Software and Systems Modeling (2021) 20:447–467
https://doi.org/10.1007/s10270-020-00810-9

REGULAR PAPER

Modeling cultures of the embedded software industry: feedback from the field

Deniz Akdur[1,2] · **Bilge Say**[3] · **Onur Demirörs**[4]

Received: 25 January 2019 / Revised: 29 April 2020 / Accepted: 30 May 2020 / Published online: 25 June 2020

Abstract
Engineering of modern embedded systems requires complex technical, managerial and operational processes. To cope with the complexity, modeling is a commonly used approach in the embedded software industry. The modeling approaches in embedded software vary since the characteristics of modeling such as purpose, medium type and life cycle phase differ among systems and industrial sectors. The objective of this paper is to detail the use of a characterization model MAPforES ("Modeling Approach Patterns for Embedded Software"). This paper presents the results of applying MAPforES in multiple case studies. The applications are performed in three sectors of the embedded software industry: defense and aerospace, automotive and transportation, and consumer electronics. A series of both structured and semi-structured interviews with 35 embedded software professionals were conducted as part of the case studies. The characterization model was successfully applied to these cases. The results show that identifying individual patterns provides insight for improving both individual behavior and the behavior of projects and organizations.

Keywords Software modeling · Embedded software · Modeling patterns and cultures · Characterization model · Case study

1 Introduction

Analysis, design, implementation and testing of software for modern embedded systems are constrained across different dimensions of performance and quality [1, 2]. Moreover, the increasing number of system components with different functionalities incorporated into a single system requires seamless integration of many hardware and software systems, which makes the development of embedded software more challenging [3]. Software modeling enables practitioners to manage system complexity by helping engineers work at higher levels of abstraction. It also facilitates communication [4, 5].

The modeling approaches in embedded software vary depending on many factors, including purpose, life cycle phases, roles and sector [6]. At one extreme, some stakeholders (e.g., project managers or systems engineers) use software modeling informally. In this approach, diagrams sketched on paper are used for purposes of communication. In such cases, the emphasis is on communication rather than formal specification. These diagrams may be either discarded or become inaccurate since they are not kept along with the source code up to date [7]. At the other extreme, a model may be transformed into a program using automated generation of code from models. In such cases, the models have a longer lifespan and should be archived. It is also frequently observed that different departments within the same company might use different modeling approaches for different purposes in different phases of the software development life cycle (SDLC) [8].

For software teams in an embedded software development project, deciding when to model, at what level of abstraction

Communicated by Constance Heitmeyer.

✉ Deniz Akdur
denizakdur@aselsan.com.tr

Bilge Say
bilge.say@atilim.edu.tr

Onur Demirörs
onurdemirors@iyte.edu.tr

[1] ASELSAN Inc., Ankara, Turkey

[2] Department of Information Systems, Informatics Institute, Middle East Technical University (METU), Ankara, Turkey

[3] Department of Software Engineering, Atilim University, Ankara, Turkey

[4] Department of Computer Engineering, İzmir Institute of Technology, Izmir, Turkey

and with how much modeling rigor[1] are challenging and frequently asked questions. Moreover, different modeling stakeholders might spend time and money to investigate different modeling practices that present choices in modeling processes, notations and modeling tools. Hence, projects frequently waste resources while trying out unfit or not yet mature modeling practices, which potentially increases costs.

The current literature lacks an approach that provides systematic modeling guidance for different stakeholders in embedded software development projects. Our view is that there is a need to identify the relations between the characteristics of modeling (e.g., modeling rigor, purpose, medium type used and SDLC phase) to fill this gap. Once these relations are known, a potential approach to address this gap is to identify, define and use "modeling patterns and cultures." Such an approach is analogous to the characterization model, defined and tailored for Software Process Improvement (SPI) (e.g., Software Sub-Cultures [9]). We propose a "characterization model," which describes how software modeling characteristics in an embedded software development project can assist stakeholders in identifying a suitable modeling approach based on these characteristics. The characterization model may also allow a stakeholder to compare and contrast their modeling approach with the approaches of other embedded software professionals with the similar profiles (e.g., role, industrial sector).

Accordingly, we identify and define modeling patterns and cultures in the embedded software industry using a characterization model called MAPforES (Modeling Approach Patterns for Embedded Software). The model enables a stakeholder to identify a commonsense modeling approach, by utilizing the modeling community's experiences. This characterization model not only identifies patterns and cultures of the modeling stakeholder, but also guides process and tool improvements for modeling by describing a set of commonsense industrial practices in embedded software development [10].

The objective of this paper is to report the results of applying our characterization model in different settings. The empirical study reported here is based on three case studies. In the case studies, we applied the model and conducted a series of both structured and semi-structured interviews. The conducted case studies operate in different sectors of the embedded software industry. The first study took place in a defense and aerospace organization, which is a global provider of advanced radar systems. The second study took place in an automotive and transportation organization, which develops components for railways and roads. Finally, the third study took place in a consumer electronics organization, which focuses on TV products. The case studies took place over two months and involved 35 embedded software professionals with different software engineering (SE) roles (e.g., from software developer to tester and systems engineer to project manager).

The remainder of this paper is organized as follows: Sect. 2 presents background information by summarizing the existing literature on software modeling patterns and by introducing the characterization model. Section 3 discusses the application of MAPforES by presenting our research process, findings and the potential threats to the studies' validity. Finally, Sect. 4 presents an overall conclusion and states future work directions.

2 Background

This section first summarizes the existing literature on software modeling patterns and categories. Then, as a precursor to its application, we introduce our characterization model.

2.1 Existing literature on software modeling patterns

Different definitions of "pattern" appear in the literature. It is defined as a "consistent and recurring characteristic that helps in the identification of a phenomenon or problem" [11]. In the software engineering literature, the "pattern" concept refers to proven solutions to recurrent design challenges, e.g., "software design patterns" [12, 13]. In contrast, Weinberg defines "culture" as a shared set of beliefs and goals, which shapes behaviors and activities [9]. In this study, a "modeling pattern" consists of specific characteristics of modeling (e.g., purpose, medium type used, modeling language type, SDLC phase, etc.), which helps identify the stakeholder's modeling practices, whereas a "modeling culture" is viewed as a particular group of "modeling patterns" and consists of frequently observed grouping of these characteristics in practice.

Software Process Improvement maturity models such as Capability Maturity Model Integration (CMMI) and Software Process Improvement and Capability dEtermination (SPICE) are based on similar underlying concepts that focus on organizational change [14, 15]. These models, however, do not directly provide an evaluation of modeling characteristics, and they target organizations, not individuals, for improvement.

In the literature, there are few research studies on the modeling patterns and categories. Kleppe, Warmer and Bast classify modeling usage as maturity levels by focusing on one characteristic of modeling (i.e., "modeling formality")

[1] Modeling rigor is the formality of modeling language (e.g., informal or formalized), which affects software modeling usage in varying degrees.

[16]. According to Kleppe et al., there are six Modeling Maturity Levels (MMLs) in software development projects; in each there are different types of modeling usage based on "modeling rigor." However, our empirical evidence shows that different characteristics of the modeling process need not force modeling stakeholders raise the maturity level based on a single dimension such as rigor. Instead, different modeling characteristics (such as purpose or SDLC phase) are related to different notations, tasks and roles.

In the second study, Petre focused solely on Unified Modeling Language (UML) usage categories for software developers [17]. The results showed that there are different models of what "using UML" means in practice. Five categories were identified ranging from "No UML use" to "Wholehearted use." In addition, they found that top-down UML usage is organizational (e.g., tools and culture change) and deeply embedded in all phases of the SDLC. According to the results, the majority of those interviewed in the study do not use UML at all (i.e., 70%), and those who do, use UML selectively and often informally (i.e., 22%) [17]. However, in embedded software development, many stakeholders (e.g., from software developer to tester and systems engineer to project manager) use Domain-Specific Languages (DSL), which are claimed to have more potential for model-driven development (MDD) and model-driven engineering (MDE) than UML, since UML, a graphical modeling language, is designed for documentation—not for implementation [18].

To summarize, no study exists that assigns modeling patterns of individuals and/or projects multiple dimensions independent of modeling notations. MAPforES was developed to satisfy this need. It is a complementary approach to organizational Software Process Improvement approaches and enables commonsense modeling approaches at all levels, individual, project and organization.

2.2 Overview of the characterization model: MAPforES

The MAPforES model describes software modeling characteristics for embedded software development. The characteristics include attributes such as "purpose," "medium type," "archivability," "modeling language, if any," "SDLC phase" and are organized into modeling approach patterns and cultures. MAPforES defines the characteristics in an embedded software development project and helps modeling stakeholders select a suitable modeling approach. It also allows a stakeholder to compare and contrast how he or she uses models with how embedded software professionals with similar profiles (e.g., role, industrial sector) use models [10]. The construction of the MAPforES model is described next.

The MAPforES model was constructed in a three-phase study. As a first step, a global survey was conducted to understand the state of practice of modeling in the embedded software industry [19]. The survey showed that embedded software professionals use modeling approaches to varying degrees (e.g., either as an informal sketch or a formalized model) with different modeling characteristics in different embedded software development projects. Based on the results of the survey [20, 21], the relations between modeling characteristics were investigated [6], and the preliminary version of the modeling approach pattern set was created [22]. Since some hidden patterns[2] might emerge, which could not be found out from an analysis of survey data, there was a need to validate and improve this preliminary version. Then, a series of semi-structured interviews were conducted over 8 months with 53 embedded software professionals across a variety of target industrial sectors and roles [23]. As a result, the number of patterns was increased to 12, as shown in Table 1.

