal modeling” with “UML”).

Notice that although class diagram is relevant for object-oriented programming languages (e.g., C++ or Java) and is not used in C, which is the most used programming language according to our survey result, this diagram is in third place. In other words, where applicable (i.e., if relevant for the used programming language), Class Diagram is widely used. The reason for a large usage of class diagram might be just due to the fact that it is a fundamental part of any well-formed UML diagram (i.e., if you draw a sequence diagram you need some classes to type the lifelines).

In [20], since it focused only on UML, the four most used UML diagrams were class, sequence, use-case and state machines, which were also reported so in [41,42]. Class diagrams were the most frequently used in these three surveys [20, 41, 42]. One of the most interesting result is that, although previous surveys on modeling indicate that use-case diagram usage was at one of the first places, the frequency of use case diagram usage is relatively low in our survey. Perhaps, since use-case diagram has a specific role for the analysis phase rather than design or implementation of SDLC and our pool of participants might use different types of diagrams for analysis, if needed. Moreover, use cases might not be the best way to present the requirements for an embedded system.

4.2.7. *SDLC phases in which software modeling is used (Q18)*

This multiple-response question was about SDLC phases, where software modeling is used. The majority of respondents use modeling in the “systems/software design”, “implementation” and “preliminary/systems analysis (requirements)”. “Integration” is the SDLC phase, in which modeling is used at least. The results are presented in Fig. 16. Notice that there is no categorization on modeling approach (i.e., for sketches, model-based or model-driven) while answering this question; therefore there is no distinction for either descriptive or prescriptive modeling [66].

The survey in [21] reported similar results as that dataset stated that models are mainly used for subsystem/component design, implementation, system architecture, and testing. These findings are as expected since modeling (e.g., UML) is mainly applied for design and requirements phases. Although the survey in [19] did not explicitly mentioned their ratios, it reported that MBD is mainly used in design, implementation and maintenance.

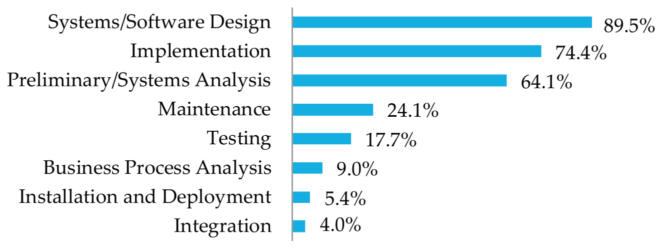


Fig. 16. SDLC phases where software modeling is used.

4.3. *Current state of MDE and its adoption (RQ2)*

4.3.1. *Degree of using MDE (Q19)*

This question investigates how often the participants use MDE. Q19 acted as a decision point in the survey in a way that the survey ended for the participants who mentioned not using MDE at all (i.e., the “Never” answers (370 respondents, 59.5% of all participants)). The survey continued for participants who said they use MDE (remaining 185 respondents, 29.5% of all participants). This decision logic was programmed into the online survey form. The results are shown in Fig. 17.

Our results show that the MDE usage ratio is slightly more than the ratio reported in [20], in which 15.8% of its participants reported knowing MDE and using it. Our study reflects a world-wide picture and ~2 years have passed after [20] was executed. We might speculate that the embedded software industry has gradually adapted the MDE practices and its usage ratio has increased. Therefore, this difference might be explained with the participants’ demographics and the possible increasing popularity of MDE practices in the embedded software industry.

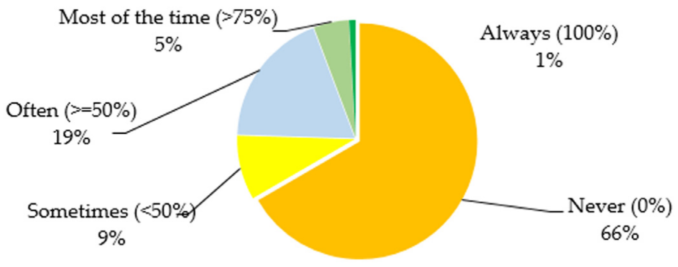


Fig. 17. Degree of using MDE.

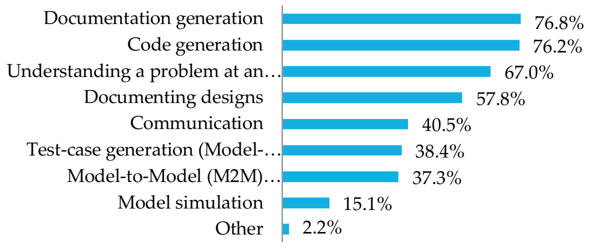


Fig. 18. What MDE is used for.

4.3.2. *What MDE is used for (Q20)*

We further asked the reasons and purposes for MDE usage as a multiple-response question. Results are shown in Fig. 18. Documentation and code generation were reported to be the most popular reasons for using MDE. Notice that we do not distinguish between descriptive and prescriptive modeling in that question (e.g., as in [18]). However, as we indicated that the purpose of the modeling and the category of software modeling (and also the media used, the lifespan and the archivability) are strongly related (See Q13). Descriptive models classify actual objects, events, and processes into categories; whereas prescriptive ones specify what is expected of systems components and how to develop them [18]. That distinction provides a formal justification between analysis and design models, which might affect the reasons for MDE usage. For example, just for “communication”, descriptive modeling might be enough (e.g., sketch), and it might not be a primary concern of MDE. Therefore, from this perspective, we suggest the future surveys to explicitly identify this distinction.

In [19], communication and early simulation of the functional model were reported as the main usage reasons of MBE. According to Agner et al. [20], communication, understanding of a problem at an abstract level and documenting designs are the most important reasons of using MDE. The survey [21] reported that models are mainly used for model simulation, code generation, test-case generation and information/documentation; hence, using models for assisting activities in the SDLC seems to be an important function as also confirmed by our survey results.

On the other hand, most participants in the survey of [20] reported that they are not conducting model-based automatic code and document generation. The authors in [20] argued that the lack of skilled professionals in MDE and also the lack of powerful and user-friendly MDE tool support are the main reasons of such a situation. They also claimed that these findings differ from results of [48], which reported that activities such as code generation, transformation models, and executable models are more used in practice. We assumed that “documentation generation”, “code generation” and “test-case generation” include some Model-to-Text (M2T) transformation; therefore we just gave “M2M” transformations in the answer set in order to get rid of any possible duplication. By focusing on the embedded software, our survey differs from Agner et al. [20]’s results since automatic artifact generation (e.g., document or code) seems to be quite popular in the embedded world for those who employ MDE.

Table 4
MDE-specific purposes’ ratio comparison with the related works [20,21].

Purpose	In [21]	In [20]	This study
MDE-specific purposes[a]	76%	23%	61%
The modeling purposes, which might be achieved without model-driven approach (e.g., with sketching or model-based)[b]	24%	77%	39%

[a] *“code generation”, “test-case generation”, “documentation generation”, “M2M transformation” and “model simulation”.*
[b] *“communication”, “understanding” and “documenting analysis & design”.*

Note that some MDE purposes in that question (e.g., “communication”) might not be specific to MDE usage and the stakeholder might achieve such purposes without MDE enforcement (e.g., strict syntax) or without using a modeling tool. If we categorize the answer set of Q20 whether the purpose is specific to MDE or not, we have two groups:

- MDE-specific purposes (i.e., “code generation”, “test-case generation”, “documentation generation”, “M2M transformation” and “model simulation”)
- The modeling purposes, which might be also achieved without model-driven approach *(*i.e*., “communication”, “understanding” and “documenting analysis & design”)*

By this way, we want to understand the relative ratios of these two derived groups in each related works as in Table 4 (Notice that in [19], there is not any percentage values for the reasons, therefore we include [20,21] as a comparison).

The majority of participants (93%) in [21] had already used model-driven techniques in their projects and software modeling is mainly used for MDE-specific purposes (76%). On the other hand, in [20], MDE activities are mainly used for the purposes, which might be also achieved by using sketching or model-based approach (77%). In our survey, there are also some participants, who just use one of the MDE-specific purposes such as “documentation generation” or “model simulation” (e.g., without “code generation”) besides having general modeling purpose(s) such as “understanding” (Note that 67% of our respondents use MDE for understanding a problem at an abstract level).

