What Do We Need from Modelling Tools for Teaching? A Survey of the Community of Higher-Education Modelling Teachers

Steffen Zschaler Timothy Lethbridge Antonio Bucchiarone
Federico Bonetti Reyhaneh Kalantari

September 22, 2025

Abstract

We report on an international survey of 59 higher-education teachers of software modelling and model-driven engineering regarding the modelling languages and tools they use, the pedagogic approaches they employ, as well as their desires for features and properties in improved modelling tools for teaching. The survey revealed divergent opinions regarding satisfaction with existing tools, with preferred teaching methods, and with currently-used modelling tools. But there was agreement on the need for better user experience in tools, more powerful capabilities, better documentation, and comprehensive libraries of examples. There was a dichotomy between a large majority who want to teach modelling using the core UML-based diagram types, versus smaller groups who want to focus either on formal languages or model transformation. The number of modeling tools in use is large, but educators are not aware of most tools, indicating a very fragmented market. We conclude that there is a need for the community to work towards a smaller set of usable and useful tools. We believe our analysis will inform the development of better tools and pedagogies for teaching modelling and model-driven engineering.

1 Introduction

Research shows that modelling greatly benefits software developers if they apply it appropriately [59]. It enables designers and engineers to efficiently explore the design space, and provides stakeholders with suitable representations of the system. Models hence help all involved parties understand, analyse, and design complex (software) systems.

Although modelling is taught in most post-secondary education programs, industrial use is lagging [32]. To ensure they adopt modelling, industrial users need to be convinced that their investment in modelling will have a positive return on investment, but if students entering the workforce are already not convinced, then their modelling education is not likely to be translated into adoption of modelling in the workforce. Therefore, to make modelling effective, both in educational and industrial practice, appropriate tools must be available to educators. These must be tools that both help the student learn modelling, and also help convince the student of the benefits of modelling.

However, most academic and industrial modelling tools currently available are not ideal for teaching modelling [13, 20, 34].

Educators choosing tools for teaching can elect to choose those used in industry. These, however have several issues: They tend to be complex and have prohibitively high pricing. They lack out-of-the-box leearner-oriented user experience, and support for teaching-related features, such as online collaboration, automated grading, and pedagogical feedback adapted to the student's level. They also evolve rapidly, and generally have short lifespans before being declared obsolete. This renders the effort of producing teaching material for them wasted.

Another choice educators may make is tools specifically designed by the educator community for modelling in the academic context. However, many of these tools suffer from low maturity and robustness, and are difficult to install. They also tend not to receive adequate maintenance due to accumulated technical debt and the lack of reward for academics who would need to invest considerable effort in a tool's development over many years. Since modelling is a large and complex task, it is also difficult for academics to produce sufficiently powerful tools that convince students that modelling can be used for serious software development.

Motivated by this situation, we organised a one-day working session on modelling tools for teaching (MTT) at the 26th International Conference on Model-Driven Engineering Languages and Systems[1]. The outcomes of the discussion during the workshop are reported in detail in [34]. In the present paper we report comprehensive results from a follow-up survey of the international community of academics teaching modelling and model-driven engineering. The survey asked about their approaches to teaching modelling and their requirements for modelling tools for teaching.

The research questions we sought to answer when designing the survey were:

- RQ1: To what extent are those teaching modelling satisfied with the current state of modelling tools for teaching?
- RQ2: What are the features and properties that those teaching modelling would like to see in tools?
- RQ3: Which modelling languages do those teaching modelling most wish to teach?
- RQ4: What teaching practices should modelling tools for teaching support?

RQ1 seeks to validate our base assumption that there is dissatisfaction with the state of the art, and quantify the level of this dissatisfaction. RQ2 is the core question we are seeking to answer so that tool designers can prioritize features and properties. RQ3 and RQ4 are aimed at helping tool developers know what types of models and teaching practices should be supported by tools.

While some results of this survey were available at the time the earlier paper [34] went into publication (so that that paper includes some initial data from the survey), the focus of [34] was on reporting qualitatively the outcomes of the workshop discussion.

Thus, our paper makes the following contributions:

1. We present a survey of the community of higher-education modelling teachers and their perspectives on modelling tools for teaching.
2. We describe detailed findings from the survey, including thematic analysis of answers to a full-text question and quantitative analysis of answers for individual questions as well as across questions.

[1] `https://modellingtoolsforteaching.github.io/`

3. We discuss our findings and draw conclusions for the further development of modelling tools for teaching.

Our paper, thus, provides quantitative and qualitative data to support the qualitative discussions in [34].

This paper is organized as follows: Section 2 discusses the methodology, including the survey design and the sampling method. Section 3 discusses demographics and background of the participants. We then give three sections covering our findings: (i) Section 4 reports descriptive statistics about the answers to individual questions where users rated their attitudes regarding tool features or qualities, as well as the importance they attribute to various languages and teaching techniques; and (ii) Section 5 reports on a thematic analysis of the open-ended question that appeared near the beginning of the survey (Q5). Section 6 synthesizes our findings, Sect. 7 discusses related work, Sect. 8 summarizes several potential threats to validity, and Sect. 9 concludes the paper.

2 Methodology

The work presented originated at a workshop at MODELS 2023. The purpose of the workshop was to engage in productive discussions regarding the requirements and necessary infrastructure for MTTs. Targeted invitations were sent to research groups actively involved in developing tools used for teaching computer science in undergraduate or graduate classes. During the workshop, the need for a survey of the wider community became evident. In the following, we describe the structure of the survey and the strategy for recruiting respondents.

2.1 Survey design

At the end of the workshop, we shared a pilot survey with workshop participants, which had been created by one of the co-authors during the workshop. This provided initial insights and, more importantly, helped us refine the final survey, data from which we present in this paper. Despite being produced within a half-hour timeframe, the survey yielded valuable insights:

- Participants expressed a keen interest in teaching using class diagrams and state machines, alongside other model types albeit with lesser importance.
- Important attributes for modelling technology in teaching included being free, having excellent user experience, having multi-platform compatibility (including having a web-based version), having a comprehensive user manual, having a library of examples, being reliable, having analysis capabilities with feedback, having fast response time, being capable of code generation, and having a textual interface available.

After the workshop, we reflected on the pilot survey and revised and extended the set of questions to produce a full survey, which we shared with the international community of higher-education modelling teachers. Ethics approval for the full survey was obtained at King's College London in January 2024 (reference number MRA-23/24-41234).

Table 1 provides a high-level summary of the questions. The table also indicates which of our top-level research questions are addressed by each survey question, and which section of the paper discusses the results.

Some of the principles we used to guide the design of the survey included:

Table 1: Summary of Survey Questions about Modelling Tools for Teaching (MTT)

Question	**Theme**	**Purpose**	**Findings**
1	Confirmation of consent and teaching experience	In-/Exclusion	
2	Number of years respondents have been teaching	Demographic	Sect. 3
3	Extent of to which they teach software modelling	Demographic	Sect. 3
4	Satisfaction with current modelling tools for teaching	RQ1	Sect. 3
5	Improvement areas for MTT (open ended)	RQ2	Sect. 5
6	Importance of teaching specific modelling languages at the undergraduate level	RQ3	Sect. 4.1
7–13	50 properties or features of MTT to be rated by importance	RQ2	Sect. 4.2
7	General user experience (UX) properties and features of MTT (9 items)		
8	Editing and collaboration features of MTT (5 items)		
9	Language and language-manipulation features of MTT (10 items)		
10	Analysis features of MTT (5 items)		
11	Transformation features of MTT (7 items)		
12	Platform capabilities for MTT (9 items)		
13	Business and economic properties of an MTT (5 items)		
14	Languages respondents would like the tool to generate	RQ2	Sect. 4.3
15	Teaching practices an MTT should support (13 items)	RQ4	Sect. 4.4
16	Modelling tools currently used (16 items)	RQ1 & RQ2	Sect. 4.5
17–19	Remaining questions for demographic analysis	Demographic	Sect. 3
17	Academic rank		
18	Continent		
19	Gender		

- Gathering a limited amount of demographic data (Q2–3 and 17–19) to help determine whether different sub-communities have different needs.
- Designing most questions (Q2–4, 6–13 and 15–16) to use rating scales with carefully considered extreme values, and a neutral point if appropriate. For example, when asking about satisfaction (Q4) choices ranged from 'very satisfied' to 'very dissatisfied'.
- For questions relating to perceived benefit of languages, tool properties or tool features (Q6–13), we not only included values ranging from 'not needed' to 'critical', but also included a negative value of 'harmful' on the scale to enable the respondents to suggest avoidance of features that might result in excess complexity.
- Adding a general open-ended question (Q5) prior to detailed rating-scale questions, to ensure that the respondent would be able to express themselves freely before they are provided with the categories the survey designers have chosen.
- Gathering categories for the questions from many sources, including the original MTT workshop [34], input from the attendees at the workshop, and items identified in two papers [2, 31].
- Adding open-ended 'other' options where there is any possibility of a respondent wanting to add additional categories or more information.

The questions and answer options are openly available from the King's College London research data repository, KORDS[2].

2.2 Sampling methods

Targeted sampling methods were employed for distributing the survey using the following steps:

1. Each attendee of the MTT workshop at MODELS'23 was tasked with ensuring that the survey reached relevant individuals within their institution, including themselves.
2. Additionally, participants were encouraged to reach out to colleagues in other institutions within their geographic region and beyond.
3. The survey was also disseminated through the LinkedIn and X accounts of several authors.

By the end of February 2024, we had gathered 59 valid responses. Only one response was excluded due to incomplete answers to the questions.

2.3 Quantitative analysis

To analyse the data from questions with rating scales (Q6-13, Q15 and Q16), we converted the responses from the rating scale to numerical values (the values used will be presented with the findings). We designed the points on the scale for each question to allow respondent to express reasonably-evenly-spaced levels of positivity about the languages, features, properties, tools and teaching techniues. We initially considered using strictly non-negative values from not needed to critical, but after reflection we decided to provide one negative point on scales for questions 6-13

[2] `https://doi.org/10.18742/25429270`

that would allow respondents to indicate that they think teaching a certain language, or having a certain feature available, might actually be harmful.

In our analysis we used basic descriptive statistics to allow ranking of responses such as languages and features. There has been some controversy about whether it is legitimate to compute the mean and standard deviation of such an ordinal scale. However, we follow the advice of Harpe [29] who analyses the measurement theoretical arguments, and concludes that it is permissible to compute such statistics when comparing many items (languages, features and so on in our case) using a scale with at least 5 points. However, we also provide the median and mode.

We also performed some differential analysis in two ways. For some questions we performed a series of T tests to see if there were any differences in the results from different sub-groups. When using multiple T-tests like this it is necessary to correct for the chance of false positives, so we performed Holm-Bonferroni correction [30]. We also use ANOVA analysis to discover if there are any significant variations in responses to each question among certain sub-groups; when variation is uncovered we use Tukey tests [1] to determine which differences are significant.

2.4 Qualitative analysis

To gain insight into free comments (coming from answers to Question 5), Thematic Analysis [10, 57] is employed. In qualitative research, this is a widely accepted method [60, 28, 16] for identifying, analyzing, and reporting emerging themes in data. In particular, it helps in capturing latent patterns across diverse participant narratives and enhancing data transparency through structured coding and categorization. Given the open nature of Question 5, this method was employed to condense data that touched upon different modelling tool features and areas of improvement. This approach requires, in order: i) familiarizing with the data; ii) initial coding; iii) generating themes; iv) assessing the validity and reliability of themes; v) defining and naming themes; vi) interpreting and reporting the results. The data was initially prepared by following a topic summary approach. Then, a thematic coding was carried out. In total, 4 authors of the present manuscript contributed to outlining and validating the themes and their reliability, across multiple sessions.

3 Background and demographics of the respondents

We asked the following questions about the participant's general background and demographics:

- **Teaching Experience and Duration (Q2):** Respondents to the survey broadly have substantial experience teaching modelling: 30% of respondents said they have been teaching modelling for 20 or more years. A further 23.3% have taught modelling for 10–19 years and 25% for 5–10 years. Only two respondents said that they "do not teach modelling (yet)".

- **Extent of Teaching Software Modelling (Q3):** The vast majority of respondents teach software modelling at a level where there expert judgment should prove valuable, with 39% heavily involved, 54% moderately involved, and only 7% teaching modelling minimally. There proved to be few significant differences in the answers to other questions between those who teach modeling heavily and those who teach it moderately or less. Later in this paper, we will report on a few exceptions to this.

