# How do I model my system? A Qualitative Study on the Challenges that Modelers Experience

Christopher Vendome
Miami University
Oxford, OH

Eric J. Rapos
Miami University
Oxford, OH

Nick DiGennaro
Miami University
Oxford, OH

## ABSTRACT

Model-Driven Software Engineering relies both on domain-expertise as well as software engineering expertise to fully grasp its representative power in modeling complex systems. As is typical in the development of any system, modelers face similar challenges to classic software developers, whether with general modeling concepts or specific features of existing tools such as the *Eclipse Modeling Framework*. In this work, we aim to understand the issues that modelers face by analyzing discussions from Eclipse's modeling tool forums, MATLAB Central, and Stack Overflow. By performing a qualitative study using an open-coding process, we created a taxonomy of common issues faced by modelers. We considered both difficulty experienced when modeling a system and issues faced using existing modeling tools; these form the basis of our two research questions. Based on the taxonomy, we propose nine suggestions and enhancements, in three overarching groups, to improve the experience of modelers, at all levels of experience.

**ACM Reference Format:**
Christopher Vendome, Eric J. Rapos, and Nick DiGennaro. 2022. How do I model my system? A Qualitative Study on the Challenges that Modelers Experience. In *Proceedings of The 30th International Conference on Program Comprehension (ICPC 2022).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

While Model-Driven Development (MDD) is a paradigm shift from traditional development processes that rely heavily on developers writing code, it has been effectively used in industrial settings [1, 2]. MDD focuses on representing software at a high-level of abstraction through representations that model a system, and from which code can be automatically generated. An important benefit of this paradigm is that it makes the development process more accessible to the stakeholders, since they can better understand the structure and design of the system without looking at its implementation.

While researchers have examined discussion forums such as Stack Overflow in order to improve the software development process [3–5], the same cannot be said for Model-Driven Software Engineering (MDSE). Even though many of the same paradigms and principles of traditional software engineering apply in MDSE, there has been little work in leveraging the community-based knowledge repositories to improve modeling processes, languages, or tools. MDSE relies on principles such as abstraction, transformation, and automation to enhance the software development process. While MDSE focuses on these aspects, there are enough similarities to traditional software engineering to indicate that the same forum mining techniques can be replicated for MDSE applications.

Since MDSE relies heavily on domain-knowledge and expertise, it stands to reason that there is more to be gained from examining community-based repositories for insights into modeling practices, environments, and tooling. By searching for patterns in commonly asked questions, it is possible to identify emergent trends in usage that were not envisioned by language designers, as well as common pitfalls that were previously unknown. Through the examination of various modeling discussion forums, we aim to highlight trends in frequently asked questions (FAQs), as well as identify suggestions to improve the modeling experience generally. This taxonomy, and resulting lessons and suggestions, should be considered a call to action to the members of the modeling community to address these issues, and make modeling a more open and accepting discipline.

To accomplish this goal, we performed a qualitative study on discussions posted to popular MDSE tool forums (*e.g.,* Eclipse and MATLAB Central) and Stack Overflow. We first mined the relevant discussions, which yielded 71,158 total discussion posts. From this dataset, we followed a formal open-coding process on a statistically significant random sample of 1,200 discussions (400 from each data source). We identified 10 high-levels categories composed of 45 sub-categories from this process that represent the difficulties that modelers face both with modeling and with utilizing existing tools. While the focus of these discussions commonly related to conceptual understanding of MDSE and tool-specific issues, interestingly, we also observed a large number of discussions that sought guidance from the modeling community (*e.g.,* how to get started in modeling or feedback on the design of a particular model).

Additionally, we distilled lessons and suggestions for improvement from the results of our qualitative analysis to provide actionable recommendations that will have a strong positive impact on the MDSE community. By understanding the issues most commonly faced by MDSE practitioners, we are better able to inform the future design of modeling language frameworks to mitigate these common pitfalls. By bringing the modeling community's opinions into the design process, we are preparing to better serve those using modeling tools. Through the creation of the taxonomy and the discussion of our findings, we aim to create a set of recommendations for the creation of modeling frameworks that are better designed to serve the modeling community. Beyond tooling suggestions, our analysis provides insights into structural suggestions to improve the modeling community as a whole. These enhancements will particularly

*ICPC 2022, May 21–22, 2022, Pittsburgh, PA, USA*
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

benefit novice modelers, with improvements that also benefit the broader community through improved tools and resources.

## 2 BACKGROUND & RELATED WORK

### 2.1 Model-Driven Software-Engineering

For the purposes of creating this Model-Driven Software Engineering (MDSE) taxonomy, we place emphasis on the aspects of MDSE related to conceptual system design, simulation, code generation, and the application of general modeling principles (such as abstraction, representation, etc.). It is through these applications that MDSE has gained popularity and increased usage; however, the increase in usage has led to an increase in inexperienced users attempting to master skills without adequate understanding or well designed tooling.

In 2017, a workshop was held that was comprised of a variety of MDSE researchers, educators, and practitioners. The purpose of the workshop was to identify significant open problems in the field, and ultimately resulted in the publication of the "Grand Challenges in Model-Driven Engineering" [6].Our work complements this work through the inclusion of a wider collection of user issues via discussion forums from which we are able to empirically derive a taxonomy that is specifically focused on issues that users (or modelers) experience.

A group of researchers has recently assembled a list of topics relevant to model-based software engineering, through community involvement and feedback cycles. The Model-Based Software Engineering Body of Knowledge (MBEBoK) [7] aims to be the standard list of topics in the field, and represents the various fields of study related to software modeling. This body of knowledge serves as one of the metrics of comparison for our resulting taxonomy, as we posit that any taxonomy representing common issues in a field ought to have specific mappings to these defined knowledge areas.

*2.1.1 Modeling Tools.* There are numerous tools that exist with the goals of leveraging MDSE; however, for the scope of this project, we aim to focus on leading tools that leverage code-generation as a primary feature in order to narrow the field of tools. One of the main reasons for this choice is the popularity of these tools, and the pervasiveness of their usage. The following tools were chosen for further investigation of their usage to help answer the research questions within this study: MATLAB Simulink, Eclipse Modeling Framework, Graphical Modeling Framework, Papyrus, and Papyrus-RT.

**MATLAB Simulink**[8] is an industrial modeling suite used to design and simulate systems (typically for embedded controllers), and ultimately generate code for deployment. It gained popularity in automotive, aerospace, robotics, and other control system domains.

**The Eclipse Modeling Framework (EMF)**[9] is primarily a meta-modeling (data-modeling) environment in which users create data models of their domain in order to leverage the model for later use in instance creation, testing, or through the creation of model editors. It is much more popular within academic research due to its open source nature than industrial alternatives (such as Simulink).

**The Graphical Modeling Framework (GMF)**[10] (alternatively branded as Graphical Modeling Project (GMP)) is an extension of EMF that allows users to create their own graphical editors and modeling tools for the data models created within EMF.

**Papyrus**[11] is a more general-purpose modeling tool that is used in both industry and academia to produce high quality models, typically, various types of Unified Modeling Language (UML) [12] models (specifically UML 2.5), as well as SysML models [13].

**Papyrus-RT**[14] is a relative of Papyrus, focused on modeling complex real-time applications. Specifically, it is an open source implementation of the UML-RT profile [15]. Papyrus-RT implements the profile in a manner that allows users to design their system (both structure and behavior) and generate executable code for use in embedded systems [16].

*2.1.2 Modeling Language Design.* Understanding what makes for a successful and effective modeling language has been a long discussed question and one that various research addresses quite well. An early look at design of modeling languages was presented by Bran Selic [17] in which he presents his personal perspective based on years of experience. While the content stems from his own experiences, his claim that modeling language design is "more of an art than a science" describes the difficulties faced by tool and framework developers that our work aims to address. Selic is not the only researcher to tackle the issues of modeling language design; the following works demonstrate the importance of language design to the effective adoption and use of MDSE.

