

Strategies to Evolve ExM Notations Extracted from a Survey with Software Engineering Professionals Perspective

Bruno Gabriel Araújo Lebtag  [Federal University of Goiás | brunogabriel@inf.ufg.br]

Paulo Gabriel Teixeira  [Federal University of Goiás | paulogabriel@inf.ufg.br]

Rodrigo Pereira dos Santos  [Federal University of the State of Rio de Janeiro | rps@uniriotec.br]

Davi Viana  [Federal University of Maranhão | davi.viana@ufma.br]

Valdemar V. Graciano Neto  [Federal University of Goiás | valdemarneto@ufg.br]

Abstract

Contemporary complex systems often exhibit dynamic structures and behaviors, several components/systems involved, and multiple interoperability links. Those systems have been exposed to fragilities of traditional software specification languages (e.g. UML and SysML), since such languages were designed to document single (not multiple interoperating) systems. Those limitations can potentially further compromise the quality of the final software product. In this context, Executable Models (ExM) technology, such as simulation models, models@runtime and executable UML, satisfy the aforementioned requirements by supporting engineers with visualization of the system structures (still at design-time) and the ability to exercise their behaviors and interactions. In our prior study, we presented the results of an exploratory study on the perceptions of those professionals (from both industry and academia) regarding the use of ExM to solve problems in their current practice. We exposed 58 professionals (researchers and practitioners) to situations to solve problems using a specific type of ExM (DEVS simulation models), based on survey research. Responses were quantitatively and qualitatively analyzed. In this article, we extended the obtained results by analyzing and compiling a list of strategies to improve ExM notations to better address the needs of software engineering professionals. Later, we assessed those strategies with software engineering researchers to confirm the importance of the proposed strategies. Results revealed that executable languages still require advances to bring them even closer to the current software engineering practice and towards a more significant adoption in the future. The proposed strategies focus on improvements on the robustness of the ExM notations, visual representation of the models, the usability of the models, and user support.

Keywords: executable models, software-intensive systems, simulation, survey research

1 Introduction

Software systems have increased their dimension and complexity and become large-scale to address emerging business needs. They have been pressured to interoperate with several other systems and to establish even ephemeral/temporary links with other systems to comply with emerging business needs (Fernandes et al., 2018). As such, those systems and the forthcoming generation of systems, as smart cities, will be composed of several independent systems with a dynamic architecture that should be analyzed, predicted, and evaluated still at design-time and, hopefully, also monitored at runtime. On the other hand, the world is experiencing a blackout of specialized labor to work in the software industry¹, which has pressured researchers and practitioners to propose solutions progressively more intuitive that enable to involve even non-specialists in the software production endeavor as earlier as possible. Emerging technologies, such as Low-Code Development (Prinz et al., 2021), and even more traditional ones as model-driven engineering (MDE) approaches - which can include modeling and simulation (M&S) practices - are emerging, reascending and becoming popular for supporting the engineering of complex systems, once they enhance the presence of models in system life cycle while leveraging the level of abstraction and offering a vi-

sual appeal to deal with the contemporary systems inherent dynamics and complexity (Mahmood et al., 2013; Bogado et al., 2014; Neis et al., 2019).

Several MDE solutions use general-purpose static languages to build systems, e.g. UML or SysML, i.e., despite the existence of some executable versions of such formalisms, their models are not natively animated to dynamically represent the systems' behaviors, for instance. Moreover, those languages were not conceived to precisely capture multiple systems, but to support the documentation of single systems with multiple models. Moreover, interoperability links among those systems can be established at runtime, which can be hard to model using only static notations. To solve that problem, one possible solution is Executable Models (ExM²) (Dahmann et al., 2017a). This technology offers the possibility of representing (and executing) the structural and behavioral attributes of the whole system through its corresponding models/documentation (Hu et al., 2014; Gray and Rumpe, 2016; Dahmann et al., 2017a; Hojaji et al., 2019), which can be very useful to deal with dynamic environments and evolving systems.

When using ExM to engineer current and forthcoming systems, ExM is demanded as means to provide (i) continuous monitoring of the systems status, (ii) dynamic reasoning about its underlying architecture to underpin strategic

¹<https://www.bloomberg.com/news/newsletters/2021-05-27/labor-shortages-are-plaguing-tech-companies-too>

²Herein, the acronym ExM will be interchangeably used to express both singular and plural forms: executable model and executable models.

decisions, and (iii) a visual and intuitive perception of the whole system. However, we have observed an opposite phenomenon in the industry. Since 2011, there is evidence about a decreasing use of models such as UML and SysML (Torchiano et al., 2011; Agner et al., 2013; Gorscheck et al., 2014). Moreover, there is a resistance to MDE due to a diversity of reasons, including lack of tool support and domain-specific scope notations so that they have been applied to only a few domains (Torchiano et al., 2011; Agner et al., 2013; Whittle et al., 2017). We claim that professionals³ still lack abilities to use ExM to solve even simple problems in their practice and maybe ExM lack syntactic or visual complements that may hamper their adoption.

In this context, the goals of this study are: (i) to perform a survey with software engineering professionals to collect their perspective in the use of ExM notations to develop software, (ii) to develop a list of strategies based on the results of the survey to evolve ExM notation to be aligned with software engineering professionals needs and (iii) to assess the obtained strategies with software engineering researchers to identify their agreement. Thus, the main contribution of this paper is twofold: to report on the results of survey research on the understandability and expressiveness of a specific type of ExM (DEVS Kim and Zeigler (1987) simulation models) from the perspective of software engineering professionals and to present a set of 19 strategies to evolve ExM notations based on the results of the survey. The software engineering professionals that took part in the survey used the visual representation of the simulation from two given scenarios as a basis to solve small-scale problems similar to situations they often face in their routine. Based on this experience, we collected information to answer the established research questions: “**What are the perceptions of software engineering professionals about of adoption of DEVS executable simulation models to solve current practice problems?**” and “**How to support the evolution of ExM notations to be aligned with software engineering professionals’ needs?**”. We claim that *if software engineering professionals face difficulties to solve problems at a small-scale using ExM, they should manage even more difficulties when using ExM to deal with complex scenarios* (Boehm, 2006).

In our previous study, we conducted survey research that received 58 answers: 44 from Brazil and 14 from nine different countries. As a result, participants attest several benefits and opportunities for using ExM, such as: (i) providing a broad view of the problem and solution, (ii) serving as a communication document, and (iii) facilitating the inclusion of new members in the development team. However, participants also reported new opportunities mainly related to the improvements in ExM languages capabilities, visual support, and ExM education and training.

We extended our prior study Lebtag et al. (2020) by (i) elaborating a compilation of strategies to guide the evolution of ExM notations and (ii) the evaluation of the set of strategies with experts. The strategies were derived from the qualitative analysis conducted over the answers in our first study. The elaborated set contains 19 strategies split into

³In the scope of this article, we use the term *software engineering professionals*, or simply *professionals* to denote the academia and industry researchers and practitioners.

four categories of improvements of ExM notations: (i) notation, (ii) visual presentation, (iii) usage, and (iv) user support. We assessed these strategies with 13 software engineering researchers to validate their agreement with the proposed strategies and also discussed the implication of the results for practice and theory. The strategies had a positive reception among researchers since most of them received “agree”.

Figure 1 presents the methodology used throughout the development of this work. In the first section of the methodology, we performed survey research (described next), in which we planned a questionnaire, executed a survey with software engineering professionals, analyzed the results and elaborated a study reporting the results (Lebtag et al., 2020). Next, we re-analyzed the obtained results and looked into problems reported by participants and from those results, we elaborated and created a list of strategies to deal with those reported problems. Lastly, we elaborated a likert questionnaire and invited software engineering researchers to assess those strategies.

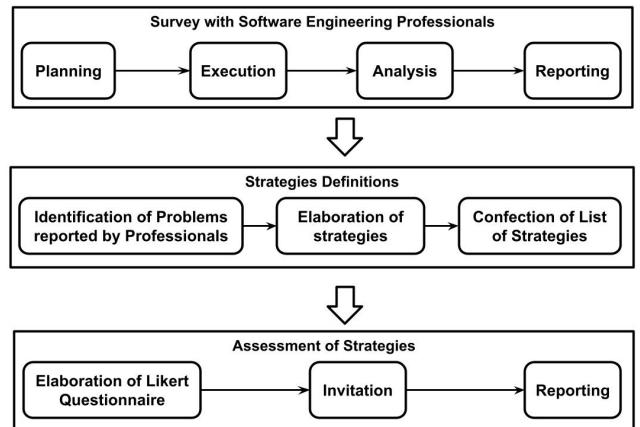


Figure 1. The process used in this article from the survey to the confection of the strategies and the assessment of the strategies.

The remainder of this article is organized as follows: Section 2 presents the paper background; Section 3 describes the survey research, including its planning, execution, results, qualitative analysis and discussion of the results; Section 4 presents a list of strategies to evolve ExM notation to be better suited to software engineers needs; Section 5 presents the assessment of the presented strategies; In Section 6, we discuss the implications to theory and practices of the presented strategies; Section 7 brings the study limitations; finally, Section 8 concludes the paper with final remarks.

2 Background

The adoption of models is prominent to support software quality assurance (Michael et al., 2011; Bogado et al., 2014). Moreover, given the increasingly complex scenarios, we must face in the forthcoming years, ExM can be used as a means to carry out their analysis and evaluation (Michael et al., 2011).

Herein, we understand an ExM as any model that can be run. ExM can be used as: (i) executable “simulation” models, i.e. the model subject to be executed works as a prototype that

is not used to directly track the final system that it reflects; and (ii) executable “design” models, i.e. the model that is executed is the same model that will be used to be part of the final product, such as *models@runtime*. Discrete-Event System Specification (DEVS) (Kim and Zeigler, 1987; Zeigler et al., 2016) and Colored Petri Nets (CPN) (Levis and Wagenhals, 2000) are examples of the former case, while fUML (OMG Executable UML, 2018) and ALF (OMG Executable UML, 2017) are expressive examples of the latter case.

Most of ExM are animated, i.e. some tools offer the possibility to animate the execution steps and states transitions so that an observer can visualize them and see the systems represented as a model (e.g. DEVS, CPNTools, IBM Rhapsody) (Dahmann et al., 2017a). In situations where visualization is unfeasible (e.g. due to the presence of thousands of interoperability links (Fernandes et al., 2018) and subsystems), ExM still offer and rely on an *execution engine*. The engine can suppress the animation but still execute the model step-by-step and deliver a diagnosis on the system at the end.

Levis and Wagenhals (2000) argue that ExM can foster intercommunication in the development team. Moreover, the authors demonstrate the capabilities provided by the software architecture, its structure and dynamics to stakeholders using ExM. An ExM behaves as like a software application by receiving inputs from external sources, processing and manipulating data, and returning generated output.

3 Survey on Executable Models

The survey is a technique to acquire knowledge by hearing the voices of professionals and observing their responses, attitudes and behaviors to offer a broader understanding of a given phenomenon of interest (Wohlin et al., 2012). Survey research has been extensively used in several knowledge areas. Particularly in software engineering, it is considered one of the most used research methods for conducting exploratory empirical investigations (Molléri et al., 2016). ExM users are frequently software engineers. Since we are interested in understanding how those professionals perceive the use of ExM in practice, we decided to use the survey as the strategy for this study. We elaborated a research protocol and a corresponding questionnaire to collect perceptions of software engineering professionals on the use of ExM to solve problems in two small-scale scenarios via an online questionnaire.

For this survey, we developed a protocol inspired by the guidelines proposed by Kasunic (Kasunic, 2005), Linåker et al. (2015) and Molléri et al. (2016). We established our methodology according to four steps: **Step 1) Planning, Step 2) Execution, Step 3) Analysis, and Step 4) Reporting**, as shown in Figure 2: **Step 1 (Planning)** is composed of (i) Identification of the research objectives, (ii) Identification & characterization of the target audience, (iii) Design of the sampling plan, and (iv) Design & elaboration of the questionnaire. **Step 2 (Execution)** was performed with (v) a Pilot study and (vi) Survey invitation. **Step 3 (Analysis)** and **Step 4 (Reporting)** were conducted as separate steps. The following sections detail those steps and how they were conducted.

3.1 Planning

In Section 1, the following main questions (MQ) were established: “**MQ1. What are the perceptions of software engineering professionals about of adoption of DEVS executable simulation models to solve current practice problems?**” and “**MQ2. How to support the evolution of ExM notations to be aligned with software engineering professionals’ needs?**”. To solve MQ1, we conducted the survey relying on the research objectives and sub-questions (RQ), as follows. From the results obtained, we extracted the 19 strategies that allowed us to answer MQ2.

1. Identification of the research objectives. The established research objectives to answer MQ1 referred to (i) collect evidence on the understandability and expressiveness of a specific type of ExM (DEVS simulation models) from the perspective of software engineering professionals; and, as a result, (ii) establish a set of 19 strategies to evolve ExM notations based on the survey results. Understandability can be measured as the capability of the professionals to correctly answer our raised technical questions using the ExM as a source for answer. Meanwhile, expressiveness is a manifold characteristic comprising strengths, facilities, visual characteristics and other attributes a language has.

