Journal of Computer Languages 63 (2021) 101030

Contents lists available at ScienceDirect

Journal of Computer Languages

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

Reviews

What do practitioners expect from the meta-modeling tools? A survey

Mert Ozkaya [a,*], Deniz Akdur [b]

[a] Yeditepe University, Istanbul, Turkey
[b] Aselsan Inc., Ankara, Turkey

ARTICLE INFO

Keywords:
Meta-modeling tools
Language workbenches
Modeling
Modeling languages
Practitioners
Survey

ABSTRACT

Modeling languages are defined with a meta-model, which are specified using the meta-modeling tools that produce the editors for specifying models in accordance with the meta-models. In this paper, we aim to understand the top-used meta-modeling tools and practitioners' expectations and challenges with respect to different requirements. So, we designed and conducted a survey, which was responded by 103 practitioners. The questionnaire considers the notation, semantics, editor services, model-transformation, validation, testing, and composability requirements. Some of the results are as follows: the top-used meta-modeling tools are Sirius and GEMS, Metaedit+, Xtext, and Microsoft DSL tools respectively. The top-preferred language visualizations are diagrammatic or textual. Practitioners prefer the editors with free-editing mode and are not familiar with projectional editing that promotes hybrid modeling. Translational semantics definition (i.e., model-to-model and model-to-text) is more popular than the interpretative semantics definition. Importing/exporting models and meta-models is highly important for facilitating the communication and collaboration. Modeling and meta-modeling with re-use and their versioning are among the top-desired features. Practitioners are willing to integrate any validation tools (e.g., model checkers) to define the languages' semantical and structural validation rules and prove the model correctness. Lastly, defining the language semantics by composition and testing the semantics definitions are crucial for many practitioners. The survey also reveals many important challenges in each type of requirements. The survey results will be useful for meta-modelers in analyzing the meta-modeling tools and tool vendors in learning practitioners' expectations.

1. Introduction

Models are considered as the abstract descriptions of any real systems and may exist in multiple forms that each address a different aspect of systems [1–4]. Models are created for various purposes, including the facilitated communications among different stakeholders, analyzing the correctness and completeness of the abstract system descriptions, test-case generation, and code generation [5]. With the abstract models, practitioners can better understand complex systems, have the chance of analyzing the possible design decisions before implementation, and obtain high-quality code that is guaranteed to satisfy the design decisions. As Selic suggests [2], for a model to be worthwhile, it needs to possess five important characteristics. *(1)* A model needs to be abstract by focussing on a particular problem of a system and suppressing the rest of the details; *(2)* a model needs also to be understandable so that one can easily grasp the complex descriptions that are modeled abstractly; *(3)* a model needs to represent a real system in an accurate way; *(4)* a model may also promote the predictiveness of a real system, as models can be used to predict the quality properties of systems before building them; and, *(5)* a model needs to be inexpensive and one should be able to create abstract models with the least cost and effort possible.

Models can be created via modeling languages, which can either be general-purpose or domain-specific [6]. While general-purpose modeling languages (e.g., UML [7]) offer high-level notation sets that can be used to specify any types of systems, domain-specific modeling languages (DSMLs) offer specialized notation sets on particular domains (e.g., AADL for the embedded domain [8]). Modeling languages are based on meta-models, which state the language concepts and the syntactic and semantic rules that the models specified with those language concepts need to satisfy [1,9,10]. The language syntax describes the elements that can be used for creating models and the rules to be followed in using these elements. On the other hand, the language semantics describes the meaning of the language elements, which can be formulated using different techniques such as operational [11], denotational [12], axiomatic [13], and algebraic semantics [14].

A meta-model for a modeling language can be defined using the meta-modeling tools (aka language workbenches) [15]. With the meta-modeling tools, one can specify the language definitions in terms of the language syntax and semantics, build a modeling editor according to

* Corresponding author.
E-mail address: mozkaya@cse.yeditepe.edu.tr (M. Ozkaya).

https://doi.org/10.1016/j.cola.2021.101030
Received 9 January 2021; Received in revised form 22 February 2021; Accepted 26 February 2021
Available online 9 March 2021

Table 1
Examples of meta-modeling tools.

Meta-modeling Tool	Web-site
ADOXX	https://www.adoxx.org
ANTLR	https://www.antlr.org
Cameo	https://www.nomagic.com/products/cameo-systems-modeler
ConceptBase	http://conceptbase.sourceforge.net
Diagen	http://www2.cs.unibw.de/tools/DiaGen
Enterprise Value Architect (EVA)	http://www.inspired.org/eva-home
Fujaba	https://web.cs.upb.de/archive/fujaba
GEMOC	http://gemoc.org
GEMS (Generic Eclipse Modeling System)	https://sourceforge.net/projects/gems/
GME (Generic Modeling Environment)	https://www.isis.vanderbilt.edu/Projects/gme/
Graphiti	https://www.eclipse.org/graphiti
JastEMF	https://bitbucket.org/jastemf/
Melange	http://melange.inria.fr/
MetaDepth	https://metadepth.org
MetaEdit+	http://metacase.com/
MetaModelAgent	http://www.metamodelagent.com
Microsoft DSL Tools	https://docs.microsoft.com/en-us/visualstudio/modeling
MPS (MetaProgrammingSystem)	https://www.jetbrains.com/mps/
Sirius	https://www.eclipse.org/sirius
Spoofax	http://www.metaborg.org
Xtext	https://www.eclipse.org/Xtext
VMTS (Visual Modeling and Transformation System)	https://www.aut.bme.hu/Pages/Research/VMTS/Introduction
WebGME	https://webgme.org

the language definitions, and even generate tools for, e.g., model analysis and code generation purposes. As presented in Table 1, there are various meta-modeling tools that can be used for defining the language meta-models and automatically producing the necessary modeling tools (e.g., editors and model transformation tools which support the meta-model definitions). The existing meta-modeling tools may vary depending on their level of support for different kinds of requirements that are concerned with the language syntax and semantics definitions, editor services, model transformation, language extensibility, model analysis and validation, and being open-source or commercial.

1.1. Motivation and goal

Software modeling is essentially considered by industry as highly crucial for developing large and complex software systems, given its support for such concepts as the abstraction, separation of concerns, and early analysability of design decisions. Thanks to the existence of meta-modeling tools, practitioners may even develop their own domain-specific modeling languages and create models that are specific to their domain problems and develop the necessary toolset for processing the models according to their specific modeling goals. However, it is not yet clear as to what extent the meta-modeling tools are adopted by practitioners in different industries; and, it is still difficult to understand practitioner's expectations from the meta-modeling tools and any challenges that practitioners face with.

As discussed in Section 2, the literature includes several attempts at understanding practitioners' perspectives towards modeling. So, one can understand, e.g., *(i)* the existing modeling languages, their usage frequencies in industry, their weak and strong points, *(ii)* practitioners' challenges on modeling and modeling languages, *(iii)* the practical application of modeling in particular domains, and *(iv)* the analysis and comparison of a set of modeling and meta-modeling tools. However, there is a gap in the existing literature to understand practitioners' perspectives towards meta-modeling. Although there are many meta-modeling tools available in the market (see Table 1), there is no study which explores the attitudes of practitioners and when, how and why meta-modeling is used with possible challenges.

In this paper, the goal is to understand the practitioners' preferences among different meta-modeling tools, their expectations, and any challenges faced with. To achieve this, we designed and conducted a practitioner survey. The survey results are expected to be useful for anyone who consider developing their own DSMLs in understanding the top-used meta-modeling tools for different domains. Also, the tool vendors could use the survey results in learning the expectations of practitioners from the meta-modeling tools and any challenges encountered.

In our survey study, we addressed a number of important requirements for the meta-modeling tools and intended to learn practitioners perspectives towards the meta-modeling tools in terms of those requirements. To determine the meta-modeling tool requirements herein, we considered Erdweg et al.'s comprehensive feature model for the meta-modeling tools [16] and extended that with further categories of requirements that we deem important after a series of pilot studies conducted with the area experts. In the rest of this section, we discuss each category of requirements separately, which are concerned with the meta-modeling tools' support for *(i)* language definitions (notation and semantics), *(ii)* modeling editors, *(iii)* model transformation, *(iv)* language validation, *(v)* language testing, and *(vi)* language composability.

1.2. Categories for meta-modeling tool requirements

1.2.1. Language notation

The language notation is concerned with models' appearances to users. Languages may support such visualizations as diagrammatic, textual, tabular, trees, matrix, map, and hybrid. The diagrammatic visualization enables the model elements to be specified using graphical symbols. The textual visualization enables the model elements to be specified in terms of texts (e.g., writing code with the programming languages). The tabular visualization enables the model elements to be specified using a table editor that can be displayed as a table and edited by simply specifying cell values. The matrix-style visualization enables the model elements to be specified and edited in two axes where each cell in the matrix essentially indicates the relationships of the elements in the two axes. The map visualization enables the model elements to be specified with their location data and the distances among the elements are of particular importance. Lastly, the hybrid visualization supports multiple visualizations (e.g., textual, graphical, and tabular) that can be used for editing the same model in a synchronized way.

1.2.2. Language semantics

The language semantics can be categorized as interpretative or translational. The interpretative semantics promotes the execution of models without performing any translations into some intermediate formats. The translational semantics promotes the definition of the model translations into an intermediate format that can be executed.

The translational semantics can be either model-to-text and model-to-model. The model-to-text translation is to do with defining the language semantics in terms of the rules for the translations into some structured text notation such as source-code (e.g., Java, C, C++, and PhP). The model-to-model translation is to do with defining the language semantics in terms of the rules for the translation into a model with a different notation set (e.g., producing entity-relationship model from a UML class diagram).

1.2.3. Editor services

The editor services are concerned with the capabilities of the modeling editors that users can create using the meta-modeling tools. The editing mode that the meta-modeling tools support can be categorized as free-form and projectional. In the free-form editing, users edit a textual or graphical model that is stored persistently, and a persistent model may then be transformed into an abstract representation that can further be transformed into an executable representation. In the projectional editing, users may edit any projections of the model's abstract representation that is stored persistently and transformed into an executable representation. Each projection may be in different formats (e.g., graphical, textual, tabular, and matrix), and unlike the free-form editing, the projections that are edited by the users are not stored persistently. Also, Erdweg et al. proposed in [16] many different syntactic and semantic editor services that may be interesting to the users. The syntactic editor services include model highlighting, navigation support, folding models, syntactic completion templates, comparing models, and auto-formatting the model appearances. The semantic editor services include semantic completion templates, model refactoring, error markers, live translation between model and generated code, and quick fix of the errors.

1.2.4. Model transformation/code-generation

Model transformation/code-generation is concerned with the transformation technologies that are supported by the meta-modeling tools and enable to develop code generators (or model transformers) for DSMLs which can be integrated into the modeling editors for transforming models. Model transformation/code-generation technologies may vary depending on the features that are supported for improving the development processes of the model transformers/code-generators. These features include the syntactic and semantic error detections while developing the model transformers/code-generators, code templates, refactoring, integration with external programs/files, AI-based model transformation, support for scalability, debugging facilities, code folding, etc.

1.2.5. Language validation

The language validation is concerned with the meta-modeling tools' support for defining validation rules for DSMLs. The validation rules for a DSML may then be executed via the modeling editor produced by the meta-modeling tool so as to validate models created via the editor. The validation rules can be categorized as the structural and semantic validation rules. The structural validation rules are concerned with the structural aspects of the language definitions, such as the multiplicities of the language elements and containment relationships between them. The semantic validation rules are concerned with the semantical aspects of the language definitions such as name/type analysis. The meta-modeling tools may further offer such features as the integration with some external validation tools (e.g., formal verification tools, theorem provers, and simulation tools, and testing tools), model animation, model debugging, and automated model validations according to the user-defined or pre-defined rules.

1.2.6. Language testing

The language testing is concerned with testing different aspects of the language development, including the syntax & semantics definitions, editor services, code-generation, and the validation rules, with regard to any functional and quality requirements. The syntax and semantics testing is to do with checking for the language definition requirements. This may include checking if *(i)* the language meta-model consists of the expected modeling elements and relationships and *(ii)* the syntax and semantics rules have been defined correctly and completely. The editor testing is to do with checking if the editor meets such quality requirements as usability and performance. Also, the editor testing may include checking if the editor enables to create models in accordance with the language syntax and semantics. The code-generator testing is to do with checking if a code-generator developed via the meta-modeling tools performs the model transformation correctly (in accordance with the transformation algorithms) and meets the quality expectations. The validation rules testing is to do with checking if the user-defined or pre-defined validation rules can be defined in accordance with the language requirements and then used for validating models correctly in a way that also meets the language requirements.

1.2.7. Language composability

The language composability is concerned with the meta-modeling tools' support for extending an existing language with some new features or unifying the parts of multiple languages for developing a new language.

Just like language testing, the language composability can be considered for different aspects of the language development. The language syntax and semantics may be composed of the syntax and semantics of any existing languages that are stored in a repository for later re-use. A modeling editor can be developed by composing multiple existing tools together such as the model versioning tool, collaboration tool, validation tool, and code-generation tool. The model transformation/code-generation tool may be developed by re-using some transformation templates, patterns, or the existing code. The validation rules can be defined by re-using and modifying the existing rules or composing multiple rules together under some conditions (e.g., logical connectives).

1.3. Structure of the paper

The remainder of this paper is structured as follows. Firstly, Section 2 discusses the related work. Section 3 presents the research methodology used to perform the survey. Section 4 presents the results with cross-factor analysis. Section 5 discusses the findings by also presenting the potential validity threats. Finally, Section 6 concludes this study and discusses the future work directions.

2. Related work

In this section, we aim to analyze the existing literature that consider practitioners' perspectives towards meta-modeling tools, as is the case with our survey discussed in this paper. However, we found out that none of the surveys in the literature do particularly focus on practitioners' perspectives on meta-modeling tools. Rather, the existing literature includes several surveys on the practical use of modeling and many analytical studies on the (meta-)modeling tools. So, below, we separately discuss the well-cited survey studies that address practitioners' perspectives towards *(i)* software modeling and *(ii)* (meta-)modeling tools. Note that the practitioners whom we refer to in the rest of this section are those, who are either researchers in academia (or in any research institutions) or hold any relevant job positions in the industry. However, concerning the modeling discussion, practitioners may be considered as those, who are interested in modeling; whereas concerning the (meta-)modeling tools discussion, practitioners are those, who are interested in using tools for modeling and meta-modeling (i.e., creating DSMLs).