- **Satisfaction with Current Technology (Q4):** We asked about satisfaction because an initial premise of our work is that existing MTTs do not provide the user experience or features

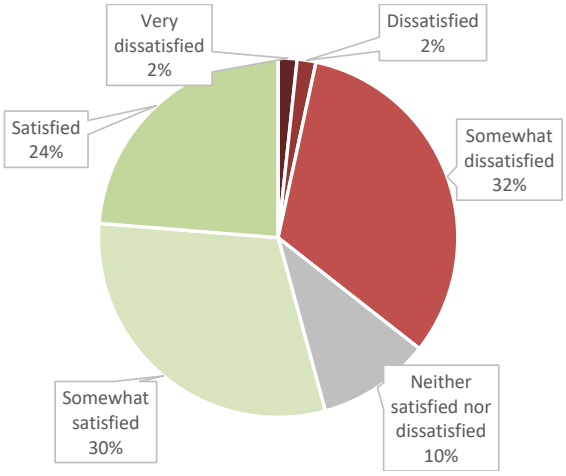


Figure 1: Overall satisfaction of respondents with existing modelling tools for teaching (Q4). No respondent used the available option "Very satisfied".

we believe modeling teachers and students would like. However, it was important to validate this in the survey.

Mixed satisfaction levels were observed (cf. Fig. 1) among respondents regarding the currently available MTTs, with no respondents indicating being very satisfied, 54% expressing satisfaction to some extent, and 36% indicating some level of dissatisfaction.

This validates, to a reasonable extent, why the remaining questions in the survey are important. We were a little surprised, however, that the percentage of dissatisfaction was not higher. Later in the paper we will point out the few questions where those who were satisfied gave significantly different responses from those who were dissatisfied. For most questions, however, there were no significant differences between these groups. This could suggest that the answers to Q4 reflect more on varying thresholds of satisfaction between respondents rather than capturing the true quality of available modelling tools for teaching.

- **Other Demographics (Q17–19):** Participants were roughly evenly divided among full professors, associate professors and assistant professors. Responses were 74% from Europe; 16% from North America, and 10% from elsewhere. Respondents were 75% male and 25% female.

4 Descriptive statistics

We begin by reporting descriptive statistics for the answers to all the questions where respondents rated items on a scale. We followed the data analysis method described in Section 2.3.

4.1 Modelling languages

Question 6 asked "To what extent do you think each of the following modeling languages should be taught to undergraduates as a required part of a computer science or software engineering program?" We listed 17 different modelling languages, ranging from 'class models' to 'Formal methods models (e.g. Alloy, NuXMV) other than OCL' ('OCL' was a separate alternative). . Many of the languages we specified are actually families of languages. For example, there are several flavours of Petri Nets and goal models, and many types of textual requirements or process modeling languages. However, listing every individual language would have been unreasonable, so we chose to list language clusters and provide an option for respondents to also add their own languages. We still refer to the clusters as languages to simplify communication.

Respondents were asked to rate each language using the following scale, where the numbers in parentheses are the values assigned to each response for the subsequent calculations, as discussed in Section 2.3:

- Harmful to include in the required curriculum (obsolete or risks adding undue complexity/clutter) (-1)
- Not needed but would not be harmful (0)
- May be useful for students but not every program should need to include it (1)
- Needed: Basic capability should be present, but it does not have to be taught extensively (2)
- Critical: Should be taught in a full-featured manner, so students have high competence (3)

Figure 2 and Table 2 provide an overview of the scores given to each of the 17 modelling languages, ordered from most to least important to teach. Figure 2 presents the results graphically, where the green bar segments highlight where respondants think that the language should be taught, and the red/pink bar segments suggest that the language should not be taught. Table 2 presents the same data numerically. It is notable that the median and mode decrease along with the mean, as would be expected; the mean simply provides a finer-grained ranking than the median or mode.[3]

Class models are clearly considered the most critical type of modelling language by respondents with only 1 respondent suggesting they would be 'harmful' and all other respondents considering them either 'needed' or 'critical'. The next most important languages are state machines, sequence diagrams, use case models, ER diagrams, and activity diagrams. Formal methods languages and Petri nets are considered the least important by our respondents.

Respondents were also able to provide a list of other modelling languages they considered important and that were not included in our list already. They were able to provide a free-text answer listing arbitrarily many additional languages. 16 respondents made use of this option. We grouped their responses by type of modelling language, resulting in 6 additional modelling languages. Figure 3 shows the distribution of responses. Process modelling languages (like BPMN [19]) were a clear omission from our list. Modelling languages for software architecture and software deployment were also requested frequently.

We performed differential analysis on the data from Question 6 to determine whether there are any differences in the responses between those who had indicated they were somewhat satisfied or satisfied with current modelling tools, as per Question 4, as compared to those who were somewhat dissatisfied to very dissatisfied. We used T tests to compare the two sub-samples. The satisfied group rated the following four languages more highly than the dissatisfied group: Entity

[3]With the exception of 'entity relationship diagrams' and 'component diagrams', where the mode is slightly higher than for the previous entry.

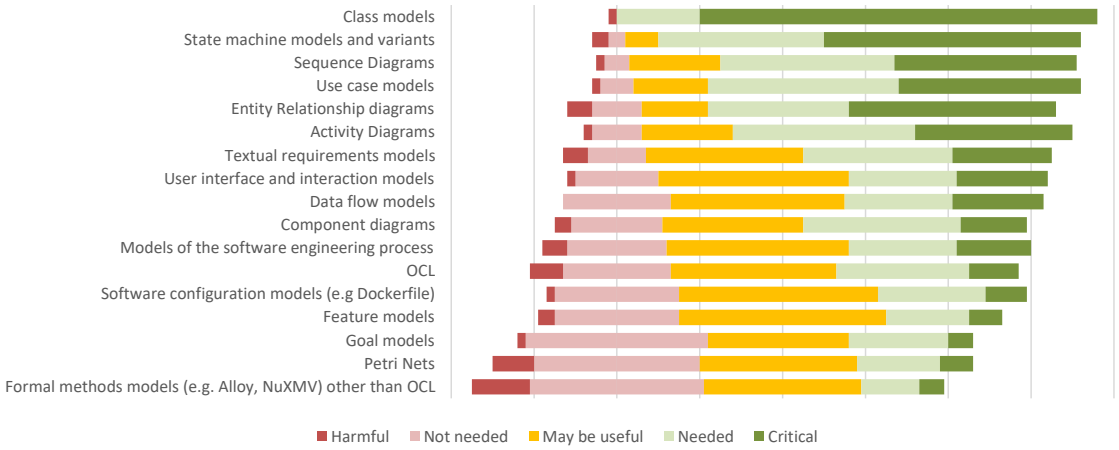


Figure 2: Responses to Question 6: "To what extent do you think each of the following modeling languages should be taught to undergraduates as a required part of a computer science or software engineering program?" - ranked by average score. X-axis shows percentage of respondents, centred around the middle of the "not needed" response.

Relationship Diagrams, Goal Models, Petri Nets and User Interface Models. Each of these T-tests would have been statistically significant on its own. However, after Hom-Bonferroni correction (as discussed in Section 2.3) none of the differences proved statistically significant. There were no differences between those who teach modeling heavily and those who teach moderately or lightly.

4.2 Modelling tool features and properties

Questions 7 to 13 asked about the importance of 50 different features or properties of modelling tools in a teaching context. They all shared a scale of possible responses, where once again we assigned the numerical values shown in parentheses:

- Harmful: would add undue complexity (-1)
- Not needed / not something I would judge a tool by (0)
- Good to have at a basic level (1)
- Important to have at a moderate level (2)
- Essential to have reasonably good capability for this (3)
- Critical: Must be as extensive and good as possible (4)

We report responses across all 7 questions, to make relative importance of features in different categories visible. As with Question 6, we again compute the means to allow fine-grained rankings; and once again we decided to have a single 'harmful' value in addition to positive values.

The results are summarised in Figure 4 and Table 3. User experience and cost aspects dominate the top-rated features, with example libraries, code generation, and the ability to provide feedback on models also very important. On the other hand, AI integration and support for mobile platforms are not considered important.

Table 2: Descriptive statistics regarding perceived desirability of teaching various modeling languages from Question 6, corresponding to Fig. 2. Values: Harmful = -1; Not needed = 0 to Critical = 3.

Modeling language	**Mean**	**Std. Deviation**	**Median**	**Mode**
Class models	2.76	0.63	3	3
State machine models and variants	2.29	0.98	3	3
Sequence Diagrams	2.03	0.97	2	3
Use case models	2.03	0.98	2	2
Entity Relationship diagrams	1.93	1.20	2	3
Activity Diagrams	1.88	1.04	2	2
Textual requirements models	1.49	1.10	2	1
User interface and interaction models	1.40	1.04	1	1
Data flow models	1.38	1.04	1	1
Component diagrams	1.35	1.06	1	2
Models of the software engineering process	1.22	1.10	1	1
OCL	1.12	1.08	1	1
Software configuration models (e.g Dockerfile)	1.10	0.95	1	1
Feature models	0.98	0.94	1	1
Goal models	0.89	0.96	1	0
Petri Nets	0.79	1.06	1	0
Formal methods models (e.g. Alloy, NuXMV) other than OCL	0.61	1.03	1	0

We analyzed the responses to Questions 7-13 by comparing the subgroups who were satisfied with existing tools, vs those who are dissatisfied, following the same method as for Question 6. The following features or properties were individually rated significantly higher among the satisfied group vs. the dissatisfied group: Access control, traceability, language engineering, formal methods capability, plagiarism detection, creation of models by AI, generation of natural language description of models, command-line access to a tool, and ability of the tool to be used commercially. None of these differences ramained significant, however, following Holm-Bonferroni correction.

We also analyzed the responses comparing those who teach modeling heavily vs. those who teach it moderately or less. Heavy users rated student collaboration, model transformation, and tools being free as significantly more important than other respondents. Heavy users rated textual interfaces as significantly less important than other users. But, as before, the statistical significance disappeared following Holm-Bonferroni correction.

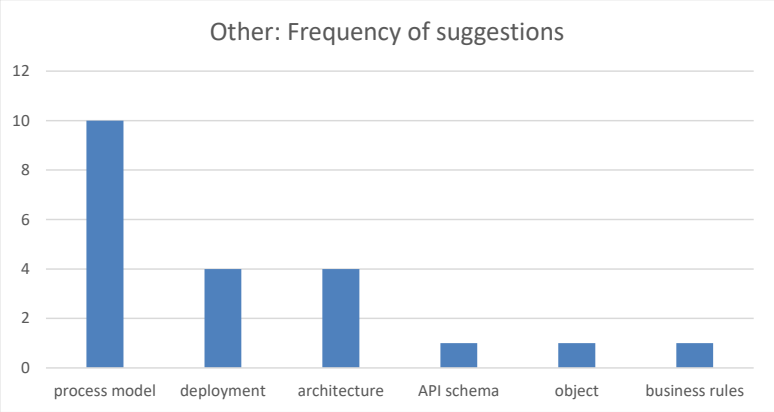


Figure 3: Additional modelling languages suggested in response to Question 6, grouped by type of language.

Table 3: Data for Questions 7 to 13 regarding desirability of properties or features, sorted by descending mean (with rare exceptions this corresponds to a sorting by median or mode). Values: Harmful = -1; Not needed = 0 to Critical = 4.

Feature or Property	**Question**	**Mean**	**Std. Dev**	**% essential / critical**	**Median**	**Mode**
Ability for students to save and load models	Q7 – UX	3.68	0.78	96.6	4	4
The tool is free to use for educators and students	Q13 – Business	3.49	1.01	86.4	4	4
Ability to run on all main laptop platforms: Windows + Mac + Linux	Q12 – Platform	3.34	0.94	83.1	4	4
General ease of use: Complies with guidelines for simplicity, good labels/icons, good feedback, undo capability, etc.	Q7 – UX	3.14	1.06	81.4	3	4
Reliability	Q7 – UX	3.00	1.14	76.3	3	4
Ability to present comprehensive error or warning messages about problems with model syntax or semantics	Q10 – Analysis	2.92	1.16	69.5	3	4

Table 3: (continued).

