Practitioners' Experiences on Developing Graphical Modeling Editors: A Survey

Mert Ozkaya[1], Kamran Musayev[2] and Mehmet Alp Kose[3]
[1]*Yeditepe University, Department of Computer Engineering, Istanbul, Turkey*
[2]*DFDS, Istanbul, Turkey*
[3]*Independent Researcher, Istanbul, Turkey*

Keywords: Survey, Practitioners, Meta-Modeling, Modeling, Graphical Modeling Editors.

Abstract: Graphical modeling editors used for modeling and processing any information can be developed using either programming technologies (e.g., software libraries and frameworks) or meta-modeling technologies. However, with the existing literature, it is not clear which technique is popular and what motivate and demotivate practitioners using those techniques. In this paper, we conducted a survey among 76 practitioners (with 52 acceptable responses) to understand their experiences on developing graphical modeling editors. The survey led to interesting results. The top motivation for developing editors is the model-driven engineering and model transformation. 62% of the participants use meta-modeling technologies for developing editors, while the rest use programming languages. Sirius is the top-used meta-modeling technology, while C# and Python are the top-used programming languages. The participants using programming languages emphasized the reduced learning-curve with programming and advanced development platforms for developing portable editors. Many of those participants have no idea about meta-modeling. The participants using meta-modeling technologies revealed the huge time and effort gain with no-code editor development. Also, enhanced maintenance of editors by just changing the meta-model without writing code is considered important. However, those practitioners state challenges on the meta-modeling technologies' support for extensibility and customisation, developers' community, and complex meta-modeling.

1 INTRODUCTION

Models are considered as the abstract representations of any real systems for understanding and reasoning about systems (Seidewitz, 2003; Kent, 2002; Kühne, 2006). With models, complex systems can be described from different viewpoints at a high-level of abstraction. Today, modeling is used in diverse industries, including defense/military, avionics, automotive, finance, telecommunications, manufacturing, and logistics, so as to improve their software development and businesses. Indeed, as Rumbaugh stated in (Rumbaugh et al., 1999), models can be used for various purposes including the precise understanding and communications of the domain knowledge, making architectural design decisions, separating design from requirements, generating various useful business products, accessing and manipulating information about complex systems, exploring alternative solutions, and managing complexity.

Models can be specified using modeling languages (Harel and Rumpe, 2000), which offer textual/visual notation sets for specifying models at varying degree of abstractions. Using meta-modeling technologies, such as Eclipse-based tools (e.g., Sirius (Viyović et al., 2014) and Xtext (Bettini, 2013)), Metaedit+ (Kelly et al., 2013) and MPS (Pech et al., 2013), one can easily define a textual/graphical domain-specific modeling language (DSML) that focuses on solving any domain-specific problem with modeling. Indeed, meta-modeling technologies provide frameworks for defining the language concepts, relationships, rules and constraints (i.e., the meta-model definition) along with the symbols corresponding to those concepts (if graphical language to be developed). Then, a modeling editor is generated automatically for specifying models in accordance with the language definitions. Also, any tools that process models can easily be developed and integrated into the editors through which models can be processed automatically for many useful operations such as transformation (e.g., code generation) and model

Ozkaya, M., Musayev, K. and Kose, M.
PractitionersâĂŹ Experiences on Developing Graphical Modeling Editors: A Survey.
DOI: 10.5220/0012062400003538
In *Proceedings of the 18th International Conference on Software Technologies (ICSOFT 2023)*, pages 276-286
ISBN: 978-989-758-665-1; ISSN: 2184-2833

analysis (e.g., checking models for pre-defined rules).

An alternative way for developing graphical modeling editors can be using programming languages (e.g., Java, C#, and Python) and their supporting technologies (e.g., advanced development platforms, GUI libraries, and frameworks) for developing graphical editors. Some examples are Java's swing library[1] and Python's tkinter[2].

Given the meta-modeling and programming technologies that can be used for developing graphical modeling editors, it is however not so clear to what extent those technologies are used in different industries. Indeed, as discussed in Section 2, the existing surveys on modeling and meta-modeling do not aid in understanding practitioners' motivations for developing graphical modeling editors, their choices of using meta-modeling technologies or programming technologies, the reasons that make practitioners prefer either of the technologies and the reasons that make them not prefer, and any challenges faced with. Therefore, we intended to design and execute a practitioners survey in this study so as to learn practitioners' experiences with developing graphical editors.

We believe that the survey results will be useful for many stakeholders. Meta-modeling tool vendors can improve their tools for better addressing the needs of practitioners and the challenges faced with. Programming technology providers (e.g., framework developers) can also get convinced about the lack of interest on the programming technologies and figure out how the existing libraries and frameworks can be improved to address the needs of practitioners. Practitioners can gain insights on alternative technologies for building editors (i.e., strong and weak points) and thus better make decisions on which technology to prefer for their problems under specific constraints.

In the rest of the paper, we firstly discuss similar works in the literature. Then, we introduce the research methodology for our survey study. Next, we give the analysis of the survey responses. Lastly, we discuss the key findings and the threats to the validity of the survey results.

