2018 44th Euromicro Conference on Software Engineering and Advanced Applications

Towards Modeling Patterns for Embedded Software Industry: Feedback from the Field

Deniz Akdur
ASELSAN Inc.,
Ankara, Turkey
denizakdur@aselsan.com.tr

Onur Demirörs
Department of Computer Engineering,
İzmir Institute of Technology, Turkey
School of Computer Science and Engineering,
University of New South Wales, Australia
onurdemirors@iyte.edu.tr

Bilge Say
Department of Software Engineering,
Atilim University, Ankara, Turkey
bilge.say@atilim.edu.tr

***Abstract*— The analysis, design, implementation and testing of software for embedded systems are not trivial. Software modeling is a commonly used approach in the embedded software industry to manage complexity of these phases. The modeling approaches vary since the characteristics of modeling such as its purpose, the medium type used, the lifecycle phase used, differ among systems and industrial sectors. Our previous research identified and defined the modeling approach patterns in embedded software development projects based on quantitative data. In this paper, to validate and improve the pre-investigated pattern set, we present a series of semi-structured interviews over eight months with 53 embedded software professionals across a variety of target industrial sectors and roles. With the help of these interviews, the different modeling approach patterns in embedded software development were better understood and the hidden patterns not evident in the previous study were identified along with a documentation of personalized modeling experiences.**

Keywords- Software modeling; model driven engineering (MDE); characteristics of a diagram; embedded software; pattern

I. INTRODUCTION

Having distinct functionalities incorporated into a single system, which require many hardware and software systems' integration, make the embedded software development challenging [1]. Software modeling helps engineers to work at higher levels of abstraction, facilitates communication and automates the generation of software development life cycle (SLDC) artifacts (e.g., code, documentation, test case) to manage the complexity of these systems.

The modeling approaches in embedded software vary since the characteristics of modeling differ among systems and sectors, e.g., consumer electronics, defense or automotive [2]. At one extreme, some stakeholders (e.g., project managers or systems engineers) use modeling informally, where diagrams are sketched on a paper in order to communicate with colleagues. At the other extreme, for some stakeholders (e.g., software developers), modeling turns into programming with automated generation of code. Moreover, different units within the same company might use different modeling approaches for different purposes [3].

Deciding when to model or in what degree and with how much rigor (e.g., as a sketch without modeling language formality or by automating software artifact generation as in model driven engineering (MDE)) are frequently asked and challenging questions for software teams. Therefore, there is a need to identify the relations between the characteristics of modeling (e.g., modeling rigor, purpose, stakeholder, medium type used, SDLC phase, etc.) to respond to these challenges. A potential approach to resolve the challenges would be to identify, define and use "modeling approach patterns", which might be analogous to the characterization models, defined and tailored for software process improvement (SPI) (e.g., Software Sub-Cultures [4]).

There are only two research studies, which have classified the modeling usage categories in the literature (Section II.B). In our previous research, we focused to fill one major part of the gap in the existing literature by identifying and defining modeling approach patterns in embedded software industry. This identification was based on the quantitative data, which we found out from our survey result [5] and our previous research, which figured out the characteristics of software modeling and the relations between them [6] (Section II.A). In order to improve what we found out from the analysis of survey, there was a need to validate these pre-investigated patterns with a more qualitative strategy. Therefore, we conducted a series of semi-structured interviews over eight months with 53 embedded software professionals across a variety of target industrial sectors and software engineering (SE) roles. With the help of these interviews, the modeling approach patterns in embedded software development were better understood and the hidden patterns[1], which could not be found out by only survey data, were identified. The goal of this paper is to report the results of these case studies and present the final set of these patterns.

The remainder of this paper is structured as follows. Section 2 gives an overview of related studies. Section 3 presents the case studies. Finally, Section 4 provides an overall conclusion and states the future work directions.