Feature or Property	Question	Mean	Std. Dev	% essential / critical	Median	Mode
Online user manual (an advanced version would have examples that can be directly loaded into the tool)	Q7 – UX	2.86	1.20	67.8	3	4
Presence of a library/repository of examples	Q7 – UX	2.61	1.27	62.7	3	3
Fast response time	Q7 – UX	2.59	1.16	59.3	3	3
Code generation from models	Q11 – Transformation	2.59	1.21	52.5	3	4
The tool is free to use for all users	Q13 – Business	2.59	1.40	58.6	3	4
Ability to have multiple diagrams showing different subsets of the model elements	Q9 – Language	2.54	1.24	61.0	3	3
Version control of models, such as integration with Git tools	Q9 – Language	2.51	1.25	50.8	3	4
Ability to run as a website (hence zero-footprint, and no installation)	Q12 – Platform	2.49	1.30	50.8	3	4
Ability to annotate, or comment on models	Q8 – Editing	2.46	1.28	52.5	3	3
Ability to hide unneeded features (e.g. to reduce complexity for students)	Q7 – UX	2.37	1.20	47.5	2	3
The tool is open source	Q13 – Business	2.37	1.35	52.5	3	3
Execution of models directly in the modeling tool (interpretation, simulation, etc.)	Q9 – Language	2.34	1.27	50.8	3	3
Strict adherence to formal standards for the supported modeling languages	Q9 – Language	2.31	1.39	45.8	2	2
Incorporation of traceability between models, or between models and generated artifacts	Q9 – Language	2.24	1.30	50.8	3	3

Table 3: (continued).

Feature or Property	Question	Mean	Std. Dev	% essential / critical	Median	Mode
Collaborative editing by multiple students	Q8 – Editing	2.12	1.35	44.1	2	3
Generation of instance models (examples of a model)	Q11 – Transformation	2.05	1.41	44.1	2	3
Ability to compare models (compute differences) and merge models	Q9 – Language	2.03	1.29	42.4	2	3
Document generation from models	Q11 – Transformation	2.00	1.34	41.4	2	3
Access control (i.e. student models can be kept private from other students)	Q7 – UX	1.97	1.46	45.8	2	3
Ability to do language engineering (creating domain-specific-models - DSLs)	Q9 – Language	1.95	1.51	40.7	2	4
Syntax-directed editing	Q8 – Editing	1.92	1.24	30.5	2	2
Collaboration by teachers in model example and curriculum preparation	Q8 – Editing	1.85	1.37	40.7	2	3
Textual interface (available, even if graphical is the main way of modeling)	Q9 – Language	1.80	1.31	33.9	2	2
Ability to embed code in models and/or vice- versa	Q9 – Language	1.78	1.37	35.6	2	3
The tool has a long history of use by many people	Q13 – Business	1.76	1.22	27.1	2	2
Generation of recommendations for model correction/improvement	Q10 – Analysis	1.75	1.12	25.4	2	2
Customizability (e.g. ability to add plugins, extensions, macros, new languages, etc.)	Q7 – UX	1.73	1.45	39.0	2	3

Table 3: (continued).

Feature or Property	Question	Mean	Std. Dev	% essential / critical	Median	Mode
Integration with other modeling tools so they can be used together, e.g. sharing or transferring models	Q12 – Platform	1.69	1.37	25.4	2	2
Reverse engineering of code to models	Q11 – Transformation	1.68	1.37	27.1	1	1
Transformation to other modeling languages	Q11 – Transformation	1.66	1.41	33.9	2	3
Separation of concerns mechanisms such as aspects, traits, mixins (for product lines or conditional code generation)	Q9 – Language	1.59	1.35	28.8	2	0
Formal mathematical verification (such as 'model checking')	Q10 – Analysis	1.59	1.34	28.8	1	0
The tool is capable of being used commercially (i.e. it is not uniquely tailored for educational use)	Q13 – Business	1.47	1.29	27.1	2	2
Automated grading of student assignments	Q10 – Analysis	1.42	1.33	22.0	1	0
Generation of natural-language descriptions of model elements	Q11 – Transformation	1.37	1.30	18.6	1	1
Ability to run as a VS-Code plugin	Q12 – Platform	1.37	1.44	23.7	1	1
Integrated plagiarism detection	Q10 – Analysis	1.32	1.36	20.3	1	1
Availability of the tool as a Docker image	Q12 – Platform	1.28	1.23	20.7	1	1
Integration of gamification in modeling tools	Q8 – Editing	1.14	1.31	18.6	1	0
Ability to run as an Eclipse plugin	Q12 – Platform	1.05	1.43	18.6	1	0
Ability to run on the commandline	Q12 – Platform	1.05	1.25	15.3	1	0

Table 3: (continued).

Feature or Property	Question	Mean	Std. Dev	% essential / critical	Median	Mode
Creation of models by artificial intelligence (LLMs etc.) e.g. using AI to transform textual requirements into a design model	Q11 – Transformation	0.83	1.43	15.3	0	0
Availability of the tool as an app in app stores	Q12 – Platform	0.73	1.19	11.9	0	0
Ability to be used on a small mobile device	Q12 – Platform	0.36	1.03	3.4	0	0

4.3 Code generation targets

Question 14 asked "If you would like code generation (model to text) to be part of your teaching process, which languages are the most important for a modeling tool to be able to generate?" Respondents were able to choose multiple answers from a predefined list of languages. They were also able to add additional languages via a free-text field and to state that they do not include code generation as part of their teaching. Figure 5 shows the distribution of choices among the languages listed, excluding responses that indicated that the responded did not currently teach code generation. Java dominates the field, with Python following at some distance. Beyond those two languages, there is a significant drop-off with SQL and C++ leading the middle field.

Notably, 34 respondents (58%) provided additional languages we had not listed explicitly. Figure 6 summarises these responses. JavaScript stands out as the language most often explicitly mentioned by respondents. Its frequency is still lower than most of the languages explicitly listed, however, it is difficult to know whether the language would have scored higher if it had been listed as an explicit option.

4.4 Teaching practices

Question 15 asked "Which of the following teaching practices are important to you when teaching modeling to undergraduates?" Respondents were able to rank teaching practices on the following scale:

- Do not use
- Use a little
- Use moderately
- Use extensively

Figure 7 summarises the responses. Teachers focus on small models, project / problem-based pedagogies, live modelling, and small modifications to models (through analysis and editing of

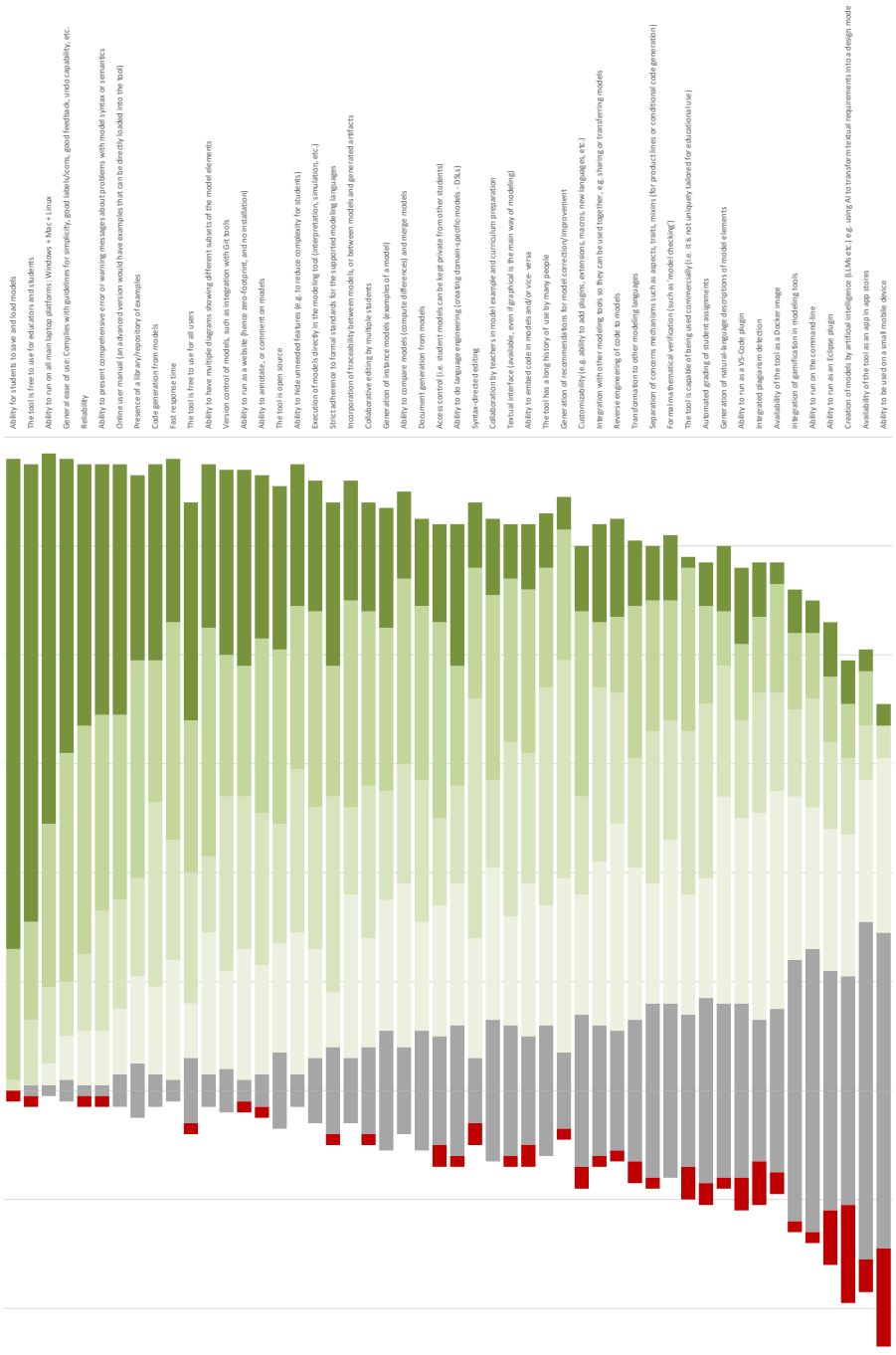


Figure 4: Visualisation of responses across Questions 7–13 asking respondents to rate tool features and properties. Bottom (dark red) = harmful; next above in grey = not needed; top (dark green) = critical. Statistics are in Table 3. X-axis shows percentage of respondents, centred around the middle of the "not needed" response.

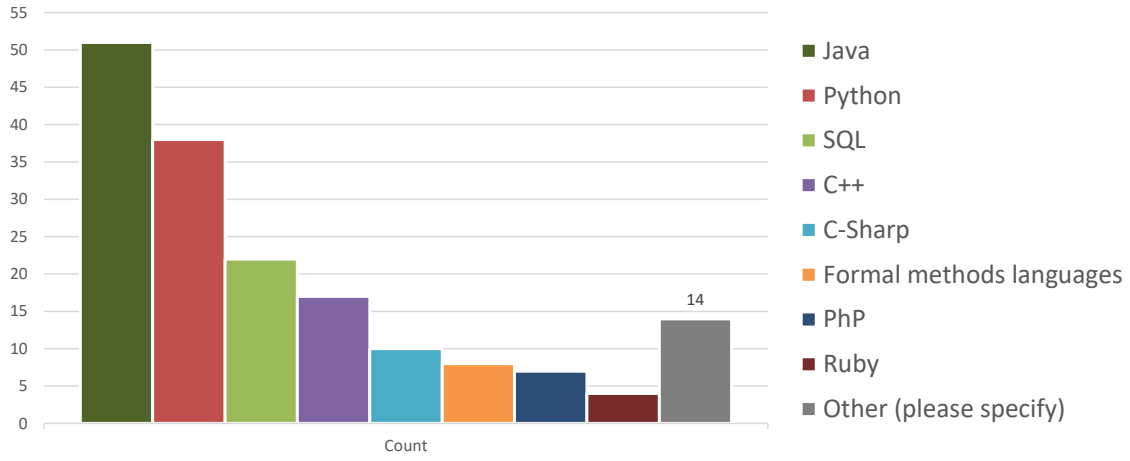


Figure 5: Frequency of code-generation target languages chosen from the list provided in response to Question 14.

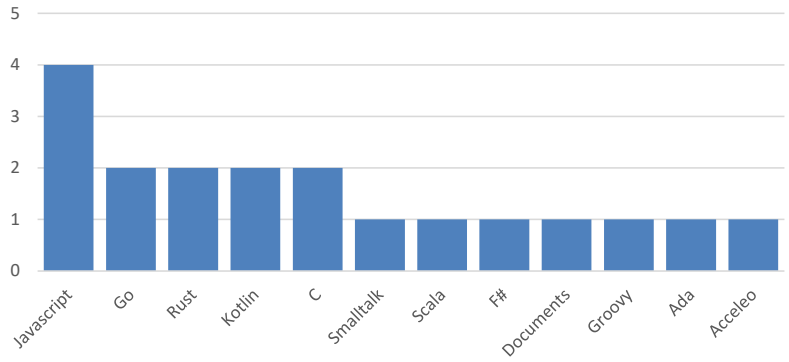


Figure 6: Frequency of code-generation target languages mentioned by respondents in response to 'Other (please specify)'.