To achieve these objectives, we collect professionals’ perceptions, possible applications they could envision, possible improvements for the language as well as what they perceive as positive and negative on the use of ExM in their current practice. RQ1 was established to target understandability and RQ2 to RQ5 address expressiveness.

For achieving the established research objectives, therefore, we derived the following research questions (RQ):

RQ1 - What is the understandability degree achieved by software engineering professionals during the use of ExM to solve the presented low-scale problems?

Rationale: In this RQ, we are using the obtained score achieved by software engineering professionals to indicate how well is the understandability of the presented ExM. Several ExM share similar features and the obtained results can offer a broad view of ExM area. This RQ is answered through a quantitative analysis of the results.

RQ2 - What are the strengths of ExM?

Rationale: This RQ was designed to collect the benefits that ExM can bring to software development. This information may help software engineers in the process of decision-making, but also support the research and industry communities with a list of perceived benefits.

RQ3 - What application opportunities do professionals envision for using ExM in their projects?

Rationale: We aim to know what opportunities for the application of ExM in their daily use are envisioned by participants. This result can support the research and industry communities with inputs for new applications and insights for ExM, such as gaps to be exploited.

RQ4 - What could be added to ExM visual perceptions

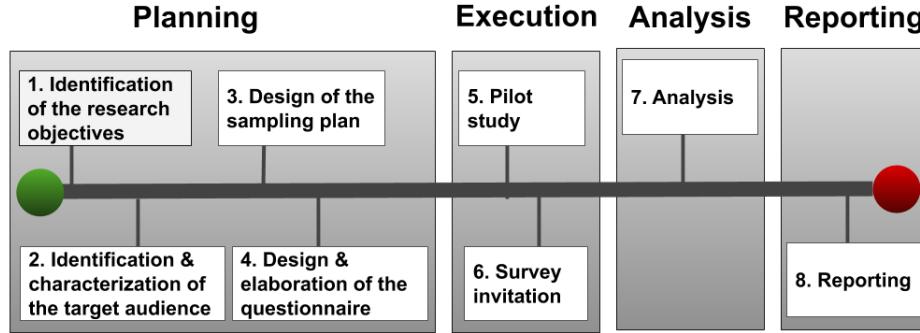


Figure 2. A survey workflow prescribed by (and adapted from) Kasunic (Kasunic, 2005) and mapped for the elaborated methodology.

in order to improve/enrich useful information for software engineers?

Rationale: ExM rely on models themselves. Consequently, models of visual aspects are important to conduct activities. We ask professionals to tell, from their perspective, what could be improved to foster an evolution in the current executable languages to make them closer to the software engineers' needs.

RQ5 - What difficulties did software engineers face during the use of ExM to solve the presented low-scale problems?

Rationale: By answering this question, we are able to identify a list of difficulties to be solved in order to improve ExM receptivity by software engineers.

RQ6 - What are the weaknesses of ExM?

Rationale: Conversely to the prior question, we also aim to collect the perceived weaknesses of ExM from the software engineers' perspective. We intend to produce a list of perceived weaknesses to support the research and practice communities with new opportunities and gaps to be explored.

2. Identification of target audience. This study's population is formed by professionals who work on software engineering in either industry or academia.

3. Design of sampling plan. The adopted sampling plan is classified by the literature as an *accidental sampling*, i.e. by sending invitations to a large number of participants and *snowballing sampling*, i.e. by asking participants to invite close professionals (Linåker et al., 2015).

4. Design and elaboration of the questionnaire.

The objective was to collect perceptions of software engineering professionals on the use of ExM to solve problems in two small-scale scenarios via an online questionnaire. Due to the current COVID-19 pandemic situation, we avoided face-to-face activities that could put participants at risk. Then, an online questionnaire was elaborated as the main instrument for the conduction of the study. The scope was the *executable simulation models*, more specifically the *DEVS formalism* (Zeigler et al., 2018). The rationale behind it is the fact that DEVS is an ExM language used to analyze and evaluate, for instance, software architectures and it is also highly used to analyze complex systems (Bogado et al.,

2014; Zeigler et al., 2016; Neto et al., 2018; Manzano et al., 2020).

In Listing 1, we provide a small fragment of code used in this study which represents a Hibernate transaction state machine. We claim that, if we want to progress to use ExM for complex systems, software engineering professionals must firstly be able to deal with ExM in simpler systems/scenarios. Therefore, we exposed the participants to two videos of DEVS simulation models (a type of ExM) being executed and running on MS4ME⁴ - each video representing a different scenario. The first video⁵ showed a scenario on how user comments on some web forums are submitted to Pluck⁶ using AJAX technologies⁷. The second video⁸ shows a Spring⁹ application using Hibernate¹⁰ for transaction management. A short explanatory text was made available together with each video. For each pair text-video, in conformance with Kasunic (2005), Linåker et al. (2015) and Molléri et al. (2016) guidelines, we asked participants to answer the study questions.

Listing 1: Code used in Hibernate JDBC Transaction for SQ2.

```
to start passivate in s0!
when in s0 and receive Commit go to s1!
when in s0 and receive Begin go to s0!
when in s0 and receive Rollback go to s0!
hold in s1 for time 1!
after s1 output Flush!
from s1 go to s0!
```

The structure of the questionnaire was conceived as illustrated in Table 1. We elaborated (i) demographic questions (DQ), i.e. characterization questions to obtain the participants' profile, (ii) study questions (SQ) to ask the participants specific information about the problems being presented in each video, and (iii) attitudinal questions (AQ), whose purpose was to collect the perceptions of participants and their opinions after being exposed to both videos and after solved problems by solely watching the running of an ExM. The SQ presented two scenarios - with two questions each - regarding the videos, in a total of four problems to be solved using ExM. We also added two questions for each problem

⁴<http://www.ms4systems.com/pages/ms4me.php>

⁵<https://youtu.be/S9a9bg4S29w>

⁶<http://directwebremoting.org/dwr/index.html>

⁷<https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX>

⁸<https://youtu.be/6jeELosgx90>

⁹<https://spring.io/>

¹⁰<http://hibernate.org/>

regarding the confidence of participants in their answers and one final open-ended question for each scenario where participants could express their difficulties.

Given the low adoption of ExM in the industry, we offered participants the opportunity to use them in a limited set-up. The main goal is not to evaluate the extension to which ExM can help participants on answering the questions, but to expose them to analyze and solve the presented problems using ExM and collect their perceptions from that experience. We elaborated closed-ended questions (CEQ) and open-ended questions (OEQ). CEQ were used to standardize the answers, facilitate reading and synthesize the required information. OEQ allowed participants to better describe their responses and expose their experience and perceptions after solving problems using the visual part of an ExM. CEQ supported DQ and SQ, as shown in Table 2 and Table 3, respectively. The aim of CEQ in SQ was to observe whether the participants were capable of objectively understanding the model and correctly answer questions related to the problem. In turn, the aim of OEQ in SQ was to collect participants' perceptions for enabling a qualitative analysis. AQs were essentially composed of OEQ to enable a qualitative analysis of the provided answers, as shown in Table 4.

Table 1. Questionnaire Layout.

Demographic Questions	Characterization Questions
Study Questions 1	Video Problem 1
	Closed-ended Questions
	Open-ended Questions
Study Questions 2	Video Problem 2
	Closed-ended Questions
	Open-ended Questions
Attitudinal Questions	Open-ended Questions

Table 2. Demographic open-ended questions (OEQ) and closed-ended questions (CEQ).

ID	Characterization Questions (CQ)
OEQ1	What is your affiliation?
OEQ2	What is your age?
OEQ3	In what country do you live?
CEQ1	What is your education level?
CEQ2	What is your main occupation?
OEQ4	What is your gender?
CEQ3	What is your professional experience in academia?
CEQ4	What is your professional experience in industry?
CEQ5	What is your experience with static models?

3.2 Execution

According to the survey workflow (Figure 2), in the execution phase, we have two activities: 5. Pilot Study, and 6. Survey invitation.

5. Pilot Study. For this study, two questionnaires were developed in two different languages: Portuguese and English. Then, we proceeded to conduct a pilot study with a small sample of the target population. The initial questionnaire was composed of questions accompanied by static and executable

Table 3. Study questions with open-ended questions (OEQ) and closed-ended questions (CEQ).

ID	Study Questions 1 (SQ1)
CEQ6	When a CAPTCHA check fails, will the user be able to post his/her comments?
CEQ7	How confident are you about your answer?
CEQ8	Which service will publish the comments?
CEQ9	How confident are you about your answer?
CEQ10	Do you feel any difficulty to understand the model?
CEQ11	Can you mentally execute/debug the model?
OEQ5	Have you felt any difficult? Please tell us.
Study Questions 2 (SQ2)	
CEQ12	What happens when saveOrUpdate() rises an exception?
CEQ13	How confident are you about your answer?
CEQ14	Is close() method called when any exception occurs?
CEQ15	How confident are you about your answer?
CEQ16	Do you feel any difficulty to understand the model?
CEQ17	Can you mentally execute/debug the model?
OEQ6	Have you felt any difficult? Please tell us.

Table 4. Attitudinal open-ended questions.

ID	Attitudinal Questions (AQ)
OEQ7	In your opinion, what were the strengths of using executable models to represent the cases?
OEQ8	In your opinion, what were the weaknesses of using executable models to represent the cases?
OEQ9	What could be added to the visual perception of the executable models to improve and/or enrich useful information for a software engineer?
OEQ10	What application opportunities do you see for using executable models in your software engineering projects?
OEQ11	Did you feel any difficulty or problem in this survey?

models. Six participants answered the initial questionnaire. We were able to identify a bias related to the use of static models. Therefore, we decided to remove the static models in the questions. A second version of the questionnaire was developed with no reference to static models. We submitted it to seven participants, which enabled us to identify some minor typographic problems. After that, the questionnaire was ready to be applied.

6. Survey Invitation. The questionnaire was sent to professionals that work with software engineering in the industry and to software engineering researchers. With the purpose of obtaining participants from different regions and nationalities, the invitation channels were a lists of e-mails from some universities in Brazil and professional chat lists in platforms such as WhatsApp¹¹ and Telegram¹² groups. We also used online IT communities as invitation channels such as Hacker

¹¹<https://www.whatsapp.com/>

¹²<http://telegram.org/>

news¹³, WhatsApp¹⁴, IT communities, Telegram¹⁵ computer and programming related groups, and LinkedIn¹⁶. Moreover, the questionnaire was sent to 292 software engineering professionals. The survey was conducted between November 14th, 2019 and April 15th, 2020. We obtained 58 answers. The number of answers in a survey it is an important factor for the validity of its results. We found other survey researches in the literature published in relevant venues (such as SBES¹⁷ and ICSE¹⁸) with a similar number of participants (Ferreira et al., 2018; Sedano et al., 2019; da Costa Carvalho et al., 2020). After the execution, we generated individual files with the content of each answer. The documents were mischaracterized to preserve the participants identities.

3.3 Analysis Procedure

The data obtained from the questionnaire were analyzed quantitatively and qualitatively. In the quantitative analysis, descriptive statistics were used to represent and describe the data to characterize the participants and to cross such information to obtain conclusions. In turn, Grounded Theory (GT) was used as the qualitative analysis procedure (Corbin, 2015). GT aims to create a theory from the collected data and it is composed of three phases: (1) open coding, (2) axial coding and (3) selective coding. In the *open coding*, break, analysis, comparison, conceptualization and categorization of the data are performed (?). A meticulous reading of the collected data is performed and each text fragment receives an expression, sentence or word forming a code or category in the first steps of this phase. In the *axial coding*, relationships among categories and subcategories are established to form dense, related categories. Finally, the *selective coding* originates the category or central idea of the study. Two researchers extracted the codes and established the relationships among them while two other researchers validated the coding process.

The coding process is finished when no new data adds new knowledge to the categorization process. Despite the GT purpose for creating theories, Cobin ? explain that a researcher can use only a few steps to reach their research goals, e.g. when researchers only need to understand a specific phenomenon or situation. In our study, we applied only phases 1 and 2 of coding to analyze the data aiming to identify the emergent topics, strengths, weaknesses, difficulties, and improvements. This type of analysis involves creativity, experience and bias from the researcher. Due to those factors, the conduction was performed together with a professional with 10 years of experience in ExM while the verification of the applied methodology was conducted by another researcher with 10 years of experience in qualitative studies.

3.4 Results

3.4.1 Quantitative Analysis

The first part of the survey comprises demographic questions. Results are communicated based on the whole set of participants. A remarkable portion (20 out of 58 participants) holds a PhD degree (corresponding to 34.48%). However, other categories were also represented, such as undergraduate students with 12 participants (20.69%), Master's students (seven, 12.06%) and PhD students with 7 participants each (12.07%). Professionals with bachelor degree's and lastly Master's degree are 5 participants (8.62%), each. Another aspect identified was the professional practice area. We verified that 27 participants (51.72%) work only in academia while 21 participants (43.10%) work only in the industry (i.e. non-academic). Seven participants were only students (i.e. not working in the industry or academia) and three participants are working in both industry and academia. These results show that the provided insights came from the academia and industry and from several knowledge degrees. We collected answers of participants studying or working from 38 different educational institutions and 12 different companies. Participants that identified themselves as female were 11 (18.97%) and 47 (81.03%) identified themselves as male.