II. OVERVIEW OF RELATED STUDIES

A. Our Previous Research

In order to understand the state-of-the-practices in modeling practices in the embedded software industry, we conducted a global survey. Participants were from 27 countries working in different subsectors of embedded software industry and SE roles [5]. The survey showed that the embedded software professionals use modeling approaches in varying degrees with different needs. All of the usages (e.g., informal sketching, model based engineering (MBE), or MDE) could be effective depending on the characteristics of modeling.

Based on the results of our findings (e.g., [1, 5]) as well as different classifications about modeling [7, 8], we presented a conceptual model [6] to better understand and identify the characteristics of software modeling. Accordingly, there are 11 main characteristics of software modeling, where some sub-characteristics affect the main ones [6].

After identifying the relations between these characteristics, we created a preliminary set. During this process, grouping 11 main characteristics on survey data, which includes ~80 attributes would generate lots of combinations [9]. In order to eliminate unnecessary combinations, we reduced the number of attributes

[1] The "hidden patterns" are the patterns of the participants, who do not know exactly their software modeling characteristics (especially about their modeling rigor and modeling language). They are unaware whether they use software modeling or MDE.

978-1-5386-7383-6/18/$31.00 ©2018 IEEE
DOI 10.1109/SEAA.2018.00030

by generating derived attributes after determining the most critical ones [10]. After visualizing these groups (e.g., with scatter and bars stacked charts), we derived the necessary characteristics, which have critical importance on the categorization: "*purpose*", "*medium type*", "*archivability*", "*modeling language, if any*" and "*SDLC phase*" [10]. Accordingly, nine modeling approach patterns were designated at the end of this multi-source research process as in Table 1 (Note that when the survey was conducted in 2015, we could not come up with better definition provided by [7]; so, "model-based" and "sketching" were in the same group (as MBE) [5]).

Table 1 Modeling approach patterns after survey data analysis

Main Pattern	**Modeling Approach Patterns Pre-investigated**				**% in survey results**	
model-driven	3.3	With DSL-like*	Purpose of the modeling includes "Code generation" or "Test case generation (MB/DT)"	With "any DSL-like" usage	16,9	29,5
	3.2	Without DSL-like		Without any "DSL-like" usage	6,5	
	3.1	Limited	Only with "Document generation", "Model simulation" or "Model to model transformation" purpose		6	
model-based	2.2	Prescriptive	SDLC phase while modeling is used includes "implementation" or "testing"		24,9	59,5
	2.1	Descriptive	SDLC phase does not include "implementation or testing"		13,7	
sketching	1.3	Archived	Purpose of the modeling includes "Documenting Analysis & Design" Media type used: Analog media usage >= Digital media usage		3,6	
	1.2	Selective	Casually & informally with some formalized modeling language (most probably, UML elements) Modeling Language set includes sketch&any formalized modeling language (e.g., UML&|DSL-like)		13,1	
	1.1	Ad-hoc	Purpose of the modeling includes only "Understanding" or "Communication" Only pen & paper / free format (e.g., without any formalized modeling language, e.g., UML) Medium type used while modeling is analog (paper or whiteboard)		4,1	
none	0	No modeling	Not using any modeling approach.		11	

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

There might be some hidden patterns, which could not be found out from the analysis of survey; hence, there was a need to validate and improve these pre-investigated patterns via in-depth interviewing. Section 3 presents these interviews' results.

B. Existing Literature on Software Modeling Usage Patterns

There are different definitions for "pattern" in the literature. It is defined as "*consistent and recurring characteristic that helps in the identification of a phenomenon*" [11]. In SE literature, there is the "pattern" concept to rely on proven solutions to recurrent design challenges such as "software design pattern" [12]. In this study, a "modeling approach pattern" consists of specific characteristics of modeling (e.g., purpose, medium type, modeling language, SDLC phase, etc.), which helps to identify stakeholder's modeling practices.