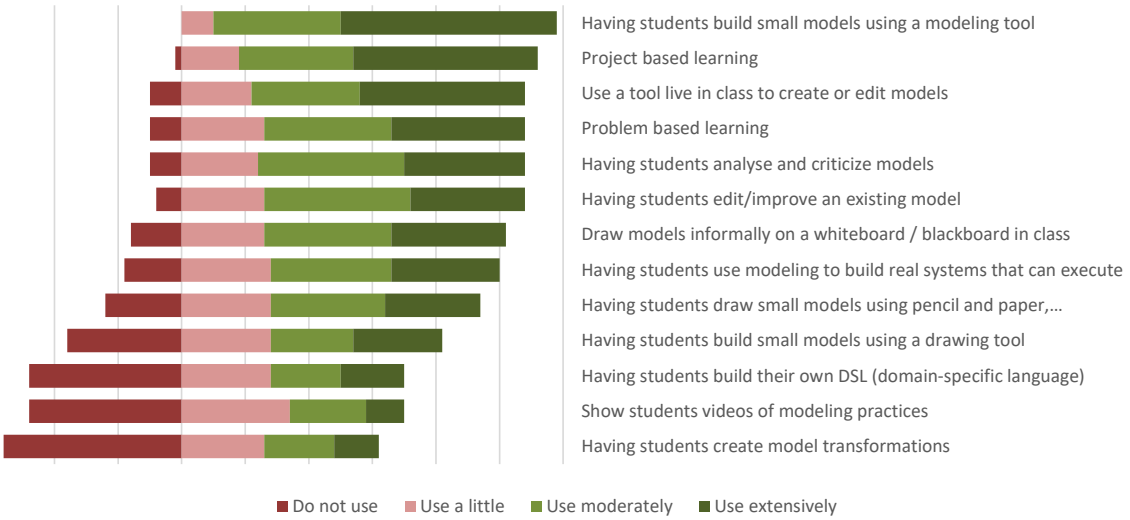


Figure 7: Responses to Question 15 on teaching practices. X-axis shows percentage of respondents, centred to separate "do not use" from the rest of the responses.

existing models). The use of modelling tools is seen as more important than the use of drawing tools. The development of domain-specific modelling languages and model transformation is comparatively rare, as is having students watch videos of modelling practice (as opposed to live modelling).

We conducted ANOVA tests to identify differences between educators who use different teaching methods for modeling (ranging from "do not use" to "use extensively") and their responses to the following categories of modeling tool features:

- **General User Experience (Q7):** How important are user experience attributes in a modeling tool for teaching?
- **Editing and Collaboration Features (Q8):** How important are features like annotation and collaboration for a modeling tool?
- **Language and Manipulation Features (Q9):** How important are features related to language and language engineering?
- **Analysis Features (Q10):** How important are the analytical capabilities of the tool?
- **Transformation Features (Q11):** How important are transformation-related features, such as model-to-model or model-to-code transformations?
- **Platform Capabilities (Q12):** How important are platform features, like cross-platform compatibility (Windows, Mac, Linux)?
- **Business or Economic Issues (Q13):** How important are business-related aspects, such as tool cost and licensing?

For the most part we encountered little difference in the preferences of groups of educators. However, the following describes some of the variations that were notable.

We consider as notable only cases where a particular group of educators rated a feature or property at least 1.5 levels higher or lower on average in our semantic difference scale, and where

this difference is both statistically significant, and interesting.

For statistical significance, we only included cases where an ANOVA followed by the Tukey test resulted in p <0.05, when comparing the responses of groups of educators.

For effect size, we only include cases where there is a 1.5 point jump or drop in our semantic difference scale. We use 'es' to indicate this in the data below. Statistically significant differences with low effect size do not allow one to draw notable conclusions. For reference, a 1-point jump (es=1) could be from 'not needed' to 'may be useful' or from 'needed' to 'critical' On the other hand a two point jump (es=2) might be from 'not needed' to 'needed'.

For interestingness, we retained only cases that were not 'obvious'. For example, we consider it not interesting that people who primarily teach with drawing tools prefer simple drawing tools to formal tools. Conversely, people who focus on teaching formal transformations would obviously be more interested in transformation languages than people who never teach transformation, so the fact that there is a statistical significance for this with high effect size is not interesting.

We boil down our comparative analysis to the following three groups of educators:

Those who teach model transformations: Not every educator teaches about model transformations, but it it is clearly very important to some educators. People who extensively teach about model transformations:

- Want more customizability (e.g., adding plugins, extensions, macros, etc.). es=1.5, p=0.027. These people are also interested in modeling tools that run in tools like VS-Code (es=1.8, p=0.004) or Eclipse, (es=2.2, $p < 0.001$) due to the plugin capabilities.
- Are more likely to want a textual interface for the modeling tool (even if graphical is the main way of modeling) es=1.5, p=0.011
- Are more interested in the ability to compare models (compute differences) and merge models. es=1.5, p=0.014
- Want tools that enable language engineering (creating domain-specific-models, DSMLs) es=1.8, p=0.004

Those who educate using video: There is a subgroup of educators who extensively use video in their pedagogical method. These people also:

- Are more interested than others in the availability of collaborative editing by multiple students (es=1.8, p=0.005).
- Want more integration of gamification (es=1.6, p=0.01).

Those who teach by having students analyse and criticize models: This group is more interested than others in the ability to embed code in models (es=1.5, p=0.002). The reason for this might be that this facilitates executing models, which is one way to analyse them.

4.5 Modelling tools

Question 16 asked about respondents' knowledge and use of specific modelling tools. Respondents were asked to rank a list of given tools on the following scale:

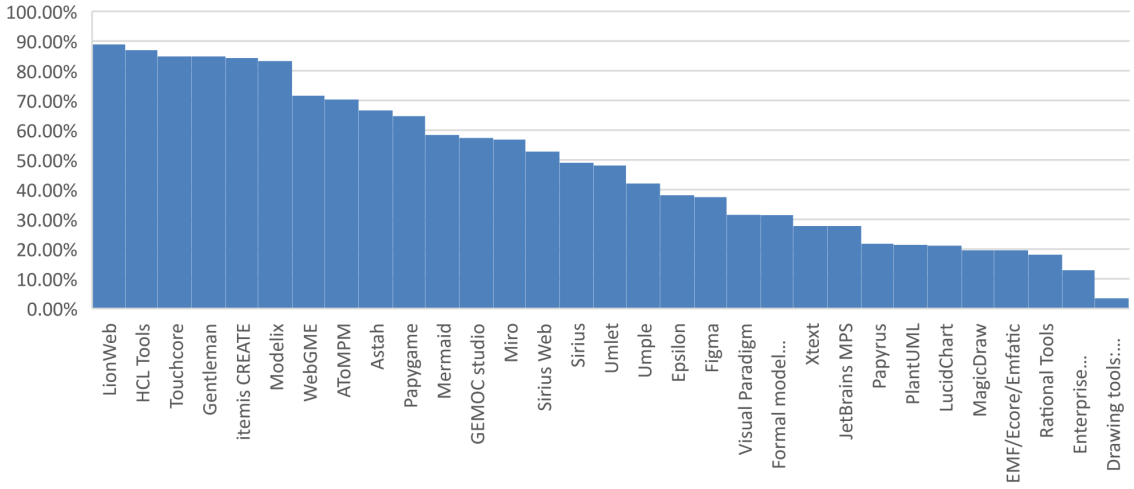


Figure 8: Percentage of respondents who claimed never to have heard of a particular modelling tool.

- I have never heard of it
- I have heard of it but have never used it in teaching modeling
- I have occasionally used it in teaching modeling
- It is one of several tools I use in teaching modeling
- It is the main tool I use in teaching modeling

Figure 8 provides an overview of the tools respondents' claimed never to have heard of. Some tools appear to only be known to a very specialist group of respondents. We corrected for the level of knowledge by re-ranking tools based on only those respondents who had heard of them before.

Figure 9 summarises the re-ranked data, where the percentages are based on only the number of responses who had heard of the tool before.

5 Thematic analysis based on the initial open-ended question

This section focused on the responses the open-ended survey question 5. We discuss our methodology in Section 2.4. The analysis began with a topic summary approach to chart the occurring themes. Then, it followed a thematic direction by examining topic co-occurrences and their relative importance, allowing for the identification of emerging patterns across the data.

We manually coded 53 raw comments provided in Q5 ("What do you think are the aspects of software modeling tools that need the most improvement to make them better for use in education?") from as many participants. From the analysis, 47 themes emerged, which were then coded into higher-order themes. We thus obtained 15 themes covering 47 sub-themes. The most frequently occurring theme was Usability/UX with 18 independent occurrences, followed by Educational Support (17), Advanced Functionality (16), Simplicity (15), and Use across platforms (14).

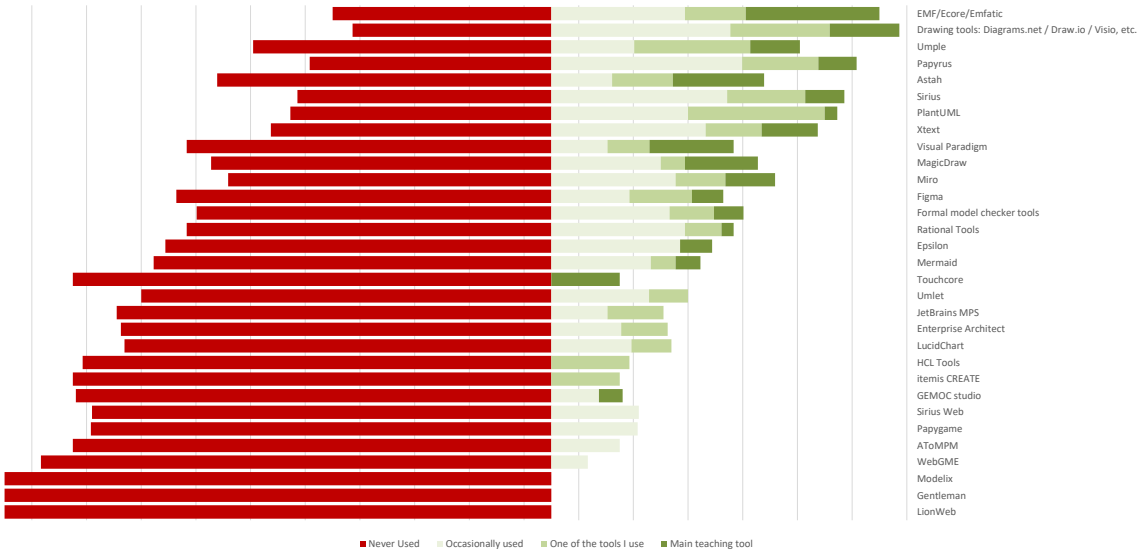


Figure 9: Level of use of modelling tools among educators who knew each tool. X-axis shows percentage of respondents, centred to separate "never used" from the rest of the responses.

We dropped sub-themes that presented only one occurrence. See Table 4 for a complete breakdown of the themes.

5.1 Theme analysis

We report below the themes and sub-themes we extracted, with example excerpts from the participants' comments (the excerpts may touch more than one theme at once).

Advanced Functionality. This theme was used to code comments that referred deeper and more complex functionalities of the tools. It included subthemes like *Deeper functionality in general*, *Execution and simulation of models*, *Language creation/metamodeling*.

> P1 - "*executable modeling tools, simple tools for generating code from models and vice versa, supporting other languages than only C, C++, [for the] entire [range] of UML diagrams*"

Usability/UX. This theme regarded usability in general, and two participants also mentioned the *undo* functionality and better direct manipulation.

> P9 - "*Easier to use, e.g., allowing direct manipulation instead of requiring users to navigate between diagrams and forms.*"

Educational Support. This theme includes considerations about diverse education-related features, such as *Guidance/assistance/tutoring for students*, *Automated grading support*, and *Value for a student CV/industrial usage*.

Table 4: Themes and their sub-themes, with the number (N) of independent occurrences. Note: sub-theme numbers may not add up to the same number of the macro-theme because of overlaps within the same comments. Sub-themes with only 1 occurrence were dropped.