Based on work by Hutchinson *et al.* [18], Cabot *et al.* [19] claim that one of causes of the limited adoption of MDSE is due to "a variety of social and technical factors but we can summarize them all in one: its benefits do not outweigh its costs". Cabot *et al.* present a similar approach to overcome some of these barriers; specifically, they propose the use of cognification to demonstrate that the benefits outweigh the costs of MDSE. Cognification is the application of large volumes of data collected from various sources (including forum posts), combined with AI approaches to boost the performance and impact of a process (in this case, MDSE). Our work differs from their approach in that we focus on the creation of a taxonomy of common issues faced by attempted practitioners and attempt to inform tool and language design to preempt these questions.

Mengerink *et al.* [20] have similarly conducted a study to obtain insights into the application of modeling, specifically in industrial settings. Their motivation lies in understanding how technologies are actually used in practice, in order to help with the development of effective tooling and environments. To this end, they produced the EMF (Meta)Model Analysis Tool (EMMA), which provides an automated analysis of modeling artifacts that provides the users with various metrics about the usage of the modeling artifacts. This work shares a similar motivation to our taxonomy; while they focus primarily on technical aspects, we aim to present a more holistic examination into the issues faced by modelers in a broader setting.

Much like programming languages, modeling languages have been the focus of educational language design. Recently, there has been an effort to create modeling languages that are designed for teaching, with the aim of avoiding many of the common pitfalls of existing modeling frameworks. While these approaches rely on pedagogy over the empirical methodologies employed in our research, the shared goals demonstrate the necessity for effective language design. One such example of educational modeling language design is the Instructional Modeling Language (IML) proposed by Rapos and Stephan [21]. IML aims to confront the issues of bloated tools and the lack of an end-to-end tooling through the introduction of a single framework for instructing modeling to students.

### 2.2 Empirical Software Engineering

Stack Overflow has been widely utilized in the Mining Software Repository community to leverage knowledge of the *crowd* for various software maintenance and evolution purposes. The 2019 Mining Software Repository (MSR) Challenge focused on utilizing the SOTorrent Dataset [22] that links Stack Overflow to GitHub projects as well as provides a version history of Stack Overflow. There have been similar qualitative studies to ours that focused on other areas like accessibility difficulty [23], Android faults [24], licensing issues [25], and documentation issues [26].

Barua *et al.* [5] utilized latent Dirichlet allocation (LDA) to automatically extract topics from Stack Overflow discussions in order to understand how trends in these discussion change over time. Similarly, Bangash *et al.* [27] used manual labeling and LDA to investigate Stack Overflow posts that are related to machine learning. *Prompter* by Ponzanelli *et al.* [4] recommends relevant Stack Overflow discussions based on the current context in the IDE. *SOFix* by Liu and Zhong [3] leveraged Stack Overflow discussions to identify repair patterns from which the authors created 13 repair templates to enhance automatic program repair. Recently, Geremia *et al.* [28] investigated the characteristics that lead developers to utilize a particular Stack Overflow post, and built a classifier that aims to identify such "quality" posts.

## 3 METHODOLOGY

The study's *goal* is to understand the difficulty that modelers face from both modeling issues and tool-related issues. To identify these issues, we analyzed discussion forums related to Simulink, EMF, GMF, Papyrus, and Papyrus-RT as well as discussions from Stack Overflow. To understand the difficulties of modelers, we investigated the following two research questions:

**RQ$_1$:** *What are the types of modeling issues that modelers face?* In this research question, we aim to better understand the modeling concepts with which modelers struggle.

**RQ$_2$:** *What are the difficulties that modelers face when utilizing existing tools for modeling?* This question focuses on understanding the limitations or problems that modelers face with existing tools.

### 3.1 Data Collection

In order to answer these research questions, we mined three different sources for discussions: i) Eclipse forums related to Modeling tools, ii) MATLAB Central, and iii) Stack Overflow.

For the Eclipse forums, we focused specifically on the forums for the four Eclipse-based tools that we selected (EMF, GMF, Papyrus, and Papyrus-RT) as the source of posts. These forums are some of the most popular on the Eclipse Community Forum page, which should provide us with a depth and breadth of modeling issues. Since each of these forums focus specifically on the tools themselves, there was no need for any search terms or tag/title filtering; all posts were deemed potentially relevant (i.e., they were considered likely relevant, but could still have false positives). Using these forums, we were able to implement a web scraper to gather: the question text, the title, number of views, number of replies, and the link of each post. This data was collected on August 26, 2019.

For Simulink, we were able to create a web crawler using MATLAB scripting to obtain all posts on MATLAB Central that were tagged with the "*Simulink*" tag. Given that Simulink is a modeling tool and to be consistent with our design for the Eclipse discussions, we did not perform any further filtering to obtain relevant posts. The crawler queried the forum for any posts with the tag, visited each post, and collected a number of attributes for each post: the title, tags, question text, number of answers, number of votes, number of views, and the link of the question on the forum. The data was collected as of August 28, 2019.

For Stack Overflow, we identified the relevant posts from the Stack Overflow March 2019 data dump. After downloading that data dump, we defined a set of modeling keywords to identify the relevant discussions. The complete list of keywords are: *"mdse"*, *"mdd"*, *"model-driven"*, *"model"*, *"modeling"*, *"simulink"*, *"papyrus"*, *"gmf"*, and *"emf"*. Then, we performed a case-insensitive keyword search on the titles, descriptions, and tags (users can specify the topic of a post with a *tag*) to extract discussion posts that were relevant to concepts in modeling or MDSE. Unlike the prior two data sources, Stack Overflow is a general QA forum that covers a broad set of computing-related questions; because of this, we required this additional step of filtering to identify relevant posts.

By mining these data sources, we extracted 20,331 posts from Eclipse, 13,504 posts tagged as Simulink from MATLAB Central, and 37,323 posts from Stack Overflow. For our study, we randomly sampled 400 posts from each of these sources totaling 1,200 discussions. We chose 400 posts from each source to make sure we had statistically significant samples. Because some of the search terms can have other meanings (*e.g.,* "model" would also return posts related to Model-View-Controller), one author performed an initial filtering to discard clear false positives. This filtering occurred until reaching the necessary 400 posts. These samples represent statistically significant samples with a 95% confidence and an interval of less than 5% (for Eclipse, the sample has a confidence of 95% ± 4.85; for Simulink, the sample has a confidence of 95% ± 4.83; for Stack Overflow, the sample has a confidence of 95% ± 4.87 when considering the pre-filtered population). We have made the data extracted from these sources available as well as the samples used in our study available [29].

### 3.2 Qualitative Analysis

For each sample, we performed an open-coding process inspired by Miles and Huberman [30] and followed a similar protocol to existing work [23]. This process was a multi-coding process to address the two research questions (*i.e.,* the modeling-related issues and the tool-related issues) as well as identifying the general modeling topic of the discussion post (we present these as part of our discussion in Section 5); thus, each discussion post contained *three* free-form codes. Since we used an open-coding process, these codes were not predefined prior to the analysis (i.e., the authors were not confined to a set of codes and could create new ones during the analysis). These codes represent the underlying difficulty that modelers experience related to the research question. For example, RQ2 focuses on the difficulty that modelers experience with an existing tool. Consider a post discussing difficulty with utilizing a signal input to an S-functions in Simulink (i.e., a tool-specific issue)[31], we could have the code *S-Function Issue* labeling this post; the rationale for this code would be that the post demonstrates difficulty with an underlying tool-specific feature. In general, a code was assigned to be representative of the underlying issue present in each post. If a discussion post was not relevant to one of these aspect, the authors were instructed to use "N/A" for the code. In this process, each discussion post was analyzed by at least two authors.