On the other hand, although it is not directly related with embedded software development and focused on only UML, the survey in [67] showed that practitioners use modeling during communication and planning of joint implementation effort. Similarly, Gorschek et al. [51] found that modeling are used primarily as a communication and collaboration mechanism where there is a need to solve problems and/or get a joint understanding of the overall design in a group.

With the comparison of these related works, we can say that there are different understanding (and also purposes) of “MDE” in the industry, which might be specific to MDE purpose or not. Our survey showed that although MDE has different benefits (Q24), it has also some drawbacks (Q25), which are not experienced in sketching or model-based approaches. Since there is a danger that resources are being wasted, deciding in what degree and with how much modeling rigor (e.g., by automating software artifact generation as in MDE with an extra tool cost) is a critical question. Moreover, while using MDE, the type of MDE-specific purpose (e.g., “code generation” or “document generation”) might affect modeling practices with respect to technology cost (e.g., selection of modeling tool). We believe that purpose is one of the important factors, which determines the most effective modeling approach (from sketching to model-driven approach) depending on stakeholder’s tasks and responsibilities in the particular project (See Section 5.2).

4.3.3. *MDE maturity levels (Q22)*

Participants were asked to describe their company’s maturity in its use of MDE. We were aware of several existing maturity models for MDE

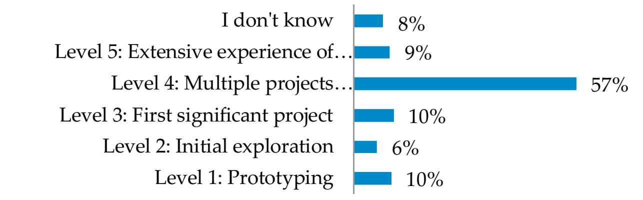


Fig. 19. Maturity of MDE usage.

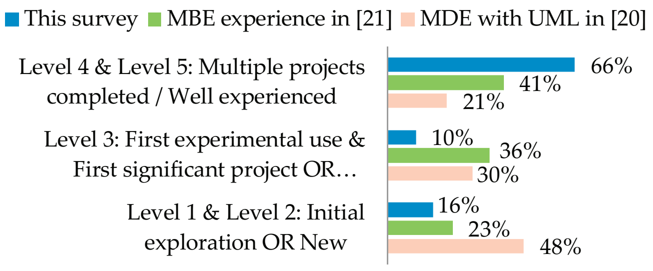


Fig. 20. MDE/MBE maturity level comparing with data categorization from [20,21].

(e.g., [68,69]). [68] seems to be the most comprehensive maturity models in this context. In choosing a maturity model to be used in our survey, we had two criteria in mind: (1) using the maturity model should not lead to having many questions which would negatively impact the response rate of our survey, and (2) the maturity model should be comparable to existing measurements in the reported surveys. Due to this, we adopted the maturity model as shown in Fig. 19.

The majority of the participants (57%) are in the Level 4, indicating that they have completed multiple MDE projects. 10% of participants reported that they have the first significant project on MDE (just finished); whereas 6% are in initial exploration phase and 10% are in the prototyping phase of MDE. On the other hand, 9% of participants reported an extensive experience of MDE on many projects and/or over many years.

According to Agner et al. [20], since it only focused on UML, 48% of the respondents confirmed its use as an initial exploration of MDE with UML and only 21% declared the development of several complete projects using UML, whereas the others confirmed its use as a first experimental use (13%) and first significant project (17%). On the other hand, concerning the MBE experience in [21], many participants (41%) are well experienced with more than 3 years of usage; whereas 36% state that they have moderate experience and only 23% are new in the field of MBE.

Since the terminologies used in these two studies are different from each other, we want to categorize them in similar groups. According to that categorization, we assume that “initial exploration” in [20] is in the same category in “*new*” in [21]; “*first experimental use and first significant project*” in [20] is in the same category in “*moderate experience*” in [21]; and finally “*several complete projects*" in [20] is in the same category in “*well experienced*” in [21] (which is our both “multiple projects completed” and “extensive experience” categories). The maturity level comparison is depicted in Fig. 20.

As it can be seen, we can say that maturity level has changed (and increased) depending on either time, generalization of geographical area (i.e., [20] was executed at 2011 in Brazil and [21] was more recent in Europe) or participant demographics. Notice that, by no means, these data indicate that the popularity and the usage of MDE have increased, but it gives an insight about its trends although all studies use different scales (and questions) and have entirely different populations.

4.3.4. *Motivations for adopting MDE (Q23)*

Participants were asked about the motivations that they and/or their companies considered for adopting MDE (Fig. 21). Since using MDE provides different types of benefits for different users, the survey provided 12 motivations to be selected according to the degree of importance. This set of motivations was synthesized from the related work (as discussed in Section 2.2).

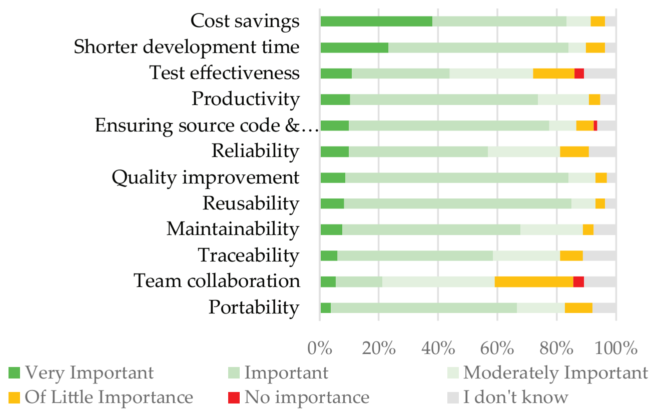


Fig. 21. Motivations for adopting MDE.

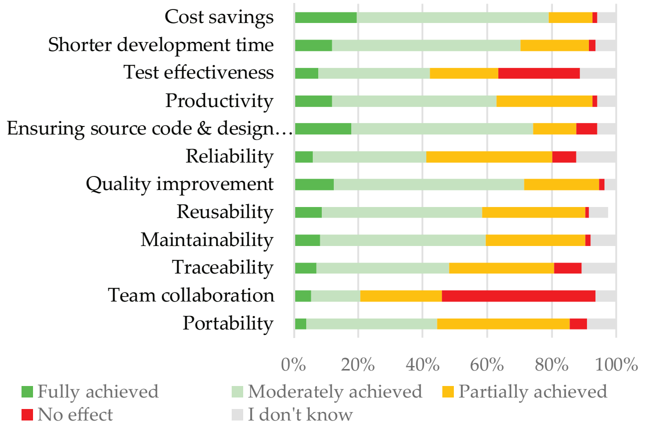


Fig. 22. Benefits of MDE in embedded software engineering.

According to results, cost savings and shorter development time were generally ranked of the highest importance. In [19], quality improvement, development of functions with high complexity and shorter development time were reported as the top three motivations for MDE. On the other hand, according to Liebel et al. [21], shorter development time, reusability and quality improvements were the most three popular motivations to introduce MBE; whereas cost savings is at sixth place in popularity while adopting MBE.

4.4. *Benefits, challenges and consequences of using MDE (RQ3)*

4.4.1. *Benefits of MDE (Q24)*

Since it is important to understand the impact of the MDE, participants were asked about the degree to which their motivations were actually achieved after using MDE (i.e., the degree to which their expectations were met). Note that the list of possible answers for question Q23 (i.e., motivations such as cost savings, shorter development time, etc.) is the same as for that question, where their ranges are different (i.e., “importance” ranges are from no importance to very important (0–4); whereas “benefit” ranges are from no effect to fully achieved (0–3)). Results are shown in Fig. 22. According to respondents, cost savings, ensuring compatibility between source code and models, shorter development time and quality improvement are the top four benefits. Generally, all the benefits are below the importance levels, denoting that

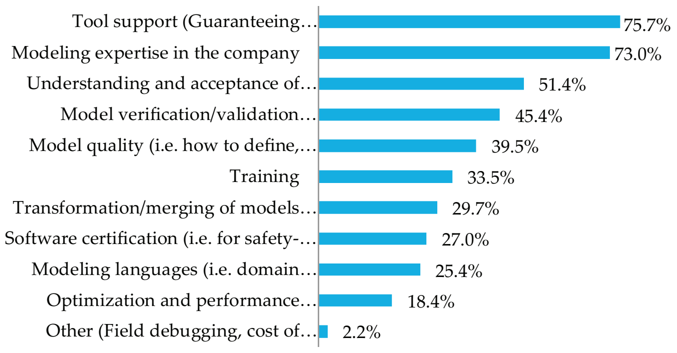


Fig. 23. Challenges of MDE in embedded software engineering.

expectations are not fully met. Please refer [63] to see what expected and gotten from MDE.

