Systematic Review of UML Diagramming Software Tools for Higher Education Software Engineering Courses

Yuting Lu
lu7560138@163.com
University of Edinburgh School of Informatics
Edinburgh, United Kindom

Cristina Adriana Alexandru
Cristina.Alexandru@ed.ac.uk
University of Edinburgh School of Informatics
Edinburgh, United Kindom

ABSTRACT

The Unified Modelling Language (UML) is a set of graphical notations underpinned by a single meta-model that aid in the description and design of software systems. Learning to draw UML diagrams is an important objective of higher education (HE) software engineering (SE) courses. To this end, UML diagramming software tools can facilitate the application of the UML notation in drawing complex diagrams collaboratively between students. Numerous UML diagramming tools are available on the market nowadays, which makes it difficult for academics and students to choose the best tool for their needs, and no thorough evaluation of such tools for education is available. This project conducted a systematic review of UML diagramming tools that are freely available or available for a trial, from the point of view of meeting desirable diagramming and collaboration features for use in HE SE courses. This paper presents the methodology and results of this systematic review, before concluding on the best tools overall and for different features, and proposing a set of guidelines for the design of better UML diagramming tools for HE SE courses.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

higher education, software engineering, UML diagramming tools, systematic review

ACM Reference Format:
Yuting Lu and Cristina Adriana Alexandru. 2023. Systematic Review of UML Diagramming Software Tools for Higher Education Software Engineering Courses. In *The United Kingdom and Ireland Computing Education Research (UKICER) conference (UKICER 2023), September 07–08, 2023, Swansea, Wales Uk*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3610969.3611123

1 INTRODUCTION

The Unified Modelling Language (UML) [51] is often seen as the de facto standard for modelling software applications [35, 45, 53]. It helps design, describe, and document software systems, but also document and communicate software or business processes, by means of diagrams. UML has seen widespread use in the software industry, and is considered one of the most popular software modelling languages [40–42]. Given increased focus on student employability in higher education (HE) [27], it has also found its way into the computer science curricula [29, 35, 49]. In particular, it plays an important role as a teaching tool in software engineering (SE) courses. Apart from preparing students to become software engineers, it is also important for teaching them basic SE concepts, most notably abstraction [29] and visual modeling [35].

The popularity of UML is partly attributed to its large tool support [42]. UML diagramming tools (also known as "UML" followed by "diagramming software", "diagram software/tools", "builder software", or "modelers/modellers") can help their users 'engineer' software artifacts by easily building and manipulating UML diagrams [25]. While drawing UML diagrams is also possible on paper, such tools can - amongst other things[1] - speed it up, make diagrams cleaner and easier to change, facilitate the incremental development of complex diagrams, support the use of the UML notation [29, 42]. For students, they can help them concentrate on the semantics, gain experience for work in the industry, and are seen as essential for a hands-on learning of SE [29, 39]. Moreover, some of them offer collaboration features. Collaboration is essential for active learning [24, 33], and in terms of teamwork in SE [39]. Moreover, it can enable students and teachers to work together if not co-located.

Despite the potential advantages of UML diagramming software tools for SE HE courses, tens of tools are available which offer various functionality [25, 42]. Previous research indicated that most of them do not offer support for accurate UML diagramming [25]. In particular, some do not respect the UML notation. Moreover, being primarily built for practitioners with experience in SE, they may not be easy to pick up for students learning SE for the first time.

Evaluations of UML diagramming tools exist, but mainly from the perspective of meeting the needs of SE practitioners. Many of these evaluations considered a small subset of tools which they compared in terms of their features (e.g. [31, 32, 46, 54]). A much more complete and recent review was offered in 2019 [42], which systematically identified 58 tools and then evaluated them from the perspective of meeting highly important requirements for SE practitioners. These requirements go far beyond what HE SE courses teach in UML, including model analysis, transformation and export, scripting, project and knowledge management. Interesting work was also presented by [23] and [50], who evaluated tools by

[1] e.g. code generation and analysis, which will not be discussed in this article

conducting experiments with students. However, they were not focusing on the use of these tools in education, but were only using students as proxies for software engineers, and only evaluated a subset of tools. All of the above evaluations included commercial tools the cost of which might be prohibitive in education. Finally, two papers that are more related to this work are [28] and [52]. The first aimed to classify UML diagramming tools: from the perspective of compliance with the UML 2.1.1 standard, without focus on the users [28]. The second was the only paper we could find focusing on HE, but merely exemplified different types of UML tools [52]. Both of these papers are now very old, and both the tools and the UML standard have been updated since then.

The absence of a systematic evaluation of UML diagramming tools from the perspective of meeting the needs of academics and students in HE SE courses makes it difficult for HE academics to choose a suitable UML diagramming tool for their SE course. Leaving the choice of tool to the students can result in poor choices causing inequality and affecting their learning and progression.

This project aimed to address the following research questions:

RQ1: What are the beneficial features that UML diagramming software tools should include for use in HE SE courses?

- **RQ1.1**: What diagram types and notation should be supported?