To perform the open-coding process, we used our own custom online tool. The tool assisted in the random selection of posts while ensuring that each post was coded by two authors. The tool displayed the original title of the post, the body of the post, and a link to entire discussion thread. The tool also provided free-form text fields for the authors to provide a code for each of the aspects investigated and validated the input to ensure codes were not left unfilled. The tool maintained a list of previously used codes for each topic and the list was displayed to users to help avoid semantically similar tags with different text. This list was updated each time a new code was added. We performed this analysis iteratively and met to discuss the codes to ensure consistency between coding sessions. Additionally, the existing codes where merged when applicable to avoid redundant codes. For each iteration, a sample of 200 posts were coded by each author, resulting in 4 iterations. In the last iteration, we observed no new codes were introduced. A final round of coding was performed to address conflicts in the coding between two authors; for each conflict, a third author was assigned to resolve the disagreement. We computed Fleiss' kappa to assess inter-rater agreement [32]. For $\mathbf{RQ}_1$, we observed $\kappa = 0.83$, which corresponds to an almost perfect agreement ($0.81 < \kappa < 1.0$); for $\mathbf{RQ}_2$, we observed $\kappa = 0.46$, which corresponds to a moderate agreement ($0.41 < \kappa < 0.60$) [33].

After the coding process, we generated a taxonomy for each research question, independently. We organized the codes into higher-level categories through a card-sorting approach [34]. The codes were iteratively clustered into groups, which represent a higher-level abstraction of the codes. We performed this card-sorting until all of the codes were assigned and converged on a common set of clusters. The number of clusters were not predefined, but were determined systematically through the iterative card-sorting process. Subsequently, we organized these groups hierarchically when there was a relationship between two or more groups. This process yielded the two-level taxonomy shown in Section 4 (implicitly, the root of this taxonomy would be *Model-Driven Development*). From this process, we generated a taxonomy with 10 top-level categories, which represent the general theme of the discussions, and 45 sub-categories, which represent the more specific concepts/topics for the clusters generated by the card-sorting. It is worth noting that not every category or sub-category was represented in both research questions, but we present these together due to space limitations. For example, discussions related to *Guidance & Understanding* could be relevant to both modeling concepts (*e.g.,* understanding best practices) as well as tools for modeling (*e.g.,* finding the appropriate tool); conversely, discussions focused on *Tool Features* were only relevant in the context of $\mathbf{RQ}_2$. In Section 4, we present the full taxonomy and further discuss these categories and sub-categories.

Beyond those classified in the taxonomy, we observed 44 discussions (3.67%) that were false positives (*i.e.,* not modeling-related) and were omitted from the taxonomy. For example, we observed some discussions asking about generic Eclipse plug-in development.

## 4 RESULTS

In Figure 1, we present the full taxonomy generated from our qualitative analysis, which is comprised of concepts that correspond to both research questions. Due to space constraints, we present the results of our card sorting for both research questions in this figure. Note that this card sorting of each research question's codes were done independently and the visualization represents an overlap of the two taxonomies. Individual visualizations are available in our replication package [29]. In the figure, the numbers on the left of each concept and category correspond to the number of posts on that topic that address $\mathbf{RQ}_1$ (top) and $\mathbf{RQ}_2$ (bottom). The images on the right relate to the implications in the discussion in Section 5. All of the categories except one (Distributed Modelling) contain forum posts that address both research questions. The top-level categories are sorted in decreasing order of the larger of these two numbers, with Modeling Tools being the largest. Within each category, the concepts of higher representation (based on number of posts) are located farther to the left.

Each concept in the taxonomy is also coded with the potential assignment of three icons relating to systematic ways of addressing the concepts. The top icon (mortarboard) relates to educational enhancements, the center (gear) relates to tool enhancements, and the bottom (presenter) relates to training and resource enhancements. More details about this process and the resulting lessons are presented in Section 5.

In the following subsections, we present each of the 10 categories and the constituent sub-categories (*i.e.,* the topics composing the categories) of forum posts through definition and example.

### 4.1 Modeling Tools

As the name implies, this category focused on difficulty with using existing modeling tools and understanding their current features. Unsurprisingly, this category was highly represented in posts relating to $\mathbf{RQ}_2$. Among the contained concepts, the most prevalent was *Simulink Block Issues*, which represented difficulty when using specific features of blocks. For example, one Simulink user asked:

> "I know how to adjust the amplitude which is simply by changing the amplitude value. But lets say if I wanna fix the period of sine wave to 5s and than change it to 10s?" [35]

As indicated, the individual already knows the necessary block and is familiar with its other functionality, but does not know how this specific feature is used. The *Simulink Block Issues* concept also consists of 26 discussions related to answering $\mathbf{RQ}_1$. These discussions focused on properly representing some specific behavior of a system with the appropriate block, or how to elicit a specific desired behavior from a block:

> "I have two signals that change with respect to time. You can say that one has a maximum amplitude of 1.37 and other one has a maximum amplitude of 0.93 at t=5. I tried using both; a function block and a subtraction block to subtract these two signals. But instead of getting the result that I expect which should be (1.37-0.93) at that maximum point, I am getting 1.73. I have no idea what is going on. I am so confused." [36]

This post demonstrates that the modeler has difficulty properly using the block, despite knowing which blocks represent the necessary behavior.