Theme	N	Sub-themes (N)
Usability/UX	18	General Usability (18)
Educational Support	17	Impose rigor (2), Guidance/assistance/tutor for students (9), Value for a student CV/industrial usage (6), Automated grading support (2)
Advanced Functionality	16	Deeper Functionality in General (12), Execution and simulation of models (4), Language creation/metamodeling (2), Support patterns/arch styles/design types/frameworks (4), Support informal/fuzzy models/intermediate states (2)
Simplicity	15	Simplicity (6), Simpler beginner UX (9)
Use across platforms	14	Installability (10), Web-based UX (6)
Transformation	8	Generate code (4), Model transformation (2), Transform to model with AI/Software 2.0 (2), Reverse engineering (2)
Collaboration Support	7	Collaboration (editing at the same time) (6), Version control/Diff-Merge/Agility support (2)
Language Support and Compliance	7	Support many modeling languages (3), Full standard compliance (5), Simpler language subset (2)
Modernization	6	Not in Eclipse (3), More modern (3), Keep up to date (3)
Analysis-Related Capabilities	6	Model validation/Error detection (4), Explanation of mistakes/Feedback (2), Traceablity (2)
Platform Versatility	5	Integration with other modeling tools (5)
Manipulation and Visualization	4	Better visualization (4)
Documentation and Learning Resources	4	Better documentation (4),
Robustness and Performance	3	Reliability/lack of bugs (3),
Availability for use	2	Free (2)

P18 - "*[...] In addition, most tools are tailored for professionals and they can be rather heavy for students and simple assignments. Freely available tools may not always support all types of diagrams.*"

Modernization. Comments included in this theme were coded into subthemes such as *More modern* and *Kept up to date*, as well as *Not in Eclipse*, a subtheme specifically referring to comments and suggestions that mentioned Eclipse being outdated.

P27 - "*Futhermore, a lot of the tools can't be run outside Eclipse (a like a lot Eclipse, but sometime[s] we just don't need it). More and more student complain about these tools. Having more modern and more easy to use tools and languages will help to improve modeling teaching.*"

Simplicity. This theme was used to code comments that referred to the unnecessary complexity of some tools, e.g., in delivering the necessary information to beginners.

P14 - "*Learning tools have too much accidental complexity (especially Eclipse), making it more difficult than necessary to explain the concepts to students.*"

Analysis-Related Capabilities. This theme deals with *Model validation/Error detection*, *Well-formedness analysis*, and related subthemes such as the analysis feedback (*Explanation of mistakes/Feedback*).

P47 - "*[...] Model verification, usually this features is not included in the Academic License of the tools. Real possibilities to applied MDE paradigm.*"

Platform Versatility. This theme includes comments about the possibility of *Integration with other modeling tools* and *Customizability.*

P48 - "*interoperability between tools and real connections between modeling languages to build up interesting outcomes*"

Language Support and Compliance. Comments from this theme were reconstructed from subthemes such as *Support many modeling languages*, and *Full standard compliance.*

P17 - "*Modelling tools typically support everything that is legal in the language, but for teaching I would like to restrict much more the concepts that students have access to, to ease them in to modelling rather than throwing them in the deep end immediately.*"

Availability for use. This theme deals with the existing opportunities to find and use free (and open source) modeling software.

P44 - "*Need for good free tools that can do reverse engineering of existing code*"

Use across platforms. Comments included in this theme referred to the possibility to work seamlessly across platforms and were coded into the following subthemes: *Installability*, *Web-based UX*, *Ability to work offline*, and *Multi-platform.*

P32 - "*Some software modeling tools are not intuitive enough, are not web-based [...]* "

Collaboration Support. This theme includes suggestions about how to have multiple people work on the same project at the same time and the support for version control and agility features.

> P18 - "*What is needed is a) Modeling tools that can be used in an agile manner. With little model changes reflected in pull requests along with the resulting code, and the ability of reviewers (and the developers) to see the 'diff' of the model from the main branch (as they can do with code.)[...]*"

Transformation. This theme about model transformation included consideration about the tools' capabilities in transformation in general (*Model transformation*, *Transform to model with AI*), as well as tangential capabilities like handling many different generation languages (*Many generation languages*).

> P52 - "*[...]b) Modeling tools that are cognizant of whatever framework is being used and can enable people to build proper software that would be considered good quality Mongo, or React, etc. Ideally it could generate code and regenerate when the model changes.[...]*"

Robustness and Performance. This theme included comments that ranged from responsiveness (*Fast response time*) to reliability (*Reliability/lack of bugs).*

> P6 - "*Robustness (they [the tools] should support a high load balance in certain periods of the semester)*"

Documentation and Learning Resources. Apart from tool documentation, this theme dealt with how well existing tools provide learning resources and examples (not only to students).

> P19 - "*[...] better documentation, integration with other systems and tools [...]*"

Manipulation and Visualization. This theme especially included functionalities for better control on models and the way they are visualized.

> P5 - "*[...] Possibility of interacting with models (e.g., while giving a lecture) that allows the teacher to highlight parts of the model of interest to the students.*"

5.2 Co-occurrence analysis

The heatmap depicted in Figure 10 represents a theme co-occurrence matrix, illustrating how often specific themes from a thematic analysis of comments appear together. Here are some key highlights:

Clusters of interest: *Usability/UX* has the strongest co-occurrences with *Educational Support* and *Use across platforms* (8, 6), and the same amount is found for the co-occurrence between *Educational Support* and *Simplicity* (8), indicating a focus on user-friendly, portable, and seamless education-oriented tools and resources.

Advanced Functionality co-occurs significantly with *Transformation* (4), *Use across platforms* (5), *Usability/UX* (6) and *Educational Support* (6), suggesting that the advancement of tools should be in the direction of easy transformations across different platforms, with more effective educational value and usability experience.

Educational Support, *Usability/UX*, *Use across platforms*, and *Advanced Functionality* are part of a broader cluster with several connections to other themes, which highlight these as central concerns in the thematic analysis. In summary, the analysis suggests a strong focus of the suggested improvements on usability, educational support, and cross-platform functionality, with some emphasis on adaptability, language support, and analytical capabilities, while aspects like availability and visualization appear less central in the discourse.

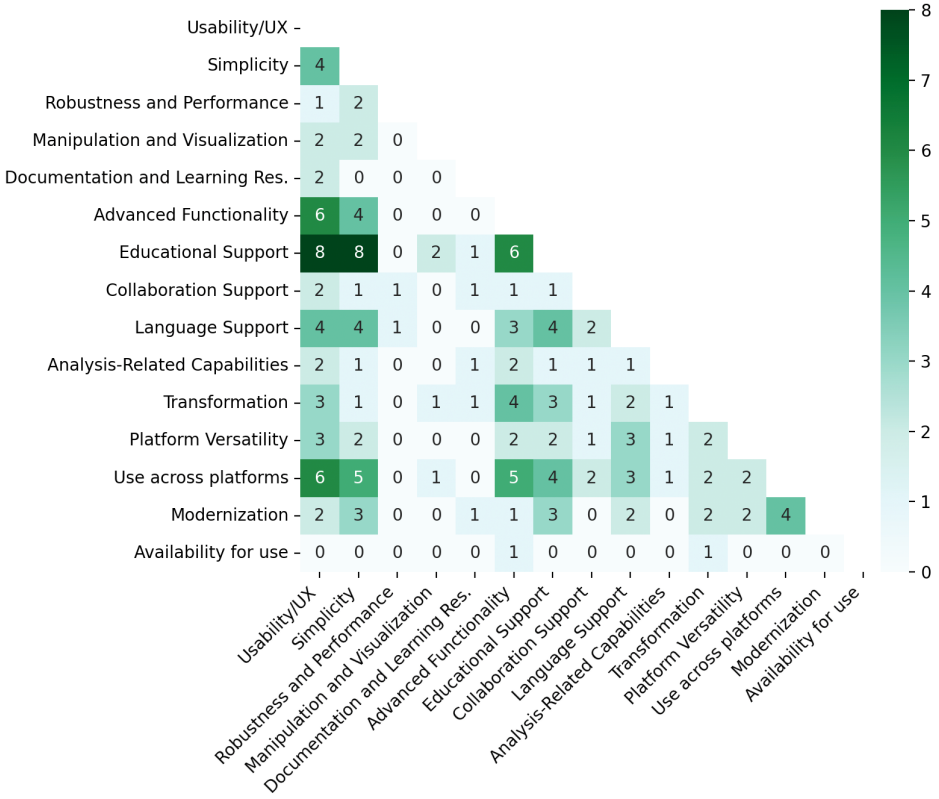
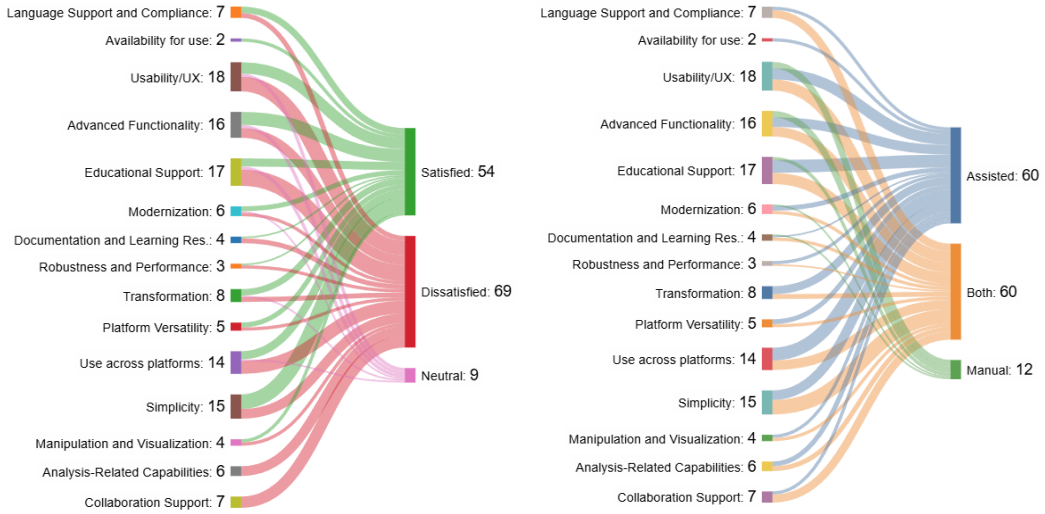


Figure 10: Theme co-occurrence analysis. Sub-themes with only 1 occurrence were dropped.



(a) Themes to satisfaction levels.

(b) Themes to teaching style categories.

Figure 11: Sankey diagrams mapping identified themes to other parts of the questionnaire.

Isolated or low co-occurrence themes: *Availability for use* shows very few or no co-occurrences with most other themes, which could mean that this theme is discussed only marginally and, when it is, it is discussed independently. Manipulation and Visualization and Modernization also have low co-occurrence with other themes, possibly indicating that these aspects are specific niche areas within the feedback.

5.3 Satisfaction analysis

As discussed earlier, in Question 4 we asked participants to rate their level of satisfaction with current modelling tools for teaching (cf. Fig. 1). Question 5 asks participants to identify areas in most need of improvement. Here we compare answers to both questions to understand whether the overall level of satisfaction has an impact on the types of improvements participants ask for.

The Sankey diagram in Fig. 11(a) illustrates how different themes connect to the overall satisfaction levels. On the right-hand side, we have categorised the responses to Question 4 into three broad categories: Satisfied (merging Very satisfied, Satisfied, and Somewhat satisfied), Dissatisfied (merging Very dissatisfied, Dissatisfied, and Somewhat dissatisfied), or Neutral (the latter was "Neither satisfied nor dissatisfied" in the original questionnaire but is reported here as "Neutral" for brevity). Each flow links features for which improvements have been requested to the overall level of satisfaction of participants.

One of the most useful insights comes from the Educational Support category, which flows primarily into the Dissatisfied segment. This appears to be a possible predictor of dissatisfaction (with the biggest flow to such category, 10 times), suggesting that shortcomings in educational scaffolding or learning resources significantly undermine the user experience. However, the flow to satisfying ratings is also significant, suggesting that this category may still vary in satisfaction, and that it is a salient category for users in general. This concentrated pattern stands out compared to

other themes, which are often associated with mixed levels of satisfaction.

In fact, many themes—such as Usability/UX, Simplicity, Advanced functionality, and Platform Versatility—are distributed fairly evenly between satisfied and dissatisfied users. This balance may indicate that satisfaction is not strictly tied to the objective presence or absence of the theme in the comments, but possibly to individual thresholds of expectation. In other words, the same (lacking) functionality may drive and predict satisfaction more or less depending on the user's personal benchmarks and intended use cases.