Such findings differ from Agner et al. [20], in which the most significant benefits are associated with quality improvement, portability, maintenance and productivity. On the other hand, according to Liebel et al. [21], the effect of introducing MBE are quality, reusability, reliability, traceability, maintainability, development time and cost, respectively (according to highly positive answers). In that sense, our results are also different from [21] since cost savings is the most significant effect of MDE. In that sense, Broy et al. [19] also says that MBD can bring significant cost savings and time savings, but only with a well-chosen approach (i.e., without manually changing auto generated code).

As in any engineering activity, embedded software projects should also be completed within anticipated budget (cost), within anticipated schedule (time) in conformance to requirements (quality) [70]. All individual quality factors (e.g., reusability, maintainability, portability) and shorter development time have significant effect on project budget, which is related with cost. Our participants experienced different benefits degrees on some specific quality attributes (e.g., moderately achieved reusability, but partially achieved productivity or vice versa) with a direct or an indirect effect on cost savings. Similarly, some of our participants achieved shorter development time, which also affects cost savings. In other words, although there might be some variations in the degree of benefit for quality attributes, improvements and shorter development time; all these resulted cost savings. This viewpoint might explain why "Cost savings" is the only benefit, which is between "Fully Achieved" and "Moderately Achieved" range according to our findings.

4.4.2. *MDE challenges (Q25)*

Participants were asked about the MDE challenges in their company as multiple-response answers. According to responses, tool support and modeling expertise in the company are the most encountered challenges (Fig. 23). Thus, we can pick those as areas for possible improvement in training, further research and tool development.

Note that, during the pilot study, we needed to modify this question pre-given answer set by combining some separate answers; but in that case we tried to make the argument clearer by including some explanations. For example, although “transformation” and “merging” models seem to be two different challenges, we combined them in a single item but include “how to integrate models in different projects?” explanation.

Although there was no explicit question on MDE challenges in [20], the reasons of not using UML diagrams was asked and the top three results were: short lead-time for the software development, lack of understanding or knowledge of UML models and existence of few people in the company who have deep knowledge of UML. Furthermore, according to Agner et al. [20], in MDE the users must have access to appropriate tools, in a way that integrates a tool suite that meets requirements such as modeling, transformations, and code generation. This supports our finding about tool support challenges in order to guarantee synchronization between software artifacts; i.e., code, document and test driver.

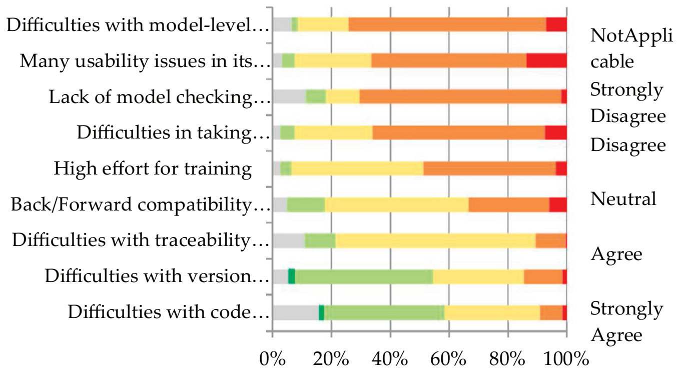


Fig. 24. Problems with MDE environments/tools in embedded software engineering.(For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

In that sense, our findings are similar to Agner et al. [20]. In addition, although it is not directly related with embedded systems, Hutchinson et al. [48] pointed out the need of a longer training period so as to overcome the lack of UML expertise, which is also in parallel with the “modeling expertise” challenge in our survey. According to Liebel et al. [21], “high effort for training of developers” and “modeling tool challenges” (which will be analyzed separately in Section 4.4.3) were also mentioned, which are similar to our findings. There was no explicit MDE-challenge question in [19], however "tool costs" and “training” were seen as a negative aspect of MDE in the automotive industry.

4.4.3. *Problems with MDE environments/tools (Q26)*

As a both multiple-response and 5-point Likert-scale question, participants were asked about the degree to which the given problems are applied to MDE environment/tool they use. All responses are shown in Fig. 24, whose x-axis indicates the response percentage. In the figure, red and orange bars indicate the existence of such a problem; whereas green-based bars indicate that there is no such an existence. On the other hand, neutral responses are depicted with yellow bar, and “not applicable” answers are depicted with grey bar. Notice that MDE environments/tools problems are directly related with what MDE is used for (Q20) hence “not applicable” answers (e.g., for the respondents who use MDE for only “documentation generation”, “difficulties with code generation capabilities” is not applicable).

According to Liebel et al. [21], tool-related problems were reported to be the following: many usability issues with the tools, difficulties with version management, difficulties of integration with legacy code, impossible/difficult to customize the tools, lack of model checking capabilities and difficulties with code generation capabilities. Such findings are quite similar to our results.

Although it is not directly related with embedded software development, a recent study in 2017 pointed out that MDE tools, which depends on technical, organizational and social factors, play a major part in the adoption of MDE [71]. Note that in that question, we focused not only on technical features of the MDE tool, but also non-technical factors such as organizational and social factors (e.g., training and difficulties in taking support from the vendor), which the respondents were also stated as impeding issues.

4.4.4. *Impacts and implications of MDE (Q27)*

This question investigated the impacts of MDE on code generation and model-based/driven testing as well as the complexity aspects of MDE. By applying a similar design to Hutchinson et al. [48]'s "*paired questions*", in which they aimed to explore the balance between the types of positive and negative effects of MDE, participants were asked about the consequences of MDE. The results are shown in Fig. 25.

Due to the growing complexity of software, it is generally agreed that the only realistic way to manage this complexity is using appropri-