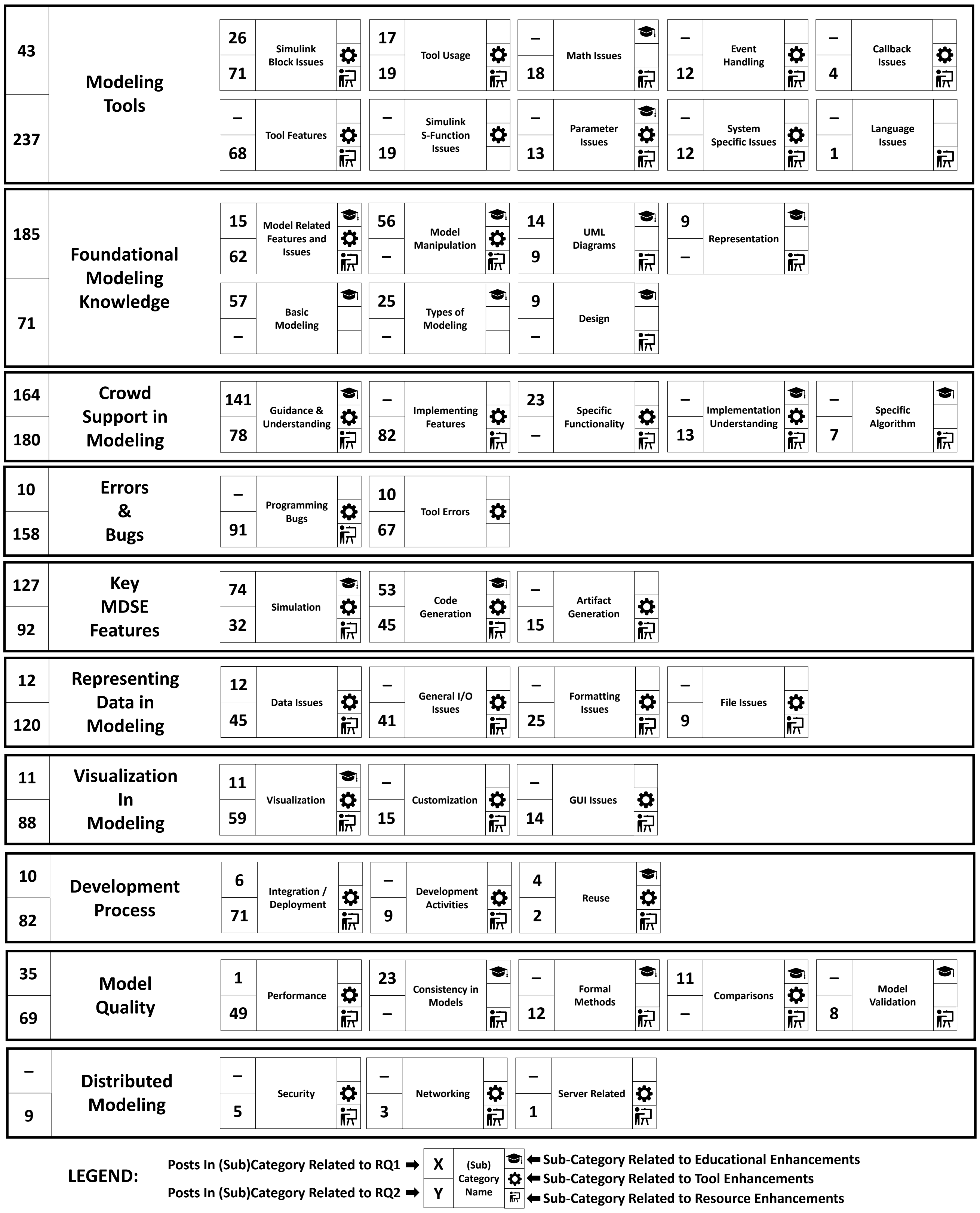

**Figure 1: Taxonomy of the challenges modelers face with modeling-specific issues (RQ1) and difficulty with existing modeling tools (RQ2). The taxonomy includes a mapping from the categories to their implications discussed in the Section 5.**

The *Tool Features* topic contains posts that focused on a specific aspect of the tool, these were general problems that would inhibit modelers from using the tool in a meaningful capacity. While some of the issues that we observed for this concept involved migrating between tools or differences between versions of tools, these discussions predominantly related to difficulty using certain features of a modeling tool. As one might expect, there were several discussions relating to *Tool Usage* that represented problems with running the tools or installing the tools. It is important to note that these could relate to a bug with the tool or some inability to use it as a whole and not just difficult with some subset of the tool. Additionally, from the modeling perspective (**RQ**$_1$), these discussions involved questions like how to collaboratively utilize multiple tools for some modeling task (*e.g.,* using a shared editing domain).

*Simulink S-Fuction Issues* related to difficulty using Simulink's S-functions that allow users to extend Simulink's functionality by creating their own custom blocks that are written in other languages and dynamically loaded to the model. For example, there were questions related to debugging S-fuctions and whether S-functions supported multi-threading capabilities. *Math Issues* focused on difficulty performing math computations, such as representing equations properly, handling matrices, or performing certain math operations like shifting a sine wave function.

The topic of *Parameter Issues* relates to setting or changing parameters of the models. For *System Specific Issues*, modelers asked questions regarding building models of specific subsystems (*e.g.,* fuzzy logic controllers or spectrum analyzers). *Event Handling* was comprised of discussions related to adding listeners to capture events, like resource changes or connection changes between nodes when using GMF. The *Callback Issues* represent discussions about difficulty using callbacks whether it is the modeler's misinterpretation regarding how callbacks work or an expected/erroneous behavior when using them. Finally, the *Language Issue* topic relates mainly to programming language issues. In the observed post, it relates to difficulty with C++ syntax with a custom block.

## 4.2 Foundational Modeling Knowledge

This category relates to difficulty that modelers experienced with the fundamental concepts and practices of modeling. Most common of these were difficulties with *Model-Related Features and Issues*. When considering **RQ**$_1$, we observe modelers having issues with model persistence. One modeler trying to modify the serialization process to avoid corruption asked the following:

> "The problem is that one of our team member had a window blue screen just when he was saving, as a result the repository was corrupted. Is there any way in EMF to secure the save process (in XMI resource), even if the VM crashes during the save?" [37]

Similarly, for **RQ**$_2$, modelers face difficulty trying to use certain features of tools. For example, one user on Stack Overflow who was building an editor posted questions about identifying nodes in GMF. Other types of difficulties that we observed relate to merging models, creating connections, and using references.

Specifically answering **RQ**$_1$, discussions on *Basic Modeling* were common with 57 discussion posts. These issues relate to understanding fundamental concepts of modeling, such as inheritance, associations, encapsulation, containment, constraints, among others. For example asking about the inheritance of stereotypes:

> "I really though Stereotypes were also inherited from the superclass, but when I implement this design in Papyrus tool, I can't see any inherited job or salary. Every subclass is the same as if it didn't have any applied Stereotype." [38]

Interestingly, the individual updated the post to indicate that the feature had existed in UML 1.3, but has since been removed in the UML 2.0 specification. Thus, these discussions are more centered on more fundamental concepts of modeling.

*Model Manipulation* discussions accounted for 56 discussions and include posts that involve model transformations, model migrations, merging models, and model evolution. For example, one modeler asked about bidirectional model transformations:

> "I would like to do bidirectional Model2Model transformations. Both models are EMF / eCore based. Actually I would prefer that one model is an editable view on the other. What are my options? Which tools and tranformation languages are avaiable and what are their restrictions?" [39]

While the replies offer suggestions and point to research on this problem, the consensus is a lack of existing support to solve this issue at the time, which is also corroborated by a reply from the original poster based on an interaction with an EMF developer.

The topic of *Type of Model* focused on issues that modelers experienced with various model types such as event modeling, functional modeling, modeling enumerations, system modeling, textual modeling, meta-modeling, and multi-view modeling.

We observed discussions where modelers experienced difficulty with *UML Diagrams* such as designing use case diagrams, handling large sequence diagrams, or when UML diagrams are applicable. These questions are not focused on the tools themselves, but understanding the principles related to properly using UML:

> "At my current employer we generally adopt the old-school approach of writing a traditional functional requirements specification and then performing a full tech design...With use-case modelling it seems that you need to gather pretty much the same information, it is just organised differently...So my question is this: what the tangible benefits of following a use-case driven approach to software development?" [40]

The discussion does not focus on how to model use cases or generate the diagram, but it focuses on *understanding its advantages*. The last two topics, tied with 9 posts each, were *Design* and *Representation*. The former discussions specifically inquired about design patterns when modeling a system. while the latter focused on how to properly represent some information within a model.

## 4.3 Crowd Support in Modeling

While the previous two categories primarily targeted one research question more than the other, *Crowd Support in Modeling* presents a more even split between the two (164 discussions related to answering **RQ**$_1$ and 180 related to answering **RQ**$_2$). These discussions sought to leverage the existing knowledge and expertise among the members of the community for both general guidance as well as problem-specific feedback.

For both **RQ**$_1$ and **RQ**$_2$, we observed that most of the discussions of this category related to *Guidance & Understanding*. This topic encompassed discussions from individuals that are new to MDSE and trying to understand its benefits or understand the best practices as well as looking for examples and tutorials:

"The communication between the domain experts (customer) and the developers is crucial to make this methodology work. What I want to know is if there is a tool suite or set of best practices that will help in the initial thrust of MDSD? Once the domain is fleshed out what about mapping that model to an ORM (or whatever)?" [41]

We also observed questions regarding choosing the right modeling language or tool for a specific application:

"I want to model this process or transaction and don't know the right modelling language for that purpose. Is the UML component model the right one?" [42]

As illustrated, these discussions aim to leverage the prior experience of modelers to either get started with modeling or use the proper existing tools to model some specific system.