After defining modeling approach patterns, six modeling cultures in embedded software development projects were identified: *None, Performed, Formalized, Archived, Prescripted and Auto-generated* [10]. Thus, a modeling culture (i.e., a particular group of modeling approach patterns) consists of different characteristics of software modeling. In this categorization, a "higher" culture can use the characteristics of the "lower" cultures, and the modeling stakeholder might, if necessary, apply the modeling practices of the stakeholders' lower-level patterns, but not vice versa. For example, a modeling stakeholder, who is at pattern 3.3, can use a medium type such as paper along with digital ones (e.g., modeling tools on a personal computer (PC)), and sketches without any modeling rigor as if being at pattern 1.1. Therefore, a "higher" culture does not necessarily entail a more "correct" or more "mature" use of modeling although a change to a "higher" culture might allow the stakeholder to more effectively use software modeling with possibly some extra costs and challenges. Since the cultures shown are based on all modeling characteristics of the individual stakeholder (e.g., purpose, task/responsibility, SDLC phase, etc.), this approach also differs from maturity models based on organizational concepts such as MML (Sect. 2.1) since it focuses on individual practices. For more details on the derived cultures, see [10].

After identifying the patterns and cultures with their characteristics, we started to create MAPforES. The characterization model has two main components: (1) the decision tree, which helps the stakeholder walk through the stakeholder's pattern and culture, and (2) the recommendation component, which

[2] "Hidden patterns" are the patterns of the participants, who do not know exactly their software modeling characteristics (especially their modeling rigor); hence their modeling patterns could not be identified by only survey data analysis. These participants are unaware as to whether they use software modeling or MDE (e.g., unaware of modeling or unaware of MDE as depicted in Table 1).

Table 1 Modeling approach patterns

Main pattern	Modeling approach patterns		
Model driven	3.3	With DSL-like	Purpose of the modeling includes "code generation" or "test case generation (e.g., model-based testing (MBT))"
	3.2	Without DSL-like	
	3.1	Limited	Only with "documentation generation," "model simulation" or "model to model transformation" purpose
	3.x	Unaware of MDE	Purpose of the modeling includes any MDE-specific purpose, but the stakeholder is unaware of MDE usage
Model based	2.2	Prescriptive	SDLC phase where modeling is used includes "implementation" or "testing"
	2.1	Descriptive	SDLC phase does not include "implementation or testing"
Sketching	1.3	Archived	Purpose of the modeling includes "documenting analysis and design" Media type used: "analog media" such as paper/whiteboard used more often than "digital media"
	1.2	Selective	Casually and informally with some formalized modeling language (most probably, UML elements) Modeling Language set includes sketch and any formalized modeling language (e.g., UML or DSL-like)
	1.1	Ad hoc	Purpose of the modeling includes only "understanding" or "communication" Only pen and paper/free format (e.g., without any formalized modeling language, e.g., UML) Medium type used while modeling is only paper or whiteboard
	1.x	Unaware of modeling	The stakeholder is unaware of modeling although there is some kind of modeling (especially sketching)
None	0.1	Bad experienced	Not using any modeling approach
	0.0	Not experienced	

"With DSL-like" means that the modeling language set of the stakeholder includes any kind of Domain-Specific Language (DSL) (e.g., any DSL [provided by tool provider or their own design] such as AUTOSAR, AADL, EAST-ADL or any UML profiles, which provides a generic extension mechanism for customizing UML diagrams such as MARTE, SysML, SoaML, any BPML or MATLAB Modeling Utilities, etc.)

As terminology, "bad experienced" pattern indicates the embedded software professionals, who don't use any kind of modeling due to disappointing and insufficient experiences of software modeling

describes what stakeholders with similar profiles in the embedded domain are doing while modeling. During this creation process, we first derived a decision tree mechanism, excluding "hidden patterns" ([10] for a detailed derivation of the tree). Feedback from 14 expert software professionals was analyzed prior to finalizing the decision tree. Accordingly, first, MAPforES takes the modeling characteristics of the stakeholder as input. Depending on these characteristics, the current modeling pattern and culture are identified. Moreover, based on these characteristics, MAPforES presents what other stakeholders with similar profiles are doing while modeling as a set of commonsense industrial practices (the data used here were from the database constructed with survey data and with findings of semi-structured interviews [24]). By comparing the similar profiles in the database with her or his input, the stakeholder learns what their competitors do in similar situations. During this process, the stakeholder gets answers to strategically important questions such as the necessary modeling approaches, languages, tools, etc. [10]. Therefore, besides identifying and defining the current pattern and culture, MAPforES captures widespread modeling practices in the embedded software industry and refers to them as commonsense practices.

The application of the characterization model, the main objective of this study, is presented next.

3 Application of MAPforES

In this section, we describe the research methodology as well as the potential threats to validity and discuss our findings. To determine the benefits of MAPforES model in practice, we performed three case studies in two different companies. One company includes two different organizations, each operating in a different subsector of the embedded software industry. Most data were gathered in applying the model and by conducting a series of structured and semi-structured one-to-one interviews to capture detailed context information and observations. The interviews were conducted over 2 months with 35 embedded software professionals. The first section describes the research methodology, and the second presents the research process and findings. The data for each case study with participants' answers are available in [25], which includes the actual documents and all other collected evidence.

3.1 Research methodology

The research methodology undertaken in the case studies provides triangulation of both quantitative and qualitative data in accordance with empirical research principles [26–29].

The goal of these case studies is to apply and evaluate the usefulness of the MAPforES model by identifying stakeholder's modeling processes and commonsense practices of embedded software development in different industrial sectors. "Applying" the MAPforES model refers to the actual methodology used to carry out the case studies, that is, the characterization of each relevant individual's modeling experience with respect to MAPforES using a series of meetings, questionnaires and interviews in an organization developing embedded software. This process makes each modeling stakeholder aware of where she or he stands in terms of the MAPforES and what else exists in MAPforES that might have possible benefits.

The interviewer acts as an evaluator, who analyzes the participants' modeling characteristics and derived recommendations (e.g., commonsense and popular modeling practices like languages, tools, etc.) based on matching demographics. Based on the above goal, the following research questions (RQs) are raised to test the hypotheses in practice:

RQ1 In what ways does MAPforES reflect or fail to reflect a stakeholder's current modeling pattern and culture?
RQ2 How does the stakeholder evaluate MAPforES's usefulness and conceptual insightfulness?

To address these RQs, an evaluation form (Appendix of [10]) is used to evaluate the result of the model based on a set of validation criteria [30].

3.2 Research process

The following main phases are applied and explained in more detail in the corresponding sections shown in Table 2.

3.2.1 Design

The case study protocol is "flexible" since it includes both structured and semi-structured parts [28]. We use interviews as a main source of evidence for two reasons: (1) to observe stakeholder's demographics and modeling practices to understand the characteristics of software modeling (i.e., the structured part) and (2) to understand the personal experience of stakeholders and confirm them by face-to-face in-depth analysis and by direct observations (i.e., the semi-structured part).

To prevent misinterpretation during data collection, a presentation on the MAPforES model was given on-site as the first step of the study. The presentation included information about the study, the model and the terminology used. In compliance with multiple case study methodology in industry [31], we used several sources of data, created a case study database and validated our data as explained below.

Questionnaires were created and used for the structured part of the research. Before the company visits, the data to be gathered are summarized on a questionnaire (Appendix of [10]) and a set of "evaluator notes" are prepared. The questionnaire is filled out after the first round of the interview. The evaluator takes notes on all given responses. Thus, the interview elicits both closed-ended and open-ended answers. During the first round of the interview, the questionnaire obtains all necessary inputs to MAPforES by eliciting necessary modeling characteristics such as *purpose*, *medium type*, *modeling language, SDLC phase* and *stakeholder profile* (e.g., university degree, role, target sector of the product) [10]. The participant answers this first part without any interaction with the evaluator. During the second part, which is conducted face to face, the responses of each participant to the questionnaire are checked to identify any misunderstanding or any missing critical information (e.g., wrong data for modeling practices caused by "unawareness" of modeling characteristics). The plan is to complete the interview process (e.g., data collection) during the first 2 days on the site.

After data collection, all answers are analyzed and MAPforES is applied to the participant's characteristics of software modeling. After this stage, the evaluator sends two forms to the participants via email. The first form summarizes the interview results (Appendix of [10], evaluator notes), and the second form is used to evaluate the model's usefulness to each participant (Appendix of [10], evaluation form). The evaluation form is based on the evaluation criteria given in Table 3, which was adopted from [30, 32]. This evaluation form, which contains five RQs, is shown in Table 4.

The last day on the site starts with a face-to-face meeting with the participants, who want to elaborate on the results of the model and suggestions sent by email. The availability of such an interview slot is announced to all participants and is performed optionally upon their request. After this session, all participants are gathered together in the meeting room for the closing meeting. In that session, all general results on the charts, which include all participants' modeling patterns and cultures, with general recommendations (e.g., the set of common industrial practices), are presented. The agenda template for these activities is presented in Table 5.