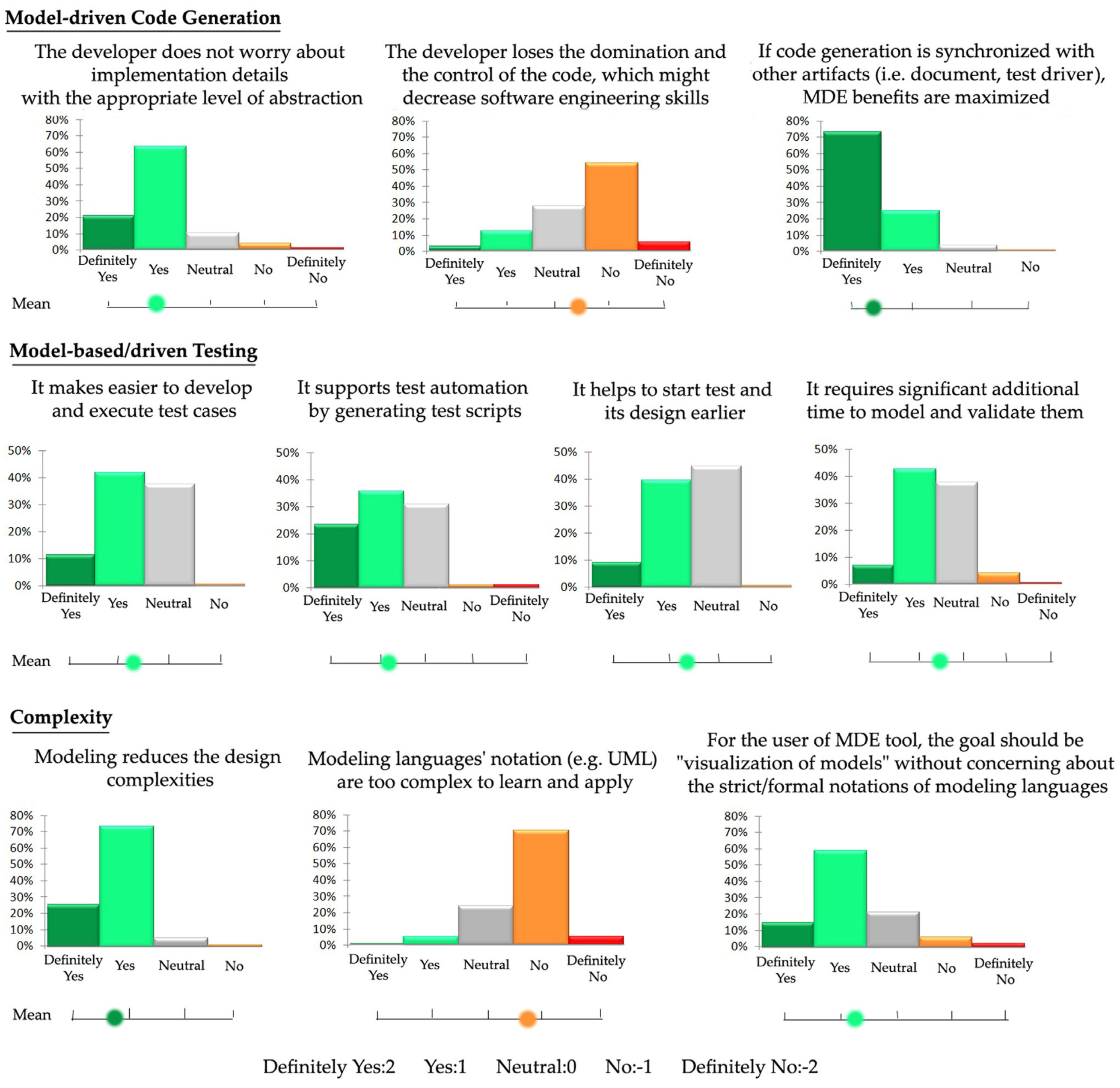


Fig. 25. Impacts and implications of MDE in embedded software engineering.

ate methods of abstraction with modeling [72]. Moreover, model-driven code generation is an important aspect to improve productivity in MDE [20]. However, an interesting result in [47] is that participants working on real-time systems are more likely to agree that their organizational culture does not endorse (like) modeling due to automatic code generation. Similarly, as in [42], UML is too complex or according to Lange et al. [43], there are lots of UML complexity problems as reported in previous studies (e.g., [73–75]). In this question, to address the balance, for example, in model-driven code generation part, the first statement mentions about the possible positive consequences of MDE on "abstraction", whereas the second statement mentions about the possible negative consequences of MDE on "abstraction". Similar approaches are applied for both model-based/driven testing and complexity. As seen in Fig. 25, all responses are depicted according to response percentage (in y-axis) and the mean value is also presented with its corresponding color at the bottom of each statement.

In terms of implications of MDE, the results showed that "abstraction" has positive impacts since the mean value of the first statement (i.e., possible positive consequence) is near to "Yes"; whereas the mean of the second statement (i.e. possible negative consequence) is between "Neutral" and "No". Moreover, similarly, respondents generally agreed that modeling reduces the design complexities as a positive consequence and they mostly did not agree that modeling languages are too complex to be learned and applied, which might be a possible negative consequence. Therefore, for these two "paired" arguments, there was no conflict (e.g., the majority of participants did not agree with the possible negative consequences; instead the negative argument supported the first one, which is the positive consequence). On the hand, many respondents believed that model-based/driven testing makes it easier to develop and execute test cases by supporting test automation (e.g., positive consequences); however, although it helps to start to test and its design earlier; it requires significant additional upfront efforts to model and validate them (e.g., negative consequence). Therefore, according to responses, there should be a "balance" while applying such an approach.

Note that the pre-given answer set for that question was also revised after pilot study. For example, the second part of the second argument for model-driven code generation was added after the first pilot study, in which three participants suggested such an argument too. Therefore, we decided to include this argument but with a probability (i.e., "*which might decrease…*").

4.5. Cross-factor analysis

One of the opportunities the survey data provided as a further study was to analyze relations among software modeling practices and practitioner demographics. To understand the effect of target sector of product(s) on the modeling practices and approaches, a cross-factor analysis was conducted. Please refer [76] for the details of this study (*Note that in that study, due to space constraint, we excluded "Government", which is the least chosen sector in the survey*). According to the results:

- "Healthcare & Biomedical" sector is using software modeling the least as being at "Sometimes" level, the other sectors is at "Often" level. However, according to MDE usage, all sectors are at "Sometimes" level, where "Finance & Banking" is the least.
  - Although "Consumer Electronics" might be probably considered as one of the sectors where innovation and time to market drives the business, MDE usage ratio is between 9–17%. MDE is a technique established to support these values at most; but it might be important to analyze what and where is the problem in this sector although its software modeling usage ratio (but not MDE usage) is high (e.g., the participants in this sector use sketching or model-based approaches, but what are the specific consumer electronics' challenges or bad experiences on MDE, which resulted such a situation?)
  - "Defense & Aerospace" sector is the one, which uses MDE at most, whose MDE usage ratio is between 24–43%. Perhaps, the project length and necessary investigation on MDE (its corresponding costs, i.e., tool, training, etc.) might be suitable for this sector.
- The dominant modeling language is UML in all sectors; however, there are interesting results based on sectors.
  - Specific modeling language for target sectors (i.e. AADL (Architecture Analysis & Design Language) for "Defense & Aerospace", EAST-ADL for "Automotive & Transportation" and Markov Chain Markup Language for "Consumer Electronics") are interesting results.
  - DSL is mostly used in "Automotive & Transportation", where AUTOSAR usage is ~15% although it was not in the pre-given answer set.
  - The usage of "Sketch/No formal modeling language" is very similar to UML usage in "Finance & Banking" sector.
- The most used diagram type according to the survey result (i.e., Sequence Diagram) is also the most used diagram for only two sectors (i.e., "IT & Telecommunications" and "Healthcare & Biomedical"); the other sectors have different most frequently used diagram types (e.g., for "Consumer Electronics" is "Flowchart/Diagram" or for "Defense & Aerospace" is "State Machine/Chart").

With the help of this cross-factor analysis, the state-of-the-practice of software modeling and MDE practices in different industrial sectors was better understood by addressing RQ1 and RQ2. Some modeling languages or diagrams are specific to some sectors or their usage ratio is different depending on their purposes and challenges [76].

5. Discussions

A summary of our findings is discussed in Section 5.1. Section 5.2 provides implications of our findings for software modeling stakeholders. Limitations, potential threats to the validity of our study and steps we have taken to minimize or mitigate them are discussed in Section 5.3.

5.1. Summary of findings

Our survey received 627 acceptable responses from 27 different countries in five continents and different industrial sectors related to embedded software. There was a good mixture of different profiles, which helped our results to be unbiased from certain types of demographics in the embedded software engineering projects. A highlight of the results is discussed next.

RQ1 – Summary of the current state of modeling

Software modeling (either informal, selective or formal) is widely used by many embedded professionals (89%). As expected, different engineers and companies use software modeling approaches in varying degrees, which usually depends on the modeling characteristics [26]. Software modeling is conducted from informal sketches (on paper) to formalized models using sophisticated modeling tools.

The majority of respondents use UML. However, depending on the type of industrial sector, a general-purpose modeling language such as UML is usually not sufficient to meet the specific requirements; therefore other modeling languages are used, e.g., the AUTOSAR language (in "Automotive & Transportation"), models based on the Markov chains (in "Consumer Electronics"), and various other DSLs (e.g., AADL for "Defense & Aerospace"). Especially, in model-driven approaches, modeling stakeholders prefer models, which can be represented in a format that is readable by a machine (as in DSL).

A variety of modeling tools are used, the most popular ones being the "Eclipse-based" family of tools, followed by "Microsoft Visio". The most used diagram types are sequence diagrams, state-machine diagram, and class diagram. The majority of respondents use modeling in the systems/software design phase, followed by implementation" and requirements/systems analysis phases of SDLC.

RQ2 – Summary of the current state of MDE adoption

Notice that 29.5% of all participants use MDE approaches (Q19). The respondents reported that they use MDE for mostly documentation and code generation, and then for understanding and analysis the problem domain at an abstract level.