- **RQ1.2**: What collaboration features should be supported?

RQ2:What are the best free (version) or free trial UML diagramming software tools in terms of meeting the beneficial features?

RQ3: What guidelines can be derived for the better design of UML diagramming tools for HE SE courses?

We focused on tools that academics could access for free, without needing to go through lengthy university licensing processes.

2 METHODOLOGY

This project consisted of the following steps:

(1) **Literature review (RQ1)**: To address RQ1, the literature was searched for UML diagram types that are useful to be taught in HE SE courses, and beneficial collaboration features for HE. For the identified diagram types, one of the authors who is an academic with over 10 years of experience in teaching UML in the School of Informatics of the University of Edinburgh contributed the typically taught notation subset, which were then checked against the UML standard [51].
(2) **Plan a systematic review of UML diagramming tools for use in HE SE courses**: This step involved detailing the methods that would be used in the systematic review, using an adaptation of the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA [38, 43]) framework for systematically reviewing software tools by Michaelidou [37]. The features corresponding to the diagram types, basic UML notation for each, and collaborative features identified in Step 1 became data items for the systematic review.
(3) **Conduct the systematic review**: This involved the identification of relevant literature sources, then the identification of free (version) or free trial UML diagramming tools from them, before evaluating these tools one-by-one using the data items from the plan, and giving them scores. Finally, total scores were calculated for each feature, each diagram type, and each tool overall, and conclusions were drawn about the best tools overall and per feature (**RQ2**).
(4) **Provide guidelines (RQ3)**. Following the results, guidelines were proposed for the design of better UML diagramming tools for HE SE courses.

3 STEP 1: LITERATURE REVIEW (RQ1)

In 2013, Reggio et al. [48] conducted a survey of books (on UML or using it as a notation), UML diagramming tools, Information Technology HE courses on UML, and tutorials on UML for practitioners to date to determine the most used types of UML diagram. They concluded that class, activity, sequence, use case and state machine diagrams were the most used, in order of priority. The following year, they extended the work with a personal opinion survey with 275 respondents from both academia and industry, and concluded that the most widely used types of diagrams in practice were sequence, class, use case, state machine, and activity diagrams, in order of popularity [47]. In 2015, Fernández-Sáez et al. [30] compared the use of different diagram types in the industry. They analysed 178 software maintenance projects from 12 countries, and concluded that the most used in order of popularity were class, use case, sequence and activity diagrams. The same appeared in the list of most produced diagram types during software development, with the difference that component diagrams were mentioned as the fourth most popular diagram. In a much more recent study from 2021, Koç et al. [34] systematically reviewed the use of UML diagrams in SE research and found that the most mentioned types in SE research between 2000 and 2019 were class, activity, use case, sequence and state machine diagrams, in order of popularity.

The above clearly suggest that *class*, *use case*, *activity*, and *sequence diagrams* have been widely used both in the SE industry and SE research. While Fernández-Sáez et al. [30] found *state machine diagrams* were used much less, this is contradicted by other and more recent findings, therefore they were also added to the list. Despite not finding surveys on most used UML diagram types in HE, the authors strongly believe that the popularity of these 5 types in SE industry and research makes them useful to be taught in HE SE courses, and thus considered for the systematic review.

The author who is an experienced academic teaching SE in the School of Informatics of the University of Edinburgh was then consulted on the typical notation used for the the 5 diagram types across the various SE courses offered by the school. These notations were checked against the UML standard [51]. They are the following:

(1) For use case diagrams: use case, actor, interaction between actors, actor with several use cases, several actors with same use case, system boundary.
(2) For class diagrams: class, association, association label, association multiplicity (exact number, range, arbitrary or undefined number), role on association, association navigability possible and not possible, attribute area in class, attribute (name and type), operations area in class, operation (operator name, parameter type, return type), generalisation, interface, interface realisation by 'lollipop' notation, interface realisation by dashed arrow from class to interface with hollow

arrowhead, class using the interface, abstract class by label on class icon, abstract class by name of class in italics, abstract operation, visibility (public, private, protected).

(3) For sequence diagrams: actor, object, lifeline, activation, processing as part of the activation, message, message name and parameters, return message, return value, found message, object sending message to itself, nested activation, object returning message to itself, optional behaviour, alternative behaviour, guard condition, iterative behaviour, iteration clause, creation of object, deletion of object.

(4) For activity diagrams: start marker/ activity, end marker/ activity, activity, activity edge, join and fork synchronisation bar, guard condition, decision diamond, swimlane.

(5) For state diagrams: start marker/ state, state, transition, event, transition keeping the object in the same state, guard condition, action as a result of a transition, area in state for actions, action always happening when entering the state, action always happening before exiting the state, end marker/ state.