Next, we aimed to identify the participants experience with static models and ExM. We asked participants to classify their experience in a scale of “I have never used them before” to “I use them in every project”. Figure 3 shows that 30 participants (51.72%) reported that they have never used ExM before, although 28 participants (48.28%) reported a certain level of experience, which reveals a balanced result. About static models, we observed that only four participants (6.90%) have never used them before and 93.10% of the participants have used them to some degree. Regarding the nationality, we obtained 14 answers from abroad (24.14%) and 44 answers from Brazilians (75.86%). The average age was 32 years, which means that most participants were young. The mean experience in academia was 5 to 10 years whereas in the industry was 1 to 5 years. Therefore, the participants had more experience in the academic environment than in the industry. We had several participants initiating their careers in industry, with 2 to 4 years of experience.

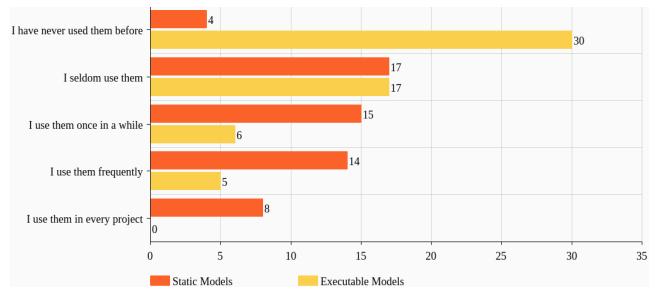


Figure 3. Adoption of static models vs executable models.

Despite the fact that the study was not focused on analyzing the responses of the SQ but rather offering the opportunity of using ExM as previously reported, many participants informed that it was their first experience with ExM. However, we also analyzed the score achieved by the participants, i.e. the percentage of correct answers when s/he was

¹³<https://news.ycombinator.com>

¹⁴<https://www.whatsapp.com>

¹⁵<http://telegram.org>

¹⁶linkedin.com

¹⁷<https://dblp.org/db/conf/sbes/index.html>

¹⁸<https://dblp.org/db/conf/icse/index.html>

requested to solve a problem using an ExM as a source (RQ1). Figure 4 shows the results obtained by the participants. We offered two videos with two different scenarios, each one with two related questions. The results we obtained are inconclusive. Participants scored better results in the second scenario, first question.

On the other hand, the results were similar for the other questions. We cannot claim for sure the reason behind this result, but a possible explanation might be the lack of experience with ExM. We also analyzed the results based on three demographic segments: (i) education level, (ii) previous experience with ExM, and (iii) origin (academia or industry). We counted the number of individuals that achieved the highest score (four questions) in the survey and identified within each demographic segment, the group that achieved the best results. In the education level segment the groups were: (PhD, PhD Student and Master's (High education level) versus Master's student, Undergraduate and Undergraduate student (initial education level). In the previous experience with ExM, the groups were: Participants that had any experience with ExM versus Participants with no experience. In the occupation segment, the groups were: academia participants versus industry participants (excluding students and participants that work in both industry and academia). Concerning to education levels, we observe that, proportionally, the group with participants with higher education levels (PhD, PhD Student and Master's) achieved higher scores (10/32 — 31.25%, i.e., ten out of the 32 participants considered scored the four questions) than participants with lower education levels (6/26 — 23.07%). When we look into experience with ExM, we observe that, again proportionally, the group with experience scored slightly better (8/28 — 28.57%) than the group with no experience (8/30 — 26.66%). When we look into occupation, the group with participants from industry (excluding the three participants from both) scored higher (8/21 — 38.09%) than participants from academia (6/27 — 22.22%). Then, in summary, among the academic participants, we had 18 PhD, 6 PhD students, 1 Master's degree and 2 master student; among the industry participants, we had 1 PhD, 4 Master's degree, 5 Master student, 7 Undergraduate, 10 Undergraduate students. We also had 2 PhD that acted in both the industry and academia and so we did not count in any group.

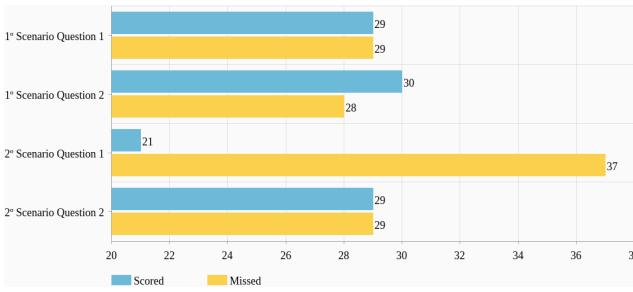


Figure 4. Score per scenarios and questions.

3.4.2 Qualitative Analysis.

The qualitative analysis was performed using questions OEQ5 to OEQ11. To do so, we applied open coding and axial coding, as mentioned in Subsection 3.3. In the open cod-

ing, the data was analyzed thoroughly to create the codes related to the participants answers. After the open coding phase, we identified the categories and established the relationships among codes, generating interrelationships. The open coding was performed in a collaborative way using the online tool Tagouette¹⁹ whilst the graphical representation for coding was created using Cmap²⁰ (Figure 5).

In your opinion, what were the weaknesses on the use of executable models to represent the cases?

R. They do not provide global overview. Skipping straight to the interesting part of the scenario is not possible. Moving back-and-forth in the chronology of events is a lacking feature. It lacks information about traceability of consequent messages (which is hidden in components). [new highlight]

Figure 5. Exemplifying open coding phase using Tagouette.

In the qualitative analysis, we identified six categories directly related to the proposed research questions: (i) advantages perceived due to the use of ExM, (ii) requirements for new technologies provided by participants during the survey, (iii) perceptions or personal opinions from the study's experience on ExM, (iv) improvements envisioned by participants regarding ExM, (v) difficulties experienced by professionals during the survey, and (vi) disadvantages identified due to the use of ExM.

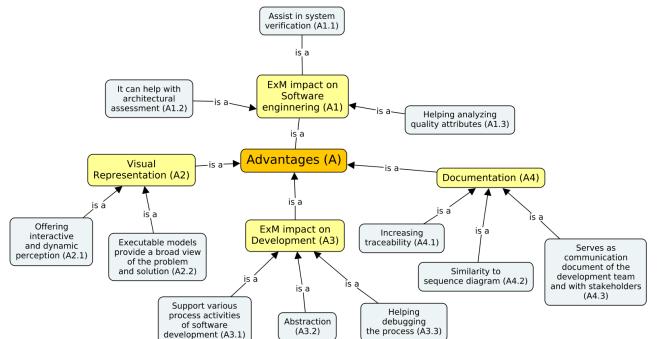


Figure 6. Some Advantages related to ExM.

Figure 6 presents the codes extracted from participants answers that correspond to the identified *Advantages (A)* category (RQ2). Several codes express advantages on ExM. After extracting the codes, we could split this category into four subcategories, which are advantages obtained from (i) ExM impact on the entire software engineering life cycle (A1), (ii) visual representation (A2), (iii) ExM impact on software development (A3), and (iv) documentation (A4). In regards to the impact on software development, participants reported that ExM offers advantages related to increasing reuse degree in the software development via model reuse. ExM could potentially improve the software understanding with the use of abstraction (A3.2). Moreover, several participants highlighted ExM potential to help on understanding scenarios (A2.1) and to help on debugging activities (A3.3).

Participants also reinforced that ExM cannot only offer benefits for coding but also support different activities during the entire software

¹⁹<https://www.tagouette.org>

²⁰<https://cmap.ihmc.us>

development process (A3.1). Participants glimpsed the adoption of ExM for software architecture evaluation (A1.2), quality attributes analysis (A1.3), and understanding the systems behaviors. They also reported strengths for systems verification and validation (A1.1), besides the potential of ExM to serve as a bridge between the architectural process and design/implementation process. Regarding ExM visual appeal, participants reported benefits related to its dynamic nature. Some participants reported on its potential to provide a broad view of the problem (A2.2).

Some participants reported that ExM are similar to sequence diagrams (A4.2) and then it can be easy to understand, reducing cognitive stress of learning new technology. Participants also pointed out ExM advantages to integrating software documentation as it can increase traceability, help manage organizational knowledge and work as a communication document among developers and other stakeholders (A4.3). The following excerpts express the potential of ExM for documentation and its potential for improving the understanding of the problem being solved:

(A2.1)

"The understanding of the model becomes clearer and more didactic." [Participant 43]

(A4.3)

"This can be useful in a team meeting to show what the software should do, so that the whole team can better understand what needs to be done." [Participant 37]

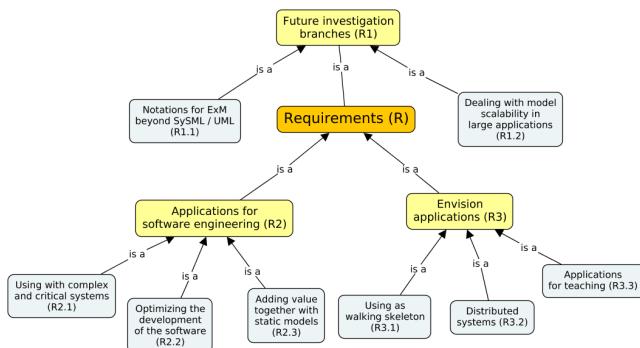


Figure 7. Some requirements related to ExM.

In Figure 7, the Requirements (R) (i.e. opportunities envisioned by professionals) category (RQ3), participants reported what was possible to envision as applications for ExM, according to their own opinions. We split this category into three subcategories: (i) future investigation branches (R1), (ii) applications for software engineering (R2), and (iii) envisioned applications (R3). In applications for software engineering, participants pointed out possibilities of use, such as using ExM to optimize software development (R2.2) by increasing the reuse of models and bringing value together with static models (R2.3). They also reported the use in complex and critical systems (R2.1).

In future investigation branches, participants reported that new technologies must deal with the problems of scalability in large applications (R1.2) and exploring other notations for ExM (not only SysML and UML) (R1.1). For envisioned applications, participants highlighted opportunities to use ExM on teaching (R3.3) and distributed systems development (R3.2). Some participants envisioned its usage with component design and for building a walking skeleton (R3.1), i.e. a prototype implementing the bare-minimum functionalities required by a company to validate the idea in the market. The following fragments exemplify this category:

(R2.1)

"You see... as we start to consider the complexity of real world software, the models should become very complex but it should also retain easiness to understand all their scenarios." [Participant 37]

(R2.2)

"With executable models we can optimize the entire software process, obtaining satisfactory results." [Participant 44]

(R3.1)

"Design of new components, mainly using Domain-Driven Development and walking skeleton." [Participant 8]

(R3.3)

"Executable models could be used for teaching." [Participant 15]

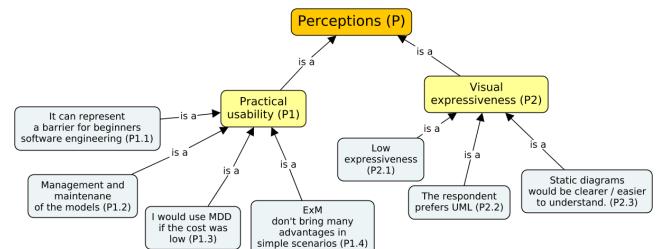


Figure 8. Some Perceptions related to ExM.

In the *Perceptions (P)* category, we separated the codes into two subcategories: (i) practical usability (P1), and (ii) visual expressiveness (P2). Figure 8 presents the perceptions mapped onto codes. Regarding *visual expressiveness*, participants informed that ExM have low expressiveness (P2.1) and visualizing the execution is limited functionality. Other participants reported that static diagrams are easier (P2.3) and that they prefer UML (P2.2). The following citation exemplifies this point:

(P2.3)

"In these cases, a good modeling of states representing the life cycle of the manipulated entities would solve the problem." [Participant 36]

Considering practical usability, participants expressed their concerns with: the models maintainability (P1.2), its applicability in the real world in an agile methodology scenario, and whether it could represent a barrier to new software engineers (P1.1). Some participants were partially susceptible to ExM, as they reported that ExM and model-driven approaches could be adopted if they could offer lower development time, cost and effort (P1.3). In turn, some participants pointed out that ExM do not bring advantage in small scenarios (P1.4) and therefore it can only reveal its value in more complex scenarios. The following fragments express those perceptions:

(P1.3)

"I would use a model-driven approach only if it could allow delivering value to the customer with the least amount of cost, time and effort possible." [Participant 30]

(P1.4)

"The scenarios given as examples are rather simple. I am not sure executable models bring that much in this case." [Participant 54]

On the *Improvement suggestions* (S) category (RQ4), we also divided it into three subcategories: (i) visual representation (S1), (ii) disposition (of elements) (S2), and (iii) notation (S3). Figure 9 presents the suggested improvements for ExM.

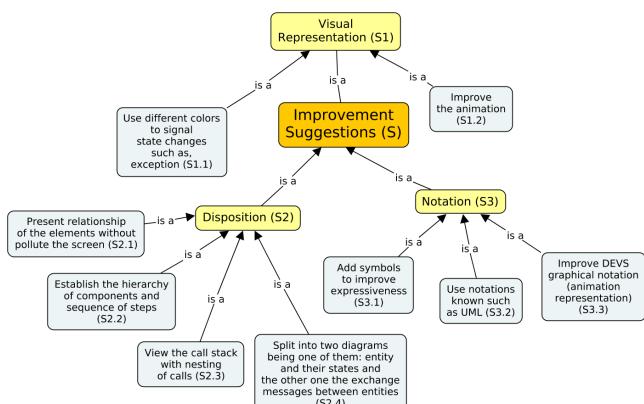


Figure 9. Some Improvement suggestions related to ExM.