To assess MDE maturity levels, we adopted from the literature a 5-level maturity model. Based on that model, we found that the majority of the participants (57%) are in the Level 4, indicating that they have completed multiple MDE projects. This is a generally good sign for the embedded software industry. The other aspect that we explored in terms of the current state of MDE and its adoption was the motivations for adopting MDE. The top motivators were "cost savings", "shorter development time", "reusability" and "quality improvements".

RQ3 – Summary of the benefits, challenges and consequences of using MDE

In terms of benefits of MDE, "cost savings", "ensuring source code & design model compatibility", and "shorter development time" were reported the most. In terms of challenges, tool support, and more specifically difficulties with model-level debugging and usability issues of tools were stated as the most impeding issues.

In terms of positive consequences and impacts, model-driven code generation was generally reported to be a beneficial outcome of MDE. Many respondents believed that model-based/driven testing makes it easier to develop and execute test cases by also supporting test automation via test scripts; however, although it helps to start to test and its design earlier; it requires significant additional upfront efforts to model and validate them. The embedded software community largely believes that modeling reduces design complexities and modeling languages are not that complex as reported in many studies.

5.2. Implications of results

Modeling captures some or all of the design decisions that comprise a system's architecture besides affecting all facets of software architecture by serving as the intellectual centerpiece of software development and evolution [77]. The survey results have shed light on the state of modeling and MDE practices in embedded software engineering projects and would provide practical benefits to various modeling stakeholders (especially software architects), by enabling them observe the latest trends in this industry and also influencing not only the system-level design (e.g., hardware/software co-design), but also other software-intensive embedded systems development aspects.

We discuss below the implications of our survey findings for practitioners, researchers, educators and tool vendors besides for the company that commissioned this study.

5.2.1. Implications for practitioners

- *Benefitting from what others are doing:* By looking at the benefits and challenges of MDE (See Section 4.4), this empirical evidence will help embedded software professionals, who are thinking about adopting MDE in their projects, to know common practices other adopted for their context. As survey results showed that there is a wide variety of practices, motivations and tools. Although we consulted with several industrial practitioners and used our personal industrial experiences when designing the closed-ended questions in the survey, we had a lot of "*Other*" answers than we expected (e.g., modeling language (Q14), programming language (Q15) or modeling tool (Q16)). This showed that there is a wide spectrum of in terms of the technology used for software modeling and our results might also help embedded software professionals to get awareness of these new technologies. In order to solve this need, a database that is formed by modeling community's prior experiences (i.e., survey data) has already constructed to guide different SE roles (e.g., software developers, software architects, systems engineers, test engineers) with respect to process and tool improvements during embedded software development [78]. By this way, this survey data helps modeling stakeholders (via this database) to know beforehand what similar profiles (e.g., similar SE positions, target sector of products, etc.) are doing while modeling and this saves time and budget before embarking on a project with alternative modeling practices.
- *Need to identify the characteristics of modeling and the relations between them:* We found that software modeling is widely used (89%), across a diverse range of embedded software industries to better handle the growing complexity of their software-intensive products. Embedded software professionals use different modeling languages, programming languages, environments with different motivations and face different challenges. In other words, different SE roles can use modeling and MDE selectively not only in implementation but also analysis, design or maintenance phase of SDLC according to the characteristics of modeling [26] (e.g., purpose). All of these approaches could be effective depending on these characteristics. As the survey results showed, MDE has certain challenges, which are not experienced in sketching or model-based approaches (e.g., MDE tool cost or automatic code generation challenges). Since there is a danger that resources are being wasted, deciding when to model or in what degree and with how much modeling rigor (e.g., as a sketch without modeling language formality or by automating software artifact generation as in MDE with strict enforcement) are frequently asked and challenging questions for software teams. Therefore, it is important for the practitioner to identify these characteristics and apply the most suitable approach for her/his tasks and responsibilities in the particular project.
- *Modeling as an effective communication tool:* Q20 revealed that modeling are also used as a communication and collaboration mechanism. Since software-intensive embedded systems include many hardware and software components, modeling is beneficial not only for software development side but also during system-level design including hardware/software co-design among all stakeholders. As the survey (Q23) also revealed that collaboration seems one of the motivations for using software modeling since it creates a common language and understanding among the teams during communication and planning of joint development. We investigated that some modeling stakeholders (mainly, systems engineers, whose responsibilities are cross-cutting with both hardware and software components of the system) use modeling (especially with sequence and activity diagrams) to convey the communication among the entities in a given system: Their purpose is a quick communication and explaining a scenario among both hardware and software stakeholders' of the system. If all modules' communications/interactions are well-depicted in a complete diagram with the necessary inputs (e.g., message interfaces) during a system scenario, every SE roles can understand the corresponding scenario without looking at the "textual description" of it, which might cause some misinterpretation; hence they could save time and effort by getting rid of unnecessary meetings between stakeholders [66]. By this way, modeling via "visualization" creates a common language for embedded software development [78]. Since modeling provides great support in the communication with other colleagues because of its possible graphical design, even colleagues from other departments or domains, who are not familiar with software development, can be involved in the software development.

5.2.2. Implications and benefits for the company that commissioned this study

- *Software modeling and MDE research group:* As survey results revealed, software modeling is not only used by software developers or architects; there are also other stakeholders such as systems engineers, test engineers or project managers. All necessary stakeholders in the company were informed about the results of this research (via presentations and meetings) to increase the awareness on the latest state-of-the-practices while modeling in the embedded software development. Then, in order to follow the latest modeling trends and apply them in a systematic manner, it was decided to form a new research group from different departments (e.g., Software Engineering Department (three software architects), Systems Engineering Department (two systems engineers), and Test Department (two test engineers)). This group is responsible for analyzing the problems in a specific context (e.g., within specific process or project) and try to find possible modeling approaches and solutions to these challenges besides working on the adoption/acceptance of related-technologies.
- *The adoption of MDE concepts & technologies & tools:* The Company had already worked with some MDE concepts; but there were some challenges in their adoption (e.g., one of the challenges in the survey - an organizational resistance). Although some teams in software department had used an MDE tool, which automatically generates code, document and test driver for communication interfaces of each component [4], some non-developer stakeholders (e.g., systems engineers and test engineers) had some concerns about using this tool for their business side. As Q27 (i.e., diagram 3 in Fig. 25) revealed that most practitioners with different SE (e.g., not only software developers or architects) believed that if code generation is synchronized with other artifacts (e.g., document, test driver), MDE benefits are maximized. After having the results (as being an empirical evidence) and with the help of newly organized research group mentioned above, the usage of this MDE tool increased not only in the software departments, which uses automatic code generation facility of that tool, but also in test and systems department, which utilize from other generated SDLC artifacts (e.g., test driver/simulator and documentation). As a domino effect (since this tool is now used by many teams), the adoption and understanding of different concepts of modeling has been positively changed in the company. Moreover, with the experience gained by MDE usage, the Company have designed and implemented various SDLC artifacts, which is currently used by many teams [16].

5.2.3. Implications for researchers

- *Need for more MDE techniques across all SDLC phases*: In Q18, we found that the majority of respondents use modeling in the systems/software design phase, implementation and analysis phases.

Modeling is used not that widely for integration and testing, although there are lots of academic advances and novel techniques in these areas. This makes us think whether there are issues which decrease the practical application of those techniques in industrial settings. Researchers are encouraged to look into these issues.
- *Addressing the MDE challenges*: Tool support and modeling expertise in the companies were the most encountered challenges. Researchers can work to develop better research-prototype tools and also collaborate with industry to improve modeling expertise of engineers.

5.2.4. Implications for educators

- *Improving the software modeling educations*: Our results also have implications for software modeling educations, e.g., [79,80], and educators. Our survey results suggest implications for the way in which software modeling is taught (from Q12). Some respondents (especially the Electrical and Electronics Engineering graduates) reported that they have mostly learned software modeling after getting the job (i.e., after graduation, during the job or with some training). Some respondents who were computer and software engineering graduates also reported that they have learned some modeling techniques during their undergraduate studies, but not at the application level in the industrial context.
- *MDE is not just the analysis and design phase*: A typical university SE course teaches a top–down fashion, in which diagrams are first developed for analysis and then iteratively refined into design, implementation and test phases of SDLC. In most software modeling courses, the students study how to design and develop a software system using software modeling techniques, but the focus is generally on the analysis and the design phases and there is a missing part while translating these diagrams into executable code. Extensions of these courses could focus on the important concepts in MDD, the requirements for setting up a model-driven approach, the state-of-the art MDE approaches, and the corresponding challenges in software modeling projects (there is an increasing number of universities, which use [23] as a SE course book and that might be a good sign for educators to understand and teach modeling trends and standards in practice). Therefore, we believe that the given courses on modeling might also be updated or enhanced after a further analysis of the results in our survey, which suggest topics that could have been widely covered or emphasized.