3.2.2 Selecting the cases and data

Our plan was to organize the data based on different target domains (e.g., consumer electronics, defense and

Table 2 Multiple case study research process

	According to [27]	According to [28]	In this study
Design	Find/develop theory	High-level design Goal, RQs and model	Section 3.1 & theoretical framework and findings
	Design data collection process	Detailed design Preparation for data collection	Section 3.2.1
	Select cases		Section 3.2.2
Plan, collect	Conduct case study	Plan Data collection	Section 3.2.3
Analyze, report	Analysis/modify the model, if necessary Reporting		Section 3.3

Table 3 Validation criteria used in the evaluation strategy

Result validation	"This criterion investigates the opinion of the potential stakeholders about the model." Does the model produce expected and relevant results? It is concerned with the quality of the model with respect to its benefits
Utility validation	"This criterion investigates whether the model is useful." Does the model produce helpful results so that the model becomes useful?
Comparison validation	"This criterion investigates whether the model provides new insight and is better than what was available before." It is related to comparing with alternative approaches (if any)

Table 4 Evaluation questions to achieve validation criteria

#	Question	Addressed RQ	Validation
1	When you think about the presentation about "modeling patterns and cultures of embedded software development project," does the model really reflect your current modeling pattern and culture? Did this model produce expected and relevant results for you?	RQ1	Result
2	Do you think that the model is helpful? Please elaborate your answer	RQ1 and RQ2	Result and utility
3	Have you ever experienced or used such a model before? Do you think that this model is better than what was available before? Please elaborate your answer	RQ2	Comparison
4	Do you think that learning what your competitors (e.g., similar demographics) are doing while modeling might affect your future modeling practices? Please elaborate your answer	RQ2	Utility
5	Do you think that the recommendations, which the model gave you, are useful or not? Please elaborate your answer	RQ2	Utility

aerospace), different business models (e.g., market or contract driven) and different customers (e.g., private, public, internal) [28]. Therefore, based on differences instead of similarities, we selected our three cases and the data in Table 6. Notice that the participants in each case study work in the same software development project but have different software engineering roles.

Org 1 and Org 2 operate independently but within the umbrella of a larger organization in the same company, whose product portfolio includes radar and electronic warfare systems, weapon systems, air defense and missile systems, transportation, traffic and automation. The total number of employees working in research and development (R&D) engineering roles in this company is more than 3500. Org1 and Org2 develop products with state of the art software development techniques like agile methods, software product lines and reusable components. These organizations have been assessed to be at CMMI level 3.

Org 1 is a global provider of advanced radar systems serving both military and civilian markets. For this study, a radar software project was chosen for Case Study A (e.g., from defense and aerospace target sector). The size of a typical software development team in Org 1, which includes different software engineering roles, is 15–25 people. In Case Study A, 17 participants were interviewed. The participants covered all roles used in this project.

The second case study, Case Study B, included participants from the same organization, but from a different target sector: automotive and transportation. Org 2 designs, develops and builds custom solutions, subsystems and critical components for the mobility of vehicles on railways, roads and public networks. The size of a typical software

Table 5 Agenda for data collection, analysis and reporting process on the organization visit

2 days	~*1 h* Acquaintance and give a presentation about MAPforES and terminology used ~*30 min* Give the questionnaire separately, let them answer this structured part individually, but after completion do not collect the forms ~**hours (30 min per participant)* Collect the forms by validating/confirming what each participant gives as answers in the questionnaire. Take notes in the questionnaire form, collect evidence (In this semi-structured part, direct observations and improvization play a critical role) *For case study A: (17 participants) → 8,5 h For case study B: (10 participants) → 5 h For case study C: (8 participants) → 4 h
Break	
Aim Analyze the answers, evaluate them and apply the model	
Subtask1 Investigate the actual modeling pattern and culture of the stakeholder (via observation and interview) and according to the model (via model inputs)	
Subtask2 Present what the stakeholders with similar profiles are doing while modeling	
Subtask3 Give recommendations for commonsense modeling practices	
Subtask4 Email the results and evaluation form to the participants to evaluate the model	
3rd day	~*2 h* Interview with the participants, who want to meet individually about the results ~*2 h* Show the general results on the chart. Repeat validatory questions about the model and make them elaborate their answers for both individual and project results ~*10 min* Thank the participants and complete the session

Table 6 Case and data selection in multiple case study

Case	Organization	Target sector	Project type	Business model/customer	Interviewee size (software project team distribution)	
A	Org1	Defense & aerospace	Radar software	Contract driven/public & private	17	10 software developers | designers | architects 3 software testers 2 systems engineers 1 project manager 1 quality assurance engineer
B	Org2	Automotive & transportation	Bus software application	Contract-driven market/public & private	10	6 software developers | designers | architects 2 software testers 1 systems engineer 1 project manager
C	Org3	Consumer electronics	TV software	Market/private	8	5 software developers | designers | architects 2 software testers 1 project manager

development team in Org 2 is 5–10 people, and our case study (i.e., bus software application project) includes 10 practitioners, who covered all software engineering roles in the project. As stated above, both Case Studies A and B belong at the same company, but with different target sectors, a different business model, and different software modeling approaches and practices.

Org 3, a subgroup of one of the largest manufacturing companies in Turkey, operates in the consumer electronics sector and is a member of a consortium for several international R&D projects. The number of employees working in R&D roles in this company is about 800. For this study, a TV software project, which produced one of the Org 3's well-known products, was chosen for Case Study C. The size of a typical software development team in Org 3 is 5–10 people. In Case Study C, eight participants were interviewed, which covered all software engineering roles in the project. This software group's main specialty is developing innovative and popular products using agile programming.

3.2.3 Collecting evidence

The agenda template (Table 5) was applied in all three case studies. First, to introduce MAPforES, a presentation on "Modeling approach patterns and cultures of embedded software development projects" was given to all participants. Moreover, the evaluator informed all interviewees about the research before the interviews to obtain initial trust and to avoid unethical issues such as disclosing possible industrial secrets. This session took approximately 1 h 15 min and included a question and answer session.

Then, the questionnaire, the main data source for the first part of the interview, was distributed to the participants to obtain individual answers. The participants filled out the questionnaire alone, which took about 30 min. Then, one by one, the semi-structured, face-to-face part of this study was carried out. The aim of this session was to validate the participants' answers. To increase data consistency, besides the interviews, any extra source of information about modeling practices (e.g., any written material, medium used) is analyzed during this process. Direct observation of modeling practices also helped us understand a participant's daily use of modeling and to capture the details, which were not taken or clarified in the first round. Note that the interviews were performed without any voice recorder since there are some confidentiality regulations for Org 1 and Org 2; nor did the participants in Org 3 want a data recorder used. However, during this session, the evaluator took notes on the questionnaire to collect evidence and detected some hidden characteristics such as Domain-Specific Language (DSL) usage or sketching as ad hoc ([25] for the evaluator notes on the original questionnaire). Thus, several sources and cross-checking these data with the questionnaire (e.g., what the evaluator observed and learned during this semi-structured session) compensated for the lack of voice recording, thus improving the studies' conclusions.

For Case Study A, the first case study, after the analysis of collected data and reporting the results, the evaluator sent the evaluation form (Appendix of [10]) to participants to evaluate the model usefulness. In the email, which gave the results (e.g., the identification of modeling patterns and cultures and the suggestions), the participants were asked to fill out these forms before the closing meeting. However, since not all participants completed this form before the session, in the second and third case studies, besides sending the form in the results email, these forms were distributed in hard copy after the completion of the interview. Therefore, the evaluation form distribution and collection procedure varied: The majority of the participants (i.e., 72%) completed the forms with their handwriting (e.g., manually) and then submitted them before the closing meeting. Five of them (i.e., 14%) completed the forms online and sent them to the evaluator via email before the closing meeting. A minority of the participants (i.e., 14%), who did not have enough time to complete the forms until the closing meeting, evaluated the model during the closing session. However, this difference did not affect the overall evaluation for these participants since they elaborated on their answers based on results sent via email during the closing meetings.

The analyses were performed on all collected evidence ([25]) during the break session before the closing meeting (for Case Study A, it took 2 days, for the other case studies, it took 1 day). During the analysis, MAPforES was applied with the modeling stakeholder's characteristics to identify the modeling patterns and cultures both from the interview and from the observations and also according to what the model predicted. Moreover, by querying these characteristics in the database constructed by survey and interview results, MAPforES presented what practitioners with the similar demographics do in commonsense industrial modeling practices (for example, based on their role, the target sector, project size, etc., MAPforES increased the awareness of commonsense practices, such as the modeling languages specific to the target sector, e.g., Architecture Analysis and Design Language (AADL), Markov Chain Modeling Language, etc.) [10].

3.3 Results and discussion

In this section, we first provide the results of the case studies. Next, we provide the participants' feedback on MAPforES and lessons learned from the results.

3.3.1 Results

In the following tables, all three case study results are presented in individual tables, in which the "unaware"[3] participants of modeling or MDE are also presented. The following tables are arranged and sorted according to descending values of modeling patterns in the "Modeling Pattern According to Model" column (e.g., 3.3, 3.2, 3.1 → 2.2, 2.1, etc.) so the way modeling languages (e.g., DSL, UML and sketches) are grouped is clear. However, online versions of these tables, in which other attributes can be chosen as sorting/filtering criteria to observe distributions of the rest, are also available [33] (Note that the abbreviations used in the tables are given in Table 7).