2.1. Surveys on practitioners' perspectives on modeling

In [17], Liebel et al. surveyed among 113 practitioners to understand the use of modeling in embedded domain and address the pros and cons of modeling in the embedded practice. In [18], Agner et al. surveyed among 209 software developers in Brazil and discuss the use of UML and modeling in embedded domain for the developers in Brazil. In [19], Torchiano et al. surveyed among 155 software professionals from Italy and discuss the use of modeling in general. Torchiano et al. address to what extent modeling and modeling languages are adopted and the problems that make practitioners stay away from modeling. In [20], Whittle et al. surveyed 450 practitioners who are experienced on modeling and further interviewed 22 practitioners in 17 companies. Whittle et al.'s survey aimed to understand practitioners' adoption on modeling in a broader sense, focussing on not only the technical aspects but also the organizational and social factors. In [21], Mohagheghi et al. surveyed among the developers working in four companies that contributed to a European Union research project called Modelplex. Mohagheghi et al. aimed in their survey to understand the current use of model-driven engineering by practitioners and address several factors including the usefulness, ease of use, compatibility, and maturity of the tools available. In [22], Tomassetti et al. surveyed among 155 software professional from Italy and aimed to understand the level of maturity in the practitioners' adoption of modeling. Tomassetti et al. focus on three dimensions of modeling, which are automation, tool, and experience. The tool dimension herein focuses on learning the meta-modeling tools and versioning tools that practitioners use and practitioners' challenges. In [23], Akdur et al. surveyed among 627 practicing embedded software engineers from 27 different countries. With the help of this study, the state-of-the-practice of software modeling and MDE were better understood by identifying to what degree, why and how it is used in embedded software industry with its possible challenges and its benefits. In [24], Bork et al. analyzed 11 well-known modeling language specifications that have been proposed by well-regarded institutions such as OMG, open group, and ISO. Bork et al. considers the language specifications from different aspects including the language specification techniques employed, language structures, the concepts used in the languages, the notation specification techniques employed, and the language interoperability and extension.

2.2. Surveys on the tools for modeling/meta-modeling

In [25], Whittle et al. interviewed two different sets of practitioners who are experienced on modeling — one group with 13 practitioners from different companies and another group with 20 practitioners from 2 companies. The goal here is to understand practitioners' perspectives towards the modeling tools with regard to three different categories (i.e., technical, organizational, and social). Whittle et al. further propose a taxonomy of the features that the modeling tools may possess in terms of these categories. In [26], Cabot et al. analyzed a set of well-known modeling tools (i.e., Poseidon, Rational Rose, MagicDraw, Objecteering/UML and Together) to compare their support for the modeling of integrity constraints and the code-generation. Cabot et al. considered the expressivity of the constraint definition language supported by the tools and efficiency of the code generated via the tools. In [27], Paige et al. consider two well-known modeling tools (i.e., VIATRA and Epsilon) and discuss the lessons learned from design, implementation, and evolution stages of those tools. In [28], Pérez-Medina et al. analyze 14 different modeling tools with regard to their support for the modeling in the human computer interaction (HCI) domain. Pérez-Medina et al. focus on the support for *(i)* the way meta-models and models are represented, *(ii)* model transformation and weaving, and *(iii)* repository and interoperability. Pérez-Medina et al. focus essentially on a set of requirements that they consider important for the HCI domain, while we consider in our work a wider range of requirements based on Erdweg et al.'s comprehensive feature model [16]. Also, Pérez-Medina et al. do not focus on the practitioners' perspectives either, which is however the main aim of our study. In [29], Popoola et al. analyze 8 different well-known modeling tools for a number of features including multi-view support, collaboration, code-generation, versioning, DSML development, merging, model-to-model transformation, and scalability. In [30], Ozkaya analyses 58 different existing modeling tools for the UML language for a number of features including modeling viewpoints, analysis, transformation & export, collaboration, tool integration, scripting, project management, and knowledge management. In [31], Kouhen et al. compared 5 meta-modeling tools for a set of requirements including graphical expressiveness, graphical completeness, customization level, openness, usability, and required resources. While Kouhen et al. consider different requirements, many requirements considered in our study are not in the scope of Kouhen et al. Moreover, unlike our work where the main focus is on learning practitioners' experiences and needs, Kouhen et al.'s study focuses on analyzing and comparing the meta-modeling tools. In [32], Kern et al. propose a comparison framework for the meta-modeling tools which is based on a number of meta-modeling concepts that are commonly existing in each tool. Kern et al. analyzed 6 meta-modeling tools with regard to their support for these meta-modeling concepts and determined the properties and values of each tool for the pre-determined list of concepts. Kern compares in another of his study [33] the existing meta-modeling tools with regard to their support for interoperability (i.e., exchanging meta-models between different tools). Kern does not however consider many other requirements of meta-modeling tools that we consider in our survey (see Section 1.2) and practitioners' actual thoughts on the meta-modeling tools' support for interoperability. In [34], Erdweg et al. analyze a set of meta-modeling tools for the feature model that Erdweg et al. has proposed in [16]. However, we essentially aim in our study to learn the practitioners' perspectives towards the meta-modeling tools in terms of Erdweg et al.'s feature model.

2.3. Summary

As discussed above, none of the existing survey studies actually aim to learn practitioners' expectations from the meta-modeling tools, which is addressed in our survey study. The existing surveys on modeling are mostly concerned with how well the practitioners in different countries and industries adopt the concept of modeling in their software development activities and the use of models in particular domains (e.g., embedded). However, meta-modeling is not the main point of concern in any of those surveys. The surveys on the modeling tools rather analyze the existing modeling and meta-modeling tools for a pre-defined set of requirements without conducting any surveys on the practitioners and aiming to learn practitioners' thoughts.

Note also that the literature includes several surveys on the use of modeling languages in practice, e.g., [35–39]. However, the surveys on the modeling languages focus on either the practitioners' perspectives towards different types of modeling languages or the analysis of the existing languages for some requirements.

3. Research methodology

In this study, we chose to use the online survey method since we wanted to obtain information from a relatively large number of practitioners in a quick manner; so, we can easily categorize and analyze these data [40].

3.1. Research questions

In this section, we discuss the research questions (RQ) addressed in this survey study so as to meet the goal explained in Section 1.1.

RQ1: What are the usage frequencies of the existing meta-modeling tools in different problem domains? In this RQ, the top-used meta-modeling tools by practitioners will be investigated. Also, the meta-modeling tools' usage frequencies for different domains (e.g., embedded, automotive, financial services, and web applications) will be revealed.

RQ2: What do practitioners expect from the meta-modeling tools? In this RQ, the goal is to determine the expectations of practitioners from the meta-modeling tools. To this end, we considered herein Erdweg et al.'s feature model for the meta-modeling tools, which has been explained in Section 1.1 and categorizes the meta-modeling tool requirements as the language notation, language semantics, editor, validation rules, language testing, and language composability. We considered for each category the pre-defined list of requirements that have been proposed by Erdweg et al. Also, we further introduced some new categories (e.g., model transformation) and new requirements for the existing categories after conducting a series of pilot studies.

RQ3: What are the challenges that practitioners face with on their meta-modeling activities? In this RQ, the goal is to investigate any challenges that practitioners face with while using/developing the meta-modeling tools. To reach this goal, we address each category of meta-modeling tool requirements considered in our study (i.e., explained in RQ2 above) and particularly aim at learning the top challenging issues that many practitioners commonly suffer from while performing any meta-modeling activities on each category.

3.2. Survey design

In designing our survey, we performed several important activities, which include *(i)* determining the correct questions that address the research questions precisely, *(ii)* structuring and ordering the questions effectively, *(iii)* working out the effective answer sets for the questions, *(iv)* deciding on the open-ended questions, multiple-choice, and single-choice questions, *(v)* conducting pilot studies and revising the survey questions accordingly, and *(vi)* analyzing the answers provided by the participants.

After creating a first draft of the survey questions, we conducted a pilot study with five industrial experts and two academics. Three of the industrial experts are those who have been involved in the development of the popular meta-modeling tools and the rest are those who have been using meta-modeling tools for developing DSLs for years. The academics are those who conduct active researches on the empirical software engineering and modeling. The participants have been communicated over e-mail and asked to provide feedback on the survey questions. To maximize the gain from the pilot study, we came up with a set of topics and asked the participants to provide feedback on those particular topics. These include *(i)* the consistencies of the research questions with the survey questions, *(ii)* the completeness of the survey questions with regard to the research questions, *(iii)* any missing, ambiguous, or redundant questions, *(iv)* any missing, ambiguous, or redundant answers, *(v)* the coherence of the survey sections, *(vi)* the correctness of the answer formats (i.e., multiple-choice, single-choice, free-text, and their combination), and *(vii)* the appropriateness of the time needed to complete the survey. The feedback from the participants have been analyzed and the necessary modifications have been performed on the survey questions.

After completing the pilot study, we have finalized the survey questions as given in Table 2. Finally, the survey consists of 25 different questions that are categorized depending on the research questions which the survey questions address and the type of answers (i.e., multiple, free-text, or single). The first five questions focus on collecting profile information from the participants — their work country, current job positions, work industries, domain of interests, and modeling experiences. The questions 6–10 focus on understanding the participants' expectations regarding the language syntax and semantics definitions that can be defined via the meta-modeling tools and besides any challenges that the participants face with while using the meta-modeling tools for the language definitions. The questions 11–15 focus on understanding the participants' expectations on the syntactic and semantic features of the modeling editors developed via the meta-modeling tools and any challenges. The questions 16–17 focus on the participants' expectations on the model transformation features of the meta-modeling tools and the challenges encountered. The questions 18–20 focus on the participants' expectations on defining the validation rules for languages via the meta-modeling tools and any challenges. The questions 21–22 focus on the participants' expectations on the language testing via the meta-modeling tools and any challenges. The questions 23–24 focus on the participants' expectations on the language composability via the meta-modeling tools and any challenges. Lastly, the participants are asked in the last question if they wish to state any other expectations apart from those given in the survey.

Concerning the question types, the questions for learning practitioners' expectations for each category are each offered with multiple-answers, which provide the options of choosing one or more answers from a pre-defined answer list and typing new answer(s) freely. The question for learning practitioners' challenges in each category and the question for learning any further expectations (i.e., the last question) offer free-text answer areas for enabling the practitioners to type their own answers. The answers provided to the free-text questions will be analyzed in detail using the coding strategy [41]. That is, each answer will be initially checked to determine whether the answer makes sense for the question and any answers that are difficult to understand will be omitted. Those free-text answers that sound close to any of the answers given in the question's answer list (if any) will be counted for that predetermined answer of the list. Otherwise, the free-text answer will be considered as a new answer to that question.

3.3. Survey execution

The survey has been made available online via the *google-forms* application and accessible in between June 2020 and August 2020. The survey link has been sent via e-mail to many different groups who are interested in modeling and meta-modeling activities. It should be noted that while our survey focuses on meta-modeling, we do not attempt at restricting the survey with the practitioners who have experience on meta-modeling. Indeed, we not only want those who create models and meta-models to be involved but also those who do not model (or meta-model) and still have something to say. This may therefore bring us valuable responses from those participants who are challenged by the meta-modeling tools.

In executing the survey, we initially used our personal contacts with whom we have collaborated. These collaborations include the consultancy jobs, R&D projects with many industrial partners from Europe (e.g., ITEA Cluster program), and past/present work experiences in different industries and countries. We sent e-mails to the groups of 6 different R&D projects and nearly 200 different individuals. In total, we sent e-mails to approximately 300 individuals. We also sent e-mails to the practitioners whom we identified via *google scholar* and who contributed to the well-regarded conference and journal papers that are associated with modeling and meta-modeling. Besides, we posted messages to several mailing-lists to attract further participations. These mailing-lists include the Eclipse modeling platforms' mailing lists (e.g., sirius-dev, graphiti-dev, papyrus-rt-dev, emf-dev, emft-dev, agileuml-dev, papyrus-ic, etc.), Netbeans mailing list, IEEE architecture

Table 2
The survey questions.

Res. Que.	Survey questions	Multiple answers	Free text	Single answer
Profile questions	1-Which country do you work in?			X
	2-What is (are) your current job position(s)?	X	X	
	3-Which industry(ies) do you work in?	X	X	
	4-Please indicate below the domain(s) in which you develop or use domain-specific modeling languages.	X	X	
	5-How many years of experience do you have in meta-modeling (i.e., developing languages)?			X
RQ1	6-Please choose the meta-modeling tool(s) that you use for developing (domain-specific) languages and their toolset.	X	X	
RQ2	7-Please indicate the type(s) of notation set that you consider in developing languages via the meta-modeling tools.	X	X	
RQ2	8-If you define the language semantics via the meta-modeling tools, please indicate below what type(s) of semantics you prefer.	X		
RQ2	9-What other meta-modeling tool features for the language definition is(are) important for you?	X	X	
RQ3	10-Do you face with any challenges while defining the language syntax and semantics with the meta-modeling tools? If so, please indicate the challenge(s) and tool(s).		X	
RQ2	11-What type(s) of editing mode do you prefer to use?	X		
RQ2	12-Which of the following syntactic editor features is(are) important for you?	X	X	
RQ2	13-Which of the following semantic editor features is(are) important for you?	X	X	
RQ2	14-What other features of modeling editors are important for you ?	X	X	
RQ3	15-Do you face with any challenges while developing and using the modeling editors? If so, please indicate the challenge(s) and tool(s).		X	
RQ2	16-Which of the following features of model transformation/code generation definition technologies are important for you.	X	X	
RQ3	17-Do you face with any challenges while using the generator definition technologies of the meta-modeling tools? If so, please indicate the challenge(s) and tool(s).		X	
RQ2	18-If you consider defining the validation rules for a language, please indicate below what types of validation you prefer.	X		
RQ2	19-Which of the following meta-modeling tools features for language validation are important for you ?	X	X	
RQ3	20-Do you face with any challenges while defining/using the language validations with the meta-modeling tools? If so, please indicate the challenge(s) and tool(s).		X	
RQ2	21-If you consider testing the languages, please indicate which of the following aspect (s) of the language development you address for the language testing.	X		
RQ3	22-Do you face with any challenges while testing languages via the meta-modeling tools? If so, please indicate the challenge(s) and tool(s).		X	
RQ2	23-If you consider composing languages (e.g., extending a language or unifying multiple languages), please indicate below which of the following aspect(s) of the language development you address for composition.	X		
RQ3	24-Do you face with any challenges while composing languages via the meta-modeling tools? If so, please indicate the challenge(s) and tool(s).		X	
RQ2	25-If you have any other expectations from the meta-modeling tools or any challenges that are not related to the survey sections above, please state your expectations/challenges here.		X	

description mailing list, AADL list. Another source of participants are the social media platforms such as *Linkedin*. So, we shared our survey on several Linkedin groups to reach as many participants as possible. These groups include domain-specific languages, domain-specific modeling, model driven architecture, model based software engineering, and the enterprise architecture network, software architecture, software developer, and software design patterns and architecture. The Linkedin groups seemed to attract many practitioners indeed, where our posts got many likes and shares. In total, our survey received 103 different participants from diverse countries and industries.[1]

3.4. Survey sampling

To create a survey sample, two methods may be applied – i.e., the probability sampling and non-probability sampling [42]. The probability sampling is based on the idea that samples are created by choosing the potential participants from the population using some probabilistic techniques. The random selection is one of the techniques for the probability sampling, which states that any two participants have the equal chance to be selected. The non-probability sampling is employed when it is not possible to determine the probability that each participant is selected for the sample. The non-probability sampling is also known as the convenience sampling, which promotes that the participants who are conveniently accessible are selected from the population.