5.2.5. Implications for MDE tool vendors (builders)

- *Need for better tool support*: Tool support is one of the most encountered MDE challenges (Q25). We have also observed several shortcomings in terms of tool support (Q26). Supporting MDE with appropriate tools increases modeling benefits. Not only for embedded software development but also for rapid prototyping for different platforms with a flexible design-space exploration, such a powerful tool is crucial. Notice that useful and usable tools not only help maximizes MDE's benefits, but also play an important role in the adoption of MDE [71]. Therefore, we suggest MDE tool vendors to invest more efforts in development and improvement of these tools and including/improving the features that practitioners mentioned in this survey (such as "increasing usability of the tool", "customization on the tool", "model verification/validation and model-level debugging feature").
- *Focusing on what industry uses the most*: Documentation, code generation and understanding of problems at higher abstract levels were reported to be the most popular reasons for using MDE with different benefits and challenges. Thus, we recommend that tool vendors work on developing more industry-relevant tools and techniques, which are not tackled by commercial tool vendors. This might be achieved with more industry-academia collaborations.

5.3. Limitations and threats to validity

We discuss the possible validity concerns based on a standard checklist [81], in terms of construct, internal, external and conclusion validity concerns, and also the steps that we have taken to minimize or mitigate them.

Construct validity: Construct validities are concerned with the extent to which the objects of study truly represents theory behind the study [81]. In other words, the issue relates to whether this survey measured software modeling approaches in embedded industry. We collected data from different sources (different countries, different industrial sectors, etc.) in order to avoid mono-operation bias.

When people feel being evaluated based on what they think, they might deflect their answers. To mitigate these, we informed participants prior to the survey that our motivation in this study was to take a snapshot of the embedded software industry and that we will not collect any identifying information so that participants will remain anonymous. Therefore, for the sake of objectiveness, the survey is completely anonymous.

In our measurement strategy, what we did was common to other survey studies (e.g., [58]) —we counted the votes for each question and then made statistical inferences. We believed that results based on such voting data can, to a certain extent, reflect the opinions of the embedded software professionals.

Although we tried to select the most suitable "*paired questions*" to figure out the balance between the positive and negative consequences of MDE (See Q27), we presented "possible positive consequence" as a first statement in this question; and the choice of this order may have led the respondents to bias in the results (e.g., if the possible negative consequences was given first, the results might have been different).

Last but not the least is the issue and definitions of MDE vs. MBE as understood by that participants. We tried to reduce this threat by making sure the participants understood and distinguished the terminologies by providing them the definitions mentioned by Brambilla et al. [23] (See [60]). In order to prevent any misunderstanding and potential threat in this terminology, we conducted a pilot phase of the survey in which several practitioners filled the survey and we met with them to assess their common understanding of the terminologies regarding MDE, MDD and MBE (See Section 2.1). However, the definition provided by Brambilla et al. [23] sadly still leave room for subjectivity and we could not come up with better definitions while designing the survey since we did not have the definition provided in [26] yet (*Notice that this definition, Brambilla et al. [23] is enriched and synthesized with the concept of sketching*). Thus, this issue stays as a potential threat, e.g., a given practitioner might in fact use MBE, even though s/he stated to use MDE or s/he does not count sketching in MBD. Moreover, although there was no specific feedback on the pre-given answer set for some items (i.e., "model checking capabilities", "M2M transformation"), as we have not explicitly specified the terms, there might be different interpretations and we could not be sure that the all respondents have the same understanding.

Internal validity: Internal validity reflects whether all causal relations are studied or if unknown factors affect the results [81]. Instrumentation was improved by using a pilot study. The survey took approximately 2–10 min to be filled out depending on the modeling usage type (e.g., for no modeling, it takes ~2 min just to take demographic data; for model-driven usage, it takes ~10 min) and was intended to be filled out once by every participant. This reduces the likelihood for learning effects and, hence, maturation effects. Moreover, since the wording and terminology used should be easily understandable to get high quality data and to prevent misunderstandings, the pilot includes embedded software professionals with different native languages (English, Turkish, French and Taiwanese), different SE roles and different experiences.

External validity: External validity is concerned with the extent to which the results of this study can be generalized [81]. In order to decrease the effect of possible dominant participant number in a specific sector due to authors' previous and current work experiences' network (i.e., defense & aerospace, consumer electronics, academia), the survey has been distributed to embedded software professionals via various social network sites in all around the world for different industrial sectors. Therefore, we have done our best to reach the subjects with a variety of different backgrounds representative for the embedded software industry. Our sample size is quite high compared to previous surveys. While we did our best to achieve an even geographical distribution, the samples were mostly based from Europe (66%), followed by Asia (17%) and then the Americas (14%). Due to researchers' location, ~40% of respondents are from Turkey, which may have led to bias in the results. Nevertheless, note that we used non-probabilistic sampling design and thus external validity is limited. To address this, we reported demographic information of the participants and companies covered in our study, and therefore the readers will be able to evaluate the applicability in different contexts.

Conclusion validity: Conclusion validity of a study deals with whether correct conclusions are reached through rigorous and repeatable treatment [81]. This study was designed by one author, who has both researcher and practitioner hat and two other researchers from two different institutions; therefore the risk for "fishing" on the results is reduced. We attempted to conclude, qualitatively, that the modeling approaches in embedded software industry have economics and organizational aspects as well as purely technical concerns. For each RQ, we attempted to reduce the bias by seeking support from the statistical results. Although we collected data from different sources (different countries, different industrial sectors, different SE roles, etc.), we, clearly, do not have any intentions to generalize our findings to all over the embedded software world since these results depend the company and practitioner demographics. Nevertheless, we reported demographic information of the participants and companies covered in our study, and therefore the readers will be able to evaluate the applicability in different contexts. Moreover, to increase transparency, the raw survey data is made available online [62] for other researchers to validate and replicate; hence, all the conclusions that we drew are strictly traceable to data. Furthermore, we improved the reliability of our survey using pilot studies prior to the survey execution.

6. Conclusion

With the help of this study, the state-of-the-practice of software modeling and MDE was better understood by identifying to what degree, why and how it is used in embedded software industry with its possible challenges and its benefits. By this way, both embedded software professionals and also researchers could benefit from our results, which would influence not only aspects related to software-intensive embedded systems development, but also the system-level design.

Different SE roles use software modeling approaches in varying degrees (e.g., from informal sketches to formal models). Our study showed that 11% of respondents do not use any software modeling approaches (neither informal nor formal); whereas the remaining 89% is somehow (partially or fully) using it in their SDLC.

This study also showed that "Sketch/No formal modeling language" is widely used in the embedded software industry (i.e.*, the second most frequently selected response after UML usage*) and this finding revealed that the formality of the modeling language is not very important while benefitting from modeling for different purposes. The formality of modeling is important when there is an auto-generation of some software artifacts (e.g., code, document or test scripts); on the other hand, for communication or understanding, this is not so crucial. We observe that the purpose and the category of software modeling (i.e., sketch, model-based or model-driven) are strongly related with the medium used. If there is no auto-generation of some software artifacts, analog media like paper is enough for communication or understanding. At that sense, our study showed that, not surprisingly, by providing modeling tools (for both sketch/model-based and model-driven) and archiving diagrams easier as being digital, PC is the most used medium. However, we think that in the near future tablet/smartphone usage ratio might increase as it provides more mobility than PC while modeling.

We observed that model-driven code generation (Q27) is an important aspect and if code generation is synchronized with other artifacts (e.g., document, test driver), the benefits are maximized as in [4]. On the other hand, the results (Q27) also showed that model-based/driven testing makes easier to develop and execute test cases by also supporting test automation via test scripts; however, although it helps to start to test and its design earlier; it requires significant additional time to model and validate them.