The results of Case Study A are depicted in Table 8.

The results show that there is a difference in modeling approach patterns for different project and software

[3] As terminology, "unaware" patterns indicate the embedded software professionals, who do not know exactly their software modeling characteristics (especially their modeling rigor); in fact, they are "hidden patterns," which could not be identified by only quantitative data analysis.

Table 7 Abbreviations used in the article

Position		University degree	
Software developer/programmer	Dev	Computer science	CS
Software designer	Desg	Electrical/electronics engineering	EE
Software architect	Arch	Computer engineering	CENG
Systems engineer	Sys	Information systems	IS
Software tester	Tstr	Software engineering	SE
Project manager	PM	Mechanical engineering	ME
Quality assurance engineer	QA		

engineering roles (This was expected and is consistent with the results of earlier studies [19, 23]); however, it is also shown that this difference is related to the tasks and responsibilities of modeling stakeholder and her/his formal education (e.g., university degree). We present different profile details from different modeling approach patterns observed in Case Study A to support this conclusion.

Participant#11 tests the UI (User Interface) modules of the radar software project and mainly writes UI test simulators in Java or C++. He described the developed simulators as "low" in terms of hardware closeness.[4] He has also used their own MDE tool (which is based on their own DSL design) to generate test cases as model-based testing (MBT). Therefore, he benefits from both UML diagrams and DSL-like diagrams during analysis, design and test phases of SDLC. Participant#12 tests the communication protocol and message interfaces between middleware and digital signal processing (DSP) modules of the radar software, which are deployed in the main processor card (not in PC). She described the simulators she developed as "medium" in terms of hardware closeness. She does not use any model-driven techniques although she studied modeling languages in earning her MSc in computer engineering (CENG). She benefits from sequence diagrams, use case diagrams and communication diagrams during the analysis and test phases of SDLC. On the other hand, participant#13, whose academic background is different from other testers (i.e., he is an electrical/electronics engineering (EE) graduate and did not take any software engineering courses on modeling), tests DSP algorithms and does not use any programming language related to modeling. Besides, he never uses any digital medium (e.g., a personal computer) while modeling although he uses some use case or sequence diagrams to communicate with other colleagues without archiving them (e.g., these diagrams are discarded shortly after the conversation).

[4] Regarding "hardware closeness," firmware or digital signal processing (DSP) software is closer to hardware than user interface (UI) or middleware software.

As stated above, although participant#11, participant#12 and participant#13 are in the same project with the same role, since their responsibilities are different (e.g., testing different modules of the same software), their modeling approach patterns are different. The formal education (e.g., university degree) of these practitioners is also different. We encountered similar situations in which the responsibilities and university degrees influence the developers' modeling approach patterns (as shown in Table 8 for participant#6, participant#3, participant#9, participant#7 and participant#4).

Moreover, in Case Study A, during the first round of the interview (based on the questionnaire), participant#8 follows pattern 3.3 (i.e., "With DSL-like" pattern, see Table 1), but during the second round of the interview, face-to-face conversation revealed that he is a modeling stakeholder, who was "unaware" of MDE (as participant#28 in Case Study C). The professionals, who were "unaware" of MDE, completed the questionnaire as if they have benefitted from automatic code generation or documentation generation with sketch and UML usage. However, we observed that they actually used DSL-like modeling languages, which categorizes them as pattern 3.x. For further details of participants' responses, see [25].

In Case Study B, almost all results after observation and interviews are compatible with what MAPforES predicted except participant#21 (Table 9). This participant (i.e., project manager (PM), whose university degree is mechanical engineering (ME) and who did not take any software engineering courses related to modeling) completed the questionnaire by describing his modeling experience as "0" and as "never" using software modeling. However, after a face-to-face interview and observation, we noticed that he understood modeling limitedly as consisting of formal UML diagrams, but, in fact, he used sketches on paper and whiteboard during meetings with the systems and software engineering teams. He mentioned that he used some sketches (boxes and lines) to understand a problem or process at an abstract level (our "ad hoc" pattern). Therefore, he is a modeling stakeholder, who is "unaware" of modeling with a hidden pattern (i.e., pattern 1.x, "unaware of modeling") [25].

Table 8 Case Study A results, defense and aerospace sector, radar software project

Participant#	Position	Degree		Experience (in years)		PL & HW closeness	Modeling language(s)	Modeling pattern		Modeling culture	
		Academic	University	Work	Modeling			Observation/ interview	According to model	Observation/ interview	According to model
6	Dev, Desg, Arch	MSc	CS, IS	10+	10+	C, C++, Java medium	Sketch, UML, UML profiles, DSL	3.3 with DSL-like	3.3 with DSL-like	Auto-generated	Auto-generated
11	Tstr	MSc	CENG	10+	10+	Java, C++ low	Sketch, UML, DSL	3.3 with DSL-like	3.3 with DSL-like	Auto-generated	Auto-generated
3	Dev, Desg, Arch	MSs	EE, CENG	10+	10+	C, C++ medium	Sketch, UML	3.2 without DSL-like	3.2 without DSL-like	Auto-generated	Auto-generated
9	Dev	MSc	EE	10+	6–10	C, MATLAB high	Sketch, UML, MATLAB	3.1 limited	3.1 limited	Auto-generated	Auto-generated
14	Sys	MSc	EE, SE	6–10	6–10	MATLAB low	Sketch, UML, MATLAB, SysML	3.1 limited	3.1 limited	Auto-generated	Auto-generated
8	Dev, Desg	MSc	EE, CENG	10+	10+	C, C++ medium	Sketch, UML, DSL*	3.x unaware of MDE	3.3 with DSL-like	Auto-generated	Auto-generated
1	Dev, Desg	MSc	EE, IS	10+	10+	C++ medium	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
7	Dev	BSc	EE	10+	10+	C, C++ medium	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
10	Dev	BSc	EE	6–10	6–10	C high	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
12	Tstr	MSc	EE, CENG	10+	10+	C++ medium	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
17	QA	MSc	CENG	10+	10+	BPEL not applicable	Sketch, UML, BPML	2.1 descriptive	2.1 descriptive	Archived	Archived
2	Dev	BSc	EE	6–10	2–5	C high	Sketch, UML	1.3 archived	1.3 archived	Archived	Archived
16	PM	MSc	ME	10+	10+	Not applicable	Sketch, UML	1.3 archived	1.3 archived	Archived	Archived
4	Dev	BSc	EE	2–5	2–5	C high	Sketch, UML	1.2 selective	1.2 selective	Formalized	Formalized
13	Tstr	BSc	EE	6–10	2–5	Not applicable	Sketch, UML	1.2 selective	1.2 selective	Formalized	Formalized
15	Sys	Bsc	EE	10+	6–10	Not applicable	Sketch	1.1 ad hoc	1.1 ad hoc	Performed	Performed
5	Dev	MSc	EE	10+	10+	C very high	–	0.1 bad experienced	0.1 bad experienced	None	None

Totally, 204 years of software development experience

*This information was obtained during the interview or direct observation after the participant's completion of the questionnaire

The same "unawareness" of modeling occurred in Case Study C, for participant#32, who also played PM role (Table 10). After the interview, we realized that he indeed uses sketching on whiteboard in an ad hoc manner during the analysis phase of the SDLC [25].

A notable finding from all of our case studies is that whenever a programming language used for modeling (i.e., "PL & HW closeness" column) goes from high level to low level (e.g., from Java, C++ to C, or to the case where the participant does not use a programming language), the use of modeling decreases (i.e., "Modeling Patterns" column). Note that some embedded software professionals in industry think that the programming language and corresponding toolset restrict them while modeling (e.g., because of the limited number of embedded modeling environments, which provides code generation facility for C, or because the number of modeling tools for high-level programming languages, such as C++ or Java, is greater than the number for lower-level programming languages such as C) [23]. Moreover, in various software development projects, different layers of the same software might use different programming languages (e.g., the DSP team uses "C," the middleware team uses "C++" and the UI team uses "Java"), and the modeling practices of different teams might differ due to differing project characteristics [34]. Although the Platform Independent Modeling (PIM) concept might achieve modeling independent from programming language [35], in some cases, the programming language choice affects both attitudes on modeling and the development and modeling process followed which depends on tool support.

Regarding the relationship between university degrees and modeling cultures, in our case studies, there were no participants in the "Auto-generated" culture, whose university degree did not include some combination of Computing Disciplines (e.g., Computer Science (CS), CENG, SE or Information Systems (IS)) except participant#9 and participant#28, who use limited MDE (e.g., without code generation or MBT) and who graduated with EE degree (e.g., participant#9 uses MATLAB for model simulation, and participant#28 uses MDE for documentation generation [25]).

Table 11 summarizes the applications of MAPforES to the line of research work that led to its construction. The percentage of identified patterns and cultures is computed comparatively for our international survey of embedded software development professionals [19], the report on interviews with a sample of embedded software development professionals [23] and finally the three case studies given in the present article.