In our survey study, it was not possible for us to reach every single meta-modeling stakeholder and therefore we performed the non-probability sampling method. Indeed, we sent e-mails to our personal contacts as discussed in Section 3.3, as those are the closest potential participants to us whom we can reach conveniently. However, to minimize any biases due the non-probability sampling method, we intended to mimic the probability sampling in our survey sampling. To this end, we shared the survey in several different mailing lists and Linkedin groups that actually enable the survey to be reached by any people enrolled in those platforms with an equal chance.

[1] The raw data for the survey responses can be accessed via the following link: https://doi.org/10.5281/zenodo.4016622.

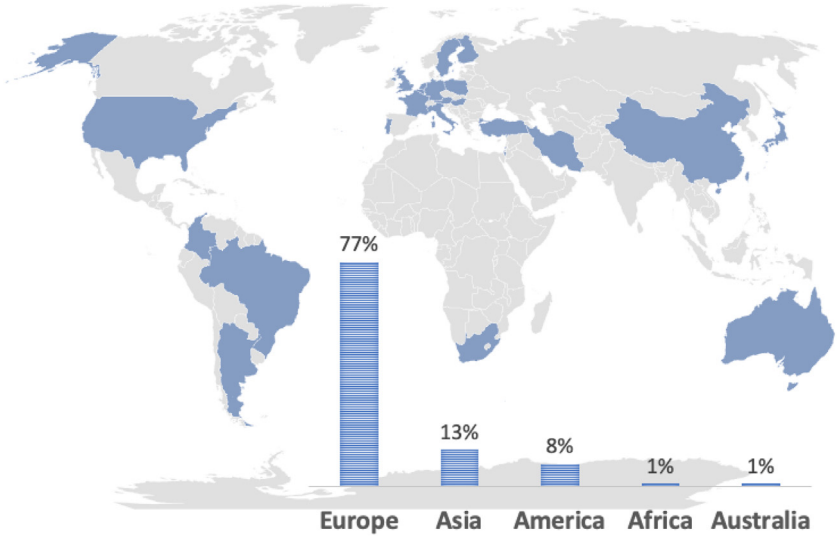

Fig. 1. The countries in which the participants work.

4. Survey results

4.1. Profile

4.1.1. Q1: Work countries

As shown in Fig. 1, the survey attracted participations from 24 different countries, most of which are from the countries in the Europe continent. The European countries include Austria, Belgium, Finland, France, Germany, Hungary, Italy, Poland, Portugal, Sweden, Turkey, the Netherlands, and United Kingdom. Some countries from the Asia and America continents also contributed to the survey, including Argentina, Brasil, China, Colombia, Japan, Iran, Israel, Taiwan, and USA. Also, one participant from Australia and another one participant from South Africa contributed to the survey.

4.1.2. Q2: Participants' job position

Fig. 2 shows the different job positions that the participants hold. Note that this is a multiple-response question, in which the participant can choose multiple positions at the same time (e.g., both software architect and software developer). So apparently, the top job positions are software architect (45%) and software developer/programmer (41%). Also, 22% of the participants hold the researcher (or research scientist) job position. While many of the rest of the job positions are held by 7%–11% of the participants, the job positions indicated as "other" (i.e., analyst, software tester, enterprise architect, system architect, and compiler/tools engineer) have each been selected by one or two participants.

4.1.3. Q3: Participants' work industries

Fig. 3 shows the industries for which the participants work. Defense/military & aviation is the top industry among the participants (30%), and that is followed by the research (26%) and IT & telecommunications (24%) industries. Some of the participants work for the healthcare and biomedical (14%), consumer electronics (14%), or automotive & transportation (15%) industries. The number of participants involved in such industries as finance & accounting, government, and software outsourcing are quite low (6%–7%). The participants from the education industry represent only 2% of the overall participants. The "other" industries such as retail, semiconductors, and digital printing are the least popular industries and have each been selected by one or two participants at most.

4.1.4. Q4: The domains for which the participants build/use DSMLs

Fig. 4 shows the domains for which the participants build/use DSML solutions. The embedded domain is the top popular one, which has been selected by more than half of the participants (56%). Following those domains, some participants consider using/developing DSML solutions for the following domains: automative (25%) and control & automation systems (24%). The domains including web applications (16%), user interface design (16%), medical device development (16%), IOT (16%), and enterprise solutions (15%). The domains such as financial services, data analytics, mobile, testing, railways systems, document engineering, real-time operating systems, and telecommunications are rarely considered in the DSML solutions (each selected by 6%–12%). The "other" domains (i.e., digital printing, aerospace, chip development, robotic, power electronics, and integrated digital television) include one or two participants at most who consider DSML solutions.

Fig. 5 shows the correlations between the participants' work industries and the domains in which the participants develop/use DSMLs. So, the research and IT & telecommunications industries are associated with the greatest number of different domains, meaning that the participants on both industries develop/use DSMLs for several different domains. Also, another important finding here is to do with which industries develop/use DSMLs for which problem domains. So apparently, the DSMLs for the automotive domain are popular in the automotive and transportation industries, the DSMLs for the embedded domain are popular in the defense/military & aviation industry, the DSMLs for the medical domain are popular in the healthcare and biomedical industry, and the DSMLs for the mobile domain are popular in the research and IT & telecommunications industries.

4.1.5. Q5: Participants' years of experience in meta-modeling (i.e., language development)

As Fig. 6 shows, 97% of the participants have varying level of experiences in meta-modeling. Among those participants, 38% have 10+ years of experience. Concerning the rest, while 18% have 6–10 years of experience, 33% have 2–5 years of experience. Just 8% of the participants have less than 2 years of experience.

4.2. Q6: Participants' meta-modeling tool usages

Fig. 7 shows the participants' usage frequencies of the meta-modeling tools. So, the top-used meta-modeling tools are Eclipse Sirius and GEMS (Generic Eclipse Modeling Environment) (43%). Those are

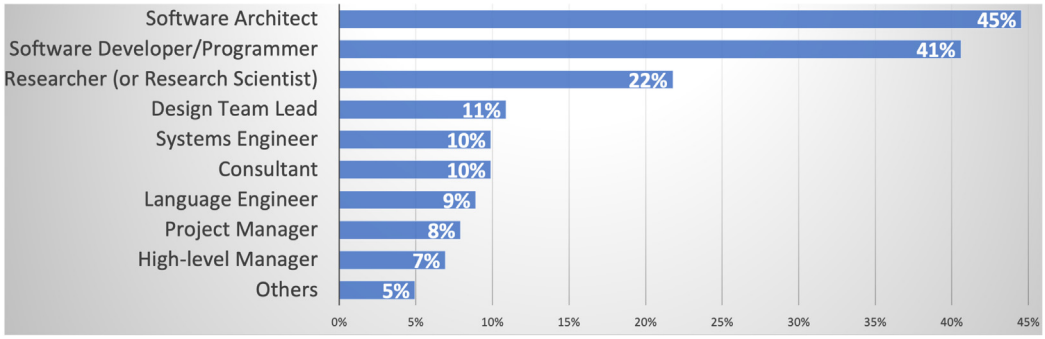

Fig. 2. Participants' job positions.

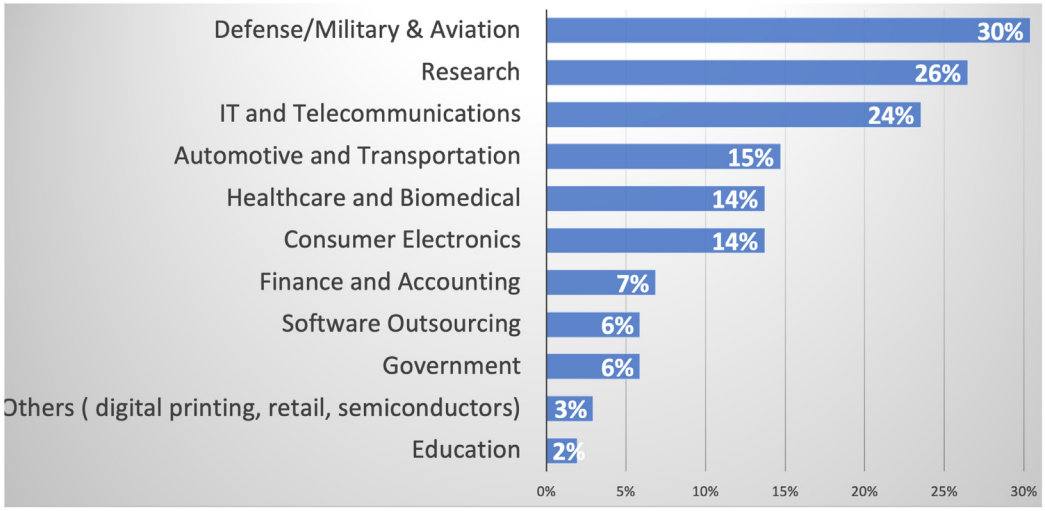

Fig. 3. Participants' work industries.

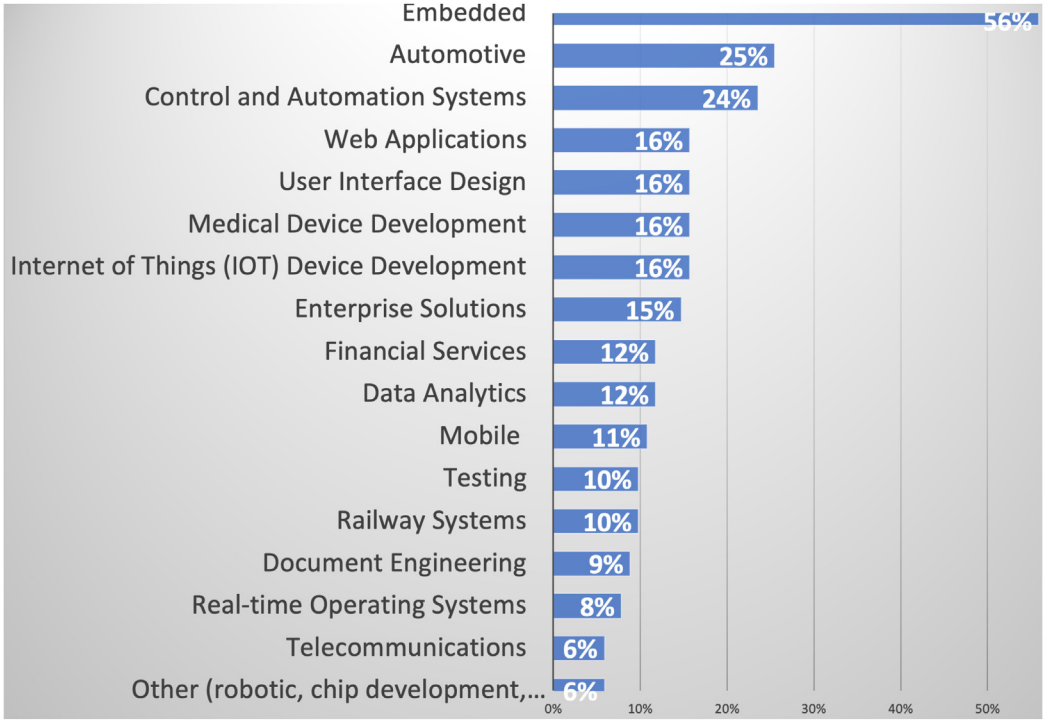

Fig. 4. The domains for which the participants build DSMLs.

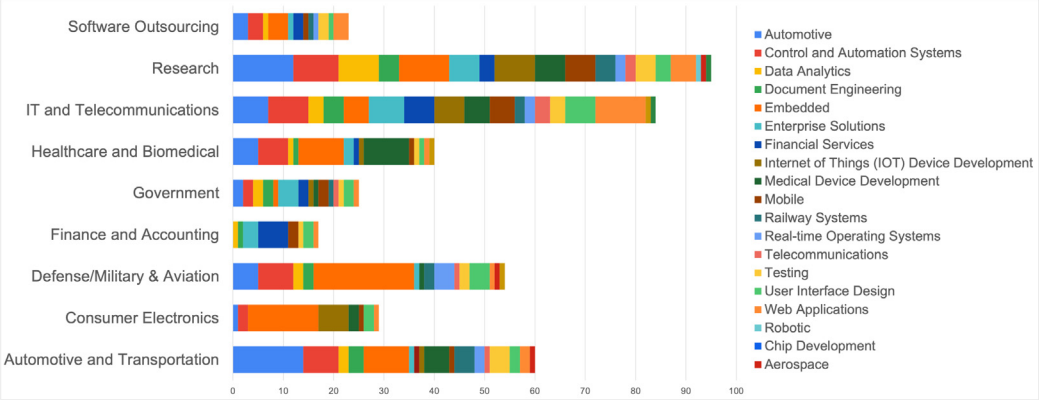

Fig. 5. The correlations between the participants' work industries and the problem domains in which the participants use/develop DSMLs.

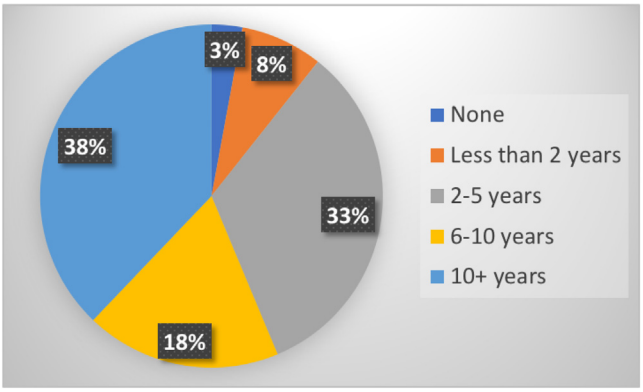

Fig. 6. Participants' years of experience in meta-modeling (i.e., language development).

followed by Metaedit+ (31%), and Xtext (29%). While some participants (18%) use the Microsoft DSL tools for meta-modeling, the rest of the meta-modeling tools are rarely used (if any). Note that 8% of the participants use their own meta-modeling tools that they build for their specific problems (i.e., in house solutions).

Fig. 8 shows the correlations between the domains for which the participants develop/use DSMLs and their choice(s) of the meta-modeling tools. So apparently, some of the meta-modeling tools such as ANTLR, GEMOC, EVA, Cameo, JastEMF, and Graphiti are essentially used for a few particular domains. While GME, Melange, and MPS are not among the frequently-used tools, those tools seem to be preferred for various domains. In almost all the domains, the top-used meta-modeling tool is Metaedit+ - except the embedded domain where the top-used meta-modeling tool is GEMS.

