# From Difficulties to Functional Requirements - Deriving Requirements from Literature about Tool-supported Teaching of UML Diagrams in Software Engineering Education


Florian Huber
Kempten University of Applied Sciences
Kempten, Germany
florian.huber@hs-kempten.de

Tobias Eigler
Kempten University of Applied Sciences
Kempten, Germany
tobias.eigler@hs-kempten.de

Georg Hagel
Kempten University of Applied Sciences
Kempten, Germany
georg.hagel@hs-kempten.de

Christian Wolff
University of Regensburg
Regensburg, Germany
Christian.Wolff@informatik.uni-regensburg.de



## ABSTRACT

With the increasing complexity of software systems, it is becoming more and more relevant for students in software engineering education to learn how to elicitate requirements from customers or stakeholders and visualize them in a diagram. This continues to present different challenges to novice modellers. To counteract these, educators and researchers have developed support tools. These offer various functionalities to address different challenges. However, it is unclear what functionality needs to be provided to address all of these issues and not just individual ones. Therefore, this paper extracts such functional requirements and summarizes them. It also shows how they can be translated into activities during an exercise lesson.



## CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Software and its engineering** → *Software creation and management*.

## KEYWORDS

Software Engineering Education, UML, Tool-Supported Teaching, Student Requirements

**ACM Reference Format:**
Florian Huber, Tobias Eigler, Georg Hagel, and Christian Wolff. 2023. From Difficulties to Functional Requirements - Deriving Requirements from Literature about Tool-supported Teaching of UML Diagrams in Software Engineering Education. In *ECSEE 2023: European Conference on Software Engineering Education (ECSEE 2023), June 19–21, 2023, Seeon/Bavaria, Germany*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3593663.3593672




*ECSEE 2023, June 19–21, 2023, Seeon/Bavaria, Germany*



ACM ISBN 978-1-4503-9956-2/23/06…$15.00
https://doi.org/10.1145/3593663.3593672


## 1 INTRODUCTION

An important part in the education of computer scientists is software engineering [16, 18]. In this context, students often receive textual requirements descriptions from an educator alongside with the task to develop a diagram that matches the specified functionalities. For diagram generation, the Unified Modeling Language (UML) is widely used [18]. However, learning how to abstract such software requirements and transfer them into a diagram is not an easy task for students [27, 40, 42]. Since many graduates will be confronted with customers' or stakeholders' requirements in their daily work, it is important for them to overcome these hurdles.

Students' misconceptions and difficulties within the modeling process have already been in focus of numerous publications (e.g. [31]). To address these issues, some educators and researchers have developed tools that are able to support their students during the modeling process. These were summarized in a systematic literature review (SLR) [17]. Along with the five most named difficulties for students in diagram generation, the various tools are explained. Despite the fact that all these tools have similar goals, they provide a variety of functionalities to tackle different problems. However, it is unclear from the SLR what functionalities a software tool should provide in order to support a students' entire learning process and not only specific parts of it. The large number of developed tools and additional publications in this topic illustrates the interest of the research community and especially educators. The contribution of this paper is a summary of the software requirements described in the papers included in the SLR. These are intended to serve as building blocks for the development of new applications that take a holistic approach and seek to address many students' problems at once. It is important to note that the focus is on functional requirements [22]. Non-functional requirements [22] (such as the response time of a server) are not described.

After the extraction process, the functional requirements are translated into activities, that either students or educators could perform during an exercise lesson.

The following sections are structured as follows: section 2 covers related work. Afterwards, functional requirements are extracted from the described list of papers in section 3. Finally a short conclusion is given in section 5.

## 2 RELATED WORK

As can be seen from the previous section, a SLR about tool-supported teaching of UML diagrams [17] serves as a starting point for this publication. The reviewed sources [1–5, 7, 8, 10, 12–15, 19–21, 23–26, 28–30, 32, 34–39, 41, 43, 45, 46] will be used to extract the functional requirements. The publications mentioned in the related work of [17] also relate to this one.