2 RELATED WORK

The literature includes several industrial surveys on modeling and meta-modeling. The existing survey studies aid in learning

(i) practitioners' experiences with modeling in particular domains, e.g., embedded systems (Akdur et al., 2018),

(ii) the extent to which modeling is used in particular countries (e.g., (Torchiano et al., 2011; Agner et al., 2013)),

(iii) the particular aspects of modeling (e.g., variability modeling (Berger et al., 2013), UML usage (Ozkaya and Erata, 2020a), and formal modeling (Ozkaya, 2018b)),

(iv) practitioners challenges in modeling and meta-modeling (Ozkaya and Akdur, 2021; Ozkaya and Erata, 2020b), and

(v) practitioners' experiences with the modeling languages (Malavolta et al., 2013).

Moreover, the literature also includes analytical studies that analyse and compare the existing modeling languages and tools with regard to some requirements of interest. In several works such as (Cabot and Teniente, 2006; Pérez-Medina et al., 2007), the authors analysed a set of well-known modeling tools for a specific sub-set of requirements. In (Ozkaya, 2019), Ozkaya analysed UML-based modeling tools for a set of requirements that are important for practitioners. In (Ozkaya, 2018a), Ozkaya analysed 124 different architectural modeling languages for the practitioners' needs. In (El Kouhen et al., 2012; Erdweg et al., 2015; Kern et al., 2011), the authors analysed the existing meta-modeling technologies.

However, none of the existing surveys and analytical studies consider practitioners' experiences with the graphical modeling editors, which is the main focus of the study discussed in this paper.

3 RESEARCH METHODOLOGY

We followed the online survey method in our study and thus reached our participants remotely over internet so as to collect their responses and analyse them in the quickest way possible (Groves et al., 2009).

3.1 Research Questions

In our survey, we investigate three research questions to achieve the paper goal introduced in Section 1.

RQ1: What motivate practitioners for developing graphical modeling editors? The goal here is to understand the reasons that lead practitioners to develop graphical modeling editors. To this end, we intend to learn why practitioners develop graphical modeling editors and which other facilities (e.g., code generation, model analysis, simulation, and collaborative modeling) practitioners need to perform. We also intend to learn the most and least important reasons.

RQ2: Which techniques and technologies practitioners prefer for the graphical modeling editor de-

[1] https://docs.oracle.com/javase/6/docs/api/

[2] https://docs.python.org/3/library/tkinter.html

velopment and their reasons for using those techniques? We consider two techniques for the graphical modeling editor development - the programming languages and meta-modeling technologies. Our goal is to understand which technique(s) are preferred more. Also, we intend to understand what make practitioners prefer (and not prefer) the meta-modeling technologies (and programming technologies).

RQ3: What are the challenges that practitioners face with while developing graphical modeling editors? In this research question, the goal is to understand any challenges that practitioners face with while using either of the technologies (i.e., programming technologies or meta-modeling technologies) for the graphical modeling editor development.

3.2 Survey Design

To design our survey, we used our expertise on modeling, modeling languages, and meta-modeling (Ozkaya, 2018b; Ozkaya and Akdur, 2021) and examined the past surveys discussed in Section 2. So, we proposed a list of questions in a draft form. We performed a pilot study with a group of practitioners and academics who are expert on modeling and meta-modeling. We got valuable feedback on the structuring of the sections and questions, the answer types (i.e., free-text, single choice, multiple choices), ambiguous questions/answers, missing questions/answers and duplicate answers. Finally, we ended up with the survey questions given in Table 1. The questions 1-4 are for learning the demographic information. The questions 5-6 are for addressing the research question RQ1 given in Section 3.1. The questions 7-10 and 12-14 are for addressing the research question RQ2. The questions 11 and 15 are for addressing the research question RQ3.

The profile question about the participants' experiences and the participants' frequency of developing editors are single-answered questions. To maximise the precision here, we ask the practitioners to choose one of the related set of answers, e.g., the question 3 for learning the participant experience (*None*, *Less than 2 years*, *2-5 years*, *6-10 years*, and *10+ years*). The questions for understanding the motivations, and reasons for preferring or not preferring any technologies are multiple-answer questions and supported with pre-determined answer lists. We determined the answer lists for each multiple-answer question through our expertise and the pilot study. Note that participants can type their own answers unless the existing answer list includes their desired answer. Lastly, the questions for understanding any challenges are free-text questions, which promote the participants to type their answers freely. We analyse any free-text answers using the coding strategy (Popping, 2015). That is, we firstly check if the answer is consistent with the question. If so, we check if the answer is close to any other free-text answers, and thus we either consider the answer as a new answer or increment the frequency of the existing answer.

3.3 Survey Execution

Our survey has been accessible online via *Google forms* for 2 months in between January 2023 and February 2023. We shared the survey link with several people involved in the graphical editor development in diverse industries. In executing the survey, we initially used our personal contacts whom we know through our consultancy services, R&D projects, and past/present work experiences. In total, we sent e-mails to approximately 180 individuals. We also used *Google Scholar* to determine any practitioners who have contributed to the well-regarded conference/journal papers that are associated with such relevant topics as software and systems engineering, modeling, meta-modeling, and modeling languages, compiler/parser design and development, and modeling toolsets. Besides, we sent posts to several mailing-lists that we think are related to the graphical modeling editor development. These include the Eclipse modeling platform mailing-lists where we could reach developers with meta-modeling experiences (e.g., sirius-dev, graphiti-dev, papyrus-rt-dev, emf-dev, emft-dev, agileuml-dev, papyrus-ic, etc.) and the mailing lists where we could reach developers (e.g., Netbeans mailing list, and IEEE architecture description). Last but not least, we left a message on several related *Linkedin* groups including DSM forum, INCOSE, domain-specific languages, software developers, engineers, and programmers, software/technology, and software design patterns and architecture, and software engineering professionals, and MBSE. We also sent hundreds of messages to the individuals who are enrolled on those linkedin groups and we think are likely to show interest on our survey.