In Fig. 9, we give the correlations between the participants' choice(s) of meta-modeling tools and their work countries. The tools that are used in the greatest number of countries are GEMS, Metaedit+, Sirius, and Xtext. GEMS is top-used in Turkey, followed by the Netherlands. The meta-modeling tools such as ANTLR, GEMOC, WebGME, Melange, JastEMF, Graphiti are each used in one or two countries. Also, some countries seem to use particular meta-modeling tools more than other tools. Indeed, USA's top used tool is Microsoft DSL tools, Finland's is Metaedit+, many European countries' top-used tools are Eclipse-based tools (e.g., GEMS, Sirius, Xtext), and the Netherlands and Turkey's is GEMS. Note that the Netherlands also use MPS quite a lot. Another interesting finding here is that many countries (i.e.,Belgium, Germany, Finland, France, Taiwan, The Netherlands, Turkey and USA) include some practitioners who essentially prefer to use in-house solutions for modeling/meta-modeling without using any of the existing meta-modeling tools.

4.3. Language definition (syntax & semantics)

4.3.1. Q7: The type(s) of notation set that the participants prefer in developing languages via the meta-modeling tools

Fig. 10 shows the different types of notation sets (i.e., visualizations) that the participants prefer in developing their languages. So, almost all the participants (88%) prefer to develop DSMLs with diagrammatic visualization. The DSML developments with a textual visualization are also highly popular among the participants (75%). Note however that developing languages with hybrid/blended notation sets that combine the textual and diagrammatic notation sets is just preferred by 26% of the participants. So, most participants choose to develop either textual or diagrammatic languages exclusively. The tree structure visualization is also quite popular, selected by 33% of the participants. The other types of visualizations (i.e., matrix, map, and tabular) are rarely preferred as depicted in Fig. 10.

Fig. 11 presents the correlations between the domain(s) for which the participants develop/use DSMLs and the participants' visualization choices. In most domains, most of the participants prefer either diagrammatic or textual visualization. Note for the railways systems, mobile, and IOT domains that the map visualization is highly popular, considering map's popularity for other domains. The diagrammatic hybrid/blended, and tree visualizations are each top-used among the participants from the embedded and automative domains. The map visualization is top-used among the participants from the IOT, mobile, and railway domains. The matrix visualization is top-used among the participants from the automotive, enterprise solutions, financial services, and medical domains. The textual visualization is top-used among the participants from the embedded and automotive domains. The tabular visualization is top-used among the participants from the embedded, enterprise solutions, and financial services domains.

4.3.2. Q8: The type(s) of semantics that the participants use in defining their language semantics

In this question, the interpretative and translational semantics definition techniques have been asked to the participants. As Fig. 12 shows, the interpretative semantics definition is not as desired as the translational semantic definition that is categorized as model-to-text and model-to-model. The greatest portion of the participants (37%) seems to be fine with either categories of the translational semantics definition; however, those participants would never use an interpretative semantics definition technique. The percentages of the participants who use the translational semantics together with the interpretative semantics is relatively small. Also, 15% of the participants stated to use the interpretative semantics definition exclusively, while another 15% use model-to-text translation exclusively and another 12% use the model-to-model translation exclusively.

Fig. 13 shows the correlations between the participants' domain and the language semantics definition techniques that the participants

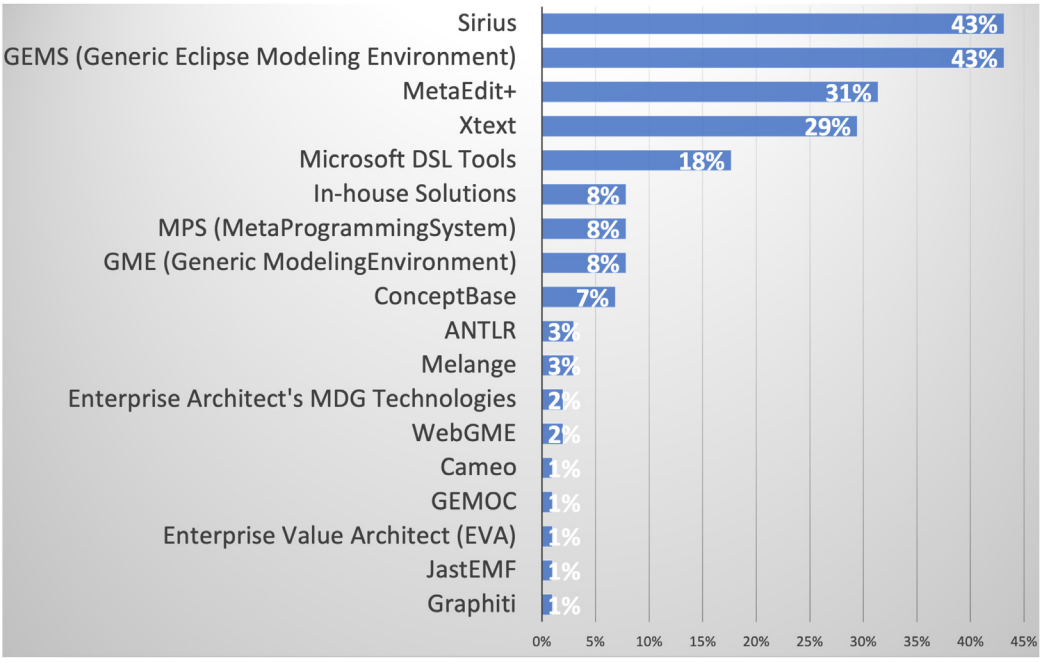

Fig. 7. Participants' meta-modeling tool usages.

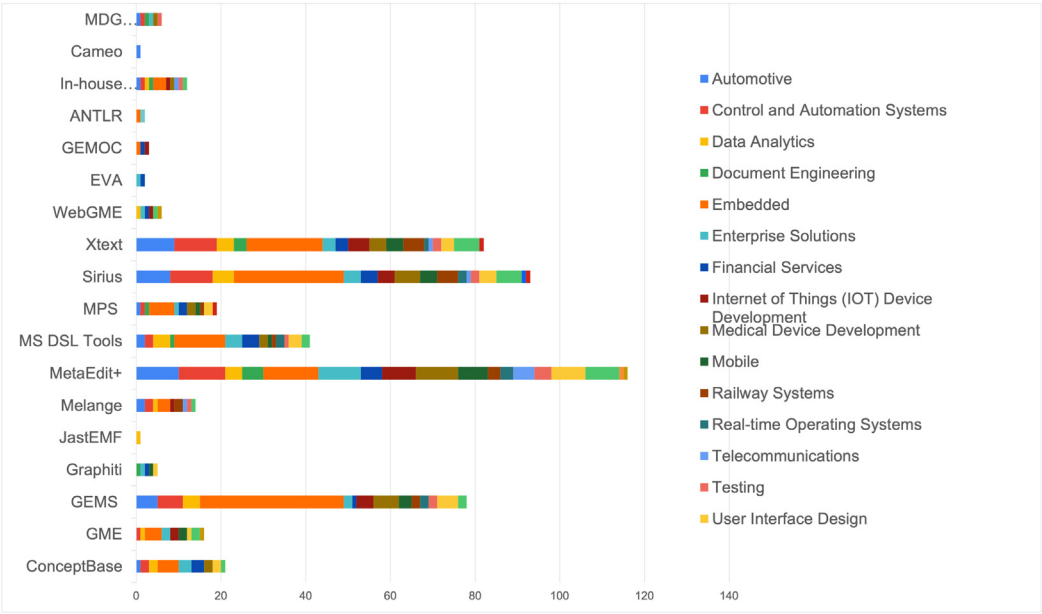

Fig. 8. The correlations between the participants' domains and meta-modeling tool choice.

prefer. The participants from some domains (e.g., web applications, aerospace, testing, telecommunications, real-time operating systems, and mobile) never use the interpretative semantics definition technique. On the other hand, the interpretative semantics definition is shown the greatest interest by the participants who develop/use DSMLs for the automotive domain. Also, in all domains, the percentages of the participants who prefer the model-to-text translation is more than those who prefer the model-to-model translation.

4.3.3. Q9: The other meta-modeling tool features that the participants wish to use in defining languages

In this question, the participants have been offered with four different features and allowed to type any other features too. Most participants (82%) are interested in importing/exporting the syntax and semantics definitions of the languages via the meta-modeling tools. By doing so, the meta-models may be shared among collaborators, documented for different purposes, or kept in a repository for later re-use. The other pre-given features such as versioning the language definitions (72%) and re-using some libraries that offer the language syntax and semantics of some popular languages (such as UML) for re-use (68%) are also quite popular. The collaborative real-time editing of language definitions is not so important for many participants however (39%).

4.3.4. Q10: The challenges that the participants face with while defining the language syntax & semantics via the meta-modeling tools

The top concerning challenge has been observed to be meta-modeling tools' steep learning curve required for defining the language

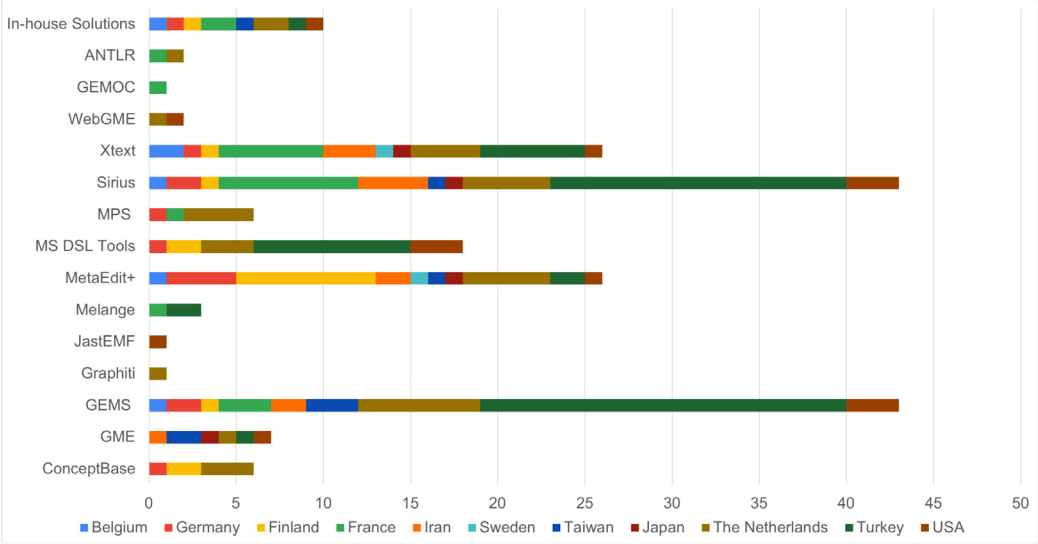

Fig. 9. The correlations between the participants' meta-modeling tool choice and their countries.

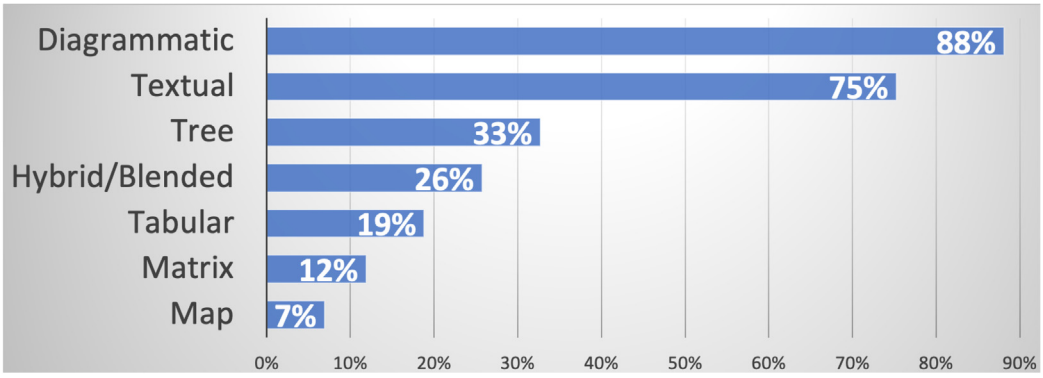

Fig. 10. The type(s) of notation set that the participants prefer in developing languages via the meta-modeling tools.

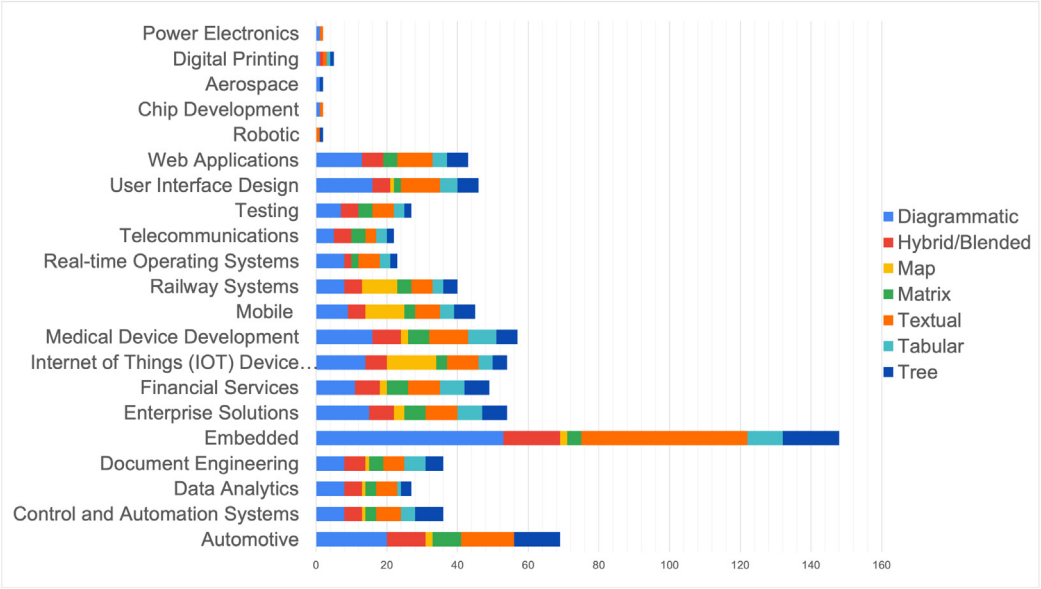

Fig. 11. The correlations between the participants' domains and the type(s) of notation sets preferred by the participants.