Unlike more general guidance with modeling, we observed 82 discussions where modelers asked for direction and advice when *Implementing Features* in their models and the difficulty relates to the underlying tools. This topic focuses on very specific features (*e.g.,* adding a context menu) as opposed to modeling a component or subsystem. One modeler asked for assistance when extending the functionality of their system:

"I'm implementing a new server type for a BPEL engine and would like to add some new items to the context menu for the modules (BPEL processes).
Is there any way to add the items to the context menu? Where should I look in the jst plugins source code for an example?" [43]

Similarly, another modeler asks about extending the functionality related to dragging and dropping files to support dragging images and having an image path attribute updating accordingly:

"How can I make my EMF Editor (Pages) support (mouse-) dropping of files (FileTransfer Type) and/or resources (ResourceTransfer Type) e.g. from the Navigatorview ?" [44]

The remaining topics were substantially less frequent. *Specific Functionality* relates to the problems that modelers face with implementing functionality, such as message handling, serialization, or logging, into their models. For example, there was a post asking about the support for serialization of Ecore objects. The topic of *Implementation Understanding* focuses on understanding the underlying tool with respect to its features (*e.g.,* supporting dynamic extensions). For *Specific Algorithms*, we observed algorithm specific questions (implementing machine learning with a model or evolutionary algorithms). For example, there was a question relating to using particle swarm optimization for speech processing.

## 4.4 Errors & Bugs

The most common problems related to typical *Programming Bugs* (91 Discussions) that are introduced by the modelers. These issues related to semantic problems, like infinite loops, exceeding bounds, and size mismatch errors with which the modelers sought help:

"When I use the DemandPopulating package registry, it has a problem with the fact that I have a model which is in a subpackage of another model. CDOPackageRegistry.putEPackage() throws an illegal argument exception. Why is this not supported?" [45]

The *Programming Bugs* discussions seek to identify the gaps of understanding that caused the bug with one of the existing tools.

As with any development tool, many existing modeling tools, while robust, contain errors. These *Tool Errors* can be difficult or misleading, since they may inaccurately suggest that there is an issue with the underlying model, or can lead to frustration with MDSE rather than the tool implementations:

"The feature 'eFactoryInstance' of 'model' contains a dangling reference 'model'
The models seem fine otherwise and generate good code. Is this a problem I shoudl worry about?" [46]

However, the follow-up indicated that this issue did not stem from the model, but was due a bug of the tool that was fixed in a later version of EMF.

## 4.5 Key MDSE Features

These discussions related to difficulty that modelers experience or gaps in knowledge with the core, intrinsic features of MDSE. We observed a large number of discussions focused on different aspects of *Simulation* ($\mathbf{RQ}_1$: 74, $\mathbf{RQ}_2$: 32). For example, one Simulink user asked about performing a large number of simulations on different parameter configurations, while another individual wanted to reduce the sampling time.

The next most prevalent issue that modelers faced in this category was *Code Generation*, which is one of the tenants of MDSE. While we observed some straightforward questions about using specific tools, there were also discussions that relate to more complex tasks and usages of code generators.

"We are thinking of writing an EMF plugin that extends the EMF code generator in such a way as to generate MOF instead of Java...What I would like to find out is, whether some one here had tried extending the EMF code generator, and how compicated was it to do so." [47]

Similarly, a Simulink forum post asks about creating traceability links between the requirements and code during code generation:

"How can i show the imported requirements (from excel) in the autosar code when this code is generated? For the traceability of the requirements, the requirements should also be found in the generateded code for autosar." [48]

These posts demonstrate that modelers require more robust support and resources to assist them with more complex aspects of code generation. Similarly, some discussions were also related to difficulty with *Artifact Generation* based on the models, like class diagrams from a custom model.

## 4.6 Representing Data in Modeling

Models at their core are abstract representations of other systems, and as such a large component of MDSE relies on effective I/O of model artifacts and related data. Any issues that relate to this topic form the category of *Representing Data in Modeling* (12 and 120 discussions related to answering $\mathbf{RQ}_1$ and $\mathbf{RQ}_2$, respectively). The majority of these posts stem from two topics *Data Issues* and *General I/O Issues*. In the former, modelers experienced difficulty handling inputs and outputs with their model:

"I have 2 embedded Matlab functions which I am using to create a Simulink model. Both functions use the output of the second function as their input. I am getting an error at the moment indicating that this is an invalid loop. Does anyone know how to implement this type of behaviour?" [49]

The accepted answer to this post notes that the desired behavior is possible, but it required a delay.

Conversely, *General I/O Issues* focused on importing and exporting models. For example, a modeler from the Papyrus forums posted a discussion regarding exporting the model diagrams programmatically. Another example of difficulty that modelers faced related to customizing the import and export of models to ensure the consistency of the models. In order to avoid users corrupting existing models, the post discusses creating a custom UUID resolver to replace the `xmi:id`.

*Formatting Issues* involved problems that developers faces with different file formats of their models and trying to convert between formats or export their model to a specific format. These issues specifically focus on the formatting of the model files, while *File Issues* are related to interacting with a file (*e.g.,* these discussions involved opening files and problems with the folder permissions).

## 4.7 Visualization in Modeling

One of the major benefits of using models to represent systems is the ability to visualize and graphically represent systems. However, this introduces further intricacies not experienced in other development environments. Posts relating to *Visualization* primarily focuses on difficulties creating custom editors for modelers to use when creating their models and questions about the appropriate visualization to represent some knowledge (*e.g.,* state or model attributes) and ensuring the information is displayed (*e.g.,* ensuring the elements are visible).

*GUI Issues* focus on problems when developing GUI-based tools with which users can create and interact with models. Mainly, these discussions were concerned with difficulty when implementing certain interactive functionality on the GUI such as drawing connections or removing links. Similarly, *Customization* discussions involved understand customizing a GUI by providing support for different color palettes and changing fonts.

## 4.8 Development Process

This category consists of discussions that are focused on the *Development Process* of the modelers when creating new models and maintaining existing models. *Integration/Deployment* discussions involve using external tools such as *git* for version control, *maven* for dependency management, and issues integrating with specific hardware and sensors. Additionally, we observed more conceptual questions related to utilizing continuous integration for models. For *Development Activities*, modelers sought help with using existing tools for debugging models as well as performing round-trip engineering. Additionally, modelers sought to *Reuse* existing models as they developed new model.

## 4.9 Model Quality

This category focused on the challenges that modelers faced with making large and robust models, and ensuring these models are correct. The most common issue that modelers faced in this category concerned *Performance* related to their models. The difficulty predominantly related to scalability issues when the tools were loading, saving, or exporting large models:

> "For large Diagrams >1MB DiagramCanonicalEditPolicy class refreshOnActivate method is taking large time for opening GMF Diagram. My Diagram file of size >1MB is taking 45 secs to load,Can some one tell me how to decrease the loading time." [50]

Similar to loading files, we observe discussions related to improving performance when saving a model:

> "Is there any 'smart' implementation, that updates existing serializations instead of rewriting the whole file? Is there anything else I can do to speed up serialization other than splitting the model into several files (I already did that to some extend)?" [51]

In the above example, the modeler finds the save time reasonable when considering the entire file, but aims to reduce the time as the model gets updated. There are also several discussions that are related to running simulations such as the performance of a math solver or refresh rate of generated diagrams.

The concept of *Formal Methods* focused finding tools that support formal evaluation of the models as opposed to performing validation. These discussions sought to identify tools that can apply constraints to models, facilitate the use of custom grammars, or support predicate logic. Conversely, *Model Validation* focused on modelers that wanted to ensure the correctness of the model. We observed discussions about testing specific blocks of models or displaying multiple failure messages for a single validation context.