In the literature, there aren't many research studies related to the modeling approach patterns. Kleppe, Warmer and Bast classified the modeling usage as maturity levels by taking only one of the characteristics of modeling (i.e., "modeling rigor") [13]. According to Kleppe et al., there are six (0 through 5) modeling maturity levels (MMLs) in software development projects, in which there are different types of modeling usage based on "rigor". However, our empirical evidence have shown that different characteristics of modeling need not necessarily force stakeholders up the maturity level with respect to a single dimension such as rigor. The variety of modeling characteristics are related with different notations, tasks and roles.

In a second study, Petre focused on Unified Modeling Language (UML) usage categories for software developers only [14]. However, within the domain of modeling in embedded software development, a variety of stakeholders (e.g., from software developer to tester and systems engineer to project manager) and patterns of Domain Specific Languages (DSL), which have been claimed to have more potential for model driven development (MDD) than UML [15] should be considered. Note that none of these studies are directly related with embedded software development.

III. Case Study

This section presents case study to validate and improve modeling approach patterns given in Table 1.

A. Research Methodology

The empirical study reported here included a series of semi-structured interviews [16], which were conducted over eight months with 53 embedded software professionals across a variety of target industrial sectors and roles to validate and improve our pre-findings on modeling approach patterns.

The main goals of this study are designed as specific as possible with the corresponding RQs:

- **RQ1**: What is main software modeling usage pattern of the interviewee (i.e., none, sketch, model-based or model-driven)?
- **RQ2**: What is the current state of software development techniques/approaches of the interviewee (e.g., programming, modeling (if any), etc.) based on main modeling usage pattern? What are the characteristics of software modeling?
- **RQ3**: Does the modeling approach pattern of the participant belong to the pre-investigated pattern set? If not (i.e., hidden pattern), what are the main characteristics?

Each of the above RQs, which are cross-cutting with our survey [5] and complementing each other, is used to derive interview questions, in which some questions were taken from the survey (i.e., demographics) and some of them were improvised and detailed during the interviews.

B. Interview Design and Execution

The semi-structured interviews were conducted mostly in face-to-face, but if it is inconvenient, on Skype as in the case of intercontinental interviews with different industrial sectors, roles, experiences and practices in embedded software industry [10]. All interviewees were promised that only anonymous data would be presented and the interviewer would take notes on what he spontaneously found relevant for analysis. During the interview, there was a clear and complete list of general topics (i.e., interview instrument), which cover about modeling usage patterns besides their success & failure stories, the attitudes to the adoption

of modeling and so on [10]. The interviewer asked the set of questions, listened to the answers and followed up answers with additional questions when necessary [16].

The interviews lasted approximately more than 1 hour and the protocol was straightforward, presenting the objectives of the interview and explaining how the data would be used. Then a set of questions were asked. After getting demographics data (as in our survey [5]), the key/critical question was "*How often do you use software modeling in your software development life cycle? (either informal and/or formalized: i.e., sketches and/or models)"*. The goal of this question was to categorize the interviewee according to main modeling usage pattern. If the answer is "*Never*", which means that the main pattern is "no modeling", the interviewee was asked a series of questions about why they don't use any software modeling during their SDLC phase. Otherwise, if the response was different from "*Never*" (e.g., sometimes, frequently, always, etc.), the interviewer tried to understand their modeling practices. Then, by giving the necessary terminology on MDE, MBE and sketching, the second key/critical question about model-driven usage was asked [10]. Since we previously found out that modeling purpose directly affects the modeling rigor (hence modeling approach) [6], the interviewer also asked "*Is there any listed purpose while you are modeling*?" by showing the model-driven specific purposes such as code generation, documentation generation, test case generation. Again, if the answer is "*Never/No*", which means that this interviewee is at "sketching" or "model-based" pattern, corresponding in-depth questions were asked. Otherwise, this means that the participant uses "model-driven" techniques -at some degree-.