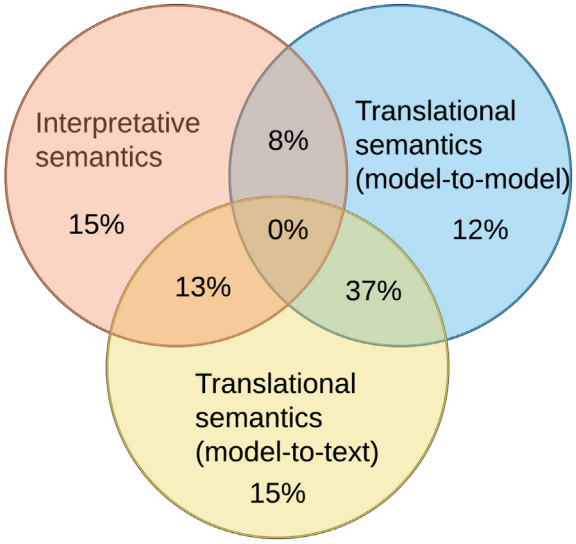

Fig. 12. The type(s) of semantics that the participants use in defining their language semantics.

syntax and semantics and the lack of support for training (e.g., tutorials, guidances, and examples on how to define the language syntax and semantics). One of those participants is especially concerned about learning the projectional editing with MPS. Some participants are concerned about the difficulties for keeping the language syntax and semantics complete and consistent as the new modeling elements are added — two participants herein are especially concerned about using the Eclipse-based meta-modeling tools. Also, a few other participants are concerned about the lack of support for *(i)* the integration with the version management tools (e.g., GIT and SVN) for versioning meta-models and *(ii)* the collaborative meta-modeling. Lastly, one participant stated to face with difficulties in using Enterprise Architect for defining the relationships between modeling elements.

4.4. Editor services

4.4.1. Q11: The type(s) of editing modes that the participants prefer

As Fig. 14 shows, 69% of the participants prefer the meta-modeling tools that support the free-form editing exclusively, which promotes the persistent storage of the editable models. So, those participants essentially prefer to use any meta-modeling tool(s) that supports developing a language with just one visualization only (e.g., Xtext for textual and Sirius for diagrammatic). Another 21% of the participants prefer the meta-modeling tools that support developing a language with the projectional editing mode, which enables the models to be edited via different projections that each supports a different visualization (e.g., graphical, textual, and tabular). Those participants may actually use the MPS meta-modeling tool for creating different projections for their language. The rest of the participants (10%) do not seem to have a particular choice here and indicated to use the meta-modeling tools that support either of the editing modes depending on their language development requirements.

Fig. 15 gives the correlations between the participants' domain and the editing modes that the participants prefer. The participants from some domains (e.g., chip dev., robotic, web app., UI design, testing, telecommunications, and real-time operating systems) do not prefer any meta-modeling tools with the projectional editing support. On the other hand, the participants from the mobile and IOT domains do not prefer the free-form editing. Also, the medical device development, enterprise solutions, and document engineering domains include many more participants who prefer the projectional editing mode.

4.4.2. Q12: The syntactic editor services that the participants prefer to use

As Fig. 16 shows, reusing models is the top-popular syntactic editor service that has been selected by 75% of the participants. Indeed, with model reuse, practitioners may keep the models created in a repository and create new models by re-using the existing models in the repository with the least effort possible. Another highly preferred syntactic editor service is the comparison of models via a diff-like tool (69%), which may enable to detect and review the differences between models and even merge them. The other editor features that are also quite popular are the syntactic completion templates that provide incomplete models/code/graph to the users (58%) and the auto formatting, restructuring, aligning, and layouting of a model's presentation (50%). Other syntactic features such as the customizable visual highlighting in models, navigation support (via, e.g., an outline view), and model folding to hide part of a model are considered by relatively fewer participants (30%–32%).

4.4.3. Q13: The semantic editor services that the participants prefer to use

Besides the modeling editor services that concern the syntax-related aspects, the modeling editors may offer services about the semantical aspects of languages. As Fig. 17 shows, the top-popular semantic editor services is the need for an error marker for highlighting a model element and any associated error messages (81%), which enables to detect the semantical errors (e.g., the violation of wellformedness rules) at modeling time. The automatic update of models upon any changes on the meta-model is the second top-popular semantic editor service (69%), which enables the modelers to detect any errors due to the semantical changes automatically. Also, the semantic completion for receiving automatic suggestions at modeling time and the refactoring of models without changing semantics (e.g., renaming and language-specific restructuring) are each desired by 60% of the participants.

The UML support for reusing and extending the UML language syntax and semantics and the live translation between the model and generated code (i.e., displaying the model and code side-by-side) are also desired by half of the participants (52%–53%). The rest of the semantic editor services such as navigation to representations (i.e., the capability to see on which representations a given semantic element is used), advanced search (i.e., the capability to find the semantic elements with the advanced search criteria), co-evolution of meta-models together with models, and quick fixes are relatively less popular among the participants.

4.4.4. Q14: The other editor services that the participants prefer to use

Besides the syntax and semantics editor services discussed above, many other editor services may exist that are not given in the above questions' lists. So, in this question, we provided a list of possible general editor services and let the participants type any other services that are not given in our list. As Fig. 18 shows, document generation (e.g., generating word, PDF, or XML files from models automatically) is the top-popular editor service herein, which is desired by 77% of the participants. Importing/exporting models for sharing models among collaborators and the version control system integration for managing the model versions are also the highly desired features for the modeling editors (72%–73%). 68% of the participants are interested in the usability of the modeling editors, which is essentially concerned with the minimum number of clicks for modeling/meta-modeling. The editor services that are also quite popular and considered by approximately half of the participants are the traceability between different view models (e.g, checking the consistencies between structural and behavioral models) (60%), the IDE integration (i.e., the integration with development environments such as Eclipse and Visual Studio) (53%) and scalability (i.e., creating and storing large number of models) (48%). The rest of the features given in Fig. 18 are relatively less popular among the participants. Also, using the "other" option, one participant wishes for the offline access to the editor, where the changes can be reflected when connecting to the internet. Another participant wishes for sharing the meta-model with any users who can use the language freely.

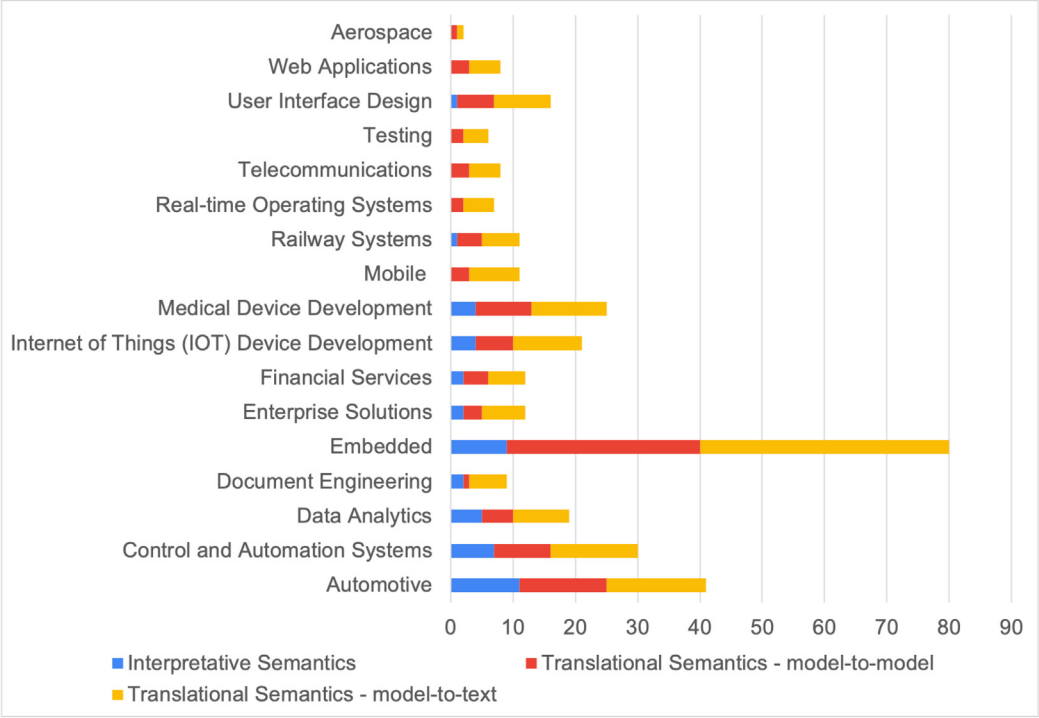

Fig. 13. The correlations between the participants' domains and the semantics definition techniques used.

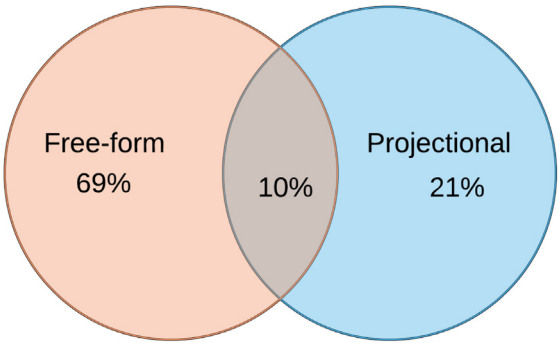

Fig. 14. The types of editing modes that the participants prefer.

4.4.5. Q15: The challenges that the participants face with on developing/using editors

The participants are most concerned about the usability of the editors they develop or use. One of those participants stated to face with some usability issues in using Eclipse Xtext/Xtend and MPS meta-modeling tools for developing editors. The same participant is also concerned about Xtext's support for scalability and traceability based on the OSLC[2] linking. Also, some of the participants are concerned about editors' lack of support for *(i)* model versioning, *(ii)* training (i.e., any learning materials to learn how to produce modeling editors), and *(iii)* integrating editors with other development technologies, such as .NET, to combine modeling and coding. Lastly, one participant is concerned about the support for web-based editors, and another one is concerned about the syntax coloring.

4.5. Model transformation/code-generation

4.5.1. Q16: The features of the model transformation/code-generation technologies that the participants prefer

As Fig. 19 shows, most participants (72%) wish to detect any syntactic and semantic errors while developing model transformation tools using the transformation technologies of the meta-modeling tools. Indeed, each transformation technology essentially offers a notation set (or a template) for developing model transformation tools, and using such notation sets may not always be so simple without strong support for error detection, as one may easily end up with developing wrong model transformation tools. The second top-desired feature is the syntax highlighting (53%), which is concerned with coloring the code for transformation and facilitates the readability and the detection of syntax errors. Scalability is another desired feature (41%), which is concerned with developing sufficiently large transformation tools that consist of hundreds of lines of code for transforming large and complex models. That is followed by the code templates (39%), and refactoring (37%) features. The features such as model-to-model transformation, integration with programming languages (e.g., Java, C++), generator debugging, and bidirectionality in model transformation (i.e., the source and target models remain consistent whenever one of them changes) are desired by 30%–33% of the participants. The rest of the features indicated in Fig. 19 are rather less important, selected by fewer participants.

4.5.2. Q17: The challenges that the participants face with while using the model transformation/code-generation technologies

A few of the participants stated some challenges on their use of the model transformation technologies. These challenges are to do with *(i)* configuring the transformation process using the Eclipse Acceleo[3] and Xtend[4] model transformation technologies, *(ii)* adapting the code generators with the changing end-user requirements, *(iii)* the training support for using the transformation technologies, *(iv)*

[2] Open Services for Lifecycle Collaboration (OSLC): https://open-services.net/.

[3] Acceleo: https://www.eclipse.org/acceleo/.

[4] Xtend: eclipse.org/xtend/.

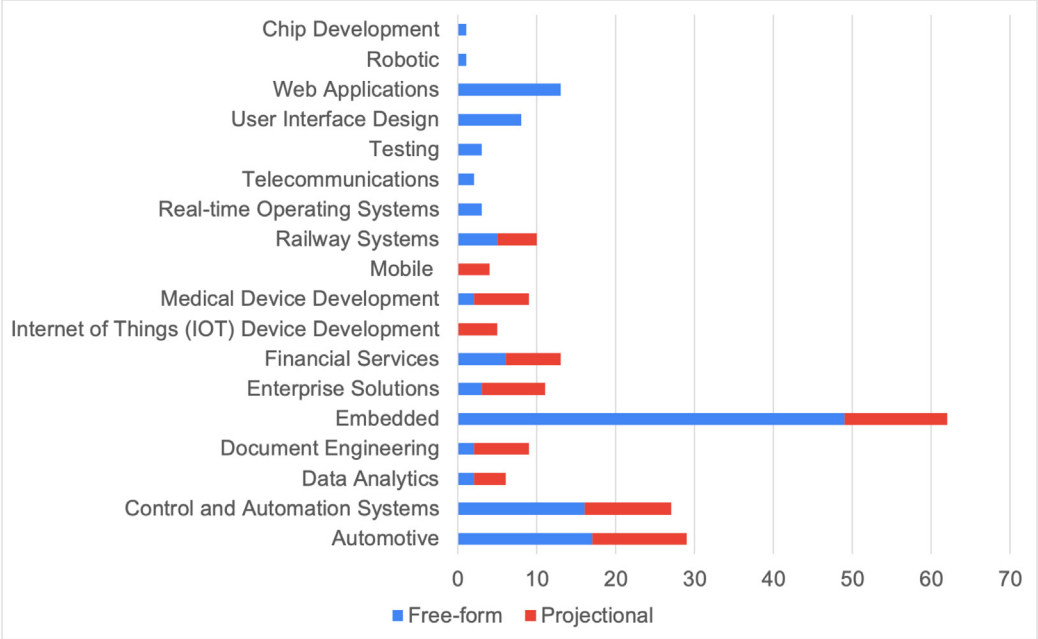

Fig. 15. The correlations between the participants' domains and the editing mode preferred.

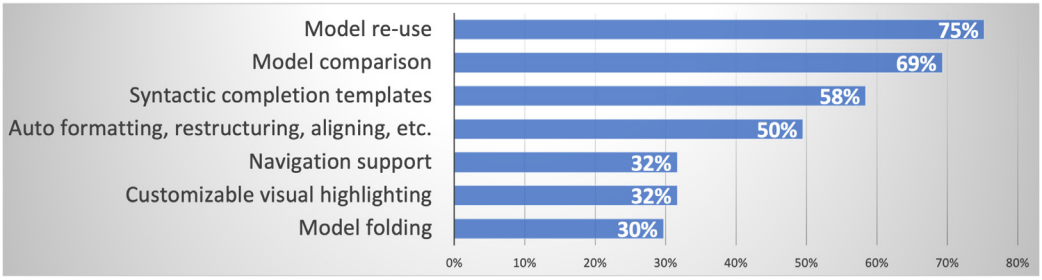

Fig. 16. The syntactic editor services that the participants prefer to use.