Nowadays software development often involves teamwork and interaction with stakeholders, and therefore effort is put into incorporating student collaboration into SE courses [26, 39]. When students work remotely, such collaboration can be organised online, using synchronous or asynchronous means. Synchrony emulates better face-to-face interaction, and there is proof that it is a better choice for students' feelings of belonging, positive affect and cognitive processes [44]. Therefore, we decided to test the UML diagramming tools for basic synchronous collaboration features, inspired by the work of Lomas et al. [36]: sharing the diagram, multiple collaborators, synchronous collaboration on different parts of the diagram, synchronous collaboration on the same part of the diagram, seeing the position of each other's cursor on the screen, displaying the list of all users who can edit the diagram, displaying the number of users who can edit the diagram, displaying the users who are currently editing, comments (leaving and replying to them), presence of modifications log (i.e. who changed what when).

4 STEP 2: PLANNING THE SYSTEMATIC REVIEW

PRISMA [38, 43] is a framework for planning and reporting systematic reviews and meta analyses, which ensures that they are adequately documented and thoroughly conducted. The "Methods" of its checklist helped lay out the plan for the systematic review, which was adapted like in Michaelidou [37] for reviewing literature sources only for the purpose of extracting names of UML diagramming tools, before reviewing the tools themselves hands-on. The following were the main items of the plan:

- **Eligibility criteria**: were split into:
  - For literature sources: being in English, reliable (articles or books published with publication name, producing company's official documents or web pages, or marketing research websites), accessible for free or using a University of Edinburgh licence.
  - For tools: being described as UML diagramming tools or their synonyms (see Section 1); being in English; being updated within the last 5 years; being free/ having a free version/ free trial; if mobile or desktop application being securely runnable (as verified by the operating system); being compatible with Windows, Linux and MacOS operating systems; being usable to draw diagrams (i.e. using shapes, symbols, text); not collecting financial information when creating an account; using the https protocol as a security guarantee when creating an account; no errors when creating an account and logging in.
- **Information sources**: Google [11] and Google Scholar [12].
- **Search strategy**: use as keywords "UML diagramming tools" and its synonyms separated by "OR" in both search engines: ”UML diagramming tools” OR ”UML diagram tools” OR ”UML diagramming software” OR ”UML diagram software” OR “UML builder software” OR “UML modeler” OR ”UML modeller”. For the search, first change the search settings to only get English results, then provide the keywords, un-tick "include citations" (only needed in Google Scholar [12]), deactivate the automatic omission of similar entries from the last page of results (only needed in Google [11]), and note the number of results now accurately shown on the last page.
- **Selection process**: To filter the identified literature sources, check them against the eligibility criteria for literature sources in their given order, including re-checking that the source is in English as the search engine language filters sometimes miss results in other languages. If any literature source eligibility criterion is not met, dismiss that source. For each approved source, use control + F or command + F on a macBook, with the terms "UML", "tool", and then "software", and check the words surrounding the results to find the names of possible UML diagramming tools. Next, filter the identified tools by checking each against the tool eligibility criteria in their order and dismissing any tools which fail any criterion.
- **Data items**: Consist first of all of the typical notation subset for the 5 popular UML diagram types together with the collaboration features identified in Step 1. For the notations, add sub-data items for the possibility of applying them and whether they are provided directly as selectable symbols (e.g. for use case diagrams, system boundary being provided as a symbol with this name rather than needing to use a rectangle symbol followed by adding a text label). Moreover, add data items for additional functionality: offering dedicated support for each of the 5 popular diagram types (i.e. having a specific area of notations for each), and allowing diagrams to be exported in different formats. The first would enhance usability as compared to notation being unstructured per diagram type, while the second is seen as critical for producing documentation. This resulted in different numbers of (sub-) data items per each diagram type, and a total of 165 (sub-) data items. Prepare one example diagram of each type containing the features from the data items list.
- **Data collection process**: Download and install each of the tools in the final list before creating an account and logging in. Attempt to reproduce each example diagram, thus checking the possibility to apply the notation data items. Then, test the export functionality by exporting one random diagram. Finally, download, install and create a second account on the tool on a different machine, and test the data items on collaboration by interacting between the two machines.

- **Synthesis of results**: For each of the notations in the example diagrams, if it can be reproduced in the tool (either as a selectable symbol or composition of other symbols), give a score of 1 for the corresponding data item under the "possibility to apply" sub-data item, if not give a score of 0. If it received 1, and the tool supports a selectable symbol for that notation, also give a score of 1 for this second sub-data item. For the 'dedicated support for the diagram type' data item, only give a score of 1 if there is a dedicated set of symbols for that diagram type (0 otherwise), but attempt representing the example diagrams even if this criterion received a 0 (e.g. symbols were not grouped, or they needed to be obtained by combining other symbols like shapes and arrows). Give the tool scores of 1 for each supported export format and collaboration feature, and 0 for each unsupported one. Synthesise the results per diagram type (out of the total score given for all of its (sub-) data items), per feature and overall (out of the total score of 165).