Some participants suggested that adding a stack call (S2.3) and improving components hierarchy and events sequence (S2.2) were needed. For conducting the study, despite the tool support for representing links between elements, we suppressed them to reduce visual pollution while we were recording the video. However, some participants missed the display of relationships among elements without polluting the screen (S2.1). Some generic suggestions for improving the ExM notation were: use known notations such as UML (S3.2), and add symbols to improve expressiveness. Participants also suggested improvements on the visual representation by improving the animation (S1.2) and adding color to signal the states change (S1.1). The following fragments endorse it:

(S1.1)

"Show in red when the first error was triggered and propagate it back as red." [Participant 19]

(S1.2)

"The animation is as bad as it could be, I suppose. So, definitely, the animation should be improved." [Participant 18]

(S3.2)

"Why not using known notation such as sequence diagram (UML)?" [Participant 8]

(S2.2)

"It should be included hierarchy and alignment of models." [Participant 1]

(S2.1)

"...however, (the relationships) should be displayed in a way that do not pollute the screen." [Participant 9]

(S2.2)

"Lines for communication paths." [Participant 48]

Still, regarding the *Improvement suggestions* category (RQ4), participants reported on the disposition of elements, their visual representations and notation. Disposition plays an important role to visual perceptions. In a static diagram, the information is usually displayed following some order. In ExM, the user can freely position the elements. However, this has caused a side effect: reducing visibility. Some participants suggested the creation of a hierarchy to present the elements on the screen (S2.2).

Another important aspect reported was the lack of visualization cues, such as colors to represent state changes (S1.1). Because users depend on the visual aspects to follow the execution process, colors could serve as a visual guide to different elements and states on the screen. Participants also highlighted that the visual notation should be improved (S3.3). A possible solution is to use well-established notation (e.g. UML) (S3.2) as a reference to improve ExM visualization techniques. It can be important to increase ExM acceptance in the software engineering practice.

In the *Difficulties* (F) category (RQ5), we were able to further split the codes into two subcategories: (i) difficulties due to the respondent the first interaction with ExM (F1), and (ii) difficulties considered inherent with the adoption of ExM (F2), as shown in Figure 10. In first interaction with ExM subcategory, participants reported on problems such as to understand the presented problem, inexperience wth ExM (F1.2) because s/he is not currently working on programming, and the presented model is complex (F1.1). Our approach consisted of presenting the same conditions a respondent is usually exposed to when they use static models. Therefore, participants had access only to a short video. Some participants felt difficulties with the chosen approach and perceived the survey as missing information to comprehend what was presented to them. However, we aimed at reducing as much bias as possible

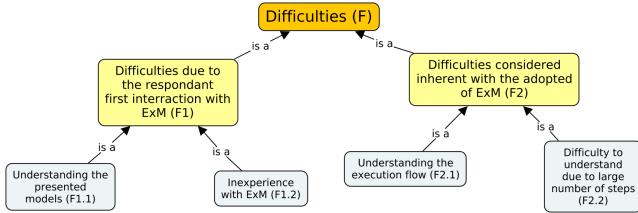


Figure 10. Some Difficulties related to ExM.

related to static models. Therefore, it is reflected in some answers.

The second subcategory refers to the difficulties participants perceived as inherent to the adoption of ExM. Some participants reported difficulties to comprehend the execution flow (F2.1) and the large number of steps to observe (F2.2). The following citation exposes these problems:

(F1.1)

"I could not follow some inputs and outputs." [Participant 58]

(F2.1)

"I had difficulty to infer which function was executed by each component." [Participant 1]

(F2.2)

"In case of referring to the algorithm steps, it is easy. Otherwise, it is hard to understand. I think this is due to the number of steps - it is too many to handle in mind." [Participant 18]

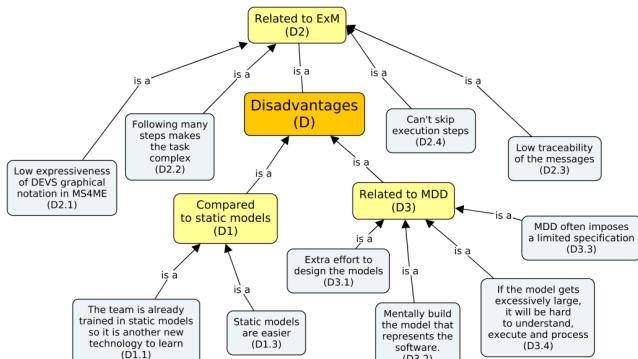


Figure 11. Some Disadvantages related to ExM.

In Figure 11, the *Disadvantage (D)* category (RQ6), we separated the codes into three subcategories: (i) disadvantages compared to static models(D1), (ii) related to ExM (D2), and (iii) related to MDE (D3). Regarding the *comparison to static models* subcategory, participants reported that their team is already trained with static models and therefore it would be one more new technology to learn (D1.1). Participants were also informed that static models are easier to understand (D1.2).

Related to ExM, some participants reported that following many steps is a complex task (D2.2) and someone cannot skip steps (D2.4). They also reported that DEVS expressiveness in the MS4ME tool

is low (D2.1) and thus it offers low traceability of the messages (D2.3). Finally, *related to MDE*, participants informed that it is not easy to create models and someone has an extra effort to conceive the models (D3.1). Participants reported that MDE often imposes a limited specification, since it is not always possible to specify everything (D3.3). They also showed concerns related to the model dimensions as if the model gets excessively large it will be hard to understand (D3.4), execute and process mentally the software (D3.2). The following citations exemplify the findings:

(D1.1)

"Thinking about the way I currently work with my team, I think it would be an additional "problem" to think about."

[Participant 23]

(D1.3)

"A well-made static model can be more easily understood."

[Participant 11]

(D3.1)

"The extra effort required to generate the models. The cost-benefit of such a technique should be assessed."

[Participant 32]

In summary, we were able to observe that participants had a bias towards the use of static models. Due to their experience with such models, they focused on difficulties to adapt themselves to use ExM. Therefore, ExM training and education can play an important role in transitioning from static models to ExM. Table 5 summarizes the findings obtained from answers for each RQ.

3.5 Discussion

The results obtained in this survey allowed us to identify: 4 difficulties associated with ExM, 9 suggestions of improvements, 11 perceived advantages, and 10 perceived disadvantages. Participants were also able to present their perceptions and envisioned opportunities. We collected 8 envisioned opportunities and 7 perceptions about ExM. As a result, we obtained 49 codes in total. In this study, we presented participants with two small scenarios. For many participants, it was their first experience with ExM. This reinforces a bleak situation currently affecting the industry, the absence of using the good practice of the software development by professionals in the industry (Torchiano et al., 2011; Agner et al., 2013; Whittle et al., 2017). This directly affects the quality of the final software. Participants also pointed out the need for special training in order to start using ExM. Modeling and Simulation (M&S) courses are rare in academic environments, mostly reserved for graduate programs. Hence, only participants with high education levels had any previous experience with M&S.

However, the industry can obtain benefits from those practices. Architectural evaluation can help prevent and reduction of defects in the software, which leads to improved quality and user satisfaction (Bass et al., 2012) and ExM can also

Table 5. RQ summarization.

RQ	Summarization
RQ1	Although participants achieved better results in the second scenario, first question, in general, the obtained results were inconclusive. Thus, we can neither confirm nor deny the degree which ExM helped participants to answer the questionnaire.
RQ2	Strengths of ExM envisioned by participants include: (i) improving reuse in software engineering, (ii) raising the abstraction level, (iii) serving as a communication document to collaborate with all involved stakeholders; and (iv) bringing benefits to all software engineering activities ranging from verification & validation, architecture, design, and implementation.
RQ3	Participants raised as opportunities for ExM: (i) optimizing the software development, (ii) increasing value by associating it with static models, and (iii) being used in distributed system development. Research opportunities include the need for professional training with ExM.
RQ4	Participants suggested to enrich ExM visual perceptions adding stack call to improve the hierarchy of components and the display of the sequence of events by highlighting different event with different colors.
RQ5	Participants reported they face difficulties due to its first interactions with ExM. Other participants felt difficulties to understand the presented problem and model. Participants also felt lost while tracking the execution flow.
RQ6	Participants identified as weaknesses for ExM the need for training their team for the use of ExM as they have already used to static models and the difficulty to have a broad view covering the entire system using only ExM. They also reported the low expressiveness and traceability in MS4ME visual notation and the extra effort to conceive the models.

be used to perform Verification and Validation (V&V), as shown in some studies (Bogado et al., 2014; Hojaji et al., 2019). Nevertheless, participants envisioned opportunities of applications for ExM in software engineering. They realized the aggregated value that ExM brings together with static models such as UML to software development. They also reported the application of ExM in distributed systems where interactions among systems are hard to track and predict in design time. Moreover, they made several suggestions to improve expressiveness in order to assist software engineering professionals in understanding ExM, such as improving components hierarchy and event sequence. This particularly represented a difficulty for participants that reported problems following the execution flow.

Regarding the obtained score as presented in Section 3.4.1, we can conclude that there was some difference among different groups. Some possible explanations for such a result is due to the fact that those groups were not disjoint. Some participants worked both in the industry and academia as well

as some participants from the industry have higher education levels. We can also argue that participants from the industry have some ability to interpret the results of the ExM due to their daily practice and the problem being solved, which is similar to those they often face. Nevertheless, we believe that previous knowledge with ExM can bring some benefits to the correct interpretation of the models.

In respect to the adopted survey approach used in this study, we argue that the specification of DEVS models (the ExM formalism we adopted) is normally written in textual format. Thus, it is really hard for someone to understand the flow of information exchanged between the entities of systems observing only the textual part of the specification. We argue that the visual representations of systems parts have the potential to improve visibility. Consequently, this was fully provided for participants through the simulation videos. In addition, common functionalities associated with simulation tools such as pausing the simulation or returning to a previous state became possible via the video controllers. The experience provided by the videos were, to some extent, similar to the one they would have if they had access to the simulation tool. The only exception is that they would have access to the source code which probably would not be very intuitive to participants. Therefore, we argue that the adopted approach probably did not impact the experience of participants.

Finally, this study contributes to paving important avenues of research and advances that are still necessary to make the ExM adoption by software engineers easier. Moreover, there is still a concern regarding the cost of programming ExM and/or model transformation. Therefore, we can conclude that ExM technology has many avenues of improvements yet to be explored in order to be fully prepared for software engineering professionals. At the same time, we observed that many professionals had never experienced ExM before. Therefore we can conclude they are not prepared to use them. It opens opportunities for further strategies for ExM notation evolution, education and training in the field.

3.6 Related Work

Particularly, simulation models as illustrative examples of ExM have certainly been adopted in software engineering research (Bogado et al., 2014; Hojaji et al., 2019; de França and Ali, 2020). Other studies on ExM in the software engineering practice have also been conducted. Hlupic (2002) presents a survey with academic and industrial users regarding the use of ExM. The study reports on the adoption of simulation to analyze application areas and user choices about the software product being produced. However, while that study offers a panorama of simulation models, our study presents perceptions of professionals extracted from an experience with a simulation model used to solve problems. Such opinions are analyzed to highlight opportunities and improvement suggestions to cope with software engineering professionals' needs.

Torchiano et al. (2011) and Agner et al. (2013) also conducted surveys on models adoption. These studies report results on the use of MDE in the Brazilian and Italian industries, respectively. Agner et al. (2013) investigated the use of UML models and MDE in the embedded software indus-

try. They investigate the level of usage of UML models and MDE as well as maturity levels within companies (Agner et al., 2013). On the other hand, Torchiano et al. (2011) report on the results of a survey with Italian software engineering professionals. The authors looked into the adoption of MDE in Italian companies, differences according to the companies sizes, used notations (e.g. UML or DSL), and the profile of those who are developing models (Torchiano et al., 2011). Those findings show differences between the two specific countries. Both studies are geographically focused on particular countries, whilst ours comprises Brazilian and other countries through the participation of international participants, besides particularly offering a panorama on the use of ExM. In the next section, we present the methodology for conducting our study as well as the conduction itself and the obtained results.

Guessi et al. (2015) present an analysis of architectural description languages (ADL) for modeling system-of-systems (SoS). They found that none of the most used ADL for modeling SoS does not fully support it. They also found that basic tool needs are still precarious. This result is related to some of our findings regarding visual representation, expressiveness power, and tool support. Some participants reported difficulties related to the limitations in current visual representations' expressiveness power. They also reported problems to understand the visual representation and limitations in the ExM tool in regards to the traceability of messages. Bork et al. (2018) also found similar problems regarding those two topics. In their study, they present a critical analysis and evaluation of ten existing modeling standards such as UML and BPMN. Bork et al. (2018) analyzed the visual expressiveness of the investigated languages and concluded that existing notations are still not very mature. Those notations still rely on simple box-and-line diagrams. They also report problems related to colorization of visual elements. In our study, participants of the survey also reported problems related to colorization and the difficulties associated with interpreting those results when the simulation is being executed.