Interestingly, some requested features like Analysis-Related Capabilities and Collaboration Support are only mentioned by dissatisfied users. This asymmetry suggests that shortcomings in these aspects may strongly predict dissatisfaction.

Features such as Simplicity and Usability/UX show a consistent alignment with positive user experiences, contributing predominantly to the Satisfied group. However, their experience remain mixed, with a significant flow to the Dissatisfied group. This may suggest that problems in simplicity and usability in current modeling tools do not drive dissatisfaction strongly, although they are frequently mentioned, and it seems advisable to improve them.

5.4 Teaching style analysis

To investigate the issues raised about the tools in relation to how teachers prefer to conduct their classes, we divided the teachers according to three categories related to their teaching styles: *Manual*, *Assisted*, and *Both*. *Manual* refers to teaching that is performed by drawing by hand, while *Assisted* refers to teaching software modeling with tools and plugins. The *Manual* category was represented by teachers with high scores in the following 4-point items from Question 15:

- **I7**. *Draw models informally on a whiteboard / blackboard in class*
- **I9**. *Having students draw small models using pencil and paper, whiteboards*

The *Assisted* category was represented by teachers with high scores in the following 4-point items:

- **I3**. *Use a tool live in class to create or edit models*
- **I10**. *Having students build small models using a modeling tool*

A threshold was used to determine a preference for one of the two: teachers were assigned to one category or the other according to the difference between the summed scores in the two pairs of questions. When the difference was not substantial ($<= 1$ point) they were categorized in *Both*. The categorization was performed and revised by two authors. We show the distribution of the themes in Fig. 11(b). All data was interpreted as meaningful (i.e., there are no cases of both categories being significantly low).

Most themes are strongly associated with preference for the *Assisted* style, with a total count of 60, and *Both*, with an equal count of 60 (see the right-hand side of Fig. 11(b)). This indicates a preference for using technological tools and plugins in software modeling education either alone or in conjunction with *Manual* methods, over teaching based on *Manual* methods alone.

The *Manual* style has only 12 total connections, highlighting a trend towards lesser use of teaching based solely on drawings. However, themes such as *Usability/UX* and *Advanced Functionality* still connect remarkably with this style, which may indicate some current tool adoption issues in

addition to the ones identified in the co-occurrence analysis. Although *Advance Functionality* has a considerable flow to *Assisted*, it also connects to the *Manual* style alone (4). This may suggest that this is a deciding category of features when adopting digital modeling tools. The theme *Educational Support* (17) also flows to some degree into the *Manual* teaching style (2). In sum, teachers might avoid adopting tools due to concerns about ease of use and insufficient functional relevance to their needs, especially with respect to educational support.

6 Discussion

We organize this section around our four research questions

6.1 RQ1: Satisfaction with the current state of MTT

As indicated in Section 4, Question 3 showed that a slight majority of our respondents are modestly satisfied with the current state of modeling tools for teaching, but none are very satisfied, and a significant percentage are dissatisfied.

Among the 50 features or properties, about which we asked in Questions 7-13, Table 3 shows that 15 are considered essential or critical by over 50% of respondents, and all but one are considered essential or critical by over 10%. Yet it is also clear that a high percentage of existing tools are missing many of the features or properties, or at least have weaknesses in these areas. Items near the top of Table 3 that seem most likely to appear as weaknesses in various current tools include comprehensive error messages (ranked #3), a comprehensive manual (#4), a library of examples (#5), code generation (#7), being free (#8), version control (#10), running as a website (#11), and model annotation (#12). It may well be the case that if these were improved a future survey might have some very satisfied respondents.

While the quantitative data highlights clear gaps between desired features and current tool support, qualitative insights further highlight educators' concerns. The thematic analysis of open-ended responses to Q5 (Section 5) highlights key areas for improvement in modeling tools for teaching. Usability, educational support, and simplicity emerged as the most frequent concerns, alongside the need for cross-platform availability and advanced features. In particular, Usability/UX has a significant interplay with Educational Support as well as with Use across platforms. Based on the co-occurrence matrix and the topic flow observable in the Sankey diagram, these themes suggest that many educators, regardless of teaching style, perceive current tools as insufficiently tailored to educational contexts, which may limit their adoption in the classroom.

These impressions are further reinforced by the satisfaction analysis presented in Fig. 11(a), which maps improvement requests to self-reported satisfaction levels. A striking pattern emerges in the Educational Support theme, whose dominant flow leads into the Dissatisfied group. This suggests that the lack of scaffolding materials and learning resources may be a primary predictor of dissatisfaction, while still being relevant to satisfied users as well, albeit to a lesser degree. In contrast, other themes like usability, simplicity, platform versatility, and advanced features show a more balanced distribution across satisfaction levels, hinting that satisfaction may not strictly correlate with the presence or absence of specific features, but with users' expectations and personal benchmarks. Interestingly, features such as Collaboration Support and Analysis-Related Capabilities are mentioned exclusively by dissatisfied users, indicating that their absence might be a particularly strong source of dissatisfaction.

As discussed earlier in the paper, following Holm-Bonferroni correction there were no significant differences in answers to Questions 6-15 among those who were broadly satisfied with existing tools as compared to those who were not satisfied. This suggests that people have developed a sense of general satisfaction with what is currently available, even though they express interest when they are presented with a list of improvements. Some may not even be aware of potential improvements, since there is a broad lack of knowledge of many existing tools, each of which tends to have its own innovations, as shown in Figure 8. A key issue might be lack of marketing or other awareness-generating activities in the community: If educators were more aware of capabilities of other tools, their overall satisfaction may drop, as they start to crave missing features

6.2 RQ2: Needed features and properties of MTT

From Fig. 4 and Table 3, we can analyze the importance of various features and properties of modeling tools. The top five features, saving and loading, being free, cross-platform compatibility, ease of use, and reliability, are fundamental attributes expected of any software. The fast response time, slightly lower on the list, also falls into this category. Although these are not modeling or education-specific concerns, they are clearly important in an education context. The user experience of modelling tools continues to be an important challenge for the MDE community [3, 13, 33], and this is reflected also in the needs of teachers of MDE. Where these attributes have been prioritized, such as in the user-friendly tool Umlet [4], the trade-off has often been limited functionality, with other desirable features missing.

The top-ranked modeling-specific features are directly tied to the learning experience: comprehensive error and warning messages (i.e., automated model analysis), an online manual with live examples, and a repository of modeling examples. These three elements are fundamental to modern learning environments. For example, programming students rely on compilers and IDEs with clear error messages, extensive documentation, a wealth of online examples, and hints on what to do next [45], but struggle even so [44]. Similar expectations exist across disciplines such as accounting [8], and chemistry [22, 55].

If modeling education is to align with modern pedagogical standards, its tools must integrate these features effectively. However, existing tools often fail in these areas. For instance, Papyrus [39], despite its capabilities, was found to be much higher in complexity than most other tools used in modeling education [2].

Further down the list, we see features more specific to modeling: code generation, visualization of model subsets, version control integration (e.g., Git), and annotation/commenting capabilities. This suggests that future modeling tools designed for education should be sure to include these features.

Some mid-tier-ranked features include adherence to standards, random instance generation, language engineering capabilities, and a textual interface. It should be noted that collaborative editing and syntax-directed editing—common in modern creative environments—did not receive higher priority. This suggests that while these features may be beneficial, they are not essential for most educators.

At the lower end of the rankings, there are some surprising results. AI-assisted model creation (e.g., generating models from requirements [48, 58]) received low ratings, despite broader research interest in these areas. Since interest in generative AI has been intense since this survey was performed in early 2024, it is possible that a more recent survey would have resulted in a higher ranking for AI integration. Another possible explanation is that educators prioritize student en-

gagement with modeling concepts over automation, ensuring that learners develop critical thinking skills rather than relying on AI-driven solutions. In programming education, similar discussions are summarised (in the context of a study of novice programmers' use of quick fixes) by Brown et al [11].

Gamification (e.g. [14]) is similarly rated low by our participants. While platforms exist that support gamification for learning modelling (e.g., [12]), we suspect that the issues in tool usability currently still dampen educators' appetite for such more complex interaction models.

The data in Figs. 5 and 6 further reinforce the strong link between modeling tools and programming language preferences. The languages most frequently taught—Java and Python [42]—are also the ones in which code generation is most desired. This correlation is expected and indicates alignment between modeling and coding practices. Building on target languages students already know reduces complexity and allows students to focus on the new modelling concepts.

6.3 RQ3: Modeling languages to be taught

Examining Fig. 2, several key insights emerge: Firstly, a significant number of participants categorized many modeling languages as critical to teach, as is evident from the rightmost dark-green portions of the bars. In particular, all languages listed were deemed essential by at least some educators. Additionally, a considerable portion of respondents marked these languages as 'needed', suggesting that approximately half of all responses indicate strong enthusiasm among modeling educators about the importance of teaching a diverse set of modeling languages.

However, the rationale behind this widespread positivity remains unclear. Several questions arise that warrant further research. Are educators basing their selections on personal (industrial) experience with these languages? Are they following textbook recommendations or adhering to curriculum standards without questioning their real-world applicability? Or is there an element of tradition at play, where educators teach the languages they themselves learned, without necessarily having contemporary (industry) insights on their usage? Knowledge of industrial use of many modelling languages is limited in the academic community—as, for example, documented by Petre in the context of UML usage [50], where there is perhaps the most significant body of empirical research (e.g., [9, 17, 23, 24, 26, 27, 37, 38, 46, 47, 49, 51, 52, 54]). Many of these languages have been in existence for more than three decades, often evolving from their predecessors. For example, class diagrams were preceded by Rumbaugh's OMT diagrams [56] and Booch's diagrams [7] in the early 1990s when object-oriented programming was becoming the dominant paradigm. The persistence of these languages in curricula may be influenced by inertia rather than a deliberate assessment of their relevance to the modern world. As pointed out by one of our anonymous reviewers, it is equally possible that respondents choose 'class models' as a key language to reflect the need for teaching modelling of data schemata – which may be done in a variety of specific languages, including entity-relationship models, JSON schemata etc. It is interesting to note, however, that 'entity relationship diagrams' (which have a similar semantics) are ranked well below 'class models' in the responses (cf. Fig. 2).

Another striking observation in Fig. 2 is that almost every model type has some educators who consider it harmful to teach. This raises critical questions: What aspects of class and state diagrams could be considered detrimental? Do some educators feel that these models have limited industry application, making other topics a better use of instructional time (see also [50, 18])? Interestingly, despite these concerns, each educator identified at least one modeling language they considered beneficial. More specifically 88 percent ranked at least one language as critical, and ten percent

gave 'needed' as their highest ranking for at least one language. Only one respondent gave 'may be useful' as their highest ranking among the languages listed.

A subset of respondents advocated for mathematically rigorous languages such as Formal Methods, Petri Nets, and OCL, presumably because they believe that these provide significant value for software specification. Conversely, they may see more popular notations like class and sequence diagrams as offering little additional value beyond what can be derived directly from examining code. Clearly, educators have diverse and sometimes conflicting opinions on which modeling languages are truly valuable and more *qualitative* research may be needed in this area.

6.4 RQ4: Teaching practices to be supported by MTT

Figure 7 sheds light on how modeling tools support or should support various teaching techniques. The dominant teaching approach is "having students build small models", perhaps to keep problem complexity contained and allow students to focus on specific modeling challenges, allowing teachers to create "desirable difficulty" [6]. Problem-based [62] and project-based [5] learning follow closely, indicating a need for tools that support larger-scale modeling exercises and more flexible exploration to allow teachers to increase complexity in controlled ways. Additionally, the emphasis on "having students use modeling to build real systems that can execute" suggests that future tools should be robust and capable of handling practical, executable models.

The emphasis on executable modeling in education suggests the need for tools that bridge theory and practice by providing immediate feedback and interactive exploration [34]. Although industrial tools—like Eclipse Papyrus and MagicDraw [51]—offer execution capabilities, their complexity can be overwhelming to students. Instead, lightweight environments, such as executable domain-specific modeling languages (xDSMLs [43]), modeling playgrounds [63, 35], and simulation-integrated tools [41, 25], can provide a more accessible learning experience. These tools allow students to incrementally build and test models, reinforcing their understanding without requiring extensive setup. Technologies like Umple [40] demonstrate how execution can be integrated into educational modeling environments, supporting problem- and project-based learning. By focusing on rapid iteration and controlled complexity, such tools can enhance engagement and help students develop a deeper understanding of modeling concepts.