The use of model-driven approaches is more significant in some sectors than in others. For example, in Case Study A, where participants worked in the defense and aerospace sector, we found that model-driven approaches were widely used. This result is consistent with earlier studies. According to [36], defense and aerospace is the sector, in which model-driven approaches are the most popular among all other sectors, and automotive and transportation was in second place. Moreover, although the research methodologies (i.e., survey, interview and case studies) and participant numbers are different (i.e., 657, 53 and 35, respectively), the percentages in the patterns are similar and compatible with one other. As shown in Fig. 1, the percentage of "Auto-generated" and "Performed" cultures in the case studies (i.e., 31.4% and 8.6%) is higher than the percentages in the survey (i.e., 29.5% and 4.5%), whereas the percentage of the "None" culture in the case studies (i.e., 5.7%) is lower than that in the survey (i.e., 11%). The reason is "hidden patterns" (e.g., participants were unaware of MDE and modeling), something that could not be identified by survey data. In fact, the participants in both studies, who do not know whether they use MDE or software modeling, might cause changes in descriptive categorization and their percentages in different research methodologies, i.e., the percentage for "unaware of modeling" pattern in the survey, might be distributed into either in "performed" or "formalized" culture, but in the qualitative interview data (e.g., face-to-face, direct observations, etc.) more robust information about "unawareness" of modeling characteristics were provided, hence a more correct categorization.

By no means, these percentages present the exact distribution of the modeling patterns and cultures in the embedded software industry, but, this result gives insight and confirms the distribution of these patterns that MAPforES identified, along with further insights as detailed below. We must also emphasize that using a single embedded professional as the unit of analysis is generally considered acceptable in the literature (e.g., [37]).

Analysis of the evaluation forms is presented next. This analysis shows how participants evaluated the utility and comparative value of MAPforES.

3.3.2 Evaluations by individual qualitative evidence

Analysis of the evaluation forms shown in Table 12 (Appendix of [10]) shows that MAPforES reflected the expected results based on validation criteria in Table 3. Note that every participant except few (Table 12) elaborated on each affirmative answer. Details of this elaboration are presented next.

All qualitative and quantitative data gathered through these forms (which we discuss below) and the attitudes of the participants during the closing meeting demonstrate that the model was useful:

- in creating and increasing the awareness of what modeling stakeholders do,

Table 9 Case Study B results, automotive and transportation sector, bus software application project

Participant#	Position	Degree		Experience (in years)		PL & HW closeness	Modeling language(s)	Modeling pattern		Modeling culture	
		Academic	University	Work	Modeling			Observation/ interview	According to model	Observation/ interview	According to model
23	Dev, Des	MSc	CENG	10+	10+	C++ medium	Sketch, UML, DSL, (AUTOSAR)	3.3 with DSL-like	3.3 with DSL-like	Auto-generated	Auto-generated
20	Dev, Des, Arch	MSc	CS, SE	10+	10+	C, C++ medium	Sketch, UML, DSL, (AUTOSAR)	3.3 with DSL-like	3.3 with DSL-like	Auto-generated	Auto-generated
18	Dev, Arch	PhD	EE, CENG	10+	10+	C++ low	Sketch, UML	3.2 without DSL-like	3.2 without DSL-like	Auto-generated	Auto-generated
27	Dev, Arch	MSc	CENG	10+	10+	C, C++ medium	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
22	Tstr	BSc	CENG	6–10	6–10	C++, Java medium	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
19	Sys	MSc	EE, CENG	6–10	2–5	Not applicable	Sketch, UML	2.1 descriptive	2.1 descriptive	Archived	Archived
26	Dev	BSc	EE	6–10	2–5	C high	Sketch, UML	1.3 archived	1.3 archived	Archived	Archived
24	Dev	BSc	EE	10+	6–10	C high	Sketch, UML	1.2 selective	1.2 selective	Formalized	Formalized
21	PM	BSc	ME	10+	>0*	Not applicable	Sketch*	1.x unaware of modeling	1.1 ad hoc	Performed	Performed
25	Tstr	BSc	EE	2–5	–	C very high	–	0.0 not experienced	0.0 not experienced	None	None

Totally, 113 years of software development experience

*This information was obtained during the interview or direct observation after the participant's completion of the questionnaire

Table 10 Case Study C results, consumer electronics sector, TV software project

Participant#	Position	Degree		Experience (in years)		PL & HW closeness	Modeling language(s)	Modeling pattern		Modeling culture	
		Academic	University	Work	Modeling			Observation/interview	According to model	Observation/interview	According to model
30	Tstr	MSc	CS, CENG	10+	10+	C, C++, Python medium	Sketch, UML, DSL (Markov Chain)	3.3 with DSL-like	3.3 with DSL-like	Auto-generated	Auto-generated
28	Dev, Arch	MSc	EE	10+	10+	C++, Python medium	Sketch, UML, DSL (own, Doxygen)*	3.x unaware of MDE	3.1 limited	Auto-generated	Auto-generated
35	Dev, Arch	MSc	CS, CENG	10+	10+	C++, medium	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
29	Dev	MSc	EE, CENG	10+	6–10	C++, medium	Sketch, UML	2.2 prescriptive	2.2 prescriptive	Prescripted	Prescripted
31	Dev	BSc	EE	6–10	6–10	C high	Sketch, UML	2.1 descriptive	2.1 descriptive	Archived	Archived
34	Tstr	BSc	EE	6–10	2–5	Not applicable	Sketch, UML	1.3 archived	1.3 archived	Archived	Archived
33	Dev	BSc	EE	6–10	2–5	C very high	Sketch, UML	1.2 selective	1.2 selective	Formalized	Formalized
32	PM	BSc	EE	10+	>0*	Not applicable	Sketch*	1.x unaware of modeling	1.1 ad hoc	Performed	Performed

Totally, 95 years of software development experience

*This information was obtained during the interview or direct observation after the participant's completion of the questionnaire

Table 11 Case study results summary: comparison with survey and interview with respect to pattern & culture percentages

Patterns		Cultures	% in survey results (657)		% in interview results (53)		% in case A (17) radar sw project		% in case B (10) bus sw project		% in case C (8) TV sw project		% in case studies (35)	
Model-driven														
3.3	With DSL/DSML	Auto-generated	15.4	29.5	15.1	32.1	11.8	35.2	20	30	12.5	25	14.3	31.4
3.2	Without DSL/DSML		6.5		7.5		5.8		10		–		5.7	
3.1	Limited		7.6		5.6		11.8		–		–		5.7	
3.x	Unaware of MDE**		–		3.7**		5.8**		–		12.5**		5.7	
Model-based														
2.2	Prescriptive	Prescripted	25.2		18.9		23.5		20		25		22.9	
2.1	Descriptive	Archived	12.7	16.2	11.3	15	5.8	17.6	10	20	12.5	25	8.6	20
1.3	Archived		3.5		3.7		11.8		10		12.5		11.5	
1.2	Selective	Formalized	13.6		13.2*		11.8		10		12.5		11.5	
1.1	Ad hoc	Performed	4.5		3.7*	9.4	5.8	5.8	–	10	–	12.5	2.9	8.6
Sketching														
1.x	Unaware of modeling*		–		5.6		–		10		12.5		5.7	
None														
0.1	Bad experienced	None	11		7.5	11.2	5.8	5.8	–	10	–	–	2.9	5.7
0.0	Not experienced				3.7		–		10		–		2.9	

*1.x pattern might be either 1.1 or 1.2; hence, its value makes "performed" and "formalized" culture increase

**3.x pattern might be either 3.1, 3.2 or 3.3; but since all of them are in "auto-generated," no need for further analysis as in 1.x pattern

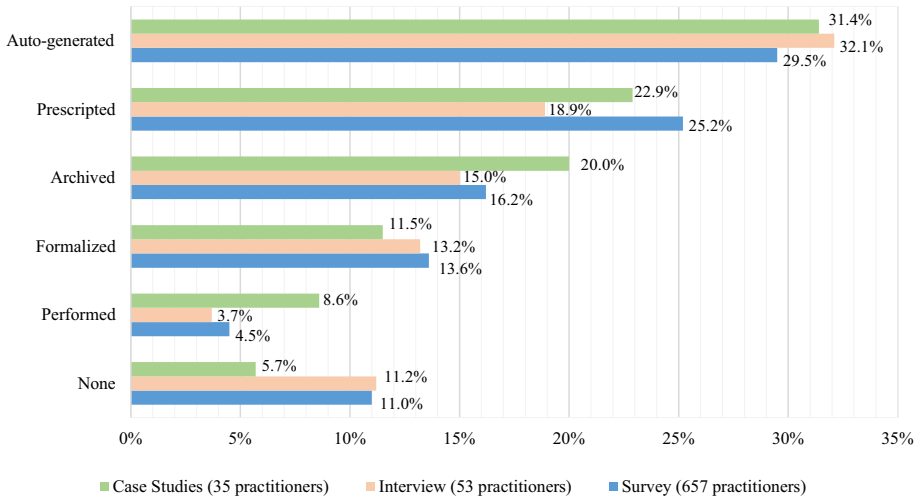

Fig. 1 Case study results: the percentage of participants in the modeling culture discussed in Sect. 2.2

- in giving an opportunity to the stakeholder to compare how software engineer with similar profiles model and also
- in suggesting useful software modeling practices [25].

Although the evaluation form is in English, four participants (~11%) answered in Turkish. Note that if the participant's answer is in English, the phrase is not corrected even if it might be grammatically incorrect on the original forms [25]. However, we have corrected these sentences in this paper to improve their understandability (with additional words added for clarity shown in brackets). Due to space constraints, selected evaluations to each question in Table 4 with verbatim quotes taken from the original evaluation forms are given below. All remaining qualitative data are available in [25].