3.4 Survey Sampling

In our survey, we performed the non-probability sampling technique as we did not have the chance for reaching each practitioner in the population (Fricker, 2008). That is, we sent e-mails to hundreds of different practitioners whom we know as discussed in Section 3.3. Also, to minimise any biases here due to the non-randomness, we shared the survey link

Table 1: The survey questions.

Res. Que.	Survey Questions	Multiple Answers	Free Text	Single Answer
Profile	1-Which industry(ies) do you work in?	X	X	
	2-What is (are) your current job position(s)?	X	X	
	3-How many years of experience do you have in the sector?			X
	4-Do you develop graphical modeling editors?			X
RQ1	5-Which other tool(s) or services do you integrate into your graphical modeling editor?	X	X	
RQ1	6-Which of the following motivation(s) make you develop a graphical modeling editor?	X	X	
RQ2	7-How do you develop a graphical modeling editor and its supplementary tools (e.g., validators, generators, exporters/importers)?			X
RQ2	8-Which programming languages do you use for developing a graphical modeling editor?	X	X	
RQ2	9-What is(are) the reason(s) that make you prefer programming languages for developing a graphical modeling editor?	X	X	
RQ2	10-What is(are) the reason(s) that make you NOT prefer meta-modeling technologies for developing a graphical modeling editor?	X	X	
RQ3	11-Please tell us any challenges that you face with while using programming languages for the graphical modeling editor development.		X	
RQ2	12-Which meta-modeling technologies do you use for developing a graphical modeling editor?	X	X	
RQ2	13-What is(are) the reason(s) that make you prefer meta-modeling technologies for developing a graphical modeling editor?	X	X	
RQ2	14-What is(are) the reason(s) that make you NOT prefer programming languages for developing a graphical modeling editor?	X	X	
RQ3	15-Please tell us any challenges that you face with while using meta-modeling technologies for the graphical modeling editor development.		X	

across several actively-used mailing lists and *Linkedin* groups where each interested practitioner who are enrolled has equal chance to participate.

3.5 Data Analysis

We have collected 76 responses from diverse industries. Among those, 2 participants stated in question-3 that they do not have any experiences in industry and thus we directed the participants to submit the form without filling any questions. Indeed, in our survey, we are interested in learning from the practitioners who have some experience about developing editors in industry. Moreover, 22 responses belong to the participants who stated in question-4 that that they have never developed a graphical editor before. So, we directed those participants to submit the survey too. So, we ended up with 52 acceptable responses that have been analysed as discussed in Section 4. To analyse the data here, we performed the voting approach for each question answer and made statistical inferences using Google form and MS Excel.

4 ANALYSIS OF RESULTS

In this section, we give the analysis of the survey responses for each question.

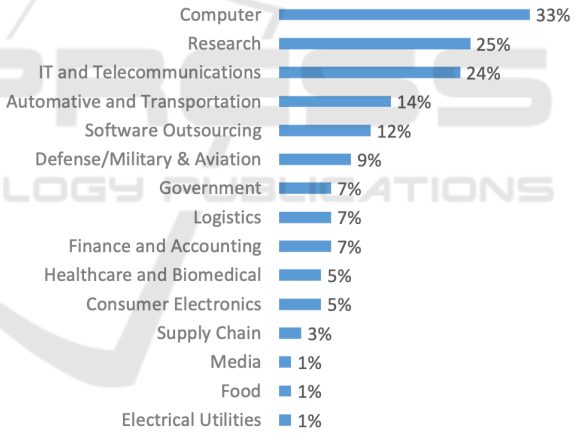

Figure 1: Participants' work industries.

4.1 Profile Questions

In this section, we analyse the responses given for the questions 1-4.

The industries in which the participants work are displayed in Figure 1. So, top popular industry that show interest on the survey is computer (33%), which is followed by the research industry (25%) including the participants from the research centers and universities and IT and Telecommunications (24%).

As shown in Figure 2, nearly half of the participants (43%) hold the software developer/programmer positions in their company. Also, 30% of the participants are researchers.

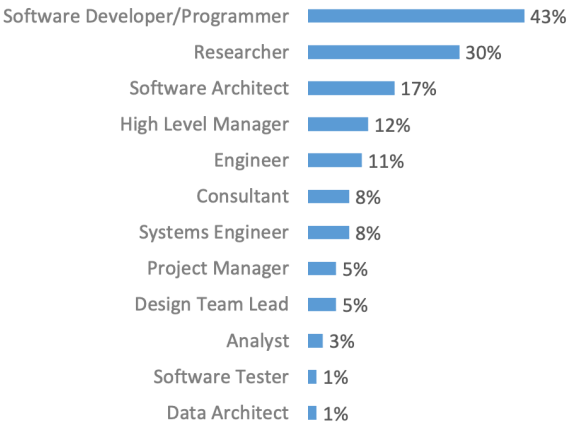

Figure 2: Participants' job positions.