5 STEP 3: CONDUCTING THE SYSTEMATIC REVIEW

Following the plan's search strategy, the initial search obtained 268 results in Google [11] and 669 results in Google Scholar [12]. Following the selection process, we obtained 138 results in Google and 108 in Google Scholar after filtering them with the literature source eligibility criteria. From them, 46 tool names were identified and recorded in a table. Checking each tool against the tool eligibility criteria reduced the list to 20 tools. Whether each tool met each eligibility criterion, the final inclusion/exclusion decision, as well as notes explaining the decision, were included in the same table. The most frequent reason for which tools were excluded was not being usable on macOS.

For the evaluation, several other tables were created: one for each diagram type, and one for the additional features to be tested. Each table contained on the rows the names of the included tools, and on the columns the data items with any sub-data items and with space for adding notes (reason why certain notation could not be represented, including that it required a paid-for tool version; how it was represented if selectable symbol unavailable). The scores were added to the (sub-) data item cells as explained in the synthesis of results, as soon as the notation from the example diagram and additional functionality were attempted. Total scores were calculated per table. Finally, overall scores were calculated for each tool.

6 RESULTS

6.1 Best Tools Overall and Per Feature (RQ2)

Table 1 presents the total scores per diagram type, additional feature, and overall, for each of the 20 tools. Overall, the top 3 best UML diagramming tools for HE SE courses were found to be Edraw [6] (score 131/165) , Visual Paradigm [21] (score 122/165) and MagicDraw [14] (score 121/165). In particular, Edraw and Visual Paradigm were the best tools for representing use cases, both gaining the maximum score. MagicDraw was the best tool for representing state diagrams, gaining the maximum score. It was also, equally with UMLetino [20], best for representing activity diagrams, with a score of 16/18. What MagicDraw was missing for activity diagrams was support for writing the label on the guard conditions (a text box needed overlaying), and for adding swimlanes (line shapes needed to be used). In UMLetino, what was missing for activity diagrams was the correct representation for start and end markers. All 3 top tools had dedicated areas of symbols for all the five tested diagram types (i.e. maximum score). Edraw supported the most types of diagram export functionalities (JPEG, PDF, HTML, Microsoft Word, SVG, XLS, PPT, Visio, VSDX, Tiff, PS, EPS). Finally, Visual Paradigm was the best at collaboration features, jointly with Lucidchart [13], both scoring 15/16. For both of them, what was missing was a modifications log, which in Lucidchart was available in the paid for version. In fact, this feature was missing in all the evaluated tools apart from Edraw which supported it. However, Edraw was missing other features like seeing the position of other users' cursor on the screen and displaying the number of current editors.

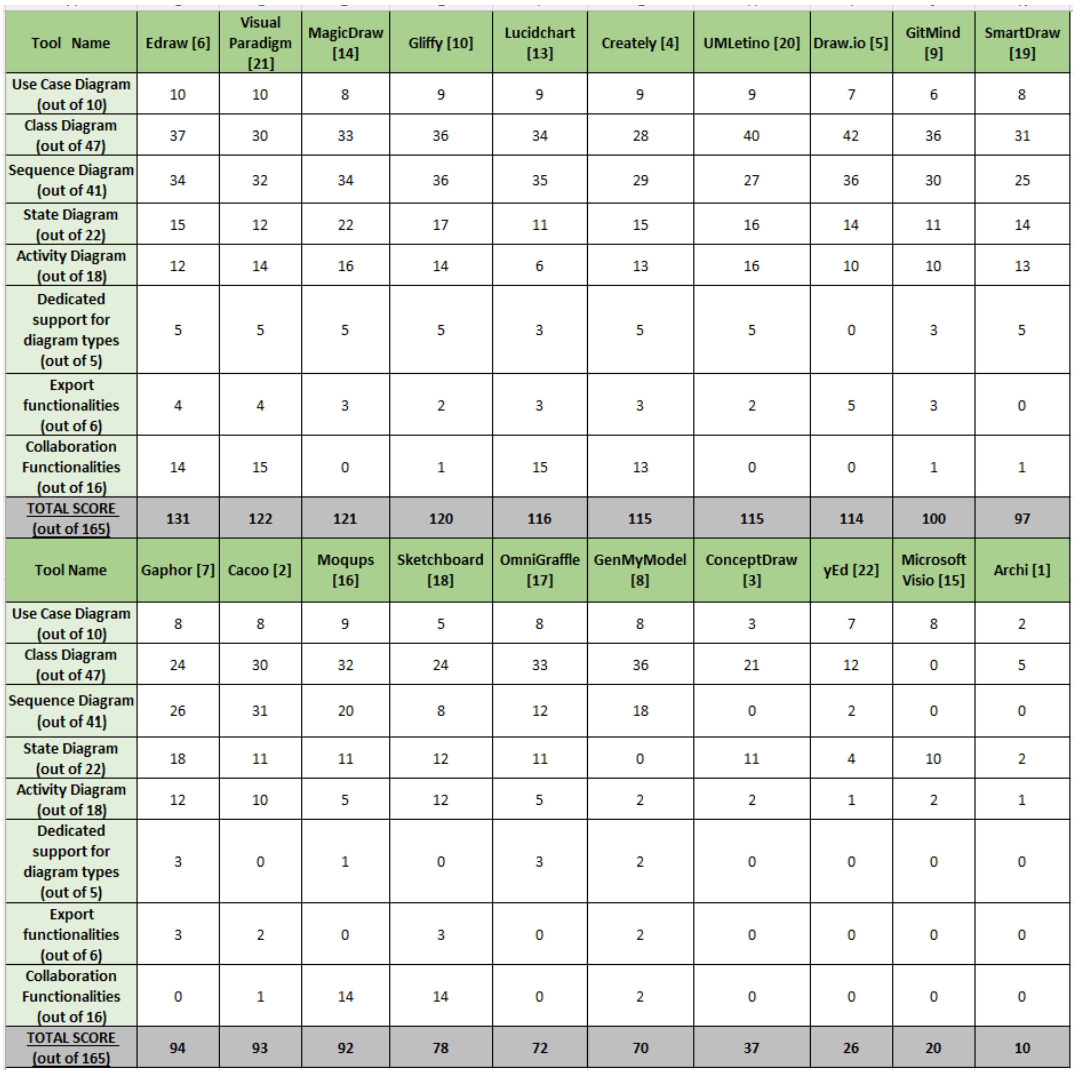
Where the top 3 tools performed worse were class diagrams and sequence diagrams. For class diagrams, Draw.io [5] was the best tool overall, with a score of 42/47. Edraw [6] was only in third position (score 37/47), preceded by UMLetino [20] (score 40/47). Draw.io was missing association labels (missing for most tools), association navigability not possible notation, a dedicated notation for abstract classes (neither of two possible notations) and operations, all of which needed to be emulated by using text boxes. For sequence diagrams, Gliffy [10] and Draw.io were on the first place, with a score of 36/41. The overall top tools Edraw and MagicDraw [14] were only in third place (score 34/41), preceded by Lucidchart (score 35/41). What was missing in Draw.io were the processing as part of the activation (not even easily representable using other symbols), correct notation for messages (parameters not supported and needed adding with a text box), support for guard conditions and iteration clauses on frames (needed adding with text box).