3.3.2.1 Evaluations on relevancy and insightfulness of MAP-forES (question 1 and question 3 of Table 4) All responses to the first question mentioned that the model produced relevant results, which addresses the "result" criterion in Table 3 (e.g., the quality of the model with respect to its

Table 12 Percentage of respondents that replied affirmatively to validity criteria

Question #	Addressed RQ #	Validation criteria	Case Study A (%)	Case Study B (%)	Case Study C (%)	Total (%)
1	RQ1	Result	100	100	100	100
2	RQ1 and RQ2	Result & utility	94.1*	100	87.5*	94.2*
3	RQ2	Comparison	100	100	100	100
4	RQ2	Utility	100	100	100	100
5	RQ2	Utility	88.2	90	100	91.5

*All participant (i.e., 100%) satisfied related criteria for this question; however, two participants (i.e., 5.8%) did not elaborate their answers and just wrote "Yes" in the forms without explicitly mentioning about the benefits of the model

benefits). One participant stated on the evaluation forms: "In fact, I really did not know whether I have been modeling; but in fact, I now realize that I have been a sketcher for more than 10 years; yes I am a modeler but part of a 'performed'" *[culture]*. A project manager indicated the benefits of the presentation as "Before [the] presentation, I didn't think that modeling was important for me; but now I can say that at least I will try to investigate these recommendations [further]." All participants explicitly affirmed the proposed benefits of the model [25]. All answers to the third question in Table 4 revealed that no participant has experienced such a characterization model before [25]. Two evaluation forms suggest an improvement (also discussed during the closing meetings): "While getting the modeling characteristics of the stakeholder [as a structured questionnaire], why do not you use any automated tool so that both filling the form and the analysis would be easier?" A positive result of manual application was the possibility of identifying unaware patterns during direct observations and face-to-face meetings. This suggestion can be implemented by dividing the process into two stages: (1) obtain the inputs with a tool and (2) evaluate and observe the stakeholder's characteristics (e.g., using the questionnaire results form). Via an automated tool, the questionnaire can be sent to several companies before the visit. This would save time (although the semi-structured part is still crucial).

3.3.2.2 Evaluations on usefulness of MAPforES (question 2, question 4 and question 5 of Table 4) The objective of the second question on the evaluation form, which affects both the "result" and "utility" criteria listed in Table 3, was to evaluate the benefits of MAPforES. Almost every participant (~94%) explicitly mentioned the need to understand different modeling patterns and cultures so that a model might help the stakeholder to follow commonsense modeling practices. According to participants, the model provides a common language for a stakeholder to share her/his modeling experience with the others in the embedded software modeling community. Note that 5.8% of participants did not elaborate their answer on the second question, i.e., "Do you think that the model is helpful? Please elaborate your answer." They just wrote "Yes" without elaborating. However, we think that this ratio (e.g., ~6%) is negligible and not a threat to validity.

Depending on the modeling stakeholder role, the evaluations varied. For example, one software developer in the first case study stated that "knowing the characteristics of what I am modeling is helpful for categorizing my modeling approach. Learning the importance of DSL in embedded software industry pushes me to investigate further a cost-effective and domain-specific (defense) solution." Another participant (i.e., systems engineer) wrote on the evaluation form: "the model is helpful to understand different modeling approaches [of different roles] such as software developers, systems engineers (such as me) and even PMs [Project Managers]. As far as I understood, all of their approaches might be the 'best' according to some criteria, so there is no just one 'best'!"

One benefit of the MAPforES model is making stakeholders aware of their modeling practices. One software developer in the third case study said: "We now know that we are using DSL in fact:)." There are many participants (83%), who mentioned that the presentation given before the interview was also beneficial since knowing the relations between the modeling characteristics has practical benefits for the modeling stakeholder.

As mentioned, MAPforES uses the results of survey and interview data to describe the modeling practices of stakeholders with similar profiles. For example, a participant presented his/her characteristics of modeling (e.g., role or target sector), and the evaluator reported the similar profiles' modeling practices to increase modeling awareness (e.g., the modeling languages specific to the target sectors such as AADL for defense and aerospace, the Markov Chain Modeling Language for consumer electronics or modeling tools). In this way, MAPforES guided process and tool improvements for modeling by describing a set of commonsense industrial practices [25]. According to the majority of participants (i.e., 74.2% of participants used "useful" explicitly in their evaluation forms for the fourth question), this

set of commonsense practices is useful because their modeling practices might be affected based on the suggestions. "Learning what the similarly profiled [embedded software practitioners] are doing is useful to analyze the approaches [before embarking on a project with modeling]; it will save time" or "Knowing alternative practices (for example modeling tools) might affect our practices. If they are cheaper than what we use, we, of course, will use and apply these practices in future" are some example quotes from participants.

Although the attitudes toward these suggestions were always positive, some participants mentioned some organizational and managerial issues. One software developer in the first case study stated: "We have an organizational decision to use a modeling tool, I don't know whether we can change this; but the managerial decision on that tool might be affected if there are cheaper alternatives." Another software architect in the same software development team wrote in the evaluation form as "Of course, 'stand on the shoulders of giants':). If some of their choices [in modeling approaches and tools] fit our organization, why not?." One participant stated: "Yes, I believe that our competitors' modeling ways could be a source for inspiration about future projects, but I am not sure about my managers' possible concerns about what our competitors are doing; and they have the last word."

The set of modeling languages used based on the stakeholder role was also appreciated by the participants. One of the systems engineers stated: "In fact, there are not many systems engineers in the industry, therefore it is very interesting to learn what they do. Specifically, I want to learn more about SysML." A software developer in the first case study also stated: "I don't know about DSL usage in our industry; I should analyze some of them like MARTE and EAST-ADL." One of the project managers commented that "Being the PM, learning the other PMs' modeling usage is very interesting; perhaps I should analyze some BPM [Business Process Modeling] diagrams to get some benefits."

Almost every participant's answers to the fifth question (~91.5%) satisfied utility validation criteria for this question, which finds out whether the model is useful or not. By using the chart for the modeling patterns and cultures with the corresponding characteristics of modeling [10], the recommendations are useful for commonsense modeling practices depending on the specific characteristics (e.g., motivation and purpose).

One test engineer stated: "Developing a company-specific [domain-specific] tool according to our needs is always a planned action for our test department. Perhaps, [based on the feedback] from this study we can accelerate this process and fully automate all our testing procedure. By this way, all testers might be in the same pattern according to your model." One of the software developers, whose pattern is 2.2 (i.e., using prescriptive modeling but not model-driven techniques), said: "The recommendations are useful with respect to [being aware of] DSL and having own modeling tool. Perhaps, we can try model-driven techniques by comparing pros and cons." A software developer in the second case study said: "I think they [the recommendations] will be useful after analyzing the suggested modeling tools and DSLs further (mainly Papyrus, Eclipse-based tools and automotive domain-specific DSLs)."

Moreover, giving these recommendations explicitly (i.e., in a written format) made some participants aware of an easy and straightforward modeling task to get practical benefits. A project manager stated: "Just taking a photo of the whiteboard screen and archiving it is a very easy and effective solution. I am wondering why I did not do that until now."

The same situation encountered in the answers to the fourth questions' responses about organizational decision-making issues was also encountered here. "Yes, [the recommendations are useful] but since I am not a decision maker, I will also forward your email to my manager" or "After trying and experiencing the suggestions, I can [personally] use them, therefore it might affect [how we work] based on the results of their feasibility analysis; but for my team, I should inform my technical lead" are some example quotes about this challenge. Amorim et al. claimed that managers should understand which modeling tools, languages and approaches are best fitted for their organization through experimenting as the best way to introduce model-based systems engineering (MBSE) [38]. In that sense, MAPforES may help to overcome organizational issues since it provides experiences of embedded software modeling community in an abstracted and compact form.

The common concern of the participants, who thought that there might be some organizational and managerial issues against a straightforward application of the recommendations (~5.8% of participants), is that MAPforES has not been a "proven" or "accepted" model in the industry yet. One software developer stated: "As in all changes here [in our company], there is always the need for a "champion" to promote this model [and its recommendations]." He continued that in order to adopt this model without any organizational resistance, the management must either see the results of this model in their own projects or learn about any success story about MAPforES. Another developer said that since MAPforES is a new model for the industry, there is a need to evaluate its recommendations in the long term for an effective adoption of MAPforES. These comments actually state the need and provide a cue for the future development of MAPforES to be a more adoptable model as detailed in concluding remarks in Sect. 4.

Qualitative data gathered through the evaluations have shown that the MAPforES model has been useful in creating awareness and guidance on software modeling in embedded software practitioners.

3.3.3 Discussion and lessons learned

The results have organizational implications since it is beneficial to identify common techniques for different modeling purposes while pinpointing the potential challenges. For example, the results showed that even stakeholders in the same software engineering roles within the same projects might have different modeling practices (e.g., in Case Study A, participant#11, participant#12 and participant#13 as being software tester). On the other hand, we also observed that different modeling stakeholders (e.g., systems engineers, test engineers and software engineers) might have common modeling approaches and they might be in the same pattern (as participant#1, participant#7, participant#10 and participant#12 are in "Descriptive" pattern although their software engineering roles are different). Such findings have organizational implications such as the need for further training in modeling or implementing practices of effective use of modeling in an organizational unit that might change from case to case. In other words, MAPforES can be utilized to decide the common approaches and practices for different software modeling characteristics such as purpose, medium type (e.g., modeling environment/tool), SDLC phases or modeling languages. For example, those stakeholders, who find UML too general or vague for their purposes [23], might actually benefit from DSL-like approaches in their choices of modeling languages depending on the sector to carry out effective MDE. A study, which presented a list of recommended practices that contribute to the increased effectiveness of modeling (e.g., with UML), showed that the use of software modeling is considered beneficial for software maintenance (as a specific SDLC activity), but needs to be tailored to its context since various practices are commonly overlooked [39]. The detection of necessary modeling patterns, corresponding practices and further training in the organization (e.g., in a specific context), can be made via MAPforES characteristics [33].