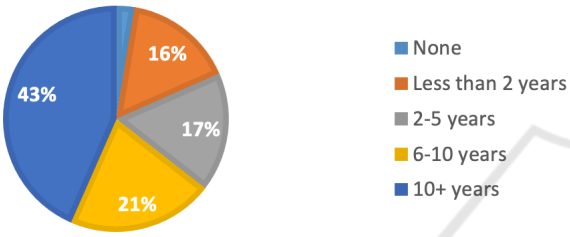

Figure 3: Participants' experience in the sector.

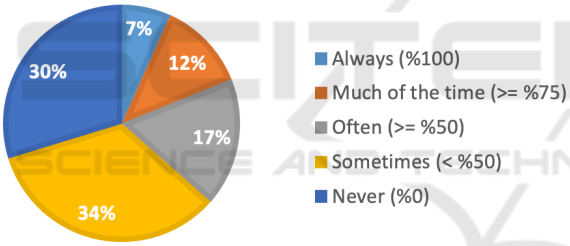

Figure 4: Participants' experience in developing graphical modeling editors.

According to Figure 3, most of the participants are highly experienced in their sector - 43% have 10+ years of experiences and 21% have 6-10 years of experiences. Note that a very small number of the participants who have no experiences in their sector have been directed to submit the survey without answering any of the technical questions so as to minimise any biases.

As shown in Figure 4, 70% of the the practitioners who participated in the survey develop (or developed) graphical modeling editors to some extent. Those participants (30%) who never develop graphical modeling editors have been directed to submit the survey so as to minimise any biases.

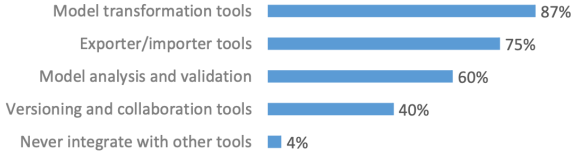

Figure 5: The types of tools that the participants integrate into their graphical modeling editors.

4.2 Developing Graphical Modeling Editors

In this section, we analyse the questions 5-7, which have been responded by the participants who indicated that they develop(ed) graphical modeling editor(s).

As indicated in Figure 5, most of the participants developing graphical modeling editors (87%) tend to integrate their editors with model transformation tools (e.g., code generators) so as to enable the automated transformation of models specified with the editors into some useful artefact. Integrating editors with some exporter/importer tools (e.g., exporting and importing models in XML, HTML file formats and exporting models in picture formats) is the second-top motivation for the participants (75%). While using model analysis and validation tools (e.g., checking user errors for incomplete, inconsistent, and incorrect models) together with the editors is also important for many participants, versioning models and accessing the models collaboratively are rarely preferred.

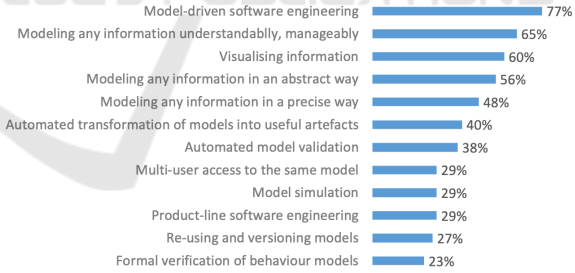

Figure 6: Participants' motivations for developing graphical modeling editors.

Figure 6 gives the participants' motivations for the graphical modeling editor development. So, most of the participants (77%) aim to perform the model-driven software engineering activities with their graphical editors, which promotes the specification of models, their analysis and transformation into useful artefact (e.g., code) (Kent, 2002). 65% of the participants develop editors because they want to model any information in an understandable and manageable way. Indeed, editors can be developed for specifying multiple-viewpoints models where different models that each focus on a separate concern can

easily be specified and related to each other (Rozanski and Woods, 2011). Also, sub-diagramming can be performed, where an element drawn can be clicked for specifying another model via the newly opening sub-editor. The third and fourth top motivations for the graphical modeling editor development are the need for visualising any information (e.g., using boxes and lines) (60%) and modeling any information abstractly (i.e., one of the main goals of modeling) (56%).

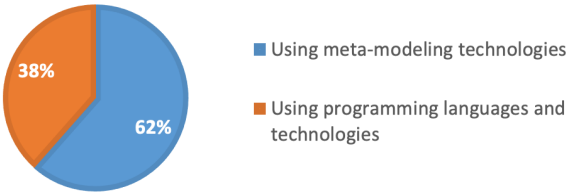

Figure 7: The technologies that the participants prefer to use for developing graphical modeling editors.

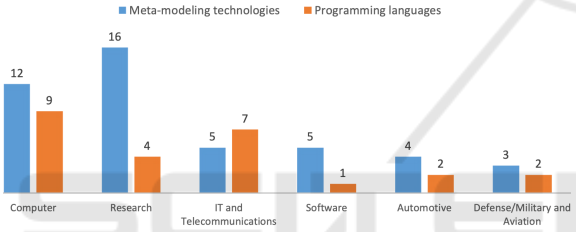

Figure 8: The industries that use the programming languages and meta-modeling technologies for developing graphical modeling editors.