Overall for diagramming, considering the summed scores for all diagram types, MagicDraw [14] came on top. Apart from being the best for state and activity diagrams, and third in sequence diagrams, its difference from the top scorers for use case diagrams was only 2/12. However, for class diagrams it performed worse, ranking sixth (33/47). In particular, it did not offer dedicated support for association labels, roles and multiplicity, notation for navigability not being possible, operations, attributes, interfaces and interface realisation, abstract classes and operations using the italics notation.

6.2 Guidelines (RQ3)

Taking detailed evaluation results for each diagram type in turn, it became apparent that support for dedicated UML notation was frequently missing and the user needed to use a combination of symbols to represent it. In particular, for use case diagrams, support for representing the interaction actor-use case and system boundary was missing for the majority of the tools. For class diagrams, there was no support available in any of the tools for abstract classes (neither of the notations), abstract operations, association navigability not being possible. Moreover, few tools supported the abstract class notation in italics (only UMLetino [20]), association labels (3/20), roles on associations (2/20), interface realisation lollipop notation (3/20), association multiplicity (6/20), class using the interface (6/20) and attributes (9/20). For sequence diagrams, no

Table 1: Evaluation results



Tool Name	Edraw [6]	Visual Paradigm [21]	MagicDraw [14]	Gliffy [10]	Lucidchart [13]	Creately [4]	UMLetino [20]	Draw.io [5]	GitMind [9]	SmartDraw [19]
Use Case Diagram (out of 10)	10	10	8	9	9	9	9	7	6	8
Class Diagram (out of 47)	37	30	33	36	34	28	40	42	36	31
Sequence Diagram (out of 41)	34	32	34	36	35	29	27	36	30	25
State Diagram (out of 22)	15	12	22	17	11	15	16	14	11	14
Activity Diagram (out of 18)	12	14	16	14	6	13	16	10	10	13
Dedicated support for diagram types (out of 5)	5	5	5	5	3	5	5	0	3	5
Export functionalities (out of 6)	4	4	3	2	3	3	2	5	3	0
Collaboration Functionalities (out of 16)	14	15	0	1	15	13	0	0	1	1
TOTAL SCORE (out of 165)	**131**	**122**	**121**	**120**	**116**	**115**	**115**	**114**	**100**	**97**