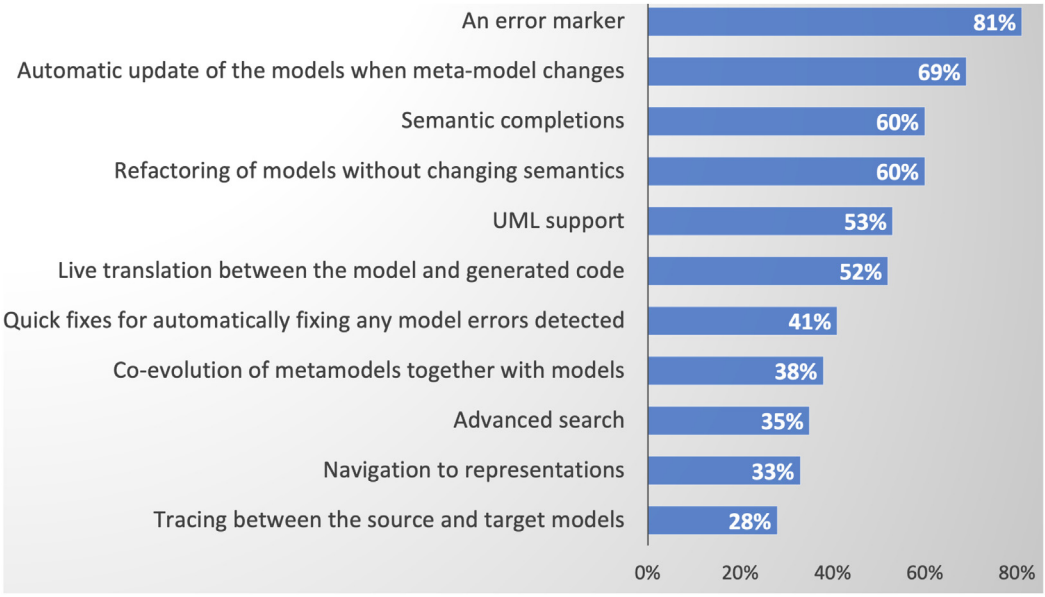

Fig. 17. The semantic editor services that the participants prefer to use.

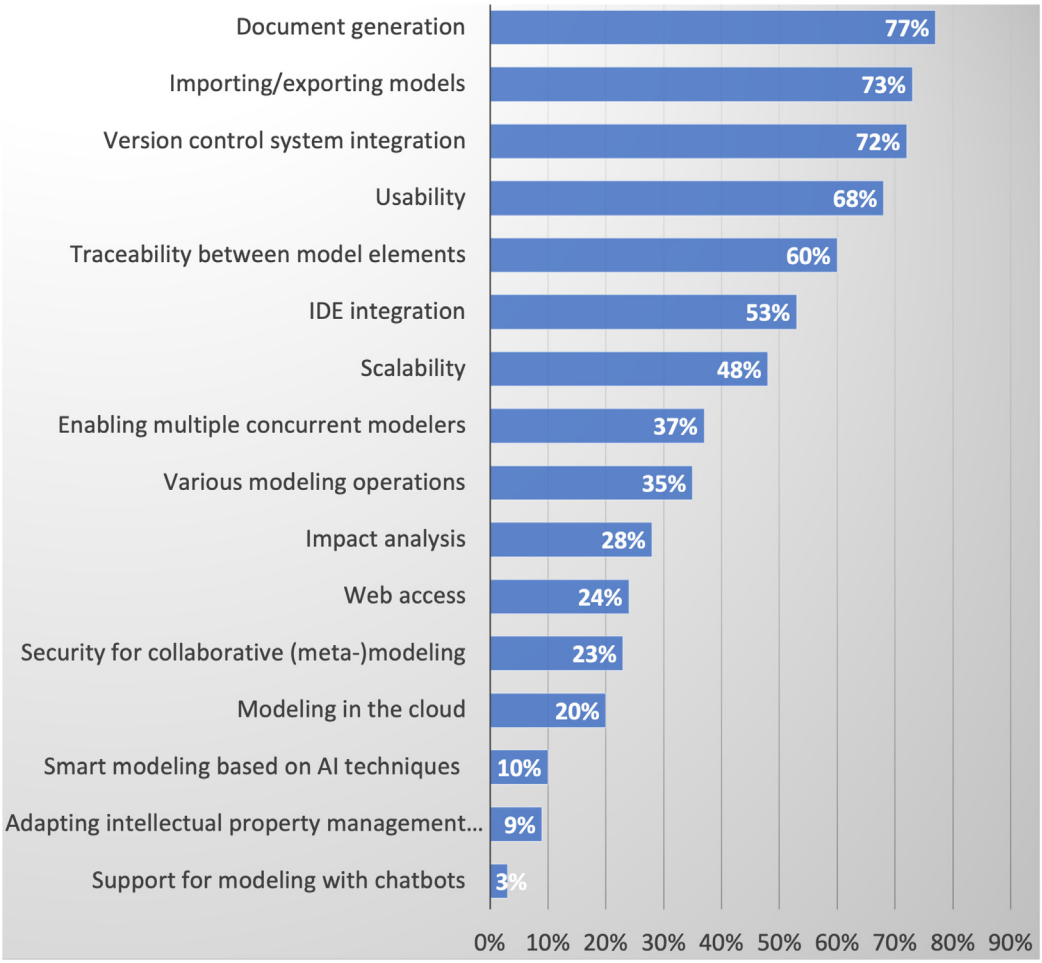

Fig. 18. The other editor services that the participants prefer to use.

integrating code generators with external Java programs, and *(v)* the transformation technologies being too Java-centric and neglecting the needs of embedded domains.

4.6. Language validation

4.6.1. Q18: The validation types that the participants prefer in defining the language validation rules

As Fig. 20 shows, most of the participants (75%) wish to define both the semantic and structural validation rules for the DSMLs that they develop via the meta-modeling tools. The percentages of the participants who prefer just semantic validation or structural validation exclusively is much lower.

4.6.2. Q19: The meta-modeling tool features that the participants prefer to use while validating their languages

As Fig. 21 shows, most practitioners (72%) prefer the meta-modeling tools to be integrated with some external tools that support the model validation. Indeed, such tools may be the formal verification and theorem proving tools that can be integrated with the modeling editors and used for formally verifying and proving the models for, e.g., consistency, completeness, and correctness. Many participants (66%) are also interested in defining type systems and rules for their languages and validating models against those user-defined validation rules (semi-)automatically. Likewise, the (semi-)automatic validation of language models for the meta-model's syntactic and semantic rules is quite important for the participants (64%). Animating models and developing and integrating model debuggers appear to be less popular features for many participants though. One of the participants further wishes the meta-modeling tools to enable the development of model debuggers.

4.6.3. Q20: The challenges that the participants face with on defining the language validation rules

While most participants did not state any challenges, a few participants pointed out meta-modeling tools' lack of support for *(i)* checking any model for the validation rules at modeling time and *(ii)* defining complex validation rules.

4.7. Testing

4.7.1. Q21: The different aspects of language development that the participants consider testing

As Fig. 22 shows, the participants show varying level of interests on what aspects of languages they wish to test via the meta-modeling tools. Most of the participants (78%) wish the meta-modeling tools to support testing the language semantics and determining any semantical issues that violate the language definition requirements. Testing the language validation rules for checking how correctly the models can be validated using the validation rules is also important for many participants (66%). Testing the model transformation/code-generation tools and language syntax definitions are each considered by around half of the participants (54%–57%). Testing the modeling editor is shown the least interest by the participants (39%).

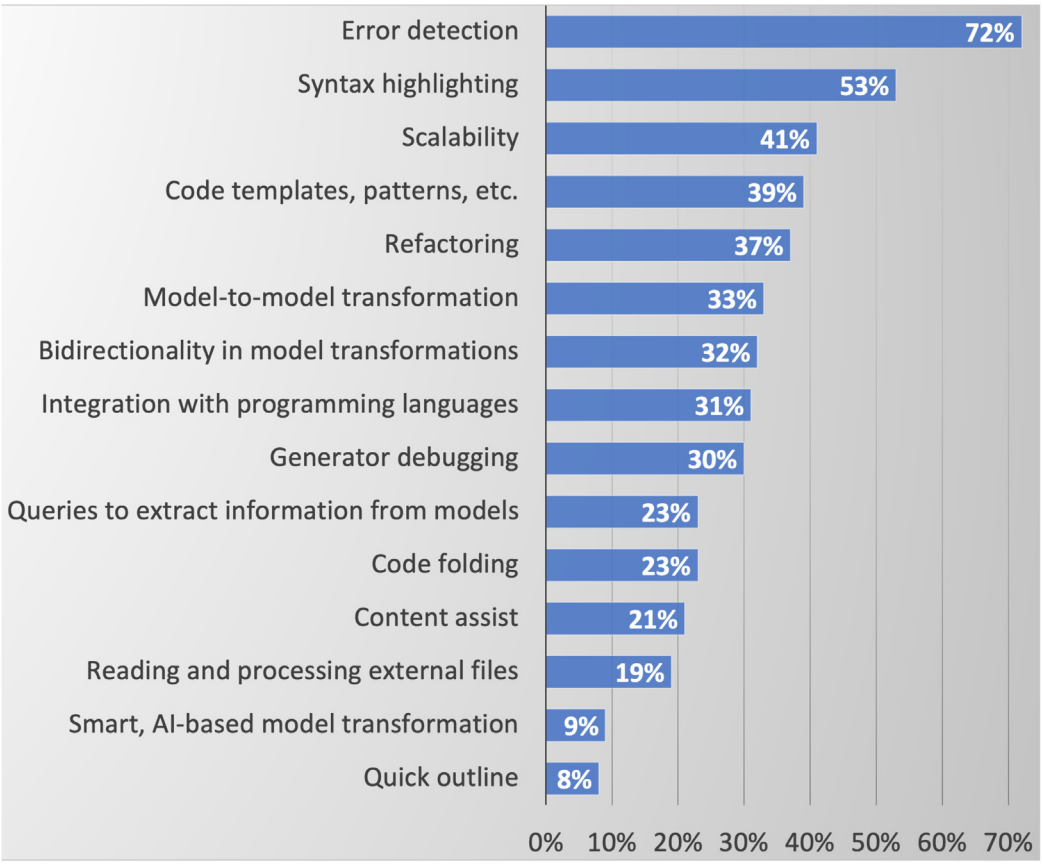

Fig. 19. The features of the model transformation/code-generation technologies that the participants prefer.

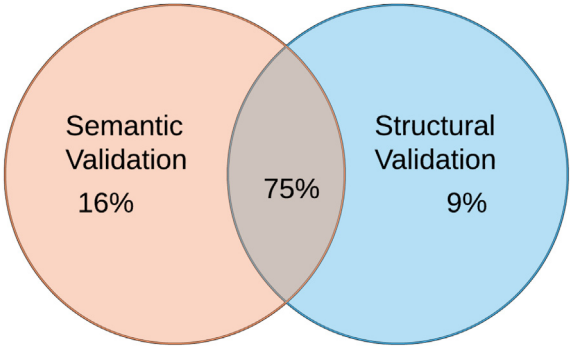

Fig. 20. The validation types that the participants prefer in defining the language validation rules.

4.7.2. Q22: The challenges that the participants face with on testing different aspects of the language development

Concerning the language testing, only a few participants stated some challenges. These challenges are to do with the lack of support for *(i)* specifying and executing test cases on different aspects of languages, *(ii)* integrating any external testing tools that can aid in the language testing, and *(iii)* the model-based testing of different aspects.

4.8. Composability

4.8.1. Q23: The different aspects of language development that the participants consider for composition

As Fig. 23 shows, most participants (84%) wish the meta-modeling tools to enable defining the language semantics by composition (e.g., re-using the semantical definitions of the existing languages such as UML). Likewise, developing model transformation/code-generation tools by composition (e.g., re-using existing code-generator code, template, or patterns) is highly desired among the participants (73%). Composing the language syntax (e.g., re-using the syntax of another language such as UML), the language validation rules, and the modeling editors (e.g., integrating different modules together such as versioning, collaboration, and code-generation modules) are each desired by 64%–66% of the participants.

4.8.2. Q24: The challenges that the participants face with on composing different languages

Just 3 participants stated some challenges herein, which are to do with *(i)* composing different languages without changing them and *(ii)* the difficulties in performing composition with textual languages.

4.9. Q25: Further remarks

A few participants further stated some other expectations from the meta-modeling tools or any challenges that are not related to the survey sections above. These include the support for the *(i)* well-documented APIs for facilitating (meta-)modeling, *(ii)* modeling-in-the-cloud, *(iii)* qualification assessment of the meta-modeling tools for the functional safety standards, *(iv)* free trials for academic use, and *(v)* usability of the meta-modeling tools to attract more practitioners.

5. Discussions

5.1. Summary of findings

In our survey study, we intended to investigate three different research questions that are explained in Section 3.1. To this end, we included 25 different questions in our survey, which are separated

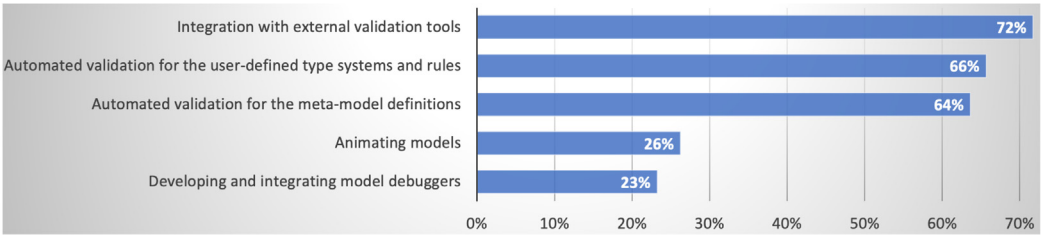

Fig. 21. The meta-modeling tool features that the participants prefer while validating their languages.

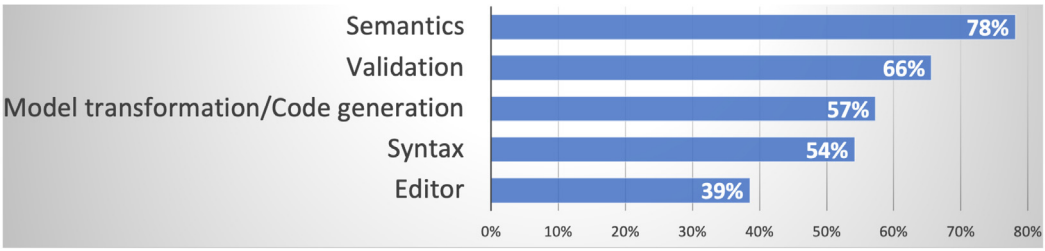

Fig. 22. The different aspects of language development that the participants consider testing.