Mincarone et al. (2018) present a systematic review and a survey with one author of each study in their systematic review. Each author evaluated the benefits of the graphical modeling languages they explored in their study for the patient care process. The authors reported that the most used notations in the studies (UML and BPMN) increased the clarification of the presentation of information in medical contexts as they are capable of representing micro and macro scenarios. Participants in our survey also represented similar answers regarding the easiness of the view representation of ExM notations to understand complex scenarios. Mendling et al. (2010) perform a study to investigate the importance of textual labels in process modeling while presenting recommendations to improve this practice. This study also presents a list of recommendations and strategies to be used in modeling as our study. Different from our study that takes a broad view regarding the visual representation and tries to propose recommendations to it, they focus on a specific part of the modeling process. Mussbacher et al. (2015) describe the results of a week-long design thinking experiment with 15 MDE experts, where they were able to discuss the benefits already achieved by MDE approaches and possible hindrances

in the path of their adoption by the industry. Some obstacles identified in their study were also perceived by software engineering professionals that participated in our survey. Obstacles were identified in the participants' answers, such as: tool support, limitations in MDE expressiveness power to address increasing demands on software, and the fact that industry does not consider MDE "cool". These results further confirms what experts in MDE identified.

4 Strategies for Evolving a Simulation ExM Notation

After obtaining the perceptions of professionals about ExM, we also extracted codes to compile a list of strategies as recommendations for improving simulation ExM so that they could be aligned with the professionals expectations. They point to directions (and in some cases restrictions) that should be followed to evolve simulation notations for encompassing software engineering professionals needs.

The strategies were organized in four general categories of improvements suggestions: (i) notation, (ii) visual presentation, (iii) usage, and (iv) user support. We established a standardized presentation for these strategies structured, as follows: (i) strategy identifier, (ii) short title summarizing the strategy, (iii) the traceability associated with the strategy, and (iv) a brief explanation of the strategies. Each strategy identifier follows the format "[Category_Character-Strategy_Number]". The tuple Category_Character represents each of the categories previously mentioned and Strategy_Number is an integer that identifies the number of the strategy in each category. The title synthesizes the core idea behind the strategy. The strategies were created straightforward from the improvements recommendations collected from the answers to the survey form. The traceability of each strategy can be seen in the table presenting the strategies and the associated code extracted from the survey based on GT procedures. Lastly, the brief explanation discusses the importance of the strategy and its consequence for the development of an ExM notation.

4.1 Strategies for ExM Notations

The first category presented is *notation improvements* (Table 6). In this category, the set of strategies recommend generic improvements on the ExM notations. We identified in the survey that participants felt the need to enhance ExM notations expressiveness power and robustness, as explained below.

[N1] Improve notation expressiveness power and robustness

ExM notations should deal with two major challenges and find a balance between: (i) the notation expressiveness power and (ii) its formality level. The former represents how much the notation can describe the real world (i.e. how much it can represent from the real world and how much it abstracts out). The latter prescribes how precise and formal the notation is,

Table 6. [N]otation Improvements.

ID	Description	Survey Code
N1	Improve notation expressiveness power and robustness	(S3.1) Add symbols to improve expressiveness (S3.2) Use notations known such as UML (S3.3) Improve DEVS graphical notation (animation representation)

but it can also hinder its understandability. ExM notation designers should pay close attention to both aspects as they are very important for the end-user. The former can impose limits to its use and it may force the end-user to drop it in favor of another notation more capable of representing the problem being modeled. The latter can create an unnecessary cognitive burden, which is exactly the opposite goal aimed when using an abstract notation.

In terms of complex systems, researchers tend to prefer to use the most suitable domain-specific notation for the problem-space (Wang and Dagli, 2011). As such, an approach such as megamodel (Favre, 2005) (i.e. combine different models and metamodels in a unified model) could be adopted. Megamodels allow software engineering professionals to work with multiple static and executable notations. The notations can be general-purpose and domain-specific notations and they can use model transformation to convert from one notation to the other whenever needed. One possible alternative that ExM notation developers could offer is the possibility for users to define their own language constructors with their operational semantics (e.g. those offered by IBM Rational Rhapsody²¹ or GEMOC²²). Then, users can adapt the language to the specific complex system domain requirements, reducing the need to use another static notation.

4.2 Strategies for ExM Visual Presentation

Table 7 contains improvement strategies for the visual presentation of the ExM notations. The improvements are focused around the problem of dealing with model traceability, model scalability and model visual presentation. The survey participants offered several suggestions to concretely enhance the visual notation.

[V1] Improve execution flow

From the survey results, we concluded that the execution flow could become hard to follow, mainly when there are many state transitions and messages being triggered simultaneously. In fact, even in small-scale scenarios as those presented to participants during the survey, we observed the participants faced difficulties in following the model execution. Therefore, new strategies must be investigated to identify better forms of presenting execution flows.

Thus, we can say that [V2] and [V3] (described below) are important strategies to be achieved first. They will

Table 7. [V]isual Presentation Improvements.

ID	Description	Survey Code
V1	Improve execution flow	(F2.1) Difficulties to understanding the execution flow (D2.2) Following many steps makes the task complex
V2	Improve visualization for a large number of elements and actions	(F2.2) Difficulty to understand due to large number of steps
V3	Use different colors to represent state changes in simulation elements	(S1.1) Use different colors to signal state changes, such as exception.
V4	Improve distribution of elements to reduce visual pollution	(D3.2) Mentally build the model that represents the software (D3.4) If the model gets excessively large, it can become very complex to understand
V5	Establish hierarchy among components	(S2.2) Establish the hierarchy of components and sequence of steps
V6	Add call stack or state transition stack	(S2.3) View the call stack with nesting of calls
V7	Offer the possibility to split the diagram into entities and messages exchange views	(S2.4) Split into two diagrams being one of them: entity and their states, and the other one the exchange messages between entities
V8	Provide an intuitive animation that illustrates data exchange and other software elements to enhance the user experience	(F2.2) Difficulty to understand due to large number of steps (S1.2) Improve the animation
V9	Display the lines of message exchange between entities without polluting the screen	(S2.1) Present relationship of the elements without polluting the screen
V10	Provide mechanisms to visualize the models such as: zoom-in, zoom-out and moving around the entire screen fluidly	(D1.3) Static models are easier
V11	Use known visual representations such as UML or SysML	(P2.2) The respondent prefers UML
V12	Add the possibility for users to provide his/her own symbols to better adapt the visual notation to the domain problem	(S3.1) Add symbols to improve expressiveness

²¹<https://www.ibm.com/products/systems-design-rhapsody>

²²<http://gemoc.org/>

increase visibility and improve the execution flow. Another possible solution to this strategy is to use conditional breakpoints (see Figure 12). Some Integrated Development Environment (IDE) offers the possibility to set up conditional breakpoints that will halt the execution when a certain condition is satisfied, which allows developers to run the software normally and only debug the exact segments of the model or their states they desire to investigate. By offering such functionality, users can investigate only the segments of execution flow they require to better understand.

```

16 if (ch != character)
Expression | Break when expression evaluates to true. 'Enter' to accept, 'esc' to cancel.
17 {
18     buffer += ch;

```

Figure 12. Conditional breakpoint offered by Visual Studio Code.

[V2] Improve visualization for a large number of elements and actions

Related to [V1], this strategy concerns the scalability of the models visualization. Real world software is made of thousands or more components/classes. If all those components/classes are converted to models and run as models instances using ExM, the model visibility can be dramatically reduced. We identify problems such as: (i) disposition of the models, (ii) visualization of the lines of message exchange among models, and (iii) decrease in animation quality due to many elements drawn on the screen.

ExM notation designers should find new ways to deal with this problem to prevent/hamper ExM adoption. Zoom-in and Zoom-out (see [V10]) or grouping elements functionalities are among participants suggestions that could help to improve visibility. The game industry faces similar problems due to a large number of elements to draw on the screen Xavier et al. (2020). Professionals use different techniques and design patterns that help deal within a large number of elements such as double buffer, spatial partition and flyweight (Nystrom, 2014). They also limit the number of elements drawn at once.

[V3] Use different colors to represent state changes in simulation elements

A feasible alternative to improve visibility is by making use of colors to distinguish elements and state transitions. Human vision evolved to perceive different colors (Bowmaker, 1998). The average number of colors the human eyes can distinguish is about 10 million different colors (Judd, 1975). This technique is widely employed in visual arts such as photography, cinema, and arts. User experience and user interface also heavily explore the use of colors (Tidwell, 2020).

Some suggestions include: ExM could employ strong colors (e.g. red or yellow) to indicate problems in the model execution, or the model has transited to an error state. Important elements could also be distinguished by colors, which can help people to focus and identify them more easily. Users could choose colors to highlight important elements or even group elements by the same color to perceive them as a unity (Figure 13).

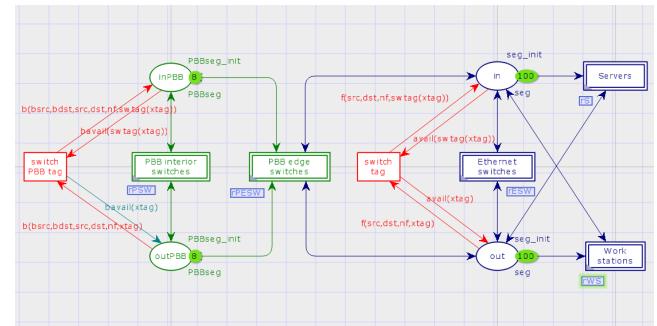


Figure 13. CPNTools is an ExM simulation tool based on Colored Petri Nets formalism that allow users to use different colors for each model, their relationships and messages exchanged (extracted from (Zaitsev et al., 2018)).

[V4] Improve distribution of elements to reduce visual pollution

As the number of models increases and become larger, their visibility decreases (Figure 14). Distributing the elements in the screen can ease visibility and improve perception. Proper size disposition and positioning of elements in the screen can improve models visibility and then the understanding of the problem being analyzed.

The disposition of elements is a topic that is particularly important within Graph Theory (Purchase et al., 1996). This problem constitutes a graph drawing problem in which the goal is to minimize the number of crossing lines to improve visibility. Nevertheless, even the best disposition algorithm from Graph Theory can fail to capture aesthetic principles and therefore, it should always be possible for humans to intervene or guide the process. We argue that ExM solutions could offer a hybrid approach to this problem using both automatic and manual alternatives to provide a viable solution for users.

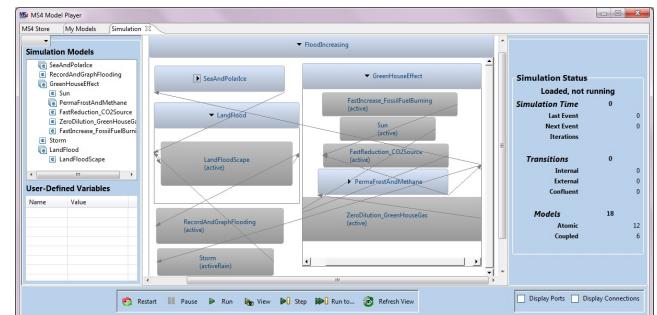


Figure 14. Disposition of elements can become difficult to understand as the number increases (extracted from MS4ME official website²³).

[V5] Establish hierarchy among components

Following [V4], another related aspect to be improved is hierarchy among components. Hierarchy and organization are common properties of systems (Bertalanffy, 2015). As such, ExM solutions should provide, natively in their notation, alternatives to mimic this behavior. DEVS, for instance, offers the possibility to group elements in a single entity named coupled models. However, in the survey, we did not

use this functionality; thus, some participants suggested the option to group elements together.

Therefore, we claim that ExM notations should offer the possibility to organize elements hierarchically and stacking them up to form hierarchical structures. Many complex systems in the real world contain hierarchical structures, so they require this capability from ExM in order to be properly captured by a model. This is also an opportunity to reduce the number of elements presented on the screen, which can improve the scalability of the models and improve their visibility.

[V6] Add call stack or state transition stack

Participants pointed out they had difficulties to keeping the state transition sequence in their memories and this hindered their reasoning. In the survey, they were exposed to videos and were asked to answer questions regarding their observations. The large number of steps to follow and to reason did not allow them to remember everything in order to answer the questions properly and they had to watch the videos many times to answer the questions. Therefore, a call stack or state transition stack could help to solve this problem.

Providing a call stack or state transition stack similar to those present in existing third-generation programming language (such as C++) tools (see Figure 15) can help users to explore the sequence of state transitions and support their reasoning during development. The stack would store the sequence of messages exchanged between components. Together with other functionalities such as conditional breakpoints already mentioned, it can reduce the need for users to remember everything and concentrate only in the required parts.

```

2  using namespace std;
3
4  int func4() {
5      return 1;
6
7  }
8
9  int func3() {
10     return func4() + 2;
11 }
12
13 int func2() {
14     return func3() + 3;
15 }
16
17 int func1() {
18     return func2() + 4;
19 }
20
21 int main(int argc, char* argv[])
22 {
23     cout << func1() << endl;
24     return 0;
25 }
```

Figure 15. Stacktrace of a C++ code as depicted in Visual Studio Code.