Tool Name	Gaphor [7]	Cacoo [2]	Moqups [16]	Sketchboard [18]	OmniGraffle [17]	GenMyModel [8]	ConceptDraw [3]	yEd [22]	Microsoft Visio [15]	Archi [1]
Use Case Diagram (out of 10)	8	8	9	5	8	8	3	7	8	2
Class Diagram (out of 47)	24	30	32	24	33	36	21	12	0	5
Sequence Diagram (out of 41)	26	31	20	8	12	18	0	2	0	0
State Diagram (out of 22)	18	11	11	12	11	0	11	4	10	2
Activity Diagram (out of 18)	12	10	5	12	5	2	2	1	2	1
Dedicated support for diagram types (out of 5)	3	0	1	0	3	2	0	0	0	0
Export functionalities (out of 6)	3	2	0	3	0	2	0	0	0	0
Collaboration Functionalities (out of 16)	0	1	14	14	0	2	0	0	0	0
TOTAL SCORE (out of 165)	**94**	**93**	**92**	**78**	**72**	**70**	**37**	**26**	**20**	**10**

tool offered support for processing as part of an activation, and few tools offered support for message names and parameters (1/20), iteration clauses (4/20), guard conditions on opt/alt frames (5/20), found messages (5/20), conditional alt (8/20), actors (8/20). For activity diagrams, few tools offered support for start marker/activity and end marker/activity (only MagicDraw [14]), guard conditions and swimlanes (only UMLetino [20]), decision diamond (7/20), join and fork synchronisation bars (9/20). Finally, in state diagrams few tools offered support for representing guard conditions and an action as a result of a transition (only MagicDraw [14]), events (2/20), action always happening before exiting the state (2/20), area in state for actions (3/20), action always happening when entering the state (3/20), transition to same state (4/20) transition between two states (5/20). Support was often deemed as missing because the symbols proposed were incorrect according to the UML standard e.g. arrow with empty arrowhead instead of normal arrow, picture of a person instead of a stick figure. Under these circumstances, it was sometimes not even possible to make use of other symbols to represent the wanted notation. This happened for representing interface realisation with the lollipop notation in class diagrams (only in 5/20 tools it was possible), and in sequence diagrams for representing processing as part of the activation (it was impossible in all tools), return message to itself (5/20 tools), found messages and conditional alt frames (8/20 tools). Therefore, ***designers should consider adding dedicated support for the notation that was emphasised as necessary for HE SE courses in this paper***, and that ***this support should involve accurate notation according to UML [51]***.

While use case diagrams were explicitly supported by more than half of the tools (13/20), this was not the case for any of the other diagram types: class diagrams by 10, sequence diagrams by 9, activity diagrams by 8 and state diagrams by 10 out of 20 tools. As all of these types are popular in practice, and explicit areas of symbols for them can enhance usability, ***designers should consider adding dedicated support for the 5 different diagram types***.

In what concerns export functionalities, we found that at least half of the tools allowed exporting diagrams in PNG (12/20), JPEG (11/20) and PDF (11/20) formats, yet they would rarely support exports to other formats such as HTML (2/20), DOC (2/20), XLS (only Edraw [6]) and PPT (only Edraw). As these diagrams are often to be used to produce documentation, and students often need to provide them in coursework reports, we would argue that ***support for more export formats would be beneficial***.

Where tools performed poorly overall was collaboration features: out of the 20 tools, less than half allowed any kind of sharing of diagrams; only 6 of them allowed multiple collaborators, displaying their list, synchronous collaboration (on different or same part of diagram), and comments (making and replying); only 4 of them allowed seeing others' cursor on the screen; only 5 of them allowed displaying the users currently editing, of which 4 also displayed their number; the modification log was only supported by Edraw [6] and the paid for version of Lucidchart [13]. As synchronous collaboration is beneficial for students in general, and in particular for SE students who need to develop teamwork, we recommend ***adding to UML diagramming tools more of the synchronous collaboration features mentioned in this paper***.

7 CONCLUSION

This project aimed to identify the beneficial features of UML diagramming software tools for use in HE SE courses (RQ1), before evaluating existing free (version) and free trial such tools based on these features to conclude on the best tools overall and per feature (RQ2) and derive a set of guidelines for the better design of such tools for HE SE courses (RQ3). To respond to RQ1, a literature review was conducted which led to the conclusion that use case, class, sequence, activity and state diagrams are used most in the industry and research, and so are also desirable to be taught in HE SE courses and be supported by these tools (RQ1.1). One of the authors who is an academic with over 10 years of experience in teaching UML in the School of Informatics of the University of Edinburgh then contributed typically taught notations for these diagram types, which were checked against the UML standard and should also be supported by the tools (RQ 1.1). Desirable collaboration features were the result of another literature review (RQ1.2). For the evaluation, a systematic review was planned using the Methods section in a version of the PRISMA statement [38, 43] modified for systematically reviewing tools [37]. The plan was closely followed to search and filter literature sources, identify names of UML diagramming tools from them, filter these tools, then download and install each tool before creating an account and trying it out to represent an example diagram of each type including all the notations, export one of the diagrams and test the collaboration features. This resulted in the evaluation of 20 tools, which were given scores depending on whether they included each of the desirable features. Overall, results for RQ2 revealed that Edraw [6], Visual Paradigm [21] and MagicDraw [14] are the top 3 best tools for HE SE courses. Edraw and Visual Paradigm scored highest for representing use cases. MagicDraw was best at representing state diagrams and activity diagrams, the latter equally with UMLetino [20]. Draw.io [5] was best at representing class diagrams, and jointly with Gliffy [10] at representing sequence diagrams. The top 3 best tools all had dedicated support for the 5 diagram types. Edraw offered the most functionality for exporting diagrams, and Visual Paradigm was the best at collaboration features. To answer to RQ3, detailed evaluation results were inspected to conclude that there is a need for more dedicated support for the 5 different diagram types, dedicated support for the UML notation taught in HE SE courses, for considering this notation accurately, for support for more export formats and more synchronous collaboration features.