*Consistency in Models* discussions were related to posts about ensuring models are consistent and synchronized (*e.g.,* updating the model based on external changes) as well discussing issues of inconsistencies in the model (*e.g.,* dangling references after deleting objects). Finally, the last topic was *Comparisons* that represented discussions about comparing models or parts of models, like how to perform model differencing or object comparisons.

## 4.10 Distributed Modeling

This category was the least prevalent among our dataset and involved communicating with remote machines for some modeling activity, or questions regarding security vulnerabilities. The *Security* topic related to the the modeling tools, like inquiring about protections for overflow vulnerabilities. *Networking* focused on the capabilities of connecting models distributed over a network. Lastly, the discussion relating to *Server Issues* involved difficulty with a model connecting and retrieving information from a remote server.

## 5 DISCUSSION

Overall, we observed a diverse set of challenges that modelers experience, and many of these challenges are influenced both by the conceptual understanding as well as limitation or difficulty with existing tools. While our work is similar to prior efforts like the *Grand Challenges* [6] and *MBEBoK* [7], our taxonomy was empirically derived from user-specific challenges; thus, we present additional depth in our taxonomy with respect to this context. We believe this makes our taxonomy as well as our lessons and recommendations complementary to these other efforts.

Using our empirically derived taxonomy (seen in Figure 1), along with a knowledge of the existing tools and support, we formulated a set of suggestions to the modeling community. These suggestions, if adopted, are likely to improve the quality of models produced by novice and experienced modelers alike, and significantly reduce the issues they face. Subsequently, we present our process for deriving these suggestions, along with a clustering of the improvements.

## 5.1 Suggestions for Improvement

After we generated the taxonomy in Figure 1, we discussed each topic to identify potential systematic methods to avoid these types of issues in the future. In other words, we sought to answer *"What can we do about these challenges to better support modelers?"*. Based on the qualitative analysis, experience, and domain knowledge, we distilled strategies that could be used to aid modelers; as seen in the taxonomy, in many cases, multiple strategies could apply to a given topic. The strategies of improvement are represented by the following three categories: *Educational Enhancements*, *Tool Enhancements*, and *Training and Resource Enhancements*. These three categories are represented in the taxonomy by three icons: the mortarboard, the gear, and the presenter, respectively. The presence of an icon for a particular topic indicates that changes of that type could help mitigate these issues.

For each of the three categories, we continued discussion to identify possible techniques that could be employed systematically to induce a reduction in the issues faced. These topics are discussed in the following sections.

### 5.1.1 *Educational Enhancements.*

A significant number of the questions discovered through our analysis could have easily been addressed through a stronger education of MDSE topics, focusing primarily on modeling basics. This is evident with *Foundational Modeling Knowledge* and *Key MDSE Features* being among the more prominent categories of the taxonomy. By focusing on the proper education and training of model-driven approaches, a number of the issues raised in the discussion forums used for this study may become non-existent. In an effort to make educational enhancements, we propose the three enhancements.

**1A: Tools Designed for Teaching:** As is the case with any new technology, misinformation and misunderstanding could be avoided through proper education. Teaching modeling has become the focus of recent research [21], and there exist several tools that focus on the education of MDSE topics (e.g. Umple [52]); however, further work in this topic is needed. There is not any specific forum post that would highlight this suggestion, as it is more of an overarching need within the modeling community.

**1B: Incorporating Modeling into Existing Curricula:** While it would be ideal for every undergraduate computer science curriculum to include education on MDSE topics through the implementation of modeling courses, this is something that is not feasible in all situations. Thus, even through MDSE tools designed for teaching, there are gaps in education that need to be addressed in order to appropriately mitigate the common issues faced by modelers. In order to realize this, it is our recommendation to directly incorporate MDSE concepts into existing curricula in order to provide sufficient understanding to avoid many of the simpler issues posed in the forum posts covered by this study. Many programs consist of courses on programming paradigms, however these courses do not typically include model-driven approaches. These courses serve as a good opportunity for incorporating modeling into existing curricula. As with the previous suggestion, there is no specific forum posts that this would address, however, a broader education in modeling, that reaches a wider audience, would reasonably reduce the barriers to understanding that cause many of the current issues.

**1C: Consolidated and High-Quality Tutorials:** A large number of posts relate to a user asking how to accomplish a certain task, the usage of tool features, and other straight forwards tasks. Each of these could be solved through the creation of high-quality tutorials that demonstrate each of these concepts.

Not only must these tutorials exist, but they ought to be part of a consolidated set of resources. Currently, tutorials exist on blogs, tool sites, textbook sites, and a multitude of other sources, and they often overlap in content, which is an ineffective use of community resources. Similar to the recommendations above regarding the modeling community, we suggest the creation of a single resource for MDSE related tutorials.

### 5.1.2 *Tool Enhancements.*

With the largest single category of issues focusing on *Modeling Tools*, it is clear that the design and implementation of effective modeling tools remains an issue within the modeling community. Based on the frequent posts requesting some of these aspects, we have identified the three recommendations presented below.

**2A: Intuitive Error Messages:** Many of the the tool issues reported in discussion questions surrounded the understandability of, and ability to react to, tool error messages. Many of the current existing tools produce run-time errors that could be avoided through more thorough design, and the errors that are handled often provide cryptic error messages. One forum post asks:

> When I try invoking the Simulink Library Browser by clicking on the icon or by typing 'simulink' on MATLAB command prompt, MATLAB crashes with the following trace." [53]

The post is accompanied with a segmentation violation that contains the stack trace; however, the issue corresponds to an inconsistency in the preferences directory. Thus, it is our recommendation to prioritize efforts in the creation of intuitive error messages to ensure the reduction of barriers for using MDSE practices.

**2B: Well-Defined Features:** Another significant number of tool related posts are of the "How do I...?" variety. This type of post is indicative of features that are not well defined. While this can sometimes be attributed to the documentation, this recommendation aims at the design of the features themselves, and how well defined they are. This could include tasks as simple as renaming tool features with more descriptive names, to tasks as complex as refactoring functionality to group similar features.

**2C: Accessible Tools:** The final area of tool-related recommendations has to do with the accessibility of the tools themselves. A significant number of posts relate to user that face issues on installation, configuration, and other non-modeling topics. With issues that occur outside of MDSE, users may form a negative opinion of modeling due to incorrect assumptions. In tool design, it is not only important to design tools that are functional and well-defined, it is imperative that as many of the external factors as possible are removed from the user control. This can be achieved in a number of ways, including the creation of web-based solutions, self-extracting installers, or platform independent development.

Regardless of the specific approach, it is clear that by removing as many barriers as possible, we are able to give MDSE a fighting chance for adoption. If users face issues installing, running, or using a tool, they will be less likely to continue its use.

### 5.1.3 *Training & Resource Enhancements.*

With *Crowd Support in Modelling* being another large concern raised through the creation of this taxonomy, it is evident that more can be done by members of the community to allow users of all experience levels to become users of MDSE techniques. The following recommendations aim to bolster the modeling community.

**3A: Model Repositories:** A large portion of the forum posts in this area were users asking how to model something specific. Often this request is not necessarily unique or difficult in nature, but it is something that a new user would like to be able to discover to continue development using the MDSE approach. This type of issue could be mitigated through the creation and maintenance of a model repository that contains example models (of all varieties) that can be queried by users. While a single model repository may be a cumbersome undertaking, it would be sufficient if each tool, community, etc. maintained their own so that users of a specific technology or technique could find the resources required.

The creation of model repositories would not only aid in the uptake of MDSE through the provision of examples, but it would also serve a number of purposes in MDSE research, such as those identified by Cabot et. al. [19]. While there are some initial efforts to create model repositories [54], a larger community effort to consolidate these models is needed.