[V7] Offer the possibility to split the diagram into entities and messages exchange views

Some participants offer the insight of separating the entities and their states from the messages being exchanged. Usually, the models, their relationship and the message exchanged between models are presented in the same view. Thus, participants have suggested providing a view with two separate screens, in one of the screens, the user would be able to see the models and their state transitions. On the other screen, it would list a sequence of message being exchanged

between models. As such, users would be able to follow in the same screen the messages being exchanged and see the models transiting between states.

This functionality could come as an alternative view from the traditional view (i.e. elements with their messages being exchanged). This specialized view, which follows a “separation of concerns” approach can help reduce the need to draw many state transitions. This can also be used in situations where offering fully animated ExM is not possible or feasible but without decreasing the understandability of the execution of the models.

[V8] Provide an intuitive animation that illustrates data exchange and other software elements to enhance the user experience

Animation is specially important for notations that offer visual capabilities such as ExM. In the survey, some participants complained that the animations were not good and sometimes even confusing. We argue that this problem may also be linked to strategies [V3], [V4] and [V5]. Selic (2008) has already highlighted this problem arguing that researchers and MDE developers tend to not pay enough attention to their tool usability, which can hamper the adoption of MDE technologies.

Therefore, we argue that proper attention should be paid to the animation of the models. This way, graph design professionals can contribute to improving animation. User interface and user experience, which are important research areas in graph design, has already a great body of knowledge in developing user interfaces that take into consideration the expectations of the users and tries to improve their experience while using the system. Thus, they could help to shape the evolution of ExM. Another research area that can contribute to ExM is games. For instance, we could use gamification to improve the experience or 3D animations to improve models simulation (when use is adequate).

[V9] Display the lines of message exchange between entities without polluting the screen

In this particular survey research, we did not display the lines of message exchanged between entities to reduce the visual pollution generated by the ExM tool we used. However, this produced an unexpected side-effect and some participants pointed out that they missed this functionality. Nevertheless, that further highlights the importance of displaying those lines so that users can understand the relationship between entities.

On the other hand, when the relationships are too dense, the visibility is compromised and it can become really hard to discern anything on the screen (which was the initial reason we removed them in the first place) (Figure 16). Offering the possibility to toggle the relationship of some or all entities can help to solve this problem. Grouping similar entities together allow the possibility of showing their relationships without polluting the screen (which could potentially violate [V4]. Thus, it may demand some trade-off analysis).

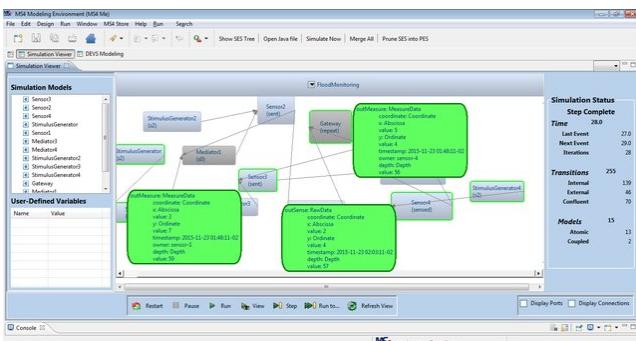


Figure 16. Even displaying the relation of a relative small number of models already hamper the visibility (extracted from (Guessi et al., 2019)).

[V10] Provide mechanisms to visualize the models such as: zoom-in, zoom-out and moving around the entire screen fluidly

Some participants in the survey mentioned that the benefit of drawing a UML diagram in a sheet of paper is that one can see all the models and all the information at once. In their opinion, it is easier to look over the sheet of paper to find the information someone needs rather than to look at the screen of the monitor and searching for the information someone needs. In fact, this is a common and known problem related to usability and immersion on electronic devices (Mangen et al., 2019).

Therefore, we argue that ExM tools should provide a mechanism that tries to be as close as possible to the behavior of seeing a model on a sheet of paper in order to improve immersion in the models. ExM should avoid usability problems in navigating through the models, which is a common issue on electronic devices and MDE solutions (Selic, 2008; Siegenthaler et al., 2010). ExM should provide zoom-in and zoom-out mechanisms that allow users to fit all the models on the screen at once the same way that would happen on a sheet of paper. At the same time, they should also provide mechanics to navigate through the models and move fluidly around the screen.

[V11] Use known visual representations such as UML or SysML

Participants were also concerned with the need to perform new training sessions with their team to learn to use new technology. Therefore, the adoption of well-established notations such as UML and SysML can help in reducing adoption resistance and eliminating the need for further training. Despite the fact these static notations are not suitable to all cases due to their semi-formal nature, these notations are well-known notations among developers. These notations have the potential to reduce the initial learning curve and accelerate ExM adoption by the industry.

However, it is also important to point out the limitations on their usage. In the context of complex systems, the need for formality is also mandatory due to the usual domain application being critical. Thus, we argue that ExM notations should aim to appeal to both concerns: (i) the robustness of formal notations, and (ii) easiness and similarity of known and well-established notations, such as UML or SysML. As previously mentioned in the strategy [S1], approaches

such as megamodels (Favre, 2005), which allows users to combine multiple models and apply model transformation between models, could help to solve this problem.

[V12] Add the possibility for users to provide his/her own symbols to better adapt the visual notation to the domain problem

This strategy complements [V3] and further improves ExM visibility. Participants mentioned their interest in providing their own symbols to be used in the notations. Then, offering the users the possibility to add his/her own symbols could increase visual perception and improve identification response-time. At the same time, it allows users to adapt the notation to better accommodate the explored domain-problem.

The human vision also evolved to perceive shapes (Singh and Hoffman, 2013). It is easier to identify a model representing a radar or sensor using the symbol or picture of a radar or sensor than a simple text or a graphical square box named “radar” and “sensor” on top of it (Figure 17). The symbols also represent an opportunity to group similar elements together and reduce visual pollution ([V9]).

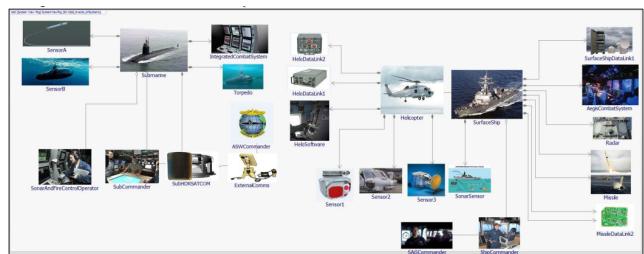


Figure 17. Using user-defined symbols has the potential to improve visibility (extracted from (Dahmann et al., 2017b)).

4.3 Strategies for ExM Usage

Table 8 represents strategies with improvements for usage.

[U1] Improve models maintenance and management

Survey participants were very interested in models maintenance and management. They were concerned with the increase in size of the models as the development progresses and how to manage such growth. We believe this is a valid concern. As the models become the main artifact in the software development, special attention must be devoted to maintenance and management of the models. Traditional programming languages already deal with this type of problem. They make use of versioning systems, most notably Git²⁴. There is also a well-established body of knowledge in software design practices and a collection of design patterns to improve management and maintenance. Because MDE usage is not widespread yet, there is no robust set of known general reusable solutions to this problem already validated and adopted by the industry (except for a few industry sectors,

²⁴<https://git-scm.com/>

Table 8. [U]sage Improvements.

ID	Description	Survey Code
U1	Improve maintenance and management of the models	(P1.3) I would use MDE if the cost was low
U2	Add the possibility for users to generate static diagrams such as UML or SysML corresponding to a “snapshot” of the executable model state in a given moment	(P2.3) Static diagrams would be clearer/easier to understand
U3	Increase level of automation in the process of creating and dealing with the models	(P1.3) I would use MDE if the cost was low
U4	Provide a mechanism in the notation to allow model reuse, such as inheritance or the possibility to build and use a library of models	(P1.3) I would use MDE if the cost was low
U5	Improve model scalability	(R1.2) Dealing with model scalability in large applications

such as automobile²⁵).

This strategy is also related to [U4]. Improving reuse can help to reduce repetitive work, which increases maintainability and improve management. When we consider complex systems, this can become a critical problem. A complex system can be composed of many models, state transitions and relationships among models. In such scenario, maintenance becomes complex and error-prone while management becomes cumbersome. It is important to investigate new solutions so we ended up not transferring complexity from traditional programming to MDE.

[U2] Add the possibility for users to generate static diagrams such as UML or SysML corresponding to a “shot” of the executable model state in a given moment

Static models are unchangeable and plain representations of the systems being modeled. They can be easier to reason compared to a constantly changing entities on the screen. Moreover, static models can be drawn or printed, which can be useful in an architecture meeting or discussion session (Siegenthaler et al., 2010; Mangen et al., 2019). This is the same difference between an e-book versus a physical book. Some people prefer one, others prefer the other.

Therefore, we argue that ExM tools could offer the possibility to generate static diagrams so they could be printed and used during discussion sessions. The digital format, which represents the main artifact, would still be the most important element in the software development, but it could also be offered a static and printable version, which

can be carried and used during meetings. We could also offer the possibility to convert the models from a different ExM notation to UML or SysML only on demand, i.e. when the user was to print the models (Figure 18).

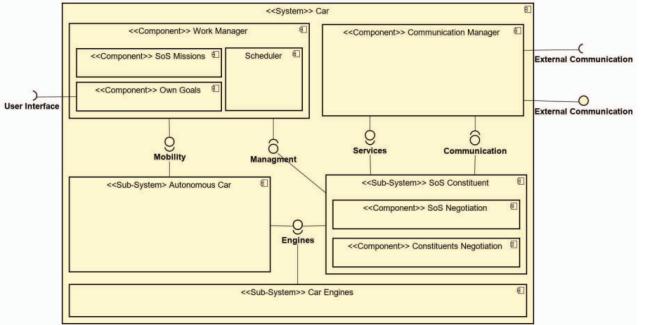


Figure 18. UML diagram are well-established static notation. They could be generated only when to be printed (extracted from (Teixeira et al., 2020)).

[U3] Increase level of automation in the process of creating and dealing with the models

We can also improve model management ([U1]) by increasing the automation in the process of creating models. Automation is always necessary as it is a way to speed up processes and reduce management costs. Computers can perform repetitive work better than humans and without errors caused by distractions.

Throughout the life cycle of the models, there are a number of activities that can also be automated. A common practice that is already widely present in traditional programming languages is macros. A macro is a native language mechanism (e.g. C and C++ macros) or implemented by third-party software that offers developers the possibility to automate repetitive work. The same concept can be adopted by ExM notations. Repetitive work such as establish relationship between groups of elements or message exchange can be fully automated by macros.

[U4] Provide a mechanism in the notation to allow model reuse, such as inheritance or the possibility to build and use a library of models

Participants mentioned the importance of models reuse to reduce cost. Although DEVS provides some reuse mechanisms, due to the reduced scale of the presented problems, we did not use them. However, participants pointed out the important of reuse, mainly in the context of agile development. Traditional programming languages offer several mechanism to increase reusability such as inheritance (for object-oriented languages) and prototyping (as in Javascript).

There are some reuse alternatives for MDE, such as model composition and algorithmic models (Keller et al., 2020). Another solution to increase reuse and development speed is for ExM solutions to offer model repository and research mechanisms (Chreyh and Wainer, 2009). These mechanisms are important because they allow fast prototyping and reuse of the models and their experimental frame, i.e. the set of experiments for which the models are valid to be used

²⁵<https://www.autosar.org/>

(Vlahovic and Ceric, 2008).

[U5] Improve model scalability

This strategy is directed connected to [V2]. In [V2], we have to deal with representing a large number of elements visually. On the other hand, in this strategy, the problem is to execute a large number of elements. Model execution engines should be scalable to deal with many state transitions and messages exchange. Since the model execution engines are the powerhouse in ExM development, they should be prepared to handle the enormous demand of real-world applications.

Again, when we analyze this problem from a complex system perspective this problem becomes evident. A complex software, which can be composed of many components/classes, when converted to an MDE approach, can generate thousands of models. We can also see this problem in terms of systems that are composed of many models such as a military system, in which a simulation of the system would require thousands of models ranging from radar, sensor, aircraft and so on. This type of simulation involves complex calculations that demand powerful computers to simulate them. In such scenarios, scalability is mandatory. One solution to this problem is using parallelism with multi-thread and multi-processor to speed up the calculations.

4.4 Strategies for ExM User Support

Table 9 presents the improvements regarding user support.

Table 9. User [S]upport Improvements.

ID	Description	Survey code
S1	Improve material and content quality for teaching	(D1.1) Another new technology to learn (R3.3) Application for teaching

[S1] Improve material and content quality for teaching

Any real ExM solution to work in the industry will be required to have a proper amount of high-quality material to support its usage. Videos, tutorials and books are the most common ones. Github allows the possibility to offer user support via issues and other means. By offering good content, it can ease its adoption and it can later be used for teaching applications as well.

Another alternative to this problem is to create a community around the ExM notation and increase a sense of sharing and contribution among participants. This can help and motivate newcomers to contribute with the evolution of the ExM notation (Steinmacher et al., 2016; Steinmacher et al., 2019). In order to motivate newcomers, it is important to take measures to ease their entrance such as providing a decent amount of initial material, create an environment of receptivity for newcomers and to popularize the ExM notation together with its key applications and benefits

(Steinmacher et al., 2019).