Authors in [31] performed a SLR about publications that cover students' errors in UML artefacts. Furthermore they provide an overview about problems that occur during the actual modeling process. Focusing on difficulties and not functionalities to solve those, this publications' intention differs from the goals described in [31].

A similar goal was pursued by [9]. The result is a catalogue of students' mistakes when modeling different types of UML diagrams. This publication also does not consider the software-based options that can reduce these errors.

Some publications focus on the challenges of teaching (UML) modeling (e.g. [6]). As software engineering education as a whole is an active field of research, there is also literature on its challenges in general (e.g. [11]). Those difficulties and challenges might have been a motivation for researchers to develop various supporting tools. However, they do not deliver insights in the actual functionalities that were implemented in order to overcome those issues. The authors of this work have not been able to find a publication that covers this topic. We follow [44] who advocates research on tool development and usage in software engineering education.

## 3 REQUIREMENTS ELICITATION

The selected papers all suggest a classroom setting, where students receive some sort of a textual requirements description provided by an educator. They have to solve the given exercise with the support of a tool. Such a scenario will serve as the basis for this work. It contains two actors: an educator and a student. Therefore, it is useful in this section to capture functional requirements for both groups. In addition, functional system requirements must be described. They include functionalities such as calculating a grade on base of a defined algorithm.

The results of the SLR show that there are two main approaches. Most of the included publications compare a students' diagram to a given set of rules or a sample solution. A smaller group uses different visualization techniques (like 3D or Virtual Reality) in order to support their students. The requirements from both groups are therefore divided into *basic* and *visualization* requirements. As already mentioned, this is a summary based on a modular principle and not a list of mandatory functionalities. A breakdown of the requirements into clusters supports this idea and makes it possible, for example, to add or omit visualizations.

For the elicitation process, all selected papers were read and the described tool functionalities listed. Once a specific functionality has been mentioned three times, it is included in our list. All requirements are formulated following the suggestions of [33].

**Table 1: Basic requirements for educators**

| ID | Requirement |
|---|---|
| BE1 | The system shall provide an educator with the ability to open a session, where students can login and work on the same exercise simultaneously. |
| BE2 | The system shall provide an educator with the ability to enter an information about the intended learning outcome alongside with the exercise that shall be imported. |
| BE3 | The system shall provide an educator with the ability to import an exercise text with UML diagram requirements descriptions. |
| BE4 | The system shall provide an educator with the ability to import a sample solution in form of a UML diagram. |
| BE5 | The system shall provide an educator with the ability to create a sample solution within a simple modeling editor. |
| BE6 | The system shall provide an educator with the ability to specify which UML components shall be compared to each other. |
| BE7 | The system shall provide an educator with the ability to view questions that students entered during a session. |
| BE8 | The system shall provide an educator with the ability to answer questions that students entered during a session. |
| BE9 | The system shall provide an educator with the ability to view a statistical overview about the students activities during the modeling process. |
| BE10 | The system should provide an educator with the ability to view submitted students' diagrams alongside with the differences to a sample solution and the grade the system calculated. |

### 3.1 Requirements from an Educators' Perspective

This subsection describes all the extracted requirements from an educators' perspective. They are summarized in table 1. Because from this point of view, there is no support required, no visualization requirements were raised.

### 3.2 Requirements from a Students' Perspective

This subsection describes all the extracted requirements from a students' perspective. Table 2 contains the basic ones, while table 3 summarizes requirements related to visualization.

### 3.3 System requirements

This subsection describes all system requirements. The basic ones are listed in table 4. Because visualizations have to be generated (for example a 2D UML diagram has to be converted into a 3D model), table 5 summarizes the visual system requirements.