**3B: Comprehensive and Intuitive Documentation:** Another community resource that would help mitigate many of the issues found through this project is enhanced documentation. By improving the quality (and sometimes quantity) of documentation, developers of MDSE tools can help improve the user experience. While many tools do boast extensive documentation it is often not in the most accessible form for novice users to obtain the desired information. Many of the posts discovered by our analysis could be solved by proper access to the right documentation. For example, comprehensive documentation could have addressed the following:

> "I'm trying to count how many times a condition is true inside a FOR loop. I declared an additional variable for the template (FOUND : Integer), and I'm trying to increment it every time the [IF] condition is "true", but the variable increments only the first time, then it gets back to its original value. Basically, if FOUND = 1 at the beginning, at every loop I get 2 in output." [55]

As the post indicates, the issue is due to variables being final in Acceleo. One major issue with current documentation is that it can be cumbersome to navigate or find the desired information. Thus as part of this recommendation, not only are we proposing effort be dedicated to the creation and maintenance of the documentation, but also that tooling be implemented to more accurately search documentation and return relevant results. While not unique to the modeling community, this is a significant barrier to adoption.

**3C: Dedicated Community for Guidance:** One of the themes we noticed throughout the analysis is that questions are often asked in multiple places, in an attempt to obtain the result quicker. This leads to multiple postings that get differing results on differing platforms, which can be a cause of confusion for novice modelers. To this end, it would be beneficial if there were a single dedicated community location for MDSE questions, rather than the distributed nature of the current existing forums. While this is more of a community-driven issue, it would help eliminate potential misinformation or duplication of efforts.

The consolidation of community efforts would only stand to improve the support of the modeling community. Through the creation of a single hub for modeling discussions and the other resources available, we could better serve the community.

#### 5.1.4 *Summary of Recommendations.*

While these recommendations are not exhaustive, they represent potential solutions to the issues addressed in the taxonomy and provide sufficient coverage of the taxonomy issues to demonstrate their potential success. The mapping below indicates the suggestions for improvement that are likely to have the most significant impact on reducing the issues contained within each category:

(1) Modeling Tools: 1A, 2A, 2B, 2C
(2) Foundational Modeling Knowledge: 1A, 1B, 1C
(3) Crowd Support in Modeling: 1C, 3A, 3B, 3C
(4) Errors & Bugs: 2A, 2C
(5) Key MDSE Features: 1A, 1B, 1C, 3B
(6) Representing Data in Modeling: 1B, 2B, 3A, 3B, 3C
(7) Visualization in Modeling: 1B, 2B, 3A, 3B, 3C
(8) Development Process: 1A, 1B, 1C, 2A, 2B, 2C, 3A, 3B, 3C
(9) Model Quality: 2A, 3A, 3C
(10) Distributed Modeling: 3A, 3C

## 5.2 Threats to Validity

*Construct validity:* Since our qualitative study relied on user created posts on the various forums, the information contained within the posts may not be complete or precise. However, as we aim to represent the issues faced by members of the community, these posts are the most direct means of collection. While new tools/approaches have been developed since the data was collected, we believe the impact would be minimal on the taxonomy, since we studied commonly used tools and found many issues permeated across them.

*Internal validity:* To reduce bias in our qualitative analysis, we relied on multiple author agreement for both the open-coding analysis and the card sorting, and we computed the inter-rater agreement showing moderate ($\mathbf{RQ}_2$) or almost perfect agreement ($\mathbf{RQ}_1$). This protocol resembles that of other similar qualitative studies.

*External validity:* The taxonomy relates to posts from a specific set of forums. Other tool forums or Q&A websites may contain issues that are not represented within our taxonomy. We do not assert that our taxonomy is comprehensive beyond the context of our study. Modelers may face difficulties that we did not capture in our study (*e.g.,* problems from other modeling tools). Further, while our random sampling of 400 posts was statistically significant, we recognize that the selection of other random posts, even from the same dataset, may have yielded slightly different results.

## 6 CONCLUSION

Through the examination of several online discussion forums targeting MDSE applications, we were able to construct a taxonomy of the common issues faced by modeling practitioners, both novice and experienced. This taxonomy contains ten top-level categories with 45 topics contained within them, and is visualized in Figure 1. We observed that the majority of the issues focused on the tools themselves, fundamental modeling concepts, and the modeling community. From our taxonomy, we generated a list of nine suggestions to address many of the issues, clustered into three groups. These suggestions target *Education Enhancements*, *Tool Enhancements*, and *Training & Resource Enhancements*, and provide adequate coverage to address the ten top-level categories of the taxonomy.

## REFERENCES

[1] K. Nicholas, Z. E. Bhatti, P. S. Roop, Model-driven development of industrial embedded systems: Challenges faced and lessons learnt, in: Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation, 2012, pp. 1–4.

[2] J.-P. Tolvanen, S. Kelly, Model-driven development challenges and solutions: Experiences with domain-specific modelling in industry, in: 2016 4th International Conference on Model-Driven Engineering and Software Development, 2016, pp. 711–719.

[3] X. Liu, H. Zhong, Mining stackoverflow for program repair, in: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering, 2018, pp. 118–129.

[4] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, M. Lanza, Mining stackoverflow to turn the ide into a self-confident programming prompter, in: Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 102–111.

[5] A. Barua, S. W. Thomas, A. E. Hassan, What are developers talking about? an analysis of topics and trends in stack overflow, Empirical Software Engineering 19 (3) (2014) 619–654.

[6] A. Bucchiarone, J. Cabot, R. F. Paige, A. Pierantonio, Grand challenges in model-driven engineering: an analysis of the state of the research, Software and Systems Modeling (2020) 1–9.

[7] L. Burgueño, F. Ciccozzi, M. Famelis, G. Kappel, L. Lambers, S. Mosser, R. F. Paige, A. Pierantonio, A. Rensink, R. Salay, et al., Contents for a model-based software engineering body of knowledge, Software and systems modeling 18 (6) (2019) 3193–3205.

[8] MathWorks, Simulink https://www.mathworks.com/products/simulink.html.

[9] Eclipse Foundation, EMF https://www.eclipse.org/modeling/emf/.

[10] Eclipse Foundation, GMP/GMF https://www.eclipse.org/modeling/gmp/,.

[11] Eclipse Foundation, Papyrus https://www.eclipse.org/papyrus/.

[12] G. Booch, The unified modeling language user guide, Pearson Education India, 2005.

[13] S. Friedenthal, A. Moore, R. Steiner, A practical guide to SysML: the systems modeling language, Morgan Kaufmann, 2014.

[14] Eclipse Foundation, Papyrus-RT https://www.eclipse.org/papyrus-rt/.

[15] B. Selic, Using uml for modeling complex real-time systems, in: Languages, compilers, and tools for embedded systems, Springer, 1998, pp. 250–260.

[16] N. Hili, J. Dingel, A. Beaulieu, Modelling and code generation for real-time embedded systems with uml-rt and papyrus-rt, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, IEEE, 2017, pp. 509–510.

[17] B. Selic, The Theory and Practice of Modeling Language Design for Model-Based Software Engineering—A Personal Perspective, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 290–321. doi:10.1007/978-3-642-18023-1_7.
URL https://doi.org/10.1007/978-3-642-18023-1_7

[18] J. Hutchinson, J. Whittle, M. Rouncefield, Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure, Science of Computer Programming 89 (2014) 144 – 161, special issue on Success Stories in Model Driven Engineering. doi:https://doi.org/10.1016/j.scico.2013.03.017.
URL http://www.sciencedirect.com/science/article/pii/S0167642313000786

[19] J. Cabot, R. Clarisó, M. Brambilla, S. Gérard, Cognifying model-driven software engineering, in: M. Seidl, S. Zschaler (Eds.), Software Technologies: Applications and Foundations, Springer International Publishing, Cham, 2018, pp. 154–160.