We also observed that the "cross-factor" correlations among the results are interesting. Some modeling languages or diagrams are specific to some sectors or their usage ratio is different depending on their purposes and challenges. For example, although the dominant modeling language is UML in all sectors; specific modeling language for target sectors (e.g., AADL for "Defense & Aerospace" and Markov Chain Markup Language for "Consumer Electronics") are interesting results as reported in Section 4.5.

The survey showed that the embedded software professionals use modeling approaches in varying degrees (e.g., either as an informal sketch or more formalized model) with different constrains depending on their needs. All of the usages could be effective depending on the software modeling characteristics in embedded software industry, but what are these significant characteristics? Based on the results of the survey and a conceptual model of software modeling usage, we have already identified these characteristics and the relations between them [26]. Then, we focused to fill one major part of the gap in the existing literature by identifying and defining modeling approach patterns in embedded software industry. In order to improve what we found out from this survey result (e.g., quantitative data), we conducted a series of semi-structured interviews over eight months with 53 embedded software professionals to get more personalized, qualitative data [66]. Based on these findings, we created a characterization model, which identifies and defines a modeling stakeholder's pattern and culture as common-sense practices by presenting what the similar profiles are doing while modeling (via the database constructed with survey data presented in this study) [78]. This characterization model is the first wide-coverage model of software modeling characteristics for embedded software development projects built on extensive input from the industry.

Acknowledgments

The authors would like to thank all embedded software professionals, who contributed to this survey.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.sysarc.2018.09.007.

References

[1] C.J. Ebert, J. Capers, Embedded software: facts, figures, and future, IEEE Comput. Soc. 42 (2009) 42–52.
[2] J. Schäuffele, T. Zurawka, Automotive Software Engineering: Principles, Processes, Methods, and Tools, SAE International, 2005.
[3] Y. Yin, B. Liu, H. Ni, Avionics embedded software modeling based on time-constrained transition equivalence class, Adv. Sci. Lett. 5 (2012) 844–847.
[4] D. Akdur, V. Garousi, Model-driven engineering in support of development, test and maintenance of communication middleware: an industrial case-study, in: Proceedings of the *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 2015.
[5] L. Jóźwiak, Advanced mobile and wearable systems, Microprocess. Microsyst. 50 (2017) 202–221.
[6] M.A. Vega-Rodríguez, Design space exploration of embedded systems: a view from diverse domains, J. Syst. Archit. 59 (2013) 1113–1114.
[7] M. Broy, Challenges in automotive software engineering, in: Proceedings of the 28th International Conference on Software engineering, Shanghai, China, 2006.
[8] J. Rushby, New challenges in certification for aircraft software, in: Proceedings of the Embedded Software (EMSOFT), 2011.
[9] C. Walls, Embedded Software, Second ed., Newnes, Oxford, 2012.