5 Strategies Assessment

The aforementioned strategies were extracted from the perception of software engineering professionals. In order to assess how suitable those strategies are regarding this perception, we decided to confront them from the perspective of software engineering researchers. As such, we also selected a survey as a method to assess how much software engineering researchers agree with the obtained strategies, what about they disagree, and what software engineering professionals missed in regards to software engineering researchers. We developed a Google forms questionnaire with all 19 strategies presented using a five Likert-scale ranging from strongly disagree to strongly agree, as it can be seen in Figure 19.

We sent out the questionnaire to researchers from Brazil and abroad from February 20th, 2021 to March 08th, 2021 and we received 13 answers. 11 participants were Brazilians, one from the United States and one from Australia. Ten participants had PhD degree and three had Master's degree. Ten participants identified themselves as male and three identified themselves as female. Their ages ranged from 31 to 48. The mean experience time in academia was 10 to 15 years and the mean experience in the industry was 5 to 10 years. The mean usage frequency of the participants with the static model was “frequent” while the mean usage frequency with ExM was “not so frequent”.

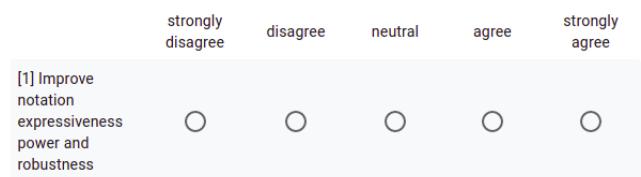


Figure 19. Likert-scale employed in the survey.

In Table 10, we observe that strategies [V4] and [V8] received equally or more “Strongly Agree” votes, as they refer to the importance of displaying elements correctly without polluting the screen and the importance of having a better animation of the entities represented in the simulation and their messages being exchanged. On the other hand, [V1], [V2], [V3], [V5], [V6], [V7], [V9], [V10], [V11], [U1], [U2], [U3], [U4], [U5] and [S1] received more “Agree” votes than “Strongly Agree” votes. Lastly, [V8] and [V12], in which [V12] deals with the possibility of users extend the existing notations to better represent the explored domain, received one and three “Disagree” votes respectively. The strategy [V8] received the worst result among the proposed strategies with two “Strongly Agree”, five “Agree”, three “Neutral”, and three “Disagree”. Finally, the strategy [V12] received the best result even though it received a negative vote with nine “Strongly Agree”, five “Agree”, one “Neutral”, and one “Disagree”.

At the end of the questionnaire, participants could also state their divergent opinions or suggestions for improvements. Only two participants offered suggestions. Participants

Table 10. Agreement level of the strategies among ExM researchers.

Strategy	Strongly Agree	Agree	Neutral agree	Disagree
[N1] Improve notation expressiveness power and robustness	3	8	2	0
[V1] Improve execution flow	4	8	1	0
[V2] Improve visualization for a large number of elements and actions	4	6	3	0
[V3] Use different colors to represent state changes in simulation elements	3	6	4	0
[V4] Improve distribution of elements to reduce visual pollution	6	6	1	0
[V5] Establish hierarchy among components	3	8	2	0
[V6] Add call stack or state transition stack	5	8	0	0
[V7] Offer the possibility to split the diagram into entities and messages exchange views	2	9	2	0
[V8] Provide an intuitive animation that illustrates data exchange and other software elements to enhance the user experience	9	5	1	1
[V9] Display the lines of message exchange between entities without polluting the screen	6	7	0	0
[V10] Provide mechanisms to visualize the models such as: zoom-in, zoom-out and moving around the entire screen fluidly	4	7	2	0
[V11] Use known visual representations such as UML or SysML	4	6	3	0
[V12] Add the possibility for users to provide his/her own symbols to better adapt the visual notation to the domain problem	2	5	3	3
[U1] Improve maintenance and management of the models	5	7	1	0
[U2] Add the possibility for users to generate static diagrams such as UML or SysML corresponding to a "shot" of the executable model state in a given moment	4	6	3	0
[U3] Increase level of automation in the process of creating the models	1	10	2	0
[U4] Provide a mechanism in the notation to allow model reuse such as inheritance or the possibility to build and use a library of models	2	7	4	0
[U5] Improve model scalability	3	8	2	0
[S1] Improve material and content quality for teaching	3	7	3	0

part one mentioned that “*the use of animations remove the users attention from the models, which it is the most important part in an MDE development*”. The participant also disagreed with the possibility of the end-user providing his/her own symbols because it shifts the focus from modeling the domain-problem to drawing the symbols of the domain-problem. We argue that visual representation is important to ExM as it is also important for a static notation to capture the explored domain-problem. Therefore, we observe the need for a visual representation with a proper user interface as re-

quired and as advocated in another study (Selic, 2008).

Participant eight suggested proposing strategies to analyze the behavior of the models (such as their goals, missions, or any other functional requirement) and the models performance. Behavior analysis is specially important in the context of complex systems due to their inherent complexity and the possibility to arise emergent and unpredictable behaviors. We can argue that models simulation already offers the possibility of analyzing the behavior of the system being modeled and their visual representation allow us to draw conclusions

and observe their behavior. Regarding the performance, we partly covered this problem in [U5], which deals with model scalability.

In the end, in this small sample, the obtained score of the strategies was positive among respondents and only received two disagreements. The disagreement is an important contribution to the strategies, which could generate new research opportunities but they do not represent a serious problem to the obtained results.

6 Implications to Theory and Practice

The analysis of the results obtained with the survey with software engineering professionals and later the assessment with software engineering researchers allowed us to observe two different point of view and concerns. On the one hand, we were able to identify the aim for more practicality from engineers in respect to ExM. For ExM to be usable from their perspective, the models have to be more practical (in the sense of being more agile), reusable and scalable. Being a simplified or abstract representation is just one side of software development. There are other equally important requirements that engineers need before they can decide to commit to another technology different from current ones. This has become even more evident after the agile movement. The urgency to deliver fast and high-quality product is a reality that engineers are constantly required by their employers to meet. In that sense, these wishes were captured in the strategies [V11], [U1], [U3], [U4], [U5], and [S1]. Therefore, further research is required on the use of ExM to investigate new avenues and maybe incorporate an agile mentality and practices into ExM practice.

On the other hand, we were able to observe the importance of ExM notation expressiveness power from software engineering researchers perspective. The researchers are particularly interested in modeling the domain problem as accurately and representative as possible. This movement is very important to software development. We can also observe such movement in the industry by practices such as domain-driven design (DDD) that focuses on the use of object-oriented programming to model and match the explored domain. DDD establishes a ubiquitous language between developers and domain experts. We claim both industry and academia have the same goal in regards to modeling the domain as precisely as possible. Thus, engineers can greatly benefit from the achievements of ExM studies. However, this can also cause tension between the two parties as they prefer to approach this problem from two different perspectives.

From the perspective of engineers, they wish to reduce their cognitive stress and reuse as much knowledge as possible. Therefore, the use of known and well-established notation as UML and SysML [V11] that does not require re-training their development team is important. From the researchers perspective, the scenario is the opposite. Researchers are interested in investigating the use of the notations at their disposal to model the studied domain and learn from this experience what went well and what could be improved. However, even from this conflict position, both par-

ties can mutually benefit. Researchers can contribute to improvement into existing notations for specific domains while engineers can provide researchers with improvement feedbacks and new opportunities for research.

Moreover, strategies to capture the visual representation of ExM are highlighted [V1-V12]. Simulation is a significant part of ExM technology. M&S is especially important for complex systems, which require mechanisms to explore possible emergent behaviors (i.e. unknown constructive or destructive behavior). Traditional software development does not usually require the use of simulation and it is only restricted to some domains. However, Industry 4.0, smart cities, IoT and digital twins impel the need for simulation (Rasheed et al., 2020). Therefore, the use of simulation and the importance of its graphical representation become important. Providing an intuitive and good user experience [V8] is necessary as it improves the understandability of the explored problems and reduce cognitive stress. Also, when the number of visual elements become a huddle, new strategies to deal with this problem are needed [V2]. The use of different colors [V3], hierarchy structures [V5], or symbols [V12] can help with reducing the problem.

We conclude by saying that these strategies have the potential to support the evolution of ExM notations, offering development opportunities and fostering the investigation of solutions to solve the problems raised and comply with the suggestions brought by the discussed strategies. By addressing the strategies, we foresee a potential to deal with the problem of ExM adoption and usage from the perspective of software engineering professionals and, at the same time, realizing and incorporating important adaptations to the better suite to the next generation of systems.

7 Limitations

In this section, we discuss the main threats to validity of steps involved in this study and how we mitigated them.

Limitations related to survey with software engineering professionals:

Population. An expressive (but still small) sample of 58 software engineering professionals was involved in the survey research. In order to maximize the number of participants, we invited as many national and international participants as we could, given a fixed time period. We tried to obtain a diversified representation of realities across different regions and countries. However, the study was certainly limited to the number of answers. This is also a difficulty in similar studies in software engineering (Smith et al., 2013). We can also highlight the number and type of the problems presented to participants as possible limitations of the study.

GT. Another limitation of this work is related to the coding phases in GT. Once the process is inherently related to human interpretations and opinions, results could certainly differ from those coded by a different team. In our study, two researchers coded and two researchers revised in order to reduce such issues.

Survey Execution. We also remark two important lim-

itations that shall be taken into account to evaluate the extension of the validity and potential of generalization of the findings communicated in this paper: (i) the portion of ExM presented and how it was presented, and (ii) specific type of ExM technology used in the study. Regarding the former factor, we provided a video to expose the participants (particularly those with no experience on ExM) to what could be considered the graphical part of an ExM in order to provide subsidies to answer the questionnaire. Moreover, we present only the graphical part (not the underlying specification model), showing how a video could also bring difficulties for the participants since they could not control the model execution as they wished. In addition, as an exploratory study, we believe the results are valuable as perceptions for future studies. Regarding the latter limitation, we only presented one specific type of ExM (a simulation model). models@runtime and ExM obtained via transformations were not in the scope of this study. Hence, further studies are still required to cope with other types of ExM. However, several findings are generic enough to be adopted in all of those technologies.

Limitations related to the definition of the strategies and their assessment:

Strategies Definition. The obtained strategies are also inherently subject to human interpretations and opinions. In order to mitigate this threat, three software engineering and ExM experts supported the elaboration and assessment of the strategies.

Strategies Assessment. In order to mitigate the elaboration of the strategies, we also assessed the obtained strategies with 13 software engineering researchers with varying experience with software engineering, in academia and also in the industry. As for the recruitment of evaluators of the strategies, in fact, the time to conduct the assessment was limited. Thus, some of the participants were partners of the research groups involved in the production of the article. And then, this can add the possibility of bias in the assessment. Another threat to the validity of the assessment is that in both the survey and the assessment of the strategies, we did not save any information regarding the identity of the respondents. Because of that, we can not cross and compare the obtained results and removed respondents that possibly participated in both studies.

8 Final Remarks

The two main contributions of this study were to communicate findings of survey research conducted to answer the research question *What are the perceptions of software engineering professionals about the expressiveness and understandability of DEVS executable simulation models?* and to develop a list of strategies to answer the research question *How to support the evolution of ExM notations to be aligned with software engineering professionals' needs?*

The answers to the survey were quantitatively and qualitatively analyzed. From a set of 58 participants, more than half (precisely 30 participants) did not have any contact with

ExM before. In addition, several participants (even the experienced ones) suggested improvements and opportunities make ExM technology closer to their current practice, making its adoption easier.

The performed analysis allowed us to identify that (i) professionals envision several advantages and opportunities for applying ExM in software engineering such as architectural assessment and documentation, (ii) the research field should provide an effective cost-benefit infrastructure for software development before its larger adoption, and (iii) any successful strategy for adoption should be integrated in the software development process smoothly, reducing cognitive stress of learning new technology.

Based on these results, we compiled a list of strategies to answer our second research question, in which, we aim to support the evolution of ExM notations elicited from the needs raised by the respondents of the survey. The strategies were organized into four categories of improvement: (i) notation, (ii) visual representation, (iii) usage, and (iv) user support. Some of the strategies aim to improve the animation of the models by adding support for colors and the possibility to users add their own symbols to better represent the explored domain. The strategies also focus on improving the scalability of the models (visual presentation and execution), the management of the models and increase the reuse of models to use in fast prototyping.

The strategies were later assessed with 13 software engineering researchers from Brazil, Australia and the United States to identify their perceptions and to contribute with improvements. In this small sample, the strategies received a positive score. Only two participants contributed with two different opinions. One participant was positioned against the importance of the visual presentation and another stated the need for strategies to deal with behavior analysis and performance. Simulating and observing the execution of the models are means to achieve behavior analysis and, in a sense, already justify the importance of having a good visual representation. The strategies for scalability also cover part of the requirement for performance. Future work includes replicating the study in a controlled environment as soon as the Covid-19 pandemic is surpassed.