Figure 7 also suggests that most educators do not require tools for DSL development or model transformation. Clearly, the majority of our respondents focus on teaching modelling rather than the development of modelling languages and tools. This is natural: developing modelling tools is a more advanced skill than producing models and is likely only taught in specialized courses in the final year of undergraduate degrees or in master's programmes [21].

Figures 8 and 9 highlight a key challenge in the modeling tool landscape: lack of awareness and adoption. Figure 8 reveals that half of the participants were unfamiliar with half of the available tools, suggesting that many tools suffer from limited visibility or overly specialized applications. Even among the more widely recognized tools, 10–20% of the educators had never encountered them. This raises concerns about whether educators have the time or resources to explore alternative tools beyond their familiar choices. Figure 11(b) shows that teacher either use technology-assisted teaching methods or both assisted and manual, but only rarely manual alone. The use of manual/analog methods for teaching could be attributed to lack in usability and advanced functionality: people within the *Manual* category often report needing improvements in these aspects. In fact, Usability, Advanced Functionality and Educational Support are tightly connected to dissatisfaction (Figure 11(a)). This figure also suggests that Analysis-Related Ca-

pabilities and Collaboration Support may be predictors of dissatisfaction with tools. One way to improve awareness and adoption of modeling tools is through better dissemination strategies and integration into educational workflows. Educational institutions and professional organizations could promote these tools through workshops, online courses, and curated repositories showcasing their capabilities. Simplifying onboarding processes with interactive tutorials, guided walkthroughs, and sandbox environments can help educators explore new tools without a steep learning curve. Additionally, fostering stronger collaboration between tool developers and educators—such as incorporating direct feedback loops, offering training sessions, and integrating tools into widely used learning management systems—could enhance visibility and usability. Open-source initiatives and community-driven documentation efforts may also play a crucial role in making these tools more accessible. Lastly, embedding modeling tools within existing curricula and providing incentives for experimentation—such as grants, professional development credits, or research collaborations—could encourage educators to step beyond their familiar tool set and adopt new, more effective solutions.

Figure 9, which excludes participants who had "never heard of" certain tools, illustrates that 18 different tools are considered the primary teaching tool by at least one respondent. However, most tools had never been used by the vast majority, even when awareness existed. This fragmentation suggests a pressing need for a community-driven effort to consolidate the best features of these tools into a smaller, more widely adopted set (the MDENet Education Platform [63] is one attempt to create such a system). Future modeling tools should aim to be widely accessible, easy to install, and user-friendly while incorporating the most desired features to ensure broad adoption among educators in the coming decades.

7 Related work

Our paper surveyed teachers of modelling and MDE about their perspectives on modelling tools for teaching. We are not aware of a similar survey in the published literature. The related survey by Ciccozzi et al. [20] focused on teaching practices rather than tools specifically. Its results complement our own work. Equally, Agner et al. [2] surveyed students about their experience with modelling tools. Again, this complements our own work by providing the student perspective.

The MDE Body of Knowledge (MDEBoK) [21, 15] provides a structured perspective of what students *should* be taught about modelling and MDE. The most recent paper [15] also incorporated results of a survey, as did our work. Clearly, modelling tools for teaching need to support MDEBoK contents. Thus, MDEBoK can provide helpful additional insights for developers of tools for teaching modelling. We analysed the content of MDEBBoK and noted several important differences: There are several categories of tool features and properties in our results that are not included in the MDEBoK. Some of these might be considered for addition to a future MDEBoK version.

- User experience aspects (Q7)
- Collaborative model editing (Q8)
- Annotating models (Q8)
- Teaching-specific aspects: Gamification in modeling tools (Q8); plagiarism detection (Q10); automatic grading (Q10)
- Embedding code in models (Q9)

- Separation of concerns in models (model traits, mixins, product lines) (Q9)
- Traceability in models (Q9)
- Generation of model improvements (Q10)
- Creation of models by AI (Q11)
- Generation of documents and natural language descriptions from models (Q11)
- Platform aspects (Q12)
- Some business and economic aspects (Q13)

Many of the items in our survey correspond to multiple MDEBoK sub-items. For example, in our survey there is a tool feature in Q10 entitled 'Ability to present comprehensive error or warning messages about problems with model syntax or semantics'. This would help with teaching of and draw knowledge from all the sub-items in MDEBoK Section 2 (Model Quality), much of MDEBoK Sections 1.1 (Syntax) and 1.2 (Semantics), as well as many of the items in MDEBoK Section 3 (Analysis). Similarly, our tool feature in Q11 entitled 'Transformation to other modeling languages' would map to much of MDEBoK section 8.1 (Model transformation languages), and MDEBoK Section 8.2 (Model transformation types). There are a few MDEBok topics that are not covered in our survey, such as 5.5 (Animation), and topics in Section 9 such as modeling in certain application domains.

In the early and mid-2010's, there was significant empirical research focusing on the use of modelling (UML in particular) in industry—often concluding that use of formal UML was rare. For example, Petre [49] undertook a survey of UML use in practice, while Osman and Chaudron [46] undertook a field study. Dobbing and Parsons [23, 24] as well as Reggio et al [53, 52, 54] focused more specifically on the use of different UML diagrams, aiming to identify more or less useful diagrams. There is some agreement here with our finding that class diagrams are the most used modelling language. Cherubini [18] reported on informal use of modelling in industrial practice. Such use of modelling without tools corresponds to our finding that a good subset of teachers appear to prefer more informal model "drawing" over more formal modelling tools. While these studies focus on industrial use of UML—very different from ou focus on the use of tools for *teaching* modelling and MDE—Boustedt [9] aimed to understand how well students understand class diagrams.

Usability has been a topic of interest for some time, and continues to be important to teachers of modelling and MDE. Auer [3] presented a very early study on the usability of UML tools. More recently, Kalantari et al. [33] systematically studied the modeler experience across MDE success stories.

Mason et al. [42] present a related survey of introductory programming courses. We note that programming is sufficiently closely related to modelling—despite the clear differences with respect to, for example, abstraction levels or domain-specificity of notations—that some insights from their survey may be transferable to the modelling-tools domain.

8 Threats to validity

The following are potential threats to validity of this work:

- We have reasonable confidence in the **validity of the survey** for several reasons: Firstly, the authors have considerable experience creating surveys about modeling tools whose results have been published (e.g. [2, 32]); two of the authors also teach about survey design. Secondly, we are deeply immersed in the use and design of modeling tools for teaching, have researched the problems with such tools, and have attempted to address them for many years; this gives us confidence that even the first version of the survey addressed our research questions. Thirdly, this was a second-generation survey, the first generation being the survey conducted at the original MTT workshop. We received feedback about the questions from the participants in that workshop, allowing us to improve survey validity. Fourthly, we piloted the penultimate version with three participants before distributing it, resulting in further fine-tuning of the questions.

  Although we put considerable effort into design and validation of the survey, there is clearly room for improvement if this survey were to be reused on a larger scale. To deepen validity, the precise wording of individual questions could be further workshopped with a sample of modeling educators, to reduce the possibility of ambiguous interpretation or misunderstanding. Additional options could be added such as those that were supplied as 'other' options for Questions 6, 14 and 16 as well the open-ended question 5.

- There may have been **bias in the sampling process** since it was not feasible to randomly sample the full population of those teaching modeling in all geographies. Bias may have resulted from the limited networks of the original workshop participants, the reachability of people particularly in Asia, and the willingness of people to respond. Indeed the full population of those who teach modeling is not clear: Modeling is a topic in SWEBOK [61] and is found throughout the ACM Computer Science Curricula [36], but we are not aware of accurate data regarding the proportion and distribution of computing programs that follow these recommendations, or computing educators who have model-teaching expertise and who use tools for such teaching.

- We note one **inconsistency** in our survey: Questions 7–13 use a 6-point scale, whereas question 5 used a very similar scale but with only 5 points. In retrospect, it would have been better to use the same scales for improved understanding. This would not have effected our conclusions however.

- Not every educator teaching software design may have a **modeling background** or believe in modeling as we have considered it in this paper, or as described by SWEBOK [61], the ACM Curricula [36], or MDEBoK [21, 15]. The survey may therefore have been over-specialized around clusters of model-teaching expertise such as those who teach UML, formal methods and model transformation. We are aware of other schools of thought around how to encourage abstract software design thinking, such as using advanced features of programming languages including Ruby or Simula, as well as the abstractions inherent in frameworks. Since we did not reach out to such educators, this may have limited the breadth of usefulness of our results.

- Modeling, and software engineering more generally, are **moving targets,** with new languages, frameworks and tools appearing each year, influencing what may need to be taught and how. Some of our conclusions may therefore become rapidly out of date. The rapid rise of AI tools is just one of the changing factors.

- We did not address **relevance** of modeling and the tools used to teach it to the students' subsequent experience **in industry.** As we mentioned in the introduction, students may not be motivated to learn modeling if the knowledge would not seem to be directly applicable in industry; also industry might make more use of modeling if they employed graduates who had learned to do it well. This does not invalidate our conclusions about what educators feel they need in a modeling tool, but it limits the work's broader applicability. It would be good if good teaching tools could also be good tools for industrial use.

9 Conclusions and outlook

We have presented the results of an international survey providing data on the way in which modelling and MDE are currently taught and the requirements that arise in consequence for modelling tools for use in teaching. This provides additional evidence to underpin the requirements captured based on workshop discussions in [34].

Our study leads to several key conclusions:

Firstly, there is significant enthusiasm among educators for teaching a diverse set of modeling languages, despite some uncertainty about their real-world applicability.

Secondly, although teaching foundational languages such as class and state diagrams remain widely supported, divergent opinions exist about the imporance of teaching more specialized or mathematically rigorous notations, and about model transformation.

Thirdly, the survey confirms our prior assessment that many educators perceive gaps in existing modeling tools, particularly in terms of usability, documentation, and pedagogical support. Essential features such as comprehensive error messages, code generation, and repository access for examples are considered critical to effective teaching. The paper's prioritized list of tool features and properties should provide useful guidance to tool developers.

Fourthly, despite the breadth of available modeling tools, awareness and adoption remain limited. Educators report that they are not even aware of many tools. This leads to a fragmented landscape where no single tool meets all educational needs.

Taken together, the above conclusions underscore the need for a community-driven effort to consolidate best practices and create tools that are accessible, user-friendly, and equipped with the most desired features.

Looking ahead, we encourage continued collaboration within the modeling education community to refine and develop tools that align with modern pedagogical practices. Future research should explore the practical impact of modeling education on industry adoption, and study modelling teachers' needs more extensively from a qualitative perspective. In addition, we must consider how modeling tools can address broader social challenges, such as sustainability, ethics in software design, and the responsible development of AI-driven systems.

Acknowledgments

Zschaler's contribution was partly funded by the UK Engineering and Physical Sciences Research Council (EPSRC) through the MDENet grant (EP/T030747/1). Lethbridge's contribution was partly funded by Canadian Natural Sciences and Engineering Research Council (NSERC) grant RGPIN-2022-04001, and by resources obtained through the Digital Research Alliance of Canada.

References

[1] Hervé Abdi and Lynne J Williams. Tukey's honestly significant difference (HSD) test. *Encyclopedia of research design*, 3(1):1–5, 2010.

[2] Luciane T. W. Agner, Timothy C. Lethbridge, and Inali W. Soares. Student experience with software modeling tools. *Software & Systems Modeling*, 18(5):3025–3047, Oct 2019.

[3] Martin Auer, Ludwig Meyer, and Stefan Biffl. Explorative UML modeling - comparing the usability of UML tools. In *Proc. 9th Int'l Conf on Enterprise Information Systems*, pages 466–473. SciTePress - Science and and Technology Publications, 2007.

[4] Martin Auer, T. Tschurtschenthaler, and Stefan Biffl. A flyweight UML modelling tool for software development in heterogeneous environments. In *Proc. 29th Euromicro Conference*. IEEE, 2003.

[5] William N. Bender. *Project-Based Learning: Differentiating Instruction for the 21st Century*. Sage, February 2021.

[6] Robert A. Bjork. Memory and metamemory considerations in the training of human beings. In *Metacognition*, pages 185–206. The MIT Press, April 1994.

[7] Grady Booch. *Object-oriented Analysis and Design with Applications*. Redwood City: Benjamin Cummings, 2nd edition, 1993.