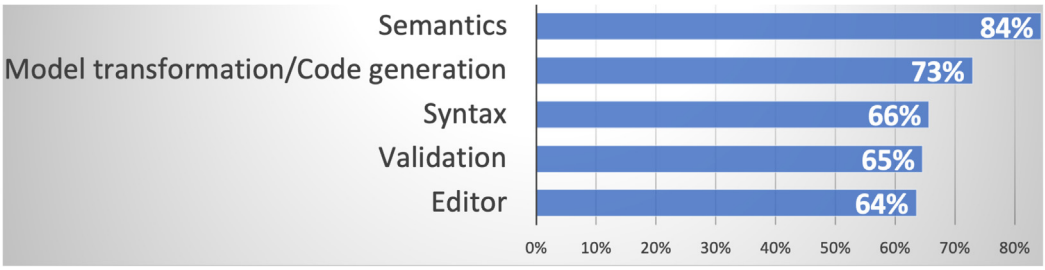

Fig. 23. The different aspects of language development that the participants consider for composition.

into eight categories: profile, the meta-modeling tool usage, language definition, editor services, model transformation, language validation, testing, and composability. Herein, the survey question on the meta-modeling tool usage led us to find the answer for our first research question (RQ1 in Section 3.1), which focus on the tools' usage frequencies. The rest of the survey questions for different categories led us to find the answers for the second and third research questions (RQ2 and RQ3) where we addressed practitioners' expectations from the meta-modeling tools and any challenges that they face with respectively. In the rest of this part, we will summarize the key findings that we obtained for each category of survey questions and thus aid in answering the research questions.

The meta-modeling tools. The top-used meta-modeling tools are Eclipse Sirius and GEMS (43%), which are followed by Metaedit+ (31%) and Xtext (29%). While the embedded domain users' top meta-modeling tool is GEMS, most of the rest of the domains' top-used meta-modeling tool is Metaedit+. Also, some countries use particular meta-modeling tools more often, such as USA using Microsoft DSL tools, Finland using Metaedit+, Turkey and the Netherlands using GEMS, and many European countries using Eclipse-based meta-modeling tools (e.g., Sirius, GEMS, and Xtext).

Language definition. The participants prefer to develop/use DSMLs with diagrammatic (88%) or textual (75%) visualizations, while DSMLs with hybrid, tabular, matrix, and map visualizations are not so popular (7%–33%). Concerning the semantics definition, most of the participants prefer the translational semantic definition techniques (i.e., model-to-model or model-to-text) and defining an interpretative semantics that is based on the direct execution of the models is not so popular among the participants. Also, the participants are highly interested in importing/exporting their language syntax (i.e., notation) and semantics definitions for, e.g., collaborative meta-modeling (82%). Besides, many participants (68%–72%) wish to define their language syntax and semantics by re-using some existing definitions and keep a repository of their definitions for versioning purposes too. Concerning the challenges on defining the language syntax and semantics, most participants complain about *(i)* the steep learning curve for defining the languages and *(ii)* the lack of support for training (e.g., tutorials, examples, guide, videos, forum, etc.) that could facilitate the use of the meta-modeling tools' utilities for language definitions.

Editor services. Most of the participants (69%) prefer to develop a modeling editor using the free-form editing mode, where a single visualization (i.e., either textual or diagrammatic) is supported and the models edited are stored persistently. The projectional editing mode that promotes hybrid modeling with multiple projections (i.e., visualizations) is rarely preferred (21%). Concerning the syntactic editor services, most participants (75%) wish the modeling editors to enable modeling via the reuse of the existing models that are stored in a repository. Also, 69% of the participants wish to compare different models with each other via some diff-like tool. Concerning the semantic editor services, most participants (81%) wish to use an error marker integrated into a modeling editor, which can highlight the semantical errors at modeling time. Also, 69% of the participants wish the modeling editors to support the automatic update of models upon any changes on the meta-model. Concerning the other editor services, most participants (77%) wish the modeling editors to generate documentations (e.g., Word, HTML) from models. The participants' top challenge here is the usability of the editors, which sometimes require considerable time and effort for the modelers to learn and use.

Model transformation/Code generation. Most participants (72%) wish the model transformation technologies to support detecting errors that may occur during the process of developing code generators

(or model transformation tools). Indeed, the model transformation technologies essentially provide a notation set with its own syntax and semantics, which needs to be used correctly so as to develop the code generators. Also, 53% of the participants wish the model transformation technologies to support syntax highlighting that can enable colored coding and thus facilitates the readability and minimize the chance of making any syntax errors. A few participants stated some challenges on the model transformation, including the lack support for *(i)* configuring the transformation process, *(ii)* adaptability for the user requirements, and *(iii)* training.

Language validation. Most participants (75%) wish to define both semantical and structural validation rules for any DSMLs that they develop. Most participants (72%) wish the meta-modeling tools to be integrated with some external tools for defining and executing language validation rules (e.g., simulators and model checkers). Also, 64%–66% of the participants wish a semi-automatic tool support for checking models against the validation rules. On the other hand, a few participants are concerned about the meta-modeling tools' inadequate support for *(i)* developing modeling editors that can aid in checking the models against the validation rules at modeling time and *(ii)* defining complex validation rules.

Testing. The top-important aspect of the language development that the participants wish to test is the language semantics definitions (78%), which could enable to determine any violations of the language definition requirements. A few participants are concerned about the meta-modeling tools' inadequate support for *(i)* specifying and executing test cases, *(ii)* integrating any tools that facilitate the test execution, and *(iii)* model-based testing.

Composability. The top-important aspect of the language development that the participants wish to define by composition is the language semantics (84%), which could actually be defined by re-using the existing language semantics (e.g., re-using the semantics of UML diagram). A few participants are concerned about *(i)* composing different languages without changing them and *(ii)* difficulties in performing composition with textual languages.

5.2. Lessons learned

We learned from the survey results that modeling (and thus meta-modeling) is essentially the concerns of several different industries where practitioners work on large and complex problems that can better be managed with the advantages of modeling such as abstraction, enhanced communication, separation of concerns, early analysis of design decisions, and generating code from models. While the survey attracted the greatest number of participants from the defense/military & aviation industry, many other industries have been participated in the survey to indicate their perspectives towards the meta-modeling tools including IT & telecommunications, automotive & transportation, healthcare & biomedical, consumer electronics, finance, government, education, etc. Also, modeling (or meta-modeling) is considered by those industries for solving problems at various different domains that include embedded, automotive, control and automation systems, web, user interface design, medical device development, IOT, enterprise solutions, data analytics, railway systems, testing, robotic, mobile, chip development, document engineering, and real-time OSs.

The survey results revealed that most of the practitioners from different industries use either of the five different meta-modeling tools which are Sirius, GEMS, Metaedit+, Xtext, and Microsoft DSL tools. Other meta-modeling tools such as ANTLR, ConceptBase, Melange, GEMOC, Graphiti, WebGME, Cameo, EVA, and JastEMF are rarely used. Among the top-five meta-modeling tools, Sirius, Xtext, and GEMS are all actually Eclipse-based tools that are based on the Eclipse Modeling Framework (EMF) and can be used as an Eclipse plug-in. It should also be noted that Xtext is the only meta-modeling tool among the top-five, which offers a textual visualization. Moreover, some countries seem to use particular meta-modeling tools. Indeed, USA prefer Microsoft DSL tools, Finland prefer Metaedit+, and most European countries prefer Eclipse-based tools and Metaedit+.

Another lesson learned about the meta-modeling tools is that some participants (12%) seem to consider EMF (Eclipse Modeling Framework) as a meta-modeling tool. Indeed, those participants used the "other" free-text area of the corresponding survey answer and typed "EMF" for their meta-modeling tool. However, EMF is essentially not a meta-modeling tool but an Eclipse modeling project whose goal is to propose a modeling framework that can be used for building a modeling editor for a given data model [43]. EMF is used by many popular Eclipse-based meta-modeling tools, including Xtext, Sirius, GEMS, Graphiti. So, these meta-modeling tools benefit the EMF utilities by enabling the users to develop textual/visual DSMLs with the EMF framework.

Concerning the lessons about the language definitions, while most practitioners prefer either diagrammatical or textual visualizations for their DSML notations, developing/using DSMLs with hybrid visualization that promote the use of multiple visualizations in a synchronized manner is neglected. One of the underlying reasons is meta-modeling tools' lack of technology support for developing hybrid visualizations. Indeed, most meta-modeling tools support textual or diagrammatic visualizations exclusively. We also observed that practitioners wish to define the language semantics by means of translations into text (e.g., source-code) or model (e.g., formal process algebra) and most of them ignore the interpretative semantics definition techniques (e.g., denotational semantics). This might be due to the fact that the meta-modeling tools essentially provide transformation technologies for defining and executing the translational semantics. Defining the interpretative semantics for notations seems to be rather addressed by the researchers in academia. Also, collaborative meta-modeling seems to be highly important for many practitioners, who indicated in the survey that importing/exporting the language definitions and versioning the definitions in a repository for later re-use are among their top-interests for meta-modeling.

Practitioners have been observed to be reluctant to develop/use DSMLs for hybrid modeling that could enable multiple stakeholders with different profiles to edit the same model using different visualizations. Indeed, as the survey results reveal, most practitioners are not interested in hybrid modeling and they essentially prefer the meta-modeling tools with the free-editing mode rather than projectional editing which promotes the same model to be edited by different projections (i.e., each considering a different visualization). This could be attributed to the fact that most practitioners are unfamiliar with the projectional editing, Indeed, the MPS meta-modeling tool that supports the projectional editing seems to be rarely used.

Another important lesson learned is about the model and meta-model re-usabilities. According to the survey results, re-usability is one of the biggest concerns for practitioners for both modeling and meta-modeling. Indeed, practitioners are highly interested in defining the language syntax and semantics by re-using the definitions of the existing languages that are kept in a repository (e.g., the UML diagram notations) without having to define the meta-models from scratch. Also, practitioners wish the modeling editors produced by the meta-modeling tools to enable re-using the existing models for creating new, extended models.

Besides reusability, another important requirement for practitioners is observed to be the model and meta-model versioning that is concerned with keeping different states of the models and language definitions (i.e., syntax and semantics) in a repository for later access and management. Indeed, many practitioners wish the meta-modeling editors to be integrated with version control systems (e.g., GIT) and provide support for creating different versions of their models (or meta-models) that can even be compared and merged.

Concerning the language validation, we learned that defining validation rules is highly important for practitioners most of whom wish the meta-modeling tools to enable defining both the semantical and

structural validation rules. Practitioners also wish the meta-modeling tools to be integrated with some validation tools such as the formal verification and model simulation tools. By doing so, the validation rules can be defined and executed on the models specified in such a way that the models can be proved (i.e., formally verified) for the validation rules.

We learned some lessons about the survey questions for understanding the challenges. Indeed, despite that each category of the survey includes a question for learning the challenges which the participants face with in that category, many participants omitted the challenge questions. We attribute this to the fact that the challenge questions are free-text and the participants have been prompt to type their own answers. Given also the feedback we got from the participants that the survey is quite longer than they expect, those participants seemed reluctant to separate further time to state their challenges. In the future, we will try to come up with a set of potential challenges and prompt the participants to choose among the pre-defined list of challenges.

Lastly, while we asked the participants a set of questions for each category of requirements and received their responses, we did not actually focus on associating the participants' meta-modeling tool choice(s) with their responses for the questions. This is firstly because the questions have been intended for learning practitioners' expectations regardless of their tool preferences and thus presented to the participants accordingly. Also, most participants seem to have chosen multiple meta-modeling tools and therefore one may not accurately associate the responses with the tool usages. Therefore, we could not discuss in our paper some potentially interesting outcomes about the relationships between the meta-modeling tools and the question answers for different tool requirements (e.g., those who prefer web-access for meta-modeling actually use the tool(s) with web-access support).

5.3. Limitations and threats to validity

In this section, we present the possible validity concerns and also the steps that we have taken to minimize them.

Construct validities of this study are concerned with the extent to which the issue relates to whether this survey measured the opinions of practitioners about meta-modeling tools [44]. We tried to collect data from different sources in order to avoid mono-operation bias (e.g., different countries, different industrial sectors, different software engineering roles, etc.). In order to mitigate the feeling of being evaluated, we informed participants prior to the survey that our motivation in this study was to take a snapshot of the industry and that we will not collect any identifying information; hence participants will remain anonymous. In our measurement strategy, what we did was common to other survey studies (e.g., [23,45]) To minimize internal validity [44], one of the important decisions during the design, is the selection of questions and the answer set of these questions. In this study, we benefitted from related works (See Section 2), our past experience on survey design, and personal experience (both industrial and academic) on meta-modeling tools. Instrumentation was improved by using a pilot study (See Section 3.2).

As Wohlin et al. presented, external validity is concerned with the extent to which the results can be generalized [44]. In order to decrease the effect of possible dominant participant number in a specific sector due to authors' network, the survey has been distributed to various practitioners via different channels in all around the world for different industrial sectors (See Section 3.3). Therefore, we have done our best to reach the subjects with a variety of different backgrounds. Our sample size is more than 100 respondents, which might help to characterize various cross-factors meaningfully. However, the results are based on the analysis of data consisting of the perceptions of our survey respondents; so we cannot claim that the same results might be obtained in other parts of the world though we believe that our approach can be replicated in other contexts. According to our respondent demographics, the embedded domain is the top popular one (56%), which might have some impact on the results drawn out of our dataset. Also, note that 22% of the participants are researchers, who work in universities (or in any other research institutions) and thus it may be considered as a possible threat for the validity of the results. However, we believe that such ratios might not affect the overall findings. Nevertheless, we also reported demographics of the participants and the readers will be able to evaluate the applicability in different contexts with the raw data.[1]

To strengthen the conclusion validity of our study, for each RQ, we attempted to reduce the bias by seeking support from the statistical results [44]. Although we collected data from different sources, we do not have any intentions to generalize our findings since these results depend the company and practitioner demographics. Since we reported demographic information covered in our study, note that all the conclusions that we drew, are strictly traceable to data.

6. Conclusion

In this study, we aimed to learn practitioners' perspectives towards the meta-modeling tools and conducted a survey among 103 practitioners from 24 different countries. The survey participants represent the different profiles of the population who differ in terms of the work industries, the problem domains, job positions, and years of experiences. Our survey investigates three important research questions, which essentially focus on the usage frequencies of the existing meta-modeling tools, practitioners' expectations from the meta-modeling tools, and any challenges that practitioners face with.