[20] J. G. M. Mengerink, A. Serebrenik, R. R. H. Schiffelers, M. G. J. van den Brand, Automated analyses of model-driven artifacts: Obtaining insights into industrial application of mde, in: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, IWSM Mensura '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 116–121. doi:10.1145/3143434.3143442.
URL https://doi.org/10.1145/3143434.3143442

[21] E. Rapos, M. Stephan, IML: Towards an instructional modeling language, in: Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2019, SCITEPRESS - Science and Technology Publications, 2019, p. 417–425. doi:10.5220/0007485204190427.
URL https://doi.org/10.5220/0007485204190427

[22] S. Baltes, L. Dumani, C. Treude, S. Diehl, Sotorrent: reconstructing and analyzing the evolution of stack overflow posts, in: A. Zaidman, Y. Kamei, E. Hill (Eds.), Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018, ACM, 2018, pp. 319–330. doi:10.1145/3196398.3196430.
URL https://doi.org/10.1145/3196398.3196430

[23] C. Vendome, D. Solano, S. Liñán, M. Linares-Vásquez, Can everyone use my app? an empirical study on accessibility in android apps, in: 2019 IEEE International Conference on Software Maintenance and Evolution, IEEE, 2019, pp. 41–52.

[24] M. Linares-Vásquez, G. Bavota, M. Tufano, K. Moran, M. Di Penta, C. Vendome, C. Bernal-Cárdenas, D. Poshyvanyk, Enabling mutation testing for android apps, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Association for Computing Machinery, New York, NY, USA, 2017, p. 233–244. doi:10.1145/3106237.3106275.
URL https://doi.org/10.1145/3106237.3106275

[25] C. Vendome, D. M. German, M. Di Penta, G. Bavota, M. Linares-Vásquez, D. Poshyvanyk, To distribute or not to distribute? why licensing bugs matter, in: 2018 IEEE/ACM 40th International Conference on Software Engineering, 2018, pp. 268–279.

[26] E. Aghajani, C. Nagy, O. L. Vega-Márquez, M. Linares-Vásquez, L. Moreno, G. Bavota, M. Lanza, Software documentation issues unveiled, in: 2019 IEEE/ACM 41st International Conference on Software Engineering, 2019, pp. 1199–1210. doi:10.1109/ICSE.2019.00122.

[27] A. A. Bangash, H. Sahar, S. Chowdhury, A. W. Wong, A. Hindle, K. Ali, What do developers know about machine learning: A study of ml discussions on stackoverflow, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories, 2019, pp. 260–264.

[28] S. Geremia, G. Bavota, R. Oliveto, M. Lanza, M. Di Penta, Characterizing leveraged stack overflow posts, in: 2019 19th International Working Conference on Source Code Analysis and Manipulation, 2019, pp. 141–151.

[29] Online Appendix, https://doi.org/10.6084/m9.figshare.19245915.

[30] M. B. Miles, A. M. Huberman, Qualitative Data Analysis: A Sourcebook of New Methods, Sage, Beverly Hills, CA, 1984.

[31] Simulink 73512, Bus signals as input to s-function = https://www.mathworks.com/matlabcentral/answers/73512-bus-signals-as-input-to-s-function, note = Accessed: 2022-3-9.

[32] J. L. Fleiss, Psychological Bulletin (5) 378.

[33] J. R. Landis, G. G. Koch, Biometrics (1) 159.

[34] D. Spencer, Card sorting: Designing usable categories, Rosenfeld Media, 2009.

[35] Simulink 126338, Changing the period of sine wave function in simulink?, https://www.mathworks.com/matlabcentral/answers/126338-changing-the-period-of-sine-wave-function-in-simulink, accessed: 2020-05-15.

[36] Simulink 126040, Substracting two oscillating signals with different amplitudes in simulink, https://www.mathworks.com/matlabcentral/answers/126040-substracting-two-oscillating-signals-with-different-amplitudes-in-simulink, accessed: 2020-05-15.

[37] Eclipse 201701, Securing the save process in emf, https://www.eclipse.org/forums/index.php/t/201701/, accessed: 2020-05-15.

[38] Stack Overflow 13020608, Uml stereotypes applied to class inheritance design pattern, https://stackoverflow.com/questions/13020608/uml-stereotypes-applied-to-class-inheritance-design-pattern, accessed: 2020-05-15.

[39] Stack Overflow 9004427, Bidirectional m2m transformations using eclipse emf, https://stackoverflow.com/questions/9004427/bidirectional-m2m-transformations-using-eclipse-emf, accessed: 2020-05-15.

[40] Stack Overflow 1319235, Tangible benefits of use case modelling, https://stackoverflow.com/questions/1319235/tangible-benefits-of-use-case-modelling, accessed: 2020-05-15.

[41] Stack Overflow 262279, Tools (best practices?) for model driven development?, https://stackoverflow.com/questions/262279/tools-best-practices-for-model-driven-development, accessed: 2020-05-15.

[42] Stack Overflow 5461753, https://stackoverflow.com/questions/5461753/modelling-language-for-a-process-transaction, https://stackoverflow.com/questions/5461753/modelling-language-for-a-process-transaction, accessed: 2020-05-15.

[43] Eclipse 628069, ? adding items to jst server module context menu?, https://www.eclipse.org/forums/index.php/t/628069, accessed: 2020-05-15.

[44] Eclipse 124459, Modifying standard drag'n drop behavior of the emf editor, https://www.eclipse.org/forums/index.php/t/124459/, accessed: 2020-05-15.

[45] Eclipse 135028, [cdo] "not a top level package" exception, https://www.eclipse.org/forums/index.php/t/135028/, accessed: 2020-05-15.

[46] Eclipse 126668, Ecore validate error, https://www.eclipse.org/forums/index.php/t/126668/, accessed: 2020-05-15.

[47] Eclipse 124918, Extendability of the emf code generator, https://www.eclipse.org/forums/index.php/t/124918/, accessed: 2020-05-15.

[48] Simulink 428318, How to show requirements in generated code for traceability?, https://www.mathworks.com/matlabcentral/answers/428318-how-to-show-requirements-in-generated-code-for-traceability, accessed: 2020-05-15.

[49] Stack Overflow 4320515, How can the output of a simulink block be fed back as an input, https://stackoverflow.com/questions/4320515/how-can-the-output-of-a-simulink-block-be-fed-back-as-an-input, accessed: 2020-05-15.

[50] Eclipse 287674, Gmf loading time, https://www.eclipse.org/forums/index.php/t/287674/, accessed: 2020-05-15.

[51] Eclipse 134504, Resource save performance, https://www.eclipse.org/forums/index.php/t/134504/, accessed: 2020-05-15.

[52] T. C. Lethbridge, Teaching modeling using umple: Principles for the development of an effective tool, in: 2014 IEEE 27th Conference on Software Engineering Education and Training, IEEE, 2014, pp. 23–28.

[53] Simulink 98438, Why does matlab crash when i launch simulink 7.1 (r2008a)?, https://www.mathworks.com/matlabcentral/answers/98438-why-does-matlab-crash-when-i-launch-simulink-7-1-r2008a, accessed: 2020-05-15.

[54] S. A. Chowdhury, L. S. Varghese, S. Mohian, T. T. Johnson, C. Csallner, A curated corpus of simulink models for model-based empirical studies, in: 2018 IEEE/ACM

4th International Workshop on Software Engineering for Smart Cyber-Physical Systems, IEEE, 2018, pp. 45–48.

[55] Stack Overflow 11955410, Increment a variable inside a for loop, https://stackoverflow.com/questions/11955410/increment-a-variable-inside-a-for-loop, accessed: 2020-05-15.