Figure 7 shows how the participants prefer to develop the graphical modeling editors and their supporting tools (e.g., code generators, model validators, and exporter/importer tools). So, while 62% prefer to use the meta-modeling technologies (e.g., Metaedit+, Sirius, and MPS), the rest of the participants use the programming languages and technologies (e.g., Java, Python, and C++). Figure 8 shows the correlation between the participants' work industries and the technology that they use for developing graphical modeling editors. So, the biggest portion of the participants who use meta-modeling technologies are from the research industry while the programming languages are top-used by the computer industry.

4.3 Using Programming Languages

In this section, we analyse the responses given for the questions 8-11, which are concerned with the experiences of the participants who use programming languages to develop graphical modeling editors.

Figure 9 shows the programming languages used by the participants. So, the top-preferred programming languages are C# (40%) and Python (40%), which is followed by Java (30%).

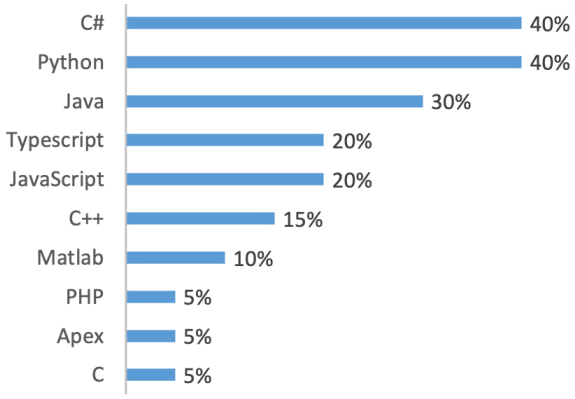

Figure 9: The programming languages that are preferred by the participants who use programming technologies for developing graphical modeling editors.

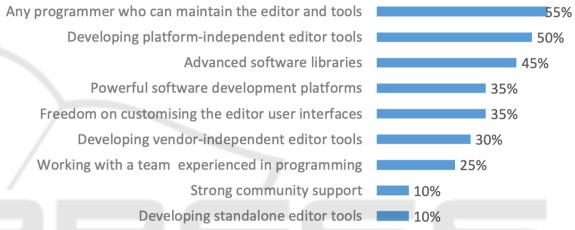

Figure 10: The reasons of the participants using programming languages for developing graphical modeling editors.

As Figure 10 shows, maintainability is the top reason of the participants using programming languages for developing graphical modeling editors (65%). Indeed, any programmers can essentially maintain the tool whenever needed and no any additional abilities (e.g., modeling and language engineering) and knowledge (e.g., meta-modeling) are needed. Also, the benefits that are gained with the programming languages such as platform-independency (e.g., jar applications) and advanced software libraries are found motivating by nearly half of the participants for the graphical editor development.

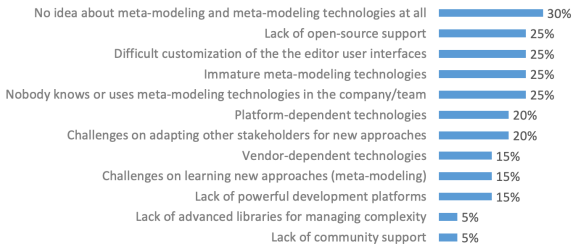

Figure 11: The reasons of the participants using programming languages for NOT preferring meta-modeling technologies.

In Figure 11, the reasons that make the participants using programming languages avoid meta-modeling technologies are addressed. 30% of the participants who use programming languages have no idea about meta-modeling technologies. 25% of the participants stated that the team in which they work have no knowledge on meta-modeling.

Some other issues selected by 25% of the participants are to do with meta-modeling technologies'

(i) lack of support for open-source development for being extended by the developers whenever needed,

(ii) lack of support for the customisation of the editor user interfaces as the meta-modeling technologies may not allow practitioners to change the user-interface layouts (e.g., changing the toolbar location in the editor) and

(iii) immaturity (e.g., being very new products).

The participants did not state any challenges regarding their experiences with the programming languages for developing graphical modeling editors.

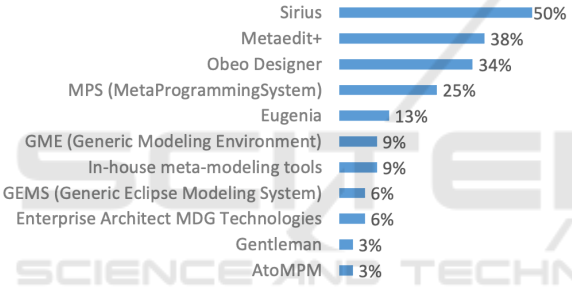

Figure 12: The meta-modeling technologies preferred by the participants for developing graphical modeling editors.

4.4 Using Meta-Modeling Technologies

In this section, we analyse the responses given for the questions 12-15, which are concerned with the experiences of the participants who use meta-modeling technologies to develop graphical modeling editors.