To the authors' knowledge, this is the first work which considers the needs of HE SE academics and students in evaluating UML diagramming tools. We hope that this paper will be used as a guide for choosing better such tools, which can facilitate students' learning of UML and improve their employability.

However, the work also has some limitations. First of all, notation features were based on the experience of a single academic. Future work could conduct a study with SE academics from universities across the UK in view of confirming and/or enriching these features. Secondly, the tools were only evaluated based on their features, while non-functional requirements like performance and usability were not considered. Future work could involve studies with academics and students to also assess such aspects. Finally, only free (version) and free trial tools were assessed, while universities with SE specialisms might consider purchasing licences for commercial tools. Future work could involve discussions with university management to decide what is a reasonable expense for such tools, before including commercial tools in the evaluation.

REFERENCES

[1] 2023. Archi. https://www.archimatetool.com/ Accessed on 23 May 2023.
[2] 2023. Cacoo. https://nulab.com/cacoo/ Accessed on 23 May 2023.
[3] 2023. ConceptDraw DIAGRAM. https://www.conceptdraw.com/products/drawing-tool Accessed on 23 May 2023.
[4] 2023. Creately. https://creately.com/ Accessed on 23 May 2023.
[5] 2023. Draw.io. https://app.diagrams.net/ Accessed on 23 May 2023.
[6] 2023. EDraw. https://www.edrawsoft.com/ Accessed on 23 May 2023.
[7] 2023. Gaphor. https://gaphor.org/en/ Accessed on 23 May 2023.
[8] 2023. GenMyModel. https://www.genmymodel.com/ Accessed on 23 May 2023.
[9] 2023. Git Mind. https://gitmind.com/ Accessed on 23 May 2023.
[10] 2023. Gliffy. https://www.gliffy.com/ Accessed on 23 May 2023.
[11] 2023. Google. https://www.google.com/ Accessed on 23 May 2023.
[12] 2023. Google Scholar. https://scholar.google.com/ Accessed on 23 May 2023.
[13] 2023. Lucidchart. https://www.lucidchart.com/pages/ Accessed on 23 May 2023.
[14] 2023. MagicDraw. https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/ Accessed on 23 May 2023.
[15] 2023. Microsoft Visio. https://www.microsoft.com/en-gb/microsoft-365/visio/flowchart-software Accessed on 23 May 2023.
[16] 2023. Moqups. https://moqups.com/ Accessed on 23 May 2023.
[17] 2023. OmniGraffle. https://www.omnigroup.com/omnigraffle Accessed on 23 May 2023.
[18] 2023. Sketchboard. https://sketchboard.io/ Accessed on 23 May 2023.
[19] 2023. SmartDraw. https://www.smartdraw.com/ Accessed on 23 May 2023.
[20] 2023. UMLetino. https://www.umletino.com/ Accessed on 23 May 2023.
[21] 2023. Visual Paradigm. https://www.visual-paradigm.com/ Accessed on 23 May 2023.
[22] 2023. yEd. https://www.yworks.com/products/yed Accessed on 23 May 2023.
[23] Anna E Bobkowska and Krzysztof Reszke. 2005. Usability of UML Modeling Tools. *Software engineering: evolution and emerging technologies* 5 (2005).
[24] Meghan E Borg, Kaitlyn M Butterfield, Eileen Wood, Huan Huan Zhang, and Sabrina Pinto. 2021. Investigating the impacts of personality on the use and perceptions of online collaborative platforms in higher education. *SN Social Sciences* 1 (2021), 1–22.
[25] Loli Burgueño, Antonio Vallecillo, and Martin Gogolla. 2018. Teaching UML and OCL models and their validation to software engineering students: an experience report. *Computer Science Education* 28, 1 (2018), 23–41.

[26] Shailey Chawla. 2019. Collaborative learning strategies in software engineering course. (2019).

[27] Ming Cheng, Olalekan Adekola, JoClarisse Albia, and Sanfa Cai. 2022. Employability in higher education: a review of key stakeholders' perspectives. *Higher Education Evaluation and Development* 16, 1 (2022), 16–31.

[28] Holger Eichelberger, Yilmaz Eldogan, and Klaus Schmid. 2009. A comprehensive survey of UML compliance in current modelling tools. *Software Engineering 2009* (2009).

[29] Gregor Engels, Jan Hendrik Hausmann, Marc Lohmann, and Stefan Sauer. 2006. Teaching UML is teaching software engineering is teaching abstraction. In *Satellite Events at the MoDELS 2005 Conference: MoDELS 2005 International Workshops Doctoral Symposium, Educators Symposium Montego Bay, Jamaica, October 2-7, 2005 Revised Selected Papers 8*. Springer, 306–319.