According to the survey results, the top-used meta-modeling tools are Eclipse Sirius and GEMS (43%), which are followed by Metaedit+ (31%) and Xtext (29%). The meta-modeling tool features have been categorized in the survey as the language definition, editor services, model transformation/code-generation, language validation, and language composability, and each category here led to very interesting results in the survey. Concerning the language definition, while the diagrammatic (88%) and textual (75%) visualizations are highly popular among the participants, other types of visualizations (e.g., tabular) and hybrid visualization are rarely preferred. Many of the participants wish to define the language semantics using the translational semantics definition technique that can either be model-to-model or model-to-text. The interpretative semantics definition is neglected. The participants are also willing to communicate their language definitions via the importing/export facility and keep the record of the language definitions that can be managed in terms of versions. The participants' main challenges on the language definition are the lack of support for training (e.g., tutorials, examples, videos, and support-desk) and the steep learning curve. Concerning the editor services, while 69% of the participants wish to use the free-form editing mode. The projectional editing mode for the hybrid modeling with multiple visualizations is not popular. The top features that are desired by the participants to be supported by the modeling editors are an error marker for semantical aspects (81%), document generation (77%), model re-use (75%), a model comparison (69%), and the automatic update of the models when meta-model changes (69%). The main challenge on the editor services is to do with the editor usability (e.g., the number of clicks needed for modeling and meta-modeling). Concerning the model transformation/code-generation, the participants wish to use a transformation technology with the error detection (72%) and syntax highlighting (53%) features. The challenges here include the lack of support for training and adapting the tools developed with the changing user requirements. Concerning the language validation, many participants (75%) are willing to define both semantic and structural validation rules for DSMLs and the top-desired tool feature for the language validation is the support for integration with the external analysis and validation tools (72%). The challenges here are to do with the real-time checking of the validation rules and complexity management. Concerning the language testing, many participants (78%)

wish to test the language semantics. The challenges on the language testing are to do with the lack of support for specifying and executing test cases, model-based testing, and integrating with test automation tools. Concerning the language composability, most participants (84%) wish to define the language semantics via composition. The challenges include the difficulties in using the composed languages and performing composition with textual languages.

We plan to validate the survey results via the international R&D projects that we are involved in. To this end, we will consider the PANORAMA project,[5] which is labeled by the European Union's EUREKA Cluster program ITEA (Information Technology for European Advancement). In PANORAMA, the Eclipse-based meta-modeling tools are being used to develop a scenario definition language for autonomous vehicles along with the supporting toolset (editor and code generator). So, we will survey and interview the practitioners who develop the language with some pre-determined list of questions that are derived from the survey results. Moreover, we will consider our Ph.D. students who use the Eclipse-based meta-modeling tools and Metaedit+ to develop DSMLs for different purposes (e.g., extending UML, design-pattern based modeling, model-based testing, etc.) and validate the survey results.

Lastly, we believe that the survey results will be highly useful for the tool vendors in determining the needs of practitioners from the meta-modeling tools. So, in the next release of their tools, the tool vendors may consider the missing features in their tools that are crucial for practitioners. Also, the survey results may be utilized by the practitioners who use or develop modeling languages. Indeed, practitioners may learn about the top-used meta-modeling tools for the domain of their interests and the important features of the meta-modeling tools from the practitioners' perspectives. So, practitioners may consider benefitting any particular meta-modeling tools for their domain and the particular features that they are interested in.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank all practitioners who have contributed to our survey. Also, we would like to thank all experts who are involved in the survey's pilot study and provide us very useful feedback about the survey design.

References

[1] E. Seidewitz, What models mean, IEEE Softw. 20 (5) (2003) 26–32, http://dx.doi.org/10.1109/MS.2003.1231147.

[2] B. Selic, The pragmatics of model-driven development, IEEE Softw. 20 (5) (2003) 19–25, http://dx.doi.org/10.1109/MS.2003.1231146.

[3] T. Kühne, Matters of (meta-)modeling, Softw. Syst. Model. 5 (4) (2006) 369–385, http://dx.doi.org/10.1007/s10270-006-0017-9.

[4] S.J. Mellor, S. Kendall, A. Uhl, D. Weise, MDA Distilled, Addison Wesley Longman Publishing Co., Inc., USA, 2004.

[5] D. Akdur, B. Say, O. Demirörs, Modeling cultures of the embedded software industry: feedback from the field, Softw. Syst. Model. (2020) http://dx.doi.org/10.1007/s10270-020-00810-9.

[6] S. Kelly, J. Tolvanen, Domain-Specific Modeling - Enabling Full Code Generation, Wiley, 2008, http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470036664.html.

[7] J. Rumbaugh, I. Jacobson, G. Booch, Unified Modeling Language Reference Manual, second ed., Pearson Higher Education, 2004.

[8] P.H. Feiler, B.A. Lewis, S. Vestal, The SAE architecture analysis & design language (AADL): A standard for engineering performance critical systems, in: IEEE Intl Symp. on Intell. Control, 2006, pp. 1206–1211, http://dx.doi.org/10.1109/CACSD.2006.285483, //aadl.info.

[9] T. Clark, P. Sammut, J.S. Willans, Applied metamodelling: A foundation for language driven development (third edition), 2015, CoRR abs/1505.00149 http://arxiv.org/abs/1505.00149.

[10] S.J. Mellor, S. Kendall, A. Uhl, D. Weise, MDA Distilled, Addison Wesley Longman Publishing Co., Inc., USA, 2004.

[11] G.D. Plotkin, The origins of structural operational semantics, J. Log. Algebr. Program. 60–61 (2004) 3–15, http://dx.doi.org/10.1016/j.jlap.2004.03.009.

[12] D.A. Schmidt, Denotational Semantics: A Methodology for Language Development, William C. Brown Publishers, Dubuque, IA, USA, 1986.

[13] S.G. Hervey, Axiomatic Semantics: A Theory of Linguistic Semantics, Scottish Academic Press, 1979.

[14] J.A. Goguen, G. Malcolm, Algebraic Semantics of Imperative Programs, MIT Press, Cambridge, MA, USA, 1996.

[15] M. Fowler, Language workbenches: The killer-app for domain specific languages, 2005, (Online; Accessed 9-July-2020) https://martinfowler.com/articles/languageWorkbench.html.

[16] S. Erdweg, T. van der Storm, M. Völter, M. Boersma, R. Bosman, W.R. Cook, A. Gerritsen, A. Hulshout, S. Kelly, A. Loh, G.D.P. Konat, P.J. Molina, M. Palatnik, R. Pohjonen, E. Schindler, K. Schindler, R. Solmi, V.A. Vergu, E. Visser, K. van der Vlist, G. Wachsmuth, J. van der Woning, The state of the art in language workbenches - conclusions from the language workbench challenge, in: M. Erwig, R.F. Paige, E.V. Wyk (Eds.), Software Language Engineering - 6th International Conference, SLE 2013, Indianapolis, in, USA, October 26-28, 2013. Proceedings, in: Lecture Notes in Computer Science, Vol. 8225, Springer, 2013, pp. 197–217, http://dx.doi.org/10.1007/978-3-319-02654-1_11.

[17] G. Liebel, N. Marko, M. Tichy, A. Leitner, J. Hansson, Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice, Softw. Syst. Model. 17 (1) (2018) 91–113, http://dx.doi.org/10.1007/s10270-016-0523-3.

[18] L.T.W. Agner, I.W. Soares, P.C. Stadzisz, J.M. SimãO, A Brazilian survey on UML and model-driven practices for embedded software development, J. Syst. Softw. 86 (4) (2013) 997–1005, http://dx.doi.org/10.1016/j.jss.2012.11.023.

[19] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, G. Reggio, Preliminary findings from a survey on the MD state of the practice, in: Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement, in: ESEM '11, IEEE Computer Society, USA, 2011, pp. 372–375, http://dx.doi.org/10.1109/ESEM.2011.51.

[20] J. Whittle, J.E. Hutchinson, M. Rouncefield, The state of practice in model-driven engineering, IEEE Softw. 31 (3) (2014) 79–85, http://dx.doi.org/10.1109/MS.2013.65.

[21] P. Mohagheghi, W. Gilani, A. Stefanescu, M.A. Fernández, An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases, Empir. Softw. Eng. 18 (1) (2013) 89–116, http://dx.doi.org/10.1007/s10664-012-9196-x.

[22] F. Tomassetti, M. Torchiano, A. Tiso, F. Ricca, G. Reggio, Maturity of software modelling and model driven engineering: A survey in the Italian industry, in: M.T. Baldassarre, M. Genero, E. Mendes, M. Piattini (Eds.), 16th International Conference on Evaluation & Assessment in Software Engineering, EASE 2012, Ciudad Real, Spain, May 14-15, 2012. Proceedings, IET - The Institute of Engineering and Technology / IEEE Xplore, 2012, pp. 91–100, http://dx.doi.org/10.1049/ic.2012.0012.

[23] D. Akdur, V. Garousi, O. Demirörs, A survey on modeling and model-driven engineering practices in the embedded software industry, J. Syst. Archit. Embed. Syst. Des. 91 (2018) 62–82, http://dx.doi.org/10.1016/j.sysarc.2018.09.007.

[24] D. Bork, D. Karagiannis, B. Pittl, A survey of modeling language specification techniques, Inf. Syst. 87 (2020) http://dx.doi.org/10.1016/j.is.2019.101425.

[25] J. Whittle, J.E. Hutchinson, M. Rouncefield, H. Burden, R. Heldal, Industrial adoption of model-driven engineering: Are the tools really the problem?, in: A. Moreira, B. Schätz, J. Gray, A. Vallecillo, P.J. Clarke (Eds.), Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings, in: Lecture Notes in Computer Science, Vol. 8107, Springer, 2013, pp. 1–17, http://dx.doi.org/10.1007/978-3-642-41533-3_1.

[26] J. Cabot, E. Teniente, Constraint support in MDA tools: A survey, in: A. Rensink, J. Warmer (Eds.), Model Driven Architecture - Foundations and Applications, 2nd European Conference, ECMDA-FA 2006, Bilbao, Spain, July 10-13, 2006, Proceedings, in: Lecture Notes in Computer Science, Vol. 4066, Springer, 2006, pp. 256–267, http://dx.doi.org/10.1007/11787044_20.

[27] R.F. Paige, D. Varró, Lessons learned from building model-driven development tools, Softw. Syst. Model. 11 (4) (2012) 527–539, http://dx.doi.org/10.1007/s10270-012-0257-9.

[28] J.L. Pérez-Medina, S. Dupuy-Chessa, A. Front, A survey of model driven engineering tools for user interface design, in: M. Winckler, H. Johnson, P.A. Palanque (Eds.), Task Models and Diagrams for User Interface Design, 6th International Workshop, TAMODIA 2007, Toulouse, France, November 7-9, 2007, Proceedings, in: Lecture Notes in Computer Science, Vol. 4849, Springer, 2007, pp. 84–97, http://dx.doi.org/10.1007/978-3-540-77222-4_8.

[5] Panorama web-site https://panorama-research.org/.

[29] S. Popoola, J.C. Carver, J. Gray, Modeling as a service: A survey of existing tools, in: L. Burgueño, J. Corley, N. Bencomo, P.J. Clarke, P. Collet, M. Famelis, S. Ghosh, M. Gogolla, J. Greenyer, E. Guerra, S. Kokaly, A. Pierantonio, J. Rubin, D.D. Ruscio (Eds.), Proceedings of MODELS 2017 Satellite Event: Workshops (ModComp, ME, EXE, COMMitMDE, MRT, MULTI, GEMOC, MoDeVVa, MDETools, FlexMDE, MDEbug), Posters, Doctoral Symposium, Educator Symposium, ACM Student Research Competition, and Tools and Demonstrations Co-Located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA, September, 17, 2017, in: CEUR Workshop Proceedings, 2019, CEUR-WS.org, 2017, pp. 360–367, http://ceur-ws.org/Vol-2019/mdetools_5.pdf.

[30] M. Ozkaya, Are the UML modelling tools powerful enough for practitioners? A literature review, IET Softw. 13 (5) (2019) 338–354, http://dx.doi.org/10.1049/iet-sen.2018.5409.

[31] A. El Kouhen, C. Dumoulin, S. Gérard, P. Boulet, Evaluation of modeling tools adaptation, Tech. rep., 2012, https://hal.archives-ouvertes.fr/hal-00706701.

[32] H. Kern, A. Hummel, S. Kühne, 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, in: SPLASH '11 Workshops, Association for Computing Machinery, New York, NY, USA, 2011, pp. 7–12, http://dx.doi.org/10.1145/2095050.2095053.

[33] H. Kern, Study of interoperability between meta-modeling tools, in: M. Ganzha, L.A. Maciaszek, M. Paprzycki (Eds.), Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, September 7-10, 2014, in: Annals of Computer Science and Information Systems, Vol. 2, 2014, pp. 1629–1637, http://dx.doi.org/10.15439/2014F255.

[34] S. Erdweg, T. van der Storm, M. Völter, L. Tratt, R. Bosman, W.R. Cook, A. Gerritsen, A. Hulshout, S. Kelly, A. Loh, G.D.P. Konat, P.J. Molina, M. Palatnik, R. Pohjonen, E. Schindler, K. Schindler, R. Solmi, V.A. Vergu, E. Visser, K. van der Vlist, G. Wachsmuth, J. van der Woning, Evaluating and comparing language workbenches: Existing results and benchmarks for the future, Comput. Lang. Syst. Struct. 44 (2015) 24–47, http://dx.doi.org/10.1016/j.cl.2015.08.007.

[35] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang, What industry needs from architectural languages: A survey, IEEE Trans. Software Eng. 39 (6) (2013) 869–891, http://dx.doi.org/10.1109/TSE.2012.74.

[36] M. Ozkaya, Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? Inf. Softw. Technol. 95 (2018) 15–33, http://dx.doi.org/10.1016/j.infsof.2017.10.008.

[37] M. Ozkaya, The analysis of architectural languages for the needs of practitioners, Softw. Pract. Exper. 48 (5) (2018) 985–1018, http://dx.doi.org/10.1002/spe.2561.

[38] P.C. Clements, A survey of architecture description languages, in: Proceedings of the 8th International Workshop on Software Specification and Design, in: IWSSD '96, IEEE Computer Society, Washington, DC, USA, 1996, p. 16.

[39] N. Medvidovic, R.N. Taylor, A classification and comparison framework for software architecture description languages, IEEE Trans. Softw. Eng. 26 (1) (2000) 70–93.

[40] R.M. Groves, F.J. Fowler Jr, M.P. Couper, J.M. Lepkowski, E. Singer, R. Tourangeau, Survey Methodology, second ed., John Wiley & Sons, 2009.

[41] R. Popping, Analyzing open-ended questions by means of text analysis procedures, Bull. Sociol. Methodol. Bull. Méthodol. Sociol. 128 (1) (2015) 23–39, http://dx.doi.org/10.1177/0759106315597389.

[42] R.D. Fricker, Sampling methods for web and e-mail surveys, in: The SAGE Handbook of Online Research Methods, Sage Publications New Delhi, India, 2008, pp. 195–216.

[43] F. Budinsky, D. Steinberg, R. Ellersick, Eclipse Modeling Framework: A Developer's Guide, Addison-Wesley Professional, 2003.

[44] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012, http://dx.doi.org/10.1007/978-3-642-29044-2.

[45] M. Ozkaya, F. Erata, A survey on the practical use of UML for different software architecture viewpoints, Inf. Softw. Technol. 121 (2020) 106275, http://dx.doi.org/10.1016/j.infsof.2020.106275.