As Figure 12 shows, the top-used meta-modeling technology for developing graphical modeling editors is Sirius[3] (50%), which is based on Eclipse modeling framework (EMF). Sirius is followed by Metaedit+ (35%), offered by Metacase[4]. Obeo Designer[5] that is based on Sirius is used by 32% of the participants and MPS[6] used by 23% of the participants. The rest of the technologies including Eugenia[7], GEMS[8], GME[9], AtoMPM[10], Gentleman[11], and Enterprise Architect's model-driven generation technologies[12] are rarely used by a few participants (1-4) only. Note that a few of the participants (9%) indicated that they build their own tools to build graphical editors.

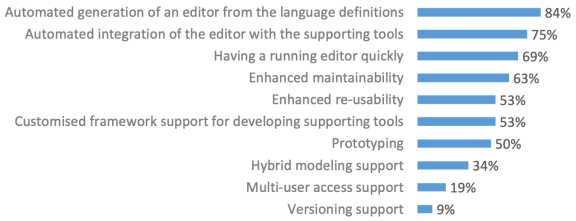

Figure 13: The reasons of the participants using meta-modeling technologies for developing graphical modeling editors.

As Figure 13 shows, the top-reason that make the participants use meta-modeling technologies is the automated generation of an editor from the language definitions (i.e., meta-model) without writing single line of code (84%). The automated integration of the editor with code generators and model validators is second top-reason here (77%). Indeed, the meta-modeling technologies offer frameworks for developing model transformation tools (e.g., code generators) and automatically integrate the transformation tools with the editor - no any effort required for integration here. Also, having a running editor quickly without coding at all for the editor implementation & testing and maintaining the editor (e.g., adding/removing concepts) without coding are quite popular reasons for using meta-modeling technologies (65-68%). On other hand, meta-modeling technologies' support for the collaborative modeling (i.e., multi-user access), hybrid modeling (i.e., modeling using different types of notation sets e.g., visual, textual, and tabular), and versioning (e.g., versioning models and meta-models) are rarely considered by the participants.

In Figure 14, we give the reasons that make the participants who use meta-modeling technologies avoid programming languages for the graphical modeling editor developments. So, most of those participants (84%) find the programming languages requiring too much time and effort for the editor development. Another crucial issue (63%) is to do with the difficulties in maintaining the editor at code level whenever some changes are required (e.g., adding new concepts and relationships).

The participants stated several of their challenges faced while using meta-modeling technologies and

[3]https://www.eclipse.org/sirius/
[4]https://www.metacase.com/
[5]https://www.obeodesigner.com/
[6]https://www.jetbrains.com/mps/
[7]https://www.eclipse.org/epsilon/doc/eugenia/
[8]https://wiki.eclipse.org/GEMS
[9]https://webgme.org/
[10]https://atompm.github.io
[11]https://github.com/geodes-sms/gentleman
[12]https://sparxsystems.com/resources/mdg_tech/

Figure 14: The reasons of the participants using meta-modeling technologies for NOT preferring programming languages.

the top ones are

(i) the inadequate support for customising the editor user interfaces,

(ii) the inadequate support for specifying complex meta-model concepts and transformation algorithms, and

(iii) lack of community support (e.g., mailing lists, tutorials, forums, case-studies, etc.). Other challenges include the lack of support for

(i) the development of web-based modeling editors,

(ii) the development of editors that enable the use of textual and graphical notations together,

(iii) integrating the editors with other systems and technologies,

(iv) concurrent modeling and meta-modeling,

(v) managing changes on the meta-model,

(vi) re-using shared models acting as patterns,

(vii) defining editors using an existing editor (or its parts),

(viii) understanding the needs of the users and reflecting them on the meta-model definitions that the tools are based on,

(ix) the change management (i.e., changing meta-models).

5 DISCUSSION

Our survey attracted 52 acceptable responses from the interested practitioners of diverse industries. The top industries from which we attracted participants are computer, research, and IT and telecommunications. Nearly half of the participants work as a software developer/programmer, and 30% of the participants work as a researcher.

In the rest of this section, we discuss the key findings that we obtained with the analysis of the survey responses.

Model-Driven Engineering is the Practitioners' Top-Motivation. Model-driven software engineering - i.e., specifying, analysing, and transforming models - is the top-motivation of the participants developing graphical modeling editors (77%). Almost all the participants develop graphical modeling editors and perform the activities of model-driven engineering including model specification, model analysis, and model transformation. Also, most of the participants (87%) integrate transformation tools to the editors, revealing that model transformation is the participants' top interest concerning the activities involved in model-driven engineering.

Meta-Modeling Technologies are Used Much More Than Programming Languages. 62% of the participants who develop graphical modeling editors do so using meta-modeling technologies, while the rest use traditional programming technologies. The biggest portion of the participants who use meta-modeling technologies are from the research industry.

In the rest of this section, we discuss the practitioners' thoughts on developing graphical modeling editors. We firstly consider the practitioners who use programming languages (PP, standing for programmer practitioner) and then the practitioners using meta-modeling technologies (MP, standing for meta-modeler practitioner).