After the interview session, if possible (due to time constraints), the interviewer showed the taken notes and repeated the most critical parts to the interviewee (e.g., the critical characteristics of software modeling) to give an opportunity for clarification and expansion of their answers. During the analysis, the interviewer tried to investigate interesting key findings and observations from the informal conversations.

C. Findings

53 interviews in 14 companies within different target sectors had been carried out. Interviewees represented about different SE roles and academic background [10]. They have, cumulatively, 756 years of software development experience.

Due to space constraints, the interesting observations on the main software modeling usage patterns (i.e., "no modeling", "sketching", "model-based" and "model-driven) are given as an online appendix besides presenting informal question & answer session results with verbatim quotes of interviewees [17]. In this study, specifically, the patterns, which could not be derived from Table 1 are more focused.

Patterns in "no modeling": The survey showed that 11% of respondents have not been using any software modeling [5]. As seen from Table 1, this main pattern (i.e., "0") needs further analysis to understand why. When interview data is analyzed, there are mainly two sub-patterns, who do not use any software modeling: Some of these participants do not have any modeling experience (i.e., "not experienced"), whereas some of them do not use it although they have some experience on that (i.e., "bad experienced"[2]). There are totally six interviewees in this main pattern; two of them are "not experienced", the other four are "bad experienced". By this way, the interview divides "pattern 0" into two patterns, i.e., "pattern 0.0" and "pattern 0.1".

When the survey data was analyzed, the ones who stated that they don't use any software modeling, are mainly Industrial Engineering, Mechanical Engineering and Electrical/Electronics Engineering (EE) graduates. These respondents most probably have not learned any software modeling during university (e.g., from SE courses) and do not need it in their job history [5]. There are only two participants in the interviews, who both graduated from EE. On the other hand, the survey did not give any further information why some participants do not use any software modeling although they know it. As interviews showed that these participants have bad or poor experiences and failure stories on modeling [17]. They think that modeling is costly for their business due to hardware closeness, uniqueness and project size (i.e., the characteristics of software modeling [6]). It was interesting that these "bad experienced" professionals mentioned about modeling tools' problems, which is mandatory for "model-driven" approach, but not for "sketching" or "model-based". They had some resistances on modeling (e.g. one of the modeling challenges [5]) and it was difficult to change their negative attitude [17]. Apart from these common opinions, there are also some other issues such as "understanding the notation" of UML, cost of training (e.g., just because of training, they might not complete the project within the required time and budget), and the synchronization problem between model & code, made these interviewees (i.e., "pattern 0.1") not use any modeling.

Patterns on "sketching": To be a sketch user, modeling purpose set of the stakeholder should not include any model-driven specific purpose; but might include any general modeling purpose as communication and/or understanding [6].

During the interviews, it was observed that there exist some "sketch" users, who do not know that what they actually do is software modeling. This hidden pattern, could not be investigated from the survey results since such participants might indicate that they do not use any software modeling. With the help of this interview, this hidden pattern (i.e., "pattern 1.x) is figured out in the embedded software development projects. These embedded software professionals, whose response to the first critical question in the interview (i.e., "*How often do you use software modeling in your SDLC?)* was "*Never*", mainly use state machines, activity diagrams and sequence diagram-like drawings (e.g., includes some UML elements but informally). The common characteristic of these "unaware of modeling" (i.e., pattern 1.x) is that they have not taken any SE courses during the university and try to explain something intuitively without knowing that they actually do –somekind of- modeling [17].

Another pattern (i.e., pattern 1.1) is an obvious usage, which is only "pen & paper" with free-format (e.g., without any formalized modeling language elements). Their purposes are just communication or understanding a problem at an abstract level on an analog media such as paper or white/blackboard. Notice that it does not mean that all other usage patterns (e.g., model-driven) do not use paper or whiteboard; indeed, such analog mediums might be a quick solution for a better communication and faster idea sharing technique in some situations; but this pattern (i.e., 1.1) "only" uses such an approach as ad-hoc. Both our survey results and interviews showed that mainly systems engineers, requirement engineers and low-level (e.g., BSP, DSP) engineers are in this category [17]. Depending on the purpose, their modeling approach satisfies their motivation (e.g., for team

[2] As a terminology, "bad experienced" pattern indicates the embedded software professionals, who don't use any kind of modeling due to disappointing experiences of software modeling.

collaboration, some sketches are enough for communication); so no need for any other (e.g., more formal) approach.