The results of another study conducted in the embedded domain which investigated the barriers when adopting software modeling showed that "the lack of supporting tools" is the most possible problem preventing adoption of modeling techniques [40]. Similarly, a survey, which assessed the state of practice of model-driven approaches, showed that the main shortcomings to introduce and use these techniques are related to modeling tools and high training efforts [41]. In that sense, MAPforES might help stakeholders to find out the "right" modeling tool according to their needs by referencing to a set of commonsense industrial practices.

Modeling stakeholders are less likely to resist the challenges of modeling adoption if they can perceive its benefits. As reported in [38], the most important best practice for Model-Based Systems Engineering (MBSE)[5] adoption in the embedded software industry is related to increasing the stakeholders' motivation by making them aware of the benefits. As reported by various participants in our case studies, MAPforES increased the awareness of modeling benefits; hence, it will help overcome the challenges of modeling adoption.

Based on our results, the difference on modeling patterns might be related not only to project characteristics or software engineering roles in the project but also the tasks and responsibilities of that particular participant in that role besides formal education (e.g., university degree) of the stakeholder (as in the case of software tester roles in Case Study A).

We have already found out that educational skill set affects where/how the stakeholder learned software modeling, hence modeling approaches and its relevant practices through modeling experience [6]. For example, a stakeholder, who graduated from EE, most probably has learned software modeling after graduation with formal corporate training, or on his/her own; however, any stakeholder who graduated from a Computing Discipline has learned software modeling at the university from software engineering courses. On the other hand, for the participant, who learned software modeling at the university, a typical university software engineering course teaches in 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 model-driven approaches, which might increase the percentage of "Auto-generated" culture (Note that there is an increasing number of universities, which use "Model-driven software engineering in practice" as a software engineering course book, which covers more model-driven practices [35]). Therefore, we believe that the awareness of different modeling approaches is correlated with the discipline of university degree of the stakeholder. These case studies also showed there is a relation between the academic background and the modeling approaches if the task/responsibility of the stakeholders does not force him/her to do specific modeling practices. If the given courses on modeling might be updated or enhanced to make the embedded software professionals aware of different approaches and if

[5] Model-based systems engineering (MBSE) is one of the systems engineering methodologies that focuses on modeling as the primary means of information exchange between engineers to support analysis, design, verification and validation of the system (including software part).

the curriculum of non-Computing Disciplines (e.g., EE or ME curriculum) might be updated by including new courses about software modeling, we believe that this relationship might vary in the future.

The results also showed that there is a noticeable percentage of "unaware" participants of modeling or MDE (i.e., pattern 1.x and pattern 3.x, ~ 11%) in our embedded software community sample, which are identified during the direct observation or face-to-face semi-structured interview via a question and answer session (i.e., after the completion of structured part of the questionnaire). That is, although they practice a certain pattern in modeling, those participants are not metacognitively aware that they are using this particular pattern. According to our results, this "unawareness" related to software modeling is mainly based on stakeholder's profile (e.g., university degree, modeling experience, etc.). In other words, without a background and common terminology in modeling, the stakeholder may not be aware that she is actually using modeling in SDLC. An organization might use such outputs of MAPforES to fulfill any training need to create and increase the awareness of what their modeling stakeholders do.

3.4 Threats to validity

In our case, we should note that these case studies do not include any hardware, which is at architectural design or development stage (e.g., the hardware is robust). Thus, we assumed that hardware design and maturity problems did not present challenges to the scope of the study as the relation between software and the hardware which this software is running on is important in the embedded software development.

In our study, the following aspects are addressed [28]:

Construct validity Construct validity is concerned with the correctness of the interpretation and the theoretical constructs [42]. In this research, multiple sources of evidence were used in our case study strategy. All evidence was collected in questionnaires, written notes after interviews and direct observations and then kept in a technical report [25]. During the second round of the interview, the evaluator confirmed what the interviewee gave as responses in the questionnaire to ensure the validity of the collected data. By this way, cross-checking of what the questionnaire gave in the first round and what the evaluator observed during the second round provided more robust conclusions.

Internal validity In order to mitigate this threat [42], we focused on the study design and checked whether the results are consistent with the data. The case studies reported here are not controlled experiments; however, the stakeholders with similar profiles and modeling characteristics might be used for pattern matching with further case studies to eliminate any bias. During the first part of the interview, all participants filled out the questionnaire individually and separately so that the interviewer prevented answers of a participant to be influenced by others [43]. By this way, the interviewer avoided any information sharing between interviewees. Note that awareness of modeling or MDE is critical to feed the model with correct data. Since there is no culture difference in case of "pattern 3.x" (i.e., "pattern 3.x" might be "pattern 3.1" or "pattern 3.3" in practice, but all of them are in "auto-generated" culture), the model gives the relevant result for the corresponding culture. However, in the case of "pattern 1.x," if the input is "no modeling," the corresponding pattern would be incorrect. As seen, awareness of modeling approach (if available), hence data quality, is critical to have relevant results. Moreover, note that none of these software professionals in multiple case studies participated in the previous interviews; however, it cannot be guaranteed whether any of them participated in the survey or not. Nevertheless, note that even if they have participated in the survey, when the number of case study participants is compared to the number of participants in the survey (e.g., ~ 5.5%), a threat to internal validity would be limited.

External validity The generalizability of the results is focused to mitigate this threat [42]. Three cases and participants were selected intentionally (Sect. 3.2.2) with variation points (e.g., target domain, position, academic background, experience, hardware closeness). We cannot state that the selection is representative of all embedded software development projects. However, all three cases have similar results, and by applying the model in more case studies and projects, the generalizability shall be improved.

Reliability Reliability focuses on the replicability of the results by other researchers. This study has a case study protocol and database, which were documented and archived systematically so that the replicability and repeatability of the operation of the case study have been ensured. Note that both the questionnaires with the evaluator's notes and the evaluation form of participants were saved in case study database as a paper repository and then were digitized during the analysis when transcribed by taking the photo of each page [25]. Moreover, the draft case study design was reviewed by two academicians and three embedded software professionals. By this way, Table 5 is modified, and before the company visit, the agenda template was finalized. During the interviews, the actual (performed) progress of the case study against the planned progress (i.e., the agenda) was reviewed to determine if there are any significant differences. Moreover, the evaluator (during the second round of the interview) asked the interviewee to confirm and validate what she or he gave as responses in the questionnaire. This helped to ensure that the interview data provides a fair representation of the interviewee's opinions with correct answers.

Although the general validity of the conclusions was restricted by the limited number of case studies (i.e., three), we believe that MAPforES can be applied in different

subsectors of the embedded software industry and can be enriched with more case studies. Such a further study could strengthen the validity of the model.

4 Conclusions

In this study, we present the MAPforES model to identify and define modeling approach patterns and cultures used in embedded software development projects. We observed that the industrial context reflects what we presented for modeling approach patterns.

We noticed that organizations may need different modeling approach patterns for different projects or even for different individual software engineering roles within projects. Applying MAPforES provides feedback to modeling stakeholders and creates insight for individuals. The usage of the characterization model has the potential to overcome one of the most significant difficulties of top-down organizational process improvement model by enabling everyone to contribute [44, 45]. As also depicted in Heldal et al., different units within the same company might use different modeling approaches [8]. Moreover, we found that even in the same software development project, the same software engineering roles might use different modeling practices depending on their tasks and responsibilities for different modeling characteristics such as different purposes in different phases of SDLC besides formal education of the stakeholder.

We found out that MAPforES is useful since the participants explicitly mentioned their satisfaction [10] in creating awareness and referencing to a set of commonsense industrial modeling practices. Qualitative data gathered through the evaluations have shown that all participants thought that the MAPforES is conceptually insightful and the majority (i.e., 74.2%) used "useful" explicitly on their evaluation forms.

MAPforES can be applied with a moderate amount of effort (i.e., about 2 h per modeling stakeholder), and its benefits easily outweigh its costs as the improvements in individual processes are accumulated in all projects to be implemented after that point in time.

MAPforES is a complementary model for process improvement approaches such as CMMI and SPICE [14, 15]. Identifying modeling patterns of individuals and/or projects before an organizational assessment of software modeling practices may be useful in pinpointing the potential threats for institutionalization such as the diversity of techniques utilized. The results can also be beneficial for identifying the common techniques for different purposes used, thereby for determining the best standardization approaches. Two organizations in the case studies have CMMI certifications and the participants found the model useful, which has increased both their awareness of their own and similar demographics' modeling practices.