PP: Programming Languages Require no Learning Curve. Concerning the participants using programming languages for developing graphical modeling editors, the top industries are the computer and IT and telecommunications. The top-preferred programming languages here are C# (40%), Python (40%), and Java (30%). The participants' top reason for using programming language is to do with the maintainability of the editors that can always be changed and corrected by any programmer without any meta-modeling knowledge (65%). Indeed, the survey results show that many participants preferring programming languages have no idea about meta-modeling technologies at all and consider meta-modeling technologies as causing difficult-to-customise editors.

PP: Programming Technologies are Found Advanced and Portable. Many of the participants who prefer programming technologies (45%) give emphasis on the advanced software libraries that come with the programming technologies and facilitate the editor development via re-usuable software modules and patterns. Also, many of the participants (50%) care about building platform-independent editors that can run on any operating systems (e.g., jar application), which could be possible if the programming technologies were used.

PP: Meta-Modeling Technologies are Difficult to Extend. The main concern of the participants who prefer programming languages over the meta-

modeling technologies is about the meta-modeling technologies' limited extension support for better addressing the stakeholder needs. Indeed, customising the editor user interfaces (e.g., changing the layout, fonts, and language) may not always be possible with the meta-modeling technologies. Also, the participants pointed out the lack of support for the open-source code availability, which could prevent the meta-modeling technologies to be extended when needed.

MP: Meta-Modeling Technologies Save a Lot of Time and Effort. The top-used meta-modeling technology by meta-modelers is Sirius (50%), followed by Metaedit+ (38%). Most of the meta-modeler practitioners (75-84%) use meta-modeling technologies as they save huge time and effort with the automation support - automated generation of editors from the language definitions and integration of the editor with any code generators and model validators.

MP: Programming Languages Require Too Much Time and Effort to Develop and Maintain Editors. Almost all the meta-modeler practitioners (84%) agree that programming languages require too much time and effort for developing editors by writing code. Also, maintaining the editor at code-level is not found easy either (63%).

MP: Challenges Faced About the Meta-Modeling Technologies. The meta-modeler participants face with some issues while using the meta-modeling technologies. The commonly faced challenges are the inadequate support for customising the editor user interfaces, specifying complex meta-model concepts and transformation algorithms, the meta-modeler community (e.g., mailing lists, tutorials, forums, case-studies, etc.).

6 THREATS TO VALIDITY

We considered the construct, internal and external threats to the validity of the survey results (Wohlin et al., 2012). To minimise any threats against the construct validity, we gave clear explanations at the beginning of the survey indicating the purpose of the survey and anonymity of the personal data. We also performed a detailed pilot study to ensure that the survey questions are consistent and complete with regard to the research questions of the study. To minimise any internal validity threats, we did our best to choose the participants as randomly as possible so as to avoid any unknown variables that could affect the results. Indeed, we shared the survey in many social platforms (e.g., diverse linkedin groups) and give each enrolled and interested participant an equal chance to participate. Lastly, the threats to the external validity were minimised by reaching practitioners with various profiles that vary in terms of their industries, job positions, and experience levels.

7 CONCLUSION

In this paper, we conducted a survey among practitioners so as to understand their experiences in developing graphical modeling editors. The survey attracted 76 participants from diverse industries holding diverse job positions and 52 of them were acceptable for analysis.

According to the survey results, model-driven engineering is the participants' top-motivation for developing graphical modeling editors and it is also revealed that most participants use editors to not only specify some models but also transform the models into useful artifacts (e.g., software code, documentation, etc.). Given two techniques for developing graphical modeling editors (i.e., programming languages vs. meta-modeling technologies), meta-modeling technologies are much more preferred by the participants (62%). Note that it is the research industry (and thus researchers) who show the greatest interest on the meta-modeling technologies. Those participants using programming languages prefer so because programming languages require no learning curve for them and are supported with advanced software libraries. Also, editors developed with programming languages can be used in any platform as a standalone application portably. The same participants find meta-modeling requiring a steep learning curve and the corresponding technologies as difficult to use due to such limitations as customising the editor user interfaces and extending the technologies when needed. The participants using meta-modeling technologies find the programming technologies as requiring too much time and effort for the graphical editor development and reducing maintainability due to the need for making all changes at code-level which is error-prone. Lastly, the participants using meta-modeling technologies face with several challenges too, including the inadequate support for editor customisation, specifying complex meta-models and transformation algorithms and the developers community (e.g., forums and tutorials).

In the near future, we are planning to validate the survey results via an empirical study. That is, we will determine two groups of practitioners where one group has intermediate level of experience on programming languages and the other group on the meta-modeling technologies. Each group will be given

the same case-study and asked to develop a graphical modeling editor. So, we will observe the effectiveness and productiveness of the groups and see to what extent the observed data justify the survey results.

REFERENCES

Agner, L. T. W., Soares, I. W., Stadzisz, P. C., and SimãO, J. M. (2013). A brazilian survey on uml and model-driven practices for embedded software development. *J. Syst. Softw.*, 86(4):997–1005.

Akdur, D., Garousi, V., and Demirörs, O. (2018). A survey on modeling and model-driven engineering practices in the embedded software industry. *Journal of Systems Architecture - Embedded Systems Design*, 91:62–82.

Berger, T., Rublack, R., Nair, D., Atlee, J. M., Becker, M., Czarnecki, K., and Wasowski, A. (2013). A survey of variability modeling in industrial practice. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, VaMoS '13, pages 7:1–7:8, New York, NY, USA. ACM.