In the survey, the respondents, who state that they were doing informal modeling, make the sketches, which include some essences of UML (e.g., some elements of UML, but without strict rules) as reported in [14]. These participants (i.e., pattern 1.2) answered this survey question by selecting some diagram types (e.g., some participants, who use "Sketch/No formal modeling language", draw a use case or sequence diagram) [5]. Interviews showed that most people use sequence diagrams informally to convey the communication among the entities in a given system. Their purpose is just a quick communication and understanding a scenario [17]. This sketching might have occurred either on analog or digital media, but without any documentation purpose (e.g., without documenting analysis & design).

Another pattern (i.e., pattern 1.3) is based on the purpose of modeling (i.e., documenting) and the medium type while modeling (i.e., digital or analog), which affects "archivability" and "lifespan" of the diagrams [6]. In this pattern, although there is a "documenting analysis & design" purpose, analog media usage is more frequent than to the digital ones and there might be some media type transitions, after the modeling process (e.g., during documenting). These transitions (e.g., from analog to digital) were occurred to achieve more archivability and hence lifespan. For example, some analog diagrams (e.g., either in paper or in white/blackboard) are archived by saving a digital picture or by redrawing them digitally for the customer requirement [17].

Patterns on "model-based": Almost all interviewees in this pattern use UML selectively and often informally. The differentiation point is SDLC phase(s), where modeling is used, which is also related with the difference between descriptive and prescriptive modeling [6]. It was realized that the patterns are originated and depend on whether SDLC set include "Implementation and/or Testing". If the diagram might be an input for implementation and/or testing phase, this approach is close to prescriptive; otherwise descriptive. In that sense, one of the patterns (i.e., pattern 2.1) use modeling very closely to descriptive approach during analysis, design or maintenance phase of SDLC; the other pattern (i.e., pattern 2.2) uses the diagrams in implementation and/or testing phases.

The interviews also showed that in the same main modeling pattern, there are some cross-life cycle activities, in which one modeling stakeholder's input might be another's output, whose patterns are different (e.g., a systems engineer, who uses a sequence diagram in "analysis" phase, gives this input to a software developer, who uses it in "analysis and implementation" without any model-driven purpose) [17].

Although this pattern might be close to model-driven if the stakeholders would use more constraints, some of the interviewees had also bad/poor experiences on the modeling tool [17]. If there is a disappointing experience on MDE, it is also very difficult to change the attitudes and adoption. The interviews also revealed that although Platform Independent Modeling (PIM) concept might achieve modeling independent from programming language (PL), in some cases, PL choice affects both modeling attitude and also development process due to tool support. Therefore, another observation is that organizational resistance might disappear with a relevant tool support, which shows "real" model-driven benefits [17].

Patterns on "model-driven": There are some interviewees (i.e., pattern 3.x), who actually use MDE without knowing it, so they are "unaware of MDE". This hidden pattern is also derived from the interview results. They have mainly DSL-like usages (See Table 1) and benefit from model-driven concepts (e.g. with model-driven purpose(s)); however, they are a bit confused with "modeling" terminology since there is not any UML related diagrams while modeling [17].

As survey data analysis and interviews showed that some participants (i.e., pattern 3.1) use MDE in a limited way without benefiting from "code generation" or "model based/driven testing" (MB/DT). These "limited" users mainly use diagrams for "documentation generation" or "model simulation" (e.g., a systems engineer, who uses simulation in designing an algorithm; but not sharing this with any software engineer during implementation; or a software developer, who just wants to generate documents from the models) [17].