[30] Ana M Fernández-Sáez, Danilo Caivano, Marcela Genero, and Michel RV Chaudron. 2015. On the use of UML documentation in software maintenance: Results from a survey in industry. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 292–301.

[31] Heena and Ranjna Garg. 2011. A comparative study of UML tools.. In *ACAI*. 1–4.

[32] Lena Khaled. 2009. A comparison between UML tools. In *2009 second international conference on environmental and computer science*. IEEE, 111–114.

[33] Ann L Kieser and Fay Ortiz Golden. 2009. Using online office applications: Collaboration tools for learning. *Distance Learning* 6, 1 (2009), 41.

[34] Hatice Koç, Ali Mert Erdoğan, Yousef Barjakly, and Serhat Peker. 2021. UML diagrams in software engineering research: a systematic literature review. In *Proceedings*, Vol. 74. MDPI, 13.

[35] Ludwik Kuzniarz and Miroslaw Staron. 2006. Best practices for teaching UML based software development. In *Satellite Events at the MoDELS 2005 Conference: MoDELS 2005 International Workshops Doctoral Symposium, Educators Symposium Montego Bay, Jamaica, October 2-7, 2005 Revised Selected Papers 8*. Springer, 320–332.

[36] Cyprien Lomas, Michael Burke, and Carie L Page. 2008. Collaboration tools. *Educause learning initiative* 2, 11 (2008).

[37] Elina Michaelidou. 2019. *Review of Web-Based Audience Response Systems Used in Higher Education*. Master's thesis. School of Informatics University of Edinburgh.

[38] David Moher, Alessandro Liberati, Jennifer Tetzlaff, Douglas G Altman, and the PRISMA Group*. 2009. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *Annals of internal medicine* 151, 4 (2009), 264–269.

[39] Jeff Offutt. 2013. Putting the engineering into software engineering education. *IEEE software* 30, 1 (2013), 96–96.

[40] Mert Ozkaya. 2016. What is software architecture to practitioners: A survey. In *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. Ieee, 677–686.

[41] Mert Ozkaya. 2018. Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? *Information and Software Technology* 95 (2018), 15–33.

[42] Mert Ozkaya. 2019. Are the UML modelling tools powerful enough for practitioners? A literature review. *IET Software* 13, 5 (2019), 338–354.

[43] Matthew J Page, Joanne E McKenzie, Patrick M Bossuyt, Isabelle Boutron, Tammy C Hoffmann, Cynthia D Mulrow, Larissa Shamseer, Jennifer M Tetzlaff, Elie A Akl, Sue E Brennan, et al. 2021. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *International journal of surgery* 88 (2021), 105906.

[44] Amy T Peterson, Patrick N Beymer, and Ralph T Putnam. 2018. Synchronous and asynchronous discussions: Effects on cooperation, belonging, and affect. *Online Learning* 22, 4 (2018), 7–25.

[45] Neil Pitman. 2005. *UML 2.0 in a Nutshell: A Desktop Quick Reference*. O'Reilly.

[46] T Rani and S Garg. 2013. Comparison of different UML tool: Tool approach. *International Journal Of Engineering And Computer Science* 2, 6 (2013), 1900–1908.

[47] Gianna Reggio, Maurizio Leotta, and Filippo Ricca. 2014. Who knows/uses what of the UML: A personal opinion survey. In *Model-Driven Engineering Languages and Systems: 17th International Conference, MODELS 2014, Valencia, Spain, September 28–October 3, 2014. Proceedings 17*. Springer, 149–165.

[48] Gianna Reggio, Maurizio Leotta, Filippo Ricca, and Diego Clerissi. 2013. What are the used UML diagrams? A Preliminary Survey.. In *EESSMod@ MoDELS*. 3–12.

[49] Rebecca Reuter, Theresa Stark, Yvonne Sedelmaier, Dieter Landes, Jürgen Mottok, and Christian Wolff. 2020. Insights in students' problems during UML modeling. In *2020 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 592–600.

[50] Safdar Aqeel Safdar, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan. 2015. Empirical evaluation of UML modeling tools–a controlled experiment. In *Modelling Foundations and Applications: 11th European Conference, ECMFA 2015, Held as Part of STAF 2015, LAquila, Italy, July 20-24, 2015. Proceedings 11*. Springer, 33–44.

[51] B Selic, C Bock, S Cook, P Rivett, T Rutt, E Seidewitz, and D Tolbert. 2015. OMG Unified Modeling Language (Version 2.5). *Object Management Group, MA, USA* (2015).

[52] Harold H Smith. 2004. On tool selection for illustrating the use of UML in system development. *Journal of Computing Sciences in Colleges* 19, 5 (2004), 53–63.

[53] Suriya Sundaramoorthy. 2022. *UML Diagramming: A Case Study Approach*. CRC Press.

[54] Weng Jie Thong and Mohamed Ariff Ameedeen. 2019. A survey of UML tools. In *Proceedings of the International Conference on Data Engineering 2015 (DaEng-2015)*. Springer, 61–70.