Our validation of MAPforES does not currently entail whether the awareness and new information acknowledged by the participants have short- and long-term positive effects in their modeling practices in embedded software development. Thus, to evaluate both the individual and organizational benefits of MAPforES, we have planned a systematic study, in which we will investigate the changes of software modeling practices in time, of the participants, via various interviews (e.g., if there is an improvement, which practices are affected and what are their consequences; if there is no change, what are the challenges of not applying MAPforES, etc.). We also plan to conduct such a study on another control group, who have not applied MAPforES. These studies will enable us to reveal the effect of MAPforES on the changes for software modeling practices.

In the multiple case studies, to take the modeling characteristics of the participant, a questionnaire was used. A recommendation system using artificial intelligence techniques might transform a more costly to implement technique such as a questionnaire into a virtual assistant for project and program managers implementing policies on software modeling based on a model such as MAPforES of community experience.

We also plan a study of technical and social factors that influence the adoption of various modeling patterns, specifically the effect of understandability and organizational resistance [46].

MAPforES is the first wide-coverage model of modeling characteristics for the embedded software sector built on extensive input from industry. The work presented in this article complements the model development effort by applying the MAPforES model successfully in three embedded software projects from two organizations. We hope that MAPforES and its applications in the field will establish a useful baseline for future individual and organizational process improvement studies in embedded systems modeling.

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

References

1. Graaf, B., Lormans, M., Toetenel, H.: Embedded software engineering: the state of the practice. Softw. IEEE **20**, 61–69 (2003)
2. Walls, C.: Embedded Software, 2nd edn. Newnes, Oxford (2012)
3. Arcelli, D., Cortellessa, V.: Assisting software designers to identify and solve performance problems. In: *Presented at the ACM International Workshop on Future of Software Architecture Design Assistants* (2015)
4. Thomas, D.: MDA: revenge of the modelers or UML utopia? Softw. IEEE **21**, 15–17 (2004)

5. Ross, J.A., Murashkin, A., Liang, J.H., Antkiewicz, M., Czarnecki, K.: Synthesis and exploration of multi-level, multi-perspective architectures of automotive embedded systems. Softw. Syst. Model. **18**, 739–767 (2017)
6. Akdur, D., Demirörs, O., Garousi, V.: Characterizing the development and usage of diagrams in embedded software systems. In: *43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Vienna, Austria (2017)
7. Dzidek, W.J., Arisholm, E., Briand, L.C.: A realistic empirical evaluation of the costs and benefits of UML in software maintenance. IEEE Trans. Softw. Eng. **34**, 407–432 (2008)
8. Heldal, R., Pelliccione, P., Eliasson, U., Lantz, J., Derehag, J., Whittle, J.: Descriptive vs prescriptive models in industry. In: *ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, France (2016)
9. Weinberg, G.M.: Quality Software Management (Vol. 1): Systems Thinking. Dorset House Publishing, New York (1992)
10. Akdur, D.: Modeling patterns and cultures of embedded software development projects. Thesis, Doctor of Philosophy (Ph.D.). Information Systems, Middle East Technical University (METU). www.researchgate.net/publication/322701453_Modeling_Patterns_and_Cultures_of_Embedded_Software_Development_Projects (2018)
11. Pattern ed Cambridge Dictionary (2017). https://dictionary.cambridge.org/
12. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston (1998)
13. Douglass, B.P.: Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Addison-Wesley, Boston (2003)
14. Dorling, A.: SPICE: software process improvement and capability determination. Inf. Softw. Technol. **35**, 404–406 (1993)
15. CMMI Institute. Available: http://cmmiinstitute.com/ (2018)
16. Kleppe, A.G., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)
17. Petre, M.: UML in practice. In: *35th International Conference on Software Engineering (ICSE)*, pp. 722–731 (2013)
18. Greenfield, J., Short, K., Cook, S., Kent, S.: Software Factories—Assembling Application with Patterns, Models, Frameworks and Tools. Wiley Publishing, New York (2004)
19. Akdur, D., Garousi, V., Demirörs, O.: A survey on modeling and model-driven engineering practices in the embedded software industry. J. Syst. Arch. **91**, 62–82 (2018)
20. Cabot, J.: Clarifying concepts: MBE vs MDE vs MDD vs MDA. Available: http://modeling-languages.com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda (2018)
21. Karagoz, N.A., Demirors, O.: Conceptual modeling notations and techniques. In: Robinson, S., Brooks, R., Kotiadis, K., Van Der Zee, D.J. (eds.) Conceptual Modeling for Discrete-Event Simulation. CRC Press, Boca Raton (2010)
22. Akdur, D., Demirors, O.: Modeling patterns and cultures of embedded software development projects: towards preliminary characterization model. In *12nd Turkish National Software Engineering Symposium (In Turkish: Ulusal Yazılım Mühendisliği Sempozyumu (UYMS))*, İstanbul, Turkey (2018)
23. Akdur, D., Demirörs, O., Say, B.: Towards modeling patterns for embedded software industry: feedback from the field. In: *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Prag, Czech Republic (2018)
24. Akdur, D., Garousi, V., Demirörs, O.: MDE in embedded SW industry-raw survey data. https://dx.doi.org/10.6084/m9.figshare.4262972 (2015). Accessed 27 Nov 2016
25. Akdur, D., Demirörs, O.: Multiple case studies to validate modeling patterns and cultures of embedded software development projects, technical report. METU, METU/II-TR-2017-90 (2017)
26. Robson, C.: Real World Research, 2nd edn. Wiley, New York (2002)
27. Yin, R.K.: Case Study Research: Design and Methods. SAGE Publications, Thousand Oaks (2003)
28. Runeson, P., Host, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering: Guidelines and Examples. Wiley Publishing, New York (2012)
29. Bratthall, L., Jørgensen, M.: Can you trust a single data source exploratory software engineering case study? Empir. Softw. Eng. **7**(1), 9–26 (2002)
30. Kitchenham, B.A., Linkman, S., Law, D.: DESMET: a methodology for evaluating software engineering methods and tools. Comput. Control Eng. J. **8**, 120–126 (1997)
31. Verner, J.M., Sampson, J., Tosic, V., Bakar, N.A.A., Kitchenham, B.A.: Guidelines for industrially-based multiple case studies in software engineering. In: *Presented at the Third International Conference on Research Challenges in Information Science*, Morocco (2009)
32. Kahraman, G., Bilgen, S.: A framework for qualitative assessment of domain-specific languages. Softw. Syst. Model. **14**, 1505–1526 (2015)
33. Akdur, D.: Online dataset: multiple case study results. Available: https://www.researchgate.net/publication/323810535_Multiple_Case_Study_Results_-_Applications_of_the_Modeling_Patterns_for_Embedded_Software_Development (2017). Accessed 16 Sept 2018
34. Akdur, D., Garousi, V.: Model-driven engineering in support of development, test and maintenance of communication middleware: an industrial case-study. In: *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, France (2015)
35. Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice. In: Baresi, L. (ed.) Synthesis Lectures on Software Engineering, vol. 1. Morgan & Claypool, San Rafael (2012)
36. Akdur, D., Garousi, V., Demirörs, O.: Cross-factor analysis of software modeling practices versus practitioner demographics in the embedded software industry. In: *6th Mediterranean Conference on Embedded Computing (MECO)*, Montenegro (2017)
37. Wallace, L., Keil, M., Rai, A.: Understanding software project risk: a cluster analysis. Inf. Manag. **42**, 115–125 (2004)
38. Amorim, T., Vogelsang, A., Pudlitz, F., Gersing, P., Philipps, J.: Strategies and best practices for model-based systems engineering adoption in embedded systems industry. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 203–212 (2019)
39. Fernández-Sáez, A.M., Chaudron, M.R.V., Genero, M.: An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles. Empir. Softw. Eng. **23**, 3281–3345 (2018)
40. Vetro, A., Bohm, W., Torchiano, M.: On the benefits and barriers when adopting software modelling and model driven techniques—an external, differentiated replication. In: *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–4 (2015)
41. Liebel, G., Marko, N., Tichy, M., Leitner, A., Hansson, J.: Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. Softw. Syst. Model. **17**, 91–113 (2018)
42. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Berlin (2012)

43. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D., El Emam, K., et al.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**, 721–734 (2002)
44. Uskarcı, A., Demirörs, O.: Do staged maturity models result in organization-wide continuous process improvement? Insight from employees. Comput. Stand. Interfaces **52**, 25–40 (2017)
45. Dikici, A., Turetken, O., Demirors, O.: Factors influencing the understandability of process models: a systematic literature review. Inf. Softw. Technol. **93**, 112–129 (2018)
46. Kılıç, Ö., Say, B., Demirörs, O.: An experimental study on the cognitive characteristics of modeling notations. In: Cipolla-Ficarra, F., Kratky, A., Pérez, M., Cipolla-Ficarra, M., Castro, C., Nicol, E. (eds.) Advances in Dynamic and Static Media for Interactive Systems: Communicability, Computer Science and Design. Blue Herons Editions, Bergamo (2011)

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 the UK 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 modeling, model-driven engineering, software engineering education and industry–academia collaborations.

Bilge Say is an assistant professor of software engineering in Atilim University, Turkey. She was previously a full-time faculty member in cognitive science department of Middle East Technical University, Turkey. She received her Ph.D. in computer engineering in 1998 from Bilkent University, Turkey. Her research interests are mainly focused on empirical cognitive studies in business process and software modeling, cognitive modeling and computational linguistics.

Onur Demirörs 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). 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.