In model-driven approach, it is not so important to have a graphical syntax to represent the model (as in UML), but these models should be represented in a format that is readable by a machine (as in DSL). UML is a general-purpose modeling language and its usage is not only restricted to modeling software [5]. "UML" was also presented as "Utopian Markup Language" since it would be utopia that UML can be used for all purposes [18]. As observed in [3], UML is not so popular for prescriptive approach since its semantics is not exactly defined. Researches claimed that for maximum benefit, there should be a customization on DSL [18]. We already figured out the importance of DSL-like usage existence (Table 1) and we also observed this distinction during the interviews. The interviewees, who do not use any DSL-like, mainly use UML (i.e., pattern 3.2). On the other hand, some model-driven users fully benefit from DSL-like usages (i.e., pattern 3.3) [17].

Result: After the interview sessions, the set of patterns were extended to 12 patterns, where the final set is given in Table 2.

Table 2. Interview results on modeling patterns by comparing survey results

Main usage pattern	Modeling Approach Patterns		interviewees (53)			% in survey results (627)	
			#	%			
model-driven	3.3	With DSL-like	8	15,1	32,1	16,9	29,5
	3.2	Without DSL-like	4	7,5		6,5	
	3.1	Limited	3	5,6		6	
	3.x	Unaware of MDE	2	3,7		-	
model-based	2.2	Prescriptive	10	18,9	30,1	24,9	59,5
	2.1	Descriptive	6	11,3		13,7	
sketching	1.3	Archived	2	3,7	26,5	3,6	
	1.2	Selective	7	13,2		13,1	
	1.1	Ad-hoc	2	3,7		4,1	
	1.x	Unaware of modeling	3	5,6		-	
none	0.1	Bad experienced	4	7,5	11,2	11	
	0.0	Not experienced	2	3,7			

Notice that, all quantitative results (i.e., survey and interview results) are depicted with their ratio in Table 2 and pattern distributions were similar in both studies.

D. Limitations and Threats to Validity

We discuss the possible validity concerns [19] and also the steps that we have taken to minimize or mitigate them.

When people feel being evaluated based on what they think, they might deflect their answers. In order to mitigate this, interviewees were informed that our motivation was to take a snapshot of the embedded software industry and that we will not collect any identifying information.

In order to prevent any misunderstanding in the terminology, we performed a pilot phase in which, we met with several practitioners to assess their common understanding of MDE, MDD, MBE and sketching. During the interview, we tried to reduce the threat related with these definitions. If we used only the terminology and did not use direct observations, this issue would stay as a potential threat, e.g., an interviewee might in fact uses sketching even though s/he states to use no modeling (as in the hidden patterns). However, face-to-face investigations and improvisations on further questions helped to understand the actual modeling approaches of the stakeholder. After the interview, if possible (due to time constraints), we showed the taken notes and repeated the critical parts to the interviewee to give an opportunity for clarification of their answers.

We collected our data from different sources (e.g., different countries, industrial sectors, SE roles etc.) in order to avoid mono-operation bias [17]. However, it cannot be guaranteed whether any of the interviewees participated the survey or not since the survey was completely anonymous [5]. Nevertheless, note that even if they have participated in the survey, when the interview participant number is compared to the survey (e.g., ~8%), a threat to internal validity would be limited.

IV. Conclusion and Future Directions

With the help of this study, the different modeling approach patterns in embedded software development projects were better understood; and the hidden patterns were identified with deeper and more personalized modeling experiences.

We observed that some interviewees (depending on their modeling characteristics such university degree, where/how modeling was learned and hardware closeness) have some resistances and misbeliefs for modeling. Some embedded software professionals think that software modeling is only done with a tool via some (formal?) UML drawings; however software modeling is not restricted with UML and it also includes sketching or DSLs without UML diagrams.