[8] Emilio Boulianne. Impact of accounting software utilization on students' knowledge acquisition. *Journal of Accounting & Organizational Change*, 10(1):22–48, Jan 2014.

[9] Jonas Boustedt. Students' different understandings of class diagrams. *Computer Science Education*, 22(1):29–62, March 2012.

[10] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[11] Neil C. C. Brown, Jamie Ford, Pierre Weill-Tessier, and Michael Kölling. Quick fixes for novice programmers: Effective but under-utilised. In *The United Kingdom and Ireland Computing Education Research (UKICER) conference*, UKICER 2023, pages 1–7. ACM, September 2023.

[12] Antonio Bucchariones, Andrea Vazquez-Ingelmo, Gianluca Schiavo, Alicia Garcia-Holgado, Francisco Garcia-Penalvo, and Steffen Zschaler. Designing learning paths with open educational resources: A case study in model-driven engineering. In *18th Iberian Conference on Information Systems and Technologies*, 2023.

[13] A. Bucchiarone, J. Cabot, R. F. Paige, et al. Grand challenges in model-driven engineering: an analysis of the state of the research. *Software Systems Modelling*, 19:5–13, 2020.

[14] Antonio Bucchiarone, Maxime Savary-Leblanc, Xavier Le Pallec, Antonio Cicchetti, Sébastien Gérard, Simone Bassanelli, Federica Gini, and Annapaola Marconi. Gamifying model-based engineering: the PapyGame experience. *Software and Systems Modeling*, 22(4):1369–1389, March 2023.

[15] Loli Burgueño, Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sebastien Mosser, Richard F. Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabriele Taentzer, Antonio Vallecillo, and Manuel Wimmer. Contents for a model-based software engineering body of knowledge. *Software and Systems Modeling*, 18(6):3193–3205, Dec 2019.

[16] Alina Cernasev and David R. Axon. Research and scholarly methods: Thematic analysis. *JACCP: Journal of the American College of Clinical Pharmacy*, 6(7):751–755, 2023.

[17] Michel R. V. Chaudron, Werner Heijstek, and Ariadi Nugroho. How effective is UML modeling? an empirical perspective on costs and benefits. *Software and Systems Modeling*, 11(4):571–580, August 2012.

[18] Mauro Cherubini, Gina Venolia, Rob Deline, and Andrew J Ko. Let's go to the whiteboard: How and why software developers use drawings. In *Proc. CHI 2007*, pages 557–566, 2007.

[19] Michele Chinosi and Alberto Trombetta. BPMN: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134, 2012.

[20] Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sebastien Mosser, Richard F. Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabi Taentzer, Antonio Vallecillo, and Manuel Wimmer. How do we teach modelling and model-driven engineering? a survey. In *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 122–129, 2018.

[21] Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sebastien Mosser, Richard F. Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabi Taentzer, Antonio Vallecillo, and Manuel Wimmer. Towards a body of knowledge for model-based software engineering. In *MODELS'18 Companion Proceedings – Educators' Symposium*, 2018.

[22] David J. Cocovi-Solberg and Manuel Miró. CocoSoft: educational software for automation in the analytical chemistry laboratory. *Analytical and Bioanalytical Chemistry*, 407(21):6227–6233, Aug 2015.

[23] Brian Dobing and Jeffrey Parsons. How UML is used. *Communications of the ACM*, 49(5):109–113, May 2006.

[24] Brian Dobing and Jeffrey Parsons. Dimensions of UML diagram use: Practitioner survey and research agenda. In *Principle Advancements in Database Management Technologies*, pages 271–290. IGI Global, 2010.

[25] Romina Eramo, Martina Nolletti, Luigi Pomante, Laura Pasquale, and Dario Pascucci. Model-driven engineering for simulation models interoperability: A case study in space industry. *Software: Practice and Experience*, 54(6):1010–1033, 2024.

[26] Ana M. Fernández-Sáez, Michel R. V. Chaudron, and Marcela Genero. An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles. *Empirical Software Engineering*, 23(6):3281–3345, March 2018.

[27] Martin Grossman, Jay E. Aronson, and Richard V. McCarthy. Does UML make the grade? insights from the software development community. *Information and Software Technology*, 47(6):383–397, April 2005.

[28] Greg Guest, Kathleen MacQueen, and Emily Namey. *Applied Thematic Analysis*. SAGE Publications, Inc., Thousand Oaks, California, May 2014.

[29] Spencer E. Harpe. How to analyze Likert and other rating scale data. *Currents in Pharmacy Teaching and Learning*, 7(6):836–850, 2015.

[30] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[31] Reyhaneh Kalantari and Timothy C. Lethbridge. Characterizing UX evaluation in software modeling tools: A literature review. *IEEE Access*, 10:131509–131527, 2022.

[32] Reyhaneh Kalantari and Timothy C. Lethbridge. Unveiling developers' mindset barriers to software modeling adoption. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 737–746, Los Alamitos, CA, USA, 2023. IEEE Computer Society.

[33] Reyhaneh Kalantari, Julian Oertel, Joeri Exelmans, Satrio Adi Rukmono, Vasco Amaral, Matthias Tichy, Katharina Juhnke, Jan-Philipp Steghöfer, and Silvia Abrahão. Systematizing modeler experience (MX) in model-driven engineering success stories. *Software and Systems Modeling*, 23(4):821–832, July 2024.

[34] Jörg Kienzle, Steffen Zschaler, William Barnett, Timur Sağlam, Antonio Bucchiarone, Silvia Abrahão, Eugene Syriani, Dimitris Kolovos, Timothy Lethbridge, Sadaf Mustafiz, and Sofia Meacham. Requirements for modelling tools for teaching. *Software and Systems Modeling*, July 2024.

[35] Dimitris Kolovos and Antonio Garcia-Dominguez. The Epsilon playground. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 131–137. Association for Computing Machinery, 2022.

[36] Amruth N. Kumar and Rajendra K. Raj. Computer science curricula 2023 (cs2023): The final report. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2*, SIGCSE 2024, page 1867–1868, New York, NY, USA, 2024. Association for Computing Machinery.

[37] C.F.J. Lange, M.R.V. Chaudron, and J. Muskens. In practice: UML software architecture and design description. *IEEE Software*, 23(2):40–46, March 2006.

[38] Philip Langer, Tanja Mayerhofer, Manuel Wimmer, and Gerti Kappel. On the usage of UML: Initial results of analyzing open UML models. In *Modellierung 2014*, pages 289–304. Gesellschaft für Informatik e.V., Bonn, 2014.

[39] Agnes Lanusse, Yann Tanguy, Huascar Espinoza, Chokri Mraidha, Sebastien Gerard, Patrick Tessier, Remi Schnekenburger, Hubert Dubois, and François Terrier. Papyrus UML: an open source toolset for MDA. In Richard F. Paige, A. Hartman, and Arend Rensink, editors, *Proc 5th European conference Model driven architecture - foundations and applications (ECMDA-FA 2009*, volume 5562 of *Lecture Notes in Computer Science*, pages 1–4, Berlin and New York, 2009. Springer.

[40] Timothy C. Lethbridge, Andrew Forward, Omar Badreddin, Dusan Brestovansky, Miguel Garzon, Hamoud Aljamaan, Sultan Eid, Ahmed Husseini Orabi, Mahmoud Husseini Orabi, Vahdat Abdelzad, Opeyemi Adesina, Aliaa Alghamdi, Abdulaziz Algablan, and Amid Zakariapour. Umple: Model-driven development for open source and education. *Science of Computer Programming*, 208:102665, 2021.

[41] Andriy Levytskyy, Hans Vangheluwe, Leon J.M. Rothkrantz, and Henk Koppelaar. MDE and customization of modeling and simulation web applications. *Simulation Modelling Practice and Theory*, 17(2):408–429, 2009.

[42] Raina Mason, Simon, Brett A. Becker, Tom Crick, and James H. Davenport. A global survey of introductory programming courses. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2024, page 799–805, New York, NY, USA, 2024. Association for Computing Machinery.

[43] Tanja Mayerhofer, Philip Langer, Manuel Wimmer, and Gerti Kappel. xMOF: Executable DSMLs based on fUML. In *Software Language Engineering*, pages 56–75. Springer International Publishing, 2013.

[44] Davin Alexander McCall and Michael Kölling. A new look at novice programmer errors. *ACM Transactions on Computing Education*, 19(4):1–30, July 2019.

[45] Marcus Messer. Detecting when a learner requires assistance with programming and delivering a useful hint. In *Proceedings of the 15th International Conference on Educational Data Mining*, 2022.

[46] Hafeez Osman and Michel R.V. Chaudron. UML usage in open source software development : A field study. In *Proc. 3rd Int'l Workshop on Experiences and Empirical Studies in Software Modeling (EESSMod 2013)*, volume 1078, pages 23–32. CEUR, 2013.

[47] Mert Ozkaya and Ferhat Erata. A survey on the practical use of UML for different software architecture viewpoints. *Information and Software Technology*, 121:106275, May 2020.

[48] Jordan Peer, Yaniv Mordecai, and Yoram Reich. NLP4ReF: Requirements classification and forecasting: From model-based design to large language models. In *2024 IEEE Aerospace Conference*, pages 1–16. IEEE, March 2024.

[49] Marian Petre. UML in practice. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 722–731. IEEE, May 2013.

[50] Marian Petre. "no shit" or "oh, shit!": responses to observations on the use of UML in professional practice. *Software and Systems Modeling*, 13(4):1225–1235, August 2014.

[51] Elena Planas and Jordi Cabot. How are UML class diagrams built in practice? a usability study of two UML tools: Magicdraw and Papyrus. *Computer Standards & Interfaces*, 67:103363, January 2020.

[52] Gianna Reggio, Maurizio Leotta, and Filippo Ricca. Who knows/uses what of the UML: A personal opinion survey. In *Model-Driven Engineering Languages and Systems*, pages 149–165. Springer International Publishing, 2014.

[53] Gianna Reggio, Maurizio Leotta, Filippo Ricca, and Diego Clerissi. What are the used UML diagrams? a preliminary survey. In *Proc. 3rd Int'l Workshop on Experiences and Empirical Studies in Software Modeling (EESSMod 2013)*, volume 1078, pages 3–12. CEUR, 2013.

[54] Gianna Reggio, Maurizio Leotta, Filippo Ricca, and Diego Clerissi. What are the used UML diagram constructs? a document and tool analysis study covering activity and use case diagrams. In *Model-Driven Engineering and Software Development*, pages 66–83. Springer International Publishing, 2015.

[55] Miguel Reina, Eduardo Gabriel Guzmán-López, César Guzmán-López, Carlos Hernández-Garciadiego, María de los Ángeles Olvera-León, Mario Alfredo Garcia-Carrillo, Marco Antonio Tafoya-Rodríguez, Victor Manuel Ugalde-Saldívar, Itzel Guerrero-Ríos, Laura Gasque, Jorge M. del Campo, Daniela Franco-Bodek, Rolando Bernal-Pérez, Milton Medeiros, Armando Marín-Becerra, Héctor García-Ortega, Jesús Gracia-Mora, and Antonio Reina. Plata: Design of an online platform for chemistry undergraduate fully automated assignments. *Journal of Chemical Education*, 101(3):1024–1035, Mar 2024.

[56] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, 1994.

[57] Jonathan A. Smith. Interpretative phenomenological analysis and the psychology of health and illness. *The Journal of Positive Psychology*, 12(3):303–304, 2017.

[58] Riad Sonbol, Ghaida Rebdawi, and Nada Ghneim. The use of NLP-based text representation techniques to support requirement engineering tasks: A systematic mapping review. *IEEE Access*, 10:62811–62830, 2022.

[59] Thomas Stahl, Markus Voelter, and Krzysztof Czarnecki. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

[60] Gareth Terry, Nikki Hayfield, Victoria Clarke, Virginia Braun, et al. Thematic analysis. *The SAGE handbook of qualitative research in psychology*, 2(17-37):25, 2017.

[61] Hironori Washizaki, Maria-Isabel Sanchez-Segura, Juan Garbajosa, Steve Tockey, and Kenneth E Nidiffer. Envisioning software engineer training needs in the digital era through the SWEBOK V4 prism. In *2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 122–126, 2023.

[62] D. F Wood. Abc of learning and teaching in medicine: Problem based learning. *BMJ*, 326(7384):328–330, February 2003.

[63] Steffen Zschaler, Will Barnett, Artur Boronat, Antonio Garcia-Dominguez, and Dimitris Kolovos. The MDENet education platform: Zero-install directed activities for learning MDE. *Software and Systems Modelling (SoSyM)*, 2025.