In conclusion, we highlight the importance of ExM, such as simulation and digital twins, mainly in the context of Industry 4.0 and smart cities where systems simulation have been applied and can bring high quality to the systems being developed henceforth (Müller-Zhang et al., 2020). By addressing the proposed strategies, we further prepare the ExM notations to deal with the challenge involved in the development of those complex systems.

Acknowledgements This article was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The 3rd author thanks to UNIRIO and FAPERJ (Proc. 211.583/2019) for partial support. The 4th author thanks to CAPES (PROCAD-Amazonia: 88887.200532/2018-0), the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0; and the State of Maranhão Research Funding Agency - FAPEMA (UNIVERSAL-00745/19;BEPP-01608/21).

Data All the data collected during this survey can be found in <https://github.com/brlebtog/JSERD-Survey>.

References

- Agner, L. T. W., Soares, I. W., Stadzisz, P. C., and Simão, J. M. (2013). A brazilian survey on UML and model-driven practices for embedded software development. *Journal of Systems and Software*, 86(4):997–1005.
- Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley Professional, 3rd edition.
- Bertalanffy, L. (2015). *General system theory : foundations, development, applications*. George Braziller, Inc, New York.
- Boehm, B. (2006). A view of 20th and 21st century software engineering. In *28th ICSE*, page 12–29. ACM.
- Bogado, V., Gonnet, S., and Leone, H. (2014). Modeling and simulation of software architecture in discrete event system specification for quality evaluation. *SIMULATION*, 90(3):290–319.
- Bork, D., Karagiannis, D., and Pittl, B. (2018). Systematic analysis and evaluation of visual conceptual modeling language notations. In *2018 12th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–11.
- Bowmaker, J. K. (1998). Evolution of colour vision in vertebrates. *Eye*, 12(3):541–547.
- Chreyh, R. and Wainer, G. (2009). Cd++ repository: An internet based searchable database of devs models and their experimental frames. In *Proceedings of the 2009 Spring Simulation Multiconference*, SpringSim ’09, San Diego, CA, USA. Society for Computer Simulation International.
- Corbin, J. (2015). *Basics of qualitative research : techniques and procedures for developing grounded theory*. SAGE.
- da Costa Carvalho, E., Malcher, P. R. C., and dos Santos, R. P. (2020). A survey research on the use of mobile applications in software project management. In *19th Brazilian Symposium on Software Quality*. ACM.
- Dahmann, J., Markina-Khusid, A., Doren, A., Wheeler, T., Cotter, M., and Kelley, M. (2017a). Sysml executable systems of system architecture definition: A working example. pages 1–6.
- Dahmann, J., Markina-Khusid, A., Doren, A., Wheeler, T., Cotter, M., and Kelley, M. (2017b). Sysml executable systems of system architecture definition: A working example.
- de França, B. B. N. and Ali, N. B. (2020). The role of simulation-based studies in software engineering research. In *Contemporary Empirical Methods in Software Engineering*, pages 263–287. Springer.
- Favre, J.-M. (2005). Megamodelling and etymology. In *Transformation Techniques in Software Engineering*.
- Fernandes, J., Graciano Neto, V. V., and Santos, R. P. d. (2018). Interoperability in systems-of-information systems: A systematic mapping study. In *17th SBQS*, page 131–140. ACM.
- Ferreira, T., Viana, D., Fernandes, J., and Santos, R. (2018). Identifying emerging topics and difficulties in software engineering education in brazil. In *32nd SBES*. ACM.
- Gorschek, T., Tempero, E., and Angelis, L. (2014). On the use of software design models in software development practice: An empirical investigation. *Journal of Systems and Software*, 95:176–193.
- Gray, J. and Rumpe, B. (2016). Models in simulation. *Software & Systems Modeling*, 15(3):605–607.
- Guessi, M., Cavalcante, E., and Oliveira, L. B. (2015). Characterizing architecture description languages for software-intensive systems-of-systems. In *2015 IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems*, pages 12–18. IEEE.
- Guessi, M., Graciano-Neto, V. V., and Nakagawa, E. Y. (2019). Architectural description of systems-of-information. In *Tópicos em Sistemas de Informação: Minicursos SBSI 2019*, pages 29–52. SBC.
- Hlupic, V. (2002). Simulation software: An operational research society survey of academic and industrial users. pages 1676–1683.
- Hojaji, F., Mayerhofer, T., Zamani, B., Hamou-Lhadj, A., and Bousse, E. (2019). Model execution tracing: a systematic mapping study. *Software and Systems Modeling*, 18(6):3461–3485.
- Hu, J., Huang, L., Cao, B., and Chang, X. (2014). Spdml: Graphical modeling language for executable architecture of systems. pages 248–255.
- Judd, D. (1975). *Color in business, science, and industry*. Wiley, New York.
- Kasunic, M. (2005). Designing an effective survey. Technical Report CMU/SEI-2005-HB-004, Carnegie Mellon Software Engineering Institute, Pittsburg, USA.
- Keller, N., Zeigler, B., Kim, D., Anderson, C., and Ceney, J. (2020). Supporting the reuse of algorithmic simulation models. In *Proceedings of the 2020 Summer Simulation Conference*, SummerSim ’20, San Diego, CA, USA. Society for Computer Simulation International.
- Kim, T. G. and Zeigler, B. P. (1987). The DEVS formalism: hierarchical, modular systems specification in an object oriented framework. In Thesen, A., Grant, H., and Kelton, W. D., editors, *Proceedings of the 19th conference on Winter simulation, WSC 1987, Atlanta, GA, USA, December 14-16, 1987*, pages 559–566. ACM.
- Lebtog, B., Teixeira, P., Santos, R., Viana, D., and Graciano Neto, V. (2020). Evaluating the understandability and expressiveness of simulation executable models with professionals – obtaining perceptions from researchers and practitioners for improving quality of models. In *Proceedings of the 19th Brazilian Symposium on Software Quality*, SBQS’20, New York, NY, USA. Association for Computing Machinery.
- Levis, A. H. and Wagenhals, L. W. (2000). C4isr architectures: I: developing a process for c4isr architecture design. *Systems Engineering*, 3(4):225–247.
- Linåker, J., Sulaman, S., Host, M., and de Mello, R. (2015). Guidelines for conducting surveys in software engineering. Technical report, Lund University, Sweden.
- Mahmood, S., Ahmed, M., and Alshayeb, M. (2013). Reuse

- environments for software artifacts: Analysis framework. In *12th ICIS*, pages 35–40.
- Mangen, A., Olivier, G., and Velay, J.-L. (2019). Comparing comprehension of a long text read in print book and on kindle: Where in the text and when in the story? *Frontiers in Psychology*, 10.
- Manzano, W., Neto, V. V. G., and Nakagawa, E. Y. (2020). Dynamic-sos: An approach for the simulation of systems-of-systems dynamic architectures. *Comput. J.*, 63(5):709–731.
- Mendling, J., Reijers, H., and Recker, J. (2010). Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, 35(4):467–482.
- Michael, J. B., Drusinsky, D., Otani, T. W., and Shing, M. (2011). Verification and validation for trustworthy software systems. *IEEE Software*, 28(6):86–92.
- Mincarone, P., Leo, C. G., Trujillo-Martín, M. D. M., Manson, J., Guarino, R., Ponzini, G., and Sabina, S. (2018). Standardized languages and notations for graphical modelling of patient care processes: a systematic review. *Int J Qual Health Care*, 30(3):169–177.
- Molléri, J. S., Petersen, K., and Mendes, E. (2016). Survey guidelines in software engineering: An annotated review. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM ’16, pages 58:1–58:6.
- Mussbacher, G., Amyot, D., Breu, R., Bruel, J.-M., Cheng, B., Collet, P., Combemale, B., France, R., Heldal, R., Hill, J., Kienzle, J., Schöttle, M., Steimann, F., Stikkolorum, D., and Whittle, J. (2015). The relevance of model-driven engineering thirty years from now. In *Proceedings of the International Conference on Model Driven Engineering Languages and Systems*, pages 1–12. Springer.
- Müller-Zhang, Z., Antonino, P. O., and Kuhn, T. (2020). Dynamic process planning using digital twins and reinforcement learning. In *25th IEEE ETFA*, volume 1, pages 1757–1764.
- Neis, P., Wehrmeister, M. A., and Mendes, M. F. (2019). Model driven software engineering of power systems applications: Literature review and trends. *IEEE Access*, 7:177761–177773.
- Neto, V. V. G., Rodriguez, L. M. G., Guessi, M., de Barros Paes, C. E., Manzano, W., Oquendo, F., and Nakagawa, E. Y. (2018). ASAS: an approach to support simulation of smart systems. In *51st HICSS*, pages 1–10.
- Nystrom, R. (2014). *Game programming patterns*. Genever Benning, United States.
- OMG Executable UML (2017). Action language for alf. Standard, Object Management Group, Massachusetts, USA.
- OMG Executable UML (2018). Semantics of fuml. Standard, Object Management Group, Massachusetts, USA.
- Prinz, N., Rentrop, C., and Huber, M. (2021). Low-code development platforms—a literature review.
- Purchase, H. C., Cohen, R. F., and James, M. (1996). Validating graph drawing aesthetics. In *Graph Drawing*, pages 435–446. Springer Berlin Heidelberg.
- Rasheed, A., San, O., and Kvamsdal, T. (2020). Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, PP:1–1.
- Sedano, T., Ralph, P., and Péreira, C. (2019). The product backlog. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 200–211.
- Selic, B. (2008). Personal reflections on automation, programming culture, and model-based software engineering. *Automated Software Engineering*, 15(3):379–391.
- Siegenthaler, E., Wurtz, P., and Groner, R. (2010). Improving the usability of e-book readers. *Journal of Usability Studies archive*, 6:25–38.
- Singh, M. and Hoffman, D. D. (2013). Natural selection and shape perception. In *Shape Perception in Human and Computer Vision*, pages 171–185. Springer London.
- Smith, E., Loftin, R., Murphy-Hill, E., and Zimmermann, T. (2013). Improving developer participation rates in surveys. pages 1–4.
- Steinmacher, I., Conte, T. U., Treude, C., and Gerosa, M. A. (2016). Overcoming open source project entry barriers with a portal for newcomers. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 273–284.
- Steinmacher, I., Treude, C., and Gerosa, M. A. (2019). Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software*, 36(4):41–49.
- Teixeira, P., Lebtag, B., dos Santos, R., Costa Fernandes, J., Mohsin, A., Kassab, M., and Graciano Neto, V. (2020). Constituent system design: A software architecture approach. pages 218–225.
- Tidwell, J. (2020). *Designing interfaces : patterns for effective interaction design*. O'Reilly Media, Sebastopol, CA.
- Torchiano, M., Ricca, F., Tiso, A., and Reggio, G. (2011). Preliminary findings from a survey on the md state of the practice. In *5th ESEM*, pages 372–375, Banff, Canada.
- Vlahovic, N. and Ceric, V. (2008). Multi-agent simulation in organizations. In *Encyclopedia of Information Science and Technology, Second Edition*, pages 2728–2733. IGI Global.
- Wang, R. and Dagli, C. (2011). Executable system architecting using systems modeling language in conjunction with colored petri nets in a model-driven systems development process. *Systems Engineering*, 14(4):383–409.
- Whittle, J., Hutchinson, J. E., Rouncefield, M., Burden, H., and Heldal, R. (2017). A taxonomy of tool-related issues affecting the adoption of model-driven engineering. *Software and Systems Modeling*, 16(2):313–331.
- Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.
- Xavier, B. L., dos Santos, R. P., and dos Santos, D. V. (2020). Software ecosystems and digital games: Understanding the financial sustainability aspect. In *Proceedings of the 22nd International Conference on Enterprise Information Systems*. SCITEPRESS - Science and Technology Publications.
- Zaitsev, D. A., Shmeleva, T. R., and Sleptsov, A. I. (2018). Reenterable colored petri net models of networks, grids, and clouds: Case study for provider backbone bridge. In

- 2018 26th Telecommunications Forum (TELFOR). IEEE.
- Zeigler, B., Sarjoughian, H. S., Duboz, R., and Soulie, J.-C. (2016). *Guide to Modeling and Simulation of Systems of Systems*. Springer Publishing Company, Incorporated, 1st edition.
- Zeigler, B. P., Muzy, A., and Kofman, E. (2018). *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. Academic Press, Inc., USA, 3rd edition.

A Acronyms

Table 11. Acronyms used throughout this article

Acronym	Definition
A	Advantages
ADL	Architectural Description Language
ALF	Action Language for Foundational UML
AQ	Atitudinal Questions
BPMN	Business Process Model and Notation
CEQ	Closed-ended Question
CPN	Colored Petri Nets
CQ	Characterization Question
DEVS	Discrete Event System Specifications
D	Disadvantage
DSL	Domain Specific Language
DQ	Demographic Question
ExM	Executable Model(s)
F	Difficulties
fUML	Foundational UML Subset
GT	Grounded Theory
I	Improvement Suggestions
IDE	Integrated Development Environment
MDE	Model-driven Engineering
MQ	Main Question
N	Notation
OEQ	Open-ended Question
UML	Unified Modeling Language
P	Perceptions
R	Requirements for new Technologies
RQ	Research Question
SoS	System of Systems
SQ	Study Question
SySML	Systems Modeling Language
V	Visual Presentation
V&V	Verification & Validation