[10] A. Gokhale, D.C. Schmidt, B. Natarajan, J. Gray, N. Wang, Model driven middleware, Middleware for Communications, Wiley, 2004.
[11] L. Jóźwiak, S.-A. Ong, Quality-driven model-based architecture synthesis for real–time embedded SoCs, J. Syst. Archit. 54 (2008) 349–368.
[12] W.J. Dzidek, E. Arisholm, L.C. Briand, A realistic empirical evaluation of the costs and benefits of UML in software maintenance, IEEE Trans. Softw. Eng. 34 (2008) 407–432.
[13] E. Linehan, S. Clarke, An aspect-oriented, model-driven approach to functional hardware verification, J. Syst. Archit. 58 (2012) 195–208.
[14] N.A. Karagoz, O. Demirors, Conceptual modeling notations and techniques, Conceptual Modeling for Discrete-Event Simulation, CRC Press, 2010.
[15] A. Dikici, O. Turetken, O. Demirors, Factors influencing the understandability of process models: a systematic literature review, Inf. Softw. Technol. 93 (2018) 112–129.
[16] D. Akdur, E. Özpolat, T. Başıbüyük, Model driven engineering of communication protocol artifact with design pattern usage in distributed and real-time embedded systems: an industrial experience, Int. J. Eng. Sci. Appl. 1 (2017) 91–98.
[17] M. Petre, UML in practice, in: Proceedings of the 35th International Conference on Software Engineering (ICSE), 2013, pp. 722–731.
[18] R. Heldal, P. Pelliccione, U. Eliasson, J. Lantz, J. Derehag, J. Whittle, Descriptive vs. prescriptive models in industry, in: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, France, 2016.
[19] M. Broy, S. Kirstan, H. Krcmar, B. Schätz, What is the benefit of a model-based design of embedded software systems in the car industry? in: Emerging Technologies for the Evolution and Maintenance of Software Models, IGI Global, 2011, pp. 343–369.
[20] L.T.W. Agner, I.W. Soares, P.C. Stadzisz, J.M. Simão, A Brazilian survey on UML and model-driven practices for embedded software development, J. Syst. Softw. 86 (2013) 997–1005.
[21] G. Liebel, N. Marko, M. Tichy, A. Leitner, J. Hansson, Assessing the state-of-practice of model-based engineering in the embedded systems domain, in: Model-Driven Engineering Languages and Systems, 8767, Springer International Publishing, 2014, pp. 166–182.
[22] B. Selic, S. Gérard, Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE: Developing Cyber-Physical Systems, Morgan Kaufmann, 2013.
[23] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice, Synthesis Lectures on Software Engineering, 1, Morgan & Claypool, 2012.
[24] J. Cabot, 2009. Available: http://modeling-languages.com/relationship-between-mdamdd-and-mde/.
[25] 2016. Available:https://www.youtube.com/watch?v=9qPbGksB3d4.
[26] D. Akdur, O. Demirörs, V. Garousi, Characterizing the development and usage of diagrams in embedded software systems, in: Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, Austria, 2017.
[27] J. Hutchinson, J. Whittle, M. Rouncefield, Model-driven engineering practices in industry: social, organizational and managerial factors that lead to success or failure, Sci. Comput. Program. 89 (Part B) (2014) 144–161.
[28] P. Liggesmeyer, M. Trapp, Trends in embedded software engineering, Softw. IEEE 26 (2009) 19–25.
[29] R. France, B. Rumpe, Model-driven development of complex software: a research roadmap, in: Proceedings of the Future of Software Engineering, 2007.
[30] B.P. Douglass, Real Time UML: Advances in the UML for Real-time Systems, Addison-Wesley, 2004.
[31] G.M. Nicolescu, Model-Based Design for Embedded Systems, CRC Press, 2009.
[32] S. Gerard, J.-P. Babau, J. Champeau, Model Driven Engineering for Distributed Real-Time Embedded Systems, Wiley-IEEE Press, 2010.
[33] , Model driven architecture – foundations and applications, in: Proceedings of the ECMDA, The Netherlands, 2009.
[34] Eclipse.org. (2012). EclipseCon Available: www.eclipsecon.org/2012.
[35] M. Guttman, J. Parodi, Real-life MDA: Solving Business Problems with Model Driven Architecture, Elsevier/Morgan Kaufmann Publishers, Amsterdam; Boston, 2007.
[36] D. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, John Wiley & Sons Inc, 2002.
[37] T. Weigert, F. Weil, Practical experiences in using model-driven engineering to develop trustworthy computing systems, in: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006.
[38] G. Karsai, S. Neema, D. Sharp, Model-driven architecture for embedded software: a synopsis and an example, Sci. Comput. Program. 73 (2008) 26–38.
[39] H. Espinoza, D. Cancila, B. Selic, S. Gérard, Challenges in combining SysML and MARTE for model-based design of embedded systems, in: R. Paige, A. Hartman, A. Rensink (Eds.), Model Driven Architecture – Foundations and Applications, 5562, Springer, Berlin Heidelberg, 2009, pp. 98–113.
[40] G. Liebel, N. Marko, M. Tichy, A. Leitner, J. Hansson, Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice, Softw. Syst. Model. 17 (2018) 91–113.
[41] M. Grossman, J.E. Aronson, R.V. McCarthy, Does UML make the grade? Insights from the software development community, Inf. Softw. Technol. 47 (2005) 383–397.
[42] B. Dobing, J. Parsons, How UML is used, Commun. ACM 49 (2006) 109–113.
[43] C.F.J. Lange, M.R.V. Chaudron, J. Muskens, In practice: UML software architecture and design description, Softw. IEEE 23 (2006) 40–46.
[44] J. Peneva, S. Ivanov, G. Tuparov, Utilization of UML in Bulgarian SME – possible training strategies, Commun. Cognit. Artif. Intell. 23 (2006) 83–88.
[45] A. Nugroho, M.R. Chaudron, A survey into the rigor of UML use and its perceived impact on quality and productivity, in: Proceedings of the ACM–IEEE Empirical Software Engineering and Measurement (ESEM), 2008, pp. 90–99.
[46] P. Fitsilis, V.C. Gerogiannis, L. Anthopoulos, Role of unified modelling language in software development in Greece – results from an exploratory study, Softw. IET 8 (2014) 143–153.
[47] A. Forward, T.C. Lethbridge, Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals, in: Proceedings of the International Workshop on Models in Software Engineering, Leipzig, Germany, 2008, pp. 27–32.
[48] J. Hutchinson, J. Whittle, M. Rouncefield, S. Kristoffersen, Empirical assessment of MDE in industry, in: Proceedings of the 33rd International Conference on Software Engineering, Waikiki, Honolulu, HI, USA, 2011, pp. 471–480.
[49] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, G. Reggio, Preliminary findings from a survey on the MD state of the practice, in: Proceedings of the Empirical Software Engineering and Measurement (ESEM), 2011, pp. 372–375.
[50] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, G. Reggio, Relevance, benefits, and problems of software modelling and model driven techniques – a survey in the Italian industry, J. Syst. Softw. 86 (2013) 2110–2126.
[51] T. Gorschek, E. Tempero, L. Angelis, On the use of software design models in software development practice: an empirical investigation, J. Syst. Softw. 95 (2014) 176–193.
[52] F. Shull, J. Singer, D.I.K. Sjoberg, Guide to Advanced Empirical Software Engineering, Springer-Verlag, Inc., New York, 2007.
[53] R.M. Groves, F.J. Fowler, M.P. Couper, J.M. Lepkowski, E. Singer, R. Tourangeau, Survey Methodology, Second, John Wiley & Sons, 2009.
[54] J. Linaker, S.M. Sulaman, R. Maiani de Mello, M. Höst, and P. Runeson, "Guidelines for Conducting Surveys in Software Engineering," 2015.
[55] V.R. Basili, G. Caldiara, H.D. Rombach, The goal question metric approach, Encyclopedia of Software Engineering, Wiley, 1994.
[56] T. Punter, M. Ciolkowski, B. Freimut, I. John, Conducting on-line surveys in software engineering, in: Proceedings of the International Symposium on Empirical Software Engineering, 2003, pp. 80–88.
[57] T.R. Lunsford, B.R. Lunsford, The research sample, part I: sampling, J. Prosthet. Orthot. 7 (1995) 105–112.
[58] V. Garousi, A. Coşkunçay, A. Betin-Can, O. Demirörs, A survey of software engineering practices in turkey, J. Syst. Softw. 108 (2015) 148–177.
[59] L. Wallace, M. Keil, A. Rai, Understanding software project risk: a cluster analysis, Inf. Manage. 42 (2004) 115–125.
[60] D. Akdur, V. Garousi, and O. Demirörs, "MDE in embedded SW industry – survey form (questions)," doi: 10.6084/m9.figshare.4262978, 2015, (Last accessed: 27 Nov ember 2016).
[61] P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, Wiley Publishing, 2012.
[62] D. Akdur, V. Garousi, and O. Demirörs, "MDE in embedded SW industry – raw survey data," doi: 10.6084/m9.figshare.4262972, 2015, (Last accessed: 27 November 2016).
[63] D. Akdur, V. Garousi, and O. Demirörs, "MDE in embedded software industry, Technical Report METU II-TR-2015-55, doi : 10.6084/m9.figshare.4262990, 2015, (Last accessed: 27 November 2016).
[64] D.A. Garvin, Managing Quality: The Strategic and Competitive Edge, Free Press, 1988.
[65] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang, What industry needs from architectural languages: a survey, IEEE Trans. Softw. Eng. 39 (2013) 869–891.
[66] D. Akdur, O. Demirörs, B. Say, Towards modeling patterns for embedded software industry: feedback from the field, in: Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prag ue , Czech Republic, 2018.
[67] T. Ho-Quang, R. Hebig, G. Robles, M.R.V. Chaudron, M.A. Fernández, Practices and perceptions of UML use in open source projects, in: Proceedings of the 39th IEEE/ACM International Conference on Software Engineering (ICSE), 2017.
[68] , MDD Maturity Models, 2014 Last accessed: September 2016. .
[69] A.G. Kleppe, J.B. Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley Professional, 2003.
[70] I. Sommerville, Software Engineering, Addison Wesley, 2010.
[71] J. Whittle, J. Hutchinson, M. Rouncefield, H. Burden, R. Heldal, A taxonomy of tool-related issues affecting the adoption of model-driven engineering, Softw Syst. Model. 16 (2017) 313–331 May 01.
[72] J. Kramer, Is abstraction the key to computing? Commun. ACM 50 (2007) 36–42.
[73] D. Thomas, MDA: revenge of the modelers or UML utopia? Softw. IEEE 21 (2004) 15–17.
[74] C. Kobryn, Will UML 2.0 be agile or awkward? Commun. ACM 45 (2002) 107–110.
[75] D. Dori, Why significant UML change is unlikely, Commun. ACM 45 (2002) 82–85.
[76] D. Akdur, V. Garousi, O. Demirörs, Cross-factor analysis of software modeling practices versus practitioner demographics in the embedded software industry, in: Proceedings of the 6th Mediterranean Conference on Embedded Computing (MECO), Montenegro, 2017.
[77] R.N. Taylor, N. Medvidovic, E.M. Dashofy, Software Architecture: Foundations, Theory, and Practice, Wiley Publishing, 2009.
[78] D. Akdur, Modeling patterns and cultures of embedded software development projects Thesis, Doctor of Philosophy (Ph.D.), Information Systems, Middle East Technical University (METU), 2018.
[79] S. Akayama, S. Kuboaki, K. Hisazumi, T. Futagami, T. Kitasuka, Development of a modeling education program for novices using model-driven development, in: Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education, Tampere, Finland, 2013.
[80] S. Flint, H. Gardner, C. Boughton, Executable/translatable UML in computing education, in: Proceedings of the Sixth Australasian Conference on Computing Education, 30, Dunedin, New Zealand, 2004.
[81] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer, Berlin Heidelberg, 2012.

Deniz Akdur is a Lead Software Engineer at ASELSAN, Inc., which is the largest Defense & Aerospace company of Turkey. Prior to that, he worked as a Software Architect for different companies in both Turkey and United Kingdom in Consumer Electronics sector. He received his BSc degree in Computer Science from Bilkent University and MSc & Ph.D. degrees in Information Systems from Middle East Technical University (METU), Ankara, Turkey. His specialties and research interests include software-intensive embedded systems, software engineering, model-driven engineering, technology acceptance, software quality management and industry-academia collaborations.

Vahid Garousi is an Associate Professor of Software Engineering in Wageningen University, the Netherlands. His research interests in software engineering include: software testing and quality assurance, model-driven development, software maintenance and empirical software engineering. He is also passionate about of development of "scientific" and engineering software (e.g., software for oil pipelines or embedded controllers). Since 2002, he has also worked as a consultant for software companies in Canada, Turkey and the Netherlands, helping them in various areas of software engineering.

Onur Demirors is a Professor of Computer Engineering at the Izmir Institute of Technology (ceng.iyte.edu.tr) and the strategy director of Bilgi Grubu Ltd. (www.bg.com.tr). He has recently joined UNSW for his sabbatical. His current research focuses on decentralized modeling and organizational change, software measurement, and management. He has leaded major research and application projects on developing improvement and modeling techniques, on establishing and implementing modeling approaches for organizations and on establishing measurement infrastructures for software organizations. He has leaded application projects for dozens of companies to improve their processes, to establish their measurement infrastructures, to create organizational knowledge structures and to identify their software needs. He continues to teach on decentralized modeling, event based systems, software project and quality management, software measurement and innovative software development approaches.