Bettini, L. (2013). *Implementing Domain-Specific Languages with Xtext and Xtend*.

Cabot, J. and Teniente, E. (2006). Constraint support in MDA tools: A survey. In Rensink, A. and Warmer, J., editors, *Model Driven Architecture - Foundations and Applications, 2nd European Conference, ECMDA-FA 2006, Bilbao, Spain, July 10-13, 2006, Proceedings*, volume 4066 of *Lecture Notes in Computer Science*, pages 256–267. Springer.

El Kouhen, A., Dumoulin, C., Gérard, S., and Boulet, P. (2012). Evaluation of Modeling Tools Adaptation. Technical report.

Erdweg, S., van der Storm, T., Völter, M., Tratt, L., Bosman, R., Cook, W. R., Gerritsen, A., Hulshout, A., Kelly, S., Loh, A., Konat, G. D. P., Molina, P. J., Palatnik, M., Pohjonen, R., Schindler, E., Schindler, K., Solmi, R., Vergu, V. A., Visser, E., van der Vlist, K., Wachsmuth, G., and van der Woning, J. (2015). Evaluating and comparing language workbenches: Existing results and benchmarks for the future. *Comput. Lang. Syst. Struct.*, 44:24–47.

Fricker, R. D. (2008). Sampling methods for web and e-mail surveys. *The SAGE handbook of online research methods*, pages 195–216.

Groves, R. M., Fowler Jr, F. J., Couper, M. P., Lepkowski, J. M., Singer, E., and Tourangeau, R. (2009). *Survey methodology*. John Wiley & Sons, 2 edition.

Harel, D. and Rumpe, B. (2000). Modeling languages: Syntax, semantics and all that stuff, part i: The basic stuff. Technical report, ISR.

Kelly, S., Lyytinen, K., and Rossi, M. (2013). Metaedit+ A fully configurable multi-user and multi-tool CASE and CAME environment. In *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*, pages 109–129.

Kent, S. (2002). Model driven engineering. In Butler, M. J., Petre, L., and Sere, K., editors, *Integrated Formal Methods, Third International Conference, IFM 2002, Turku, Finland, May 15-18, 2002, Proceedings*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer.

Kern, H., Hummel, A., and Kühne, S. (2011). Towards a comparative analysis of meta-metamodels. In *Proceedings of the Compilation of the Co-Located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*, SPLASH '11 Workshops, page 7–12, New York, NY, USA. Association for Computing Machinery.

Kühne, T. (2006). Matters of (meta-)modeling. *Software and Systems Modeling*, 5(4):369–385.

Malavolta, I., Lago, P., Muccini, H., Pelliccione, P., and Tang, A. (2013). What industry needs from architectural languages: A survey. *IEEE Trans. Software Eng.*, 39(6):869–891.

Ozkaya, M. (2018a). The analysis of architectural languages for the needs of practitioners. *Softw., Pract. Exper.*, 48(5):985–1018.

Ozkaya, M. (2018b). Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? *Information & Software Technology*, 95:15–33.

Ozkaya, M. (2019). Are the UML modelling tools powerful enough for practitioners? A literature review. *IET Softw.*, 13(5):338–354.

Ozkaya, M. and Akdur, D. (2021). What do practitioners expect from the meta-modeling tools? A survey. *J. Comput. Lang.*, 63:101030.

Ozkaya, M. and Erata, F. (2020a). A survey on the practical use of UML for different software architecture viewpoints. *Inf. Softw. Technol.*, 121:106275.

Ozkaya, M. and Erata, F. (2020b). Understanding practitioners' challenges on software modeling: A survey. *J. Comput. Lang.*, 58:100963.

Pech, V., Shatalin, A., and Voelter, M. (2013). Jetbrains mps as a tool for extending java. In *Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools*, PPPJ '13, page 165–168, New York, NY, USA. Association for Computing Machinery.

Pérez-Medina, J. L., Dupuy-Chessa, S., and Front, A. (2007). A survey of model driven engineering tools for user interface design. In Winckler, M., Johnson, H., and Palanque, P. A., editors, *Task Models and Diagrams for User Interface Design, 6th International Workshop, TAMODIA 2007, Toulouse, France, November 7-9, 2007, Proceedings*, volume 4849 of *Lecture Notes in Computer Science*, pages 84–97. Springer.

Popping, R. (2015). Analyzing open-ended questions by means of text analysis procedures. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, 128(1):23–39.

Rozanski, N. and Woods, E. (2011). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 2 edition.

Rumbaugh, J. E., Jacobson, I., and Booch, G. (1999). *The unified modeling language reference manual*. Addison-Wesley-Longman.

Seidewitz, E. (2003). What models mean. *IEEE Software*, 20(5):26–32.

Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A., and Reggio, G. (2011). Preliminary findings from a survey on the md state of the practice. In *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement*, ESEM '11, page 372–375, USA. IEEE Computer Society.

Viyović, V., Maksimović, M., and Perisić, B. (2014). Sirius: A rapid development of dsm graphical editor. In *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*, pages 233–238.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., and Regnell, B. (2012). *Experimentation in Software Engineering*. Springer.