By analyzing the common/different characteristics and applying merging techniques between some patterns in the final set (Table 2) to better guide stakeholders with necessary and sufficient process & tool improvements for an effective modeling approach, we have already defined six modeling cultures in embedded software development projects [10]. Based on these findings, we created a characterization model, which identifies and defines a modeling stakeholder's pattern and culture as commonsense practices by presenting what the similar profiles in the embedded domain is doing while modeling (via the database constructed with survey data) [10]. This characterization model is the first wide-coverage model of software modeling characteristics for embedded software development projects built on extensive input from the industry.

We would like to study technical and social factors that influence the adoption of modeling approach patterns, specifically the effect of understandability and organizational resistance [20].

Acknowledgment

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

References

[1] D. Akdur and V. Garousi, "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)*, 2015.

[2] D. Akdur, V. Garousi, and O. Demirörs, "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.

[3] R. Heldal, P. Pelliccione, U. Eliasson, J. Lantz, J. Derehag, and J. Whittle, "Descriptive vs prescriptive models in industry," in *ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, France, 2016.

[4] G. M. Weinberg, *Quality software management (Vol. 1): systems thinking*: Dorset House Publishing, 1992.

[5] D. Akdur, V. Garousi, and O. Demirörs, "MDE in embedded software industry, Technical Report," *METU II-TR-2015-55, https://dx.doi.org/10.6084/m9.figshare.4262990,* 2015, Last accessed: Nov. 27, 2016.

[6] D. Akdur, O. Demirörs, and V. Garousi, "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] J. Cabot. (2009). *Relationship between MDA,MDD and MDE*. Available: http://modeling-languages.com/relationship-between-mdamdd-and-mde/

[8] N. A. Karagoz and O. Demirors, "Conceptual Modeling Notations and Techniques," in *Conceptual Modeling for Discrete-Event Simulation*, ed, 2010.

[9] D. Akdur, V. Garousi, and O. Demirörs, "MDE in embedded SW industry-Raw survey data," *https://dx.doi.org/10.6084/m9.figshare.4262972,* 2015, Last accessed: Nov. 27, 2016.

[10] D. Akdur, "Modeling Patterns and Cultures of Embedded Software Development Projects," *Thesis, Doctor of Philosophy (PhD), Information Systems, Middle East Technical University (METU), www.researchgate.net/publication/322701453_Modeling_Patterns_and_Cultures_of_Embedded_Software_Development_Projects,* Feb. 1, 2018.

[11] "pattern," ed: Cambridge Dictionary, 2017.

[12] B. P. Douglass, *Real-Time Design Patterns : robust scalable architecture for Real-time systems*. Boston, MA: Addison-Wesley, 2003.

[13] A. G. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*: Addison-Wesley Longman Publishing Co., Inc., 2003.

[14] M. Petre, "UML in practice," in *35th International Conference on Software Engineering (ICSE)*, 2013, pp. 722-731.

[15] J. Greenfield, K. Short, S. Cook, and S. Kent, *Software Factories - Assembling Application with Patterns, Models, Frameworks and Tools*: Wiley Publishing, 2004.

[16] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*: Wiley Publishing, 2012.

[17] D. Akdur, O. Demirörs, and B. Say. (2018, Last accessed: May 24, 2018). *Online Appendix: Interviews on modeling approach patterns*. Available: www.researchgate.net/publication/325344314_Appendix_to_Towards_Modeling_Patterns_for_Embedded_Software_Industry_Feedback_from_the_Field

[18] E. Juliot, "Model Driven Software Development 2.0," in *International Advanced Topics in Software Engineering* İstanbul, Turkey, 2014.

[19] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*: Springer Berlin Heidelberg, 2012.

[20] O. Kilic, B. Say, and O. Demirors, "Cognitive aspects of error finding on a simulation conceptual modeling notation," in *23rd International Symposium on Computer and Information Sciences*, 2008, pp. 1-6.