**Table 2: Basic requirements for students**

| ID | Requirement |
|---|---|
| BST1 | The system shall provide a student with the ability to login to a session and work on a specific exercise provided by an educator. |
| BST2 | The system shall provide a student with the ability to view the intended learning outcome of the displayed exercise. |
| BST3 | The system shall provide a student with the ability to view the exercise text provided by an educator during the session. |
| BST4 | The system should provide a student with the ability to create an UML diagram within a simple editor. |
| BST5 | The system shall provide a student with the ability to ask questions about a specific exercise. |
| BST6 | The system shall be able to assess the students' diagrams during the modeling process. |
| BST7 | The system shall provide a student with the ability to view all the relevant components within the requirements descriptions. |
| BST8 | The system shall provide a student with the ability to import an UML diagram in XMI, JPG or PNG format. |
| BST9 | The system shall provide a student with the ability to submit an UML diagram for comparison with a sample solution provided by an educator. |
| BST10 | The system shall provide a student with the ability to view the detected differences in the comparison of two diagrams. |
| BST11 | After two UML diagrams have been compared to each other, the system shall provide a student with the ability to view a list of theoretical topics that should be revised in order to resolve detected mistakes. |
| BST12 | After the student received feedback, the system shall provide a student with the ability to submit an improved version of the diagram and restart the comparison process. |

**Table 3: Visualization Requirements for Students**

| ID | Requirement |
|---|---|
| VST1 | The system should provide a student with the ability to write code for a displayed exercise and automatically generate a related UML diagram. |
| VST2 | The system should provide a student with the ability to create an UML diagram and automatically generate the related code. |
| VST3 | The system should provide a student with the ability to search for keywords within the diagram. |
| VST4 | The system should provide a student with the ability to navigate through an AR, 3D or 3D VR model. |
| VST5 | The system should provide a student with the ability to run the visualized code and display the runtime states of the individual objects. |
| VST6 | The system should provide a student with the ability to interact (create, move, edit or delete) diagram elements within the 2D, AR, 3D or 3D VR view. |

**Table 4: Basic system requirements**

| ID | Requirement |
|---|---|
| BSY1 | The system shall log the students' activities during the modeling process. |
| BSY2 | The system shall be able to export a modelled UML diagram into an XMI file. |
| BSY3 | The system shall be able to convert imported diagrams into a defined structure, which allows the comparison of them. |
| BSY4 | The system shall be able to grade the students' diagrams according to a grading algorithm. |
| BSY5 | The system shall be able to detect synonyms for class, attribute and method names in a students' solution. |
| BSY6 | The system should create statistics about the differences detected in multiple comparison processes. |
| BSY7 | The system should be able to generate feedback after a comparison process. |
| BSY8 | The system shall be able to compare two imported diagrams to each other and outline the differences. |
| BSY9 | The system shall be able to apply a predefined set of rules to check an imported students' solution and outline detected deviations. |

## 4 REQUIREMENTS VISUALIZATION

Using activity diagrams, this section illustrates how the described requirements in a classroom situation could be used from an educator's and a student's perspective. This is to show when and in which order the described requirements are applicable.

### 4.1 Educators' Activities

Through which activities an educator could go during an exercise lesson, using a tool that meets all of the above requirements, is shown in Figure 1.

After starting the tool, an 'educator view' can be opened. It allows to create a new session. Educators can enter an intended learning outcome, a requirements text, a sample solution and information about which of those diagram elements shall be compared to the ones from a student's solution. When the session is started, the exercise participants can join and work on the provided contents. If questions arise, they can be entered into the tool and answered by

**Table 5: Visualization system requirements**

| ID | Requirement |
|---|---|
| VSY1 | The system should be able to create a 3D diagram from a 2D diagram created by a student. |
| VSY2 | The system should be able to visualize different levels of granularity within the diagrams (classes, attributes, methods). |
| VSY3 | The system should be able to create an AR diagram from a 2D diagram created by a student. |
| VSY4 | The system should be able to display code an related UML diagrams alongside each other. |
| VSY5 | The system should be able to create a 3D VR model from a 2D diagram created by a student. |
| VSY6 | The system should be able to highlight a selected line of code within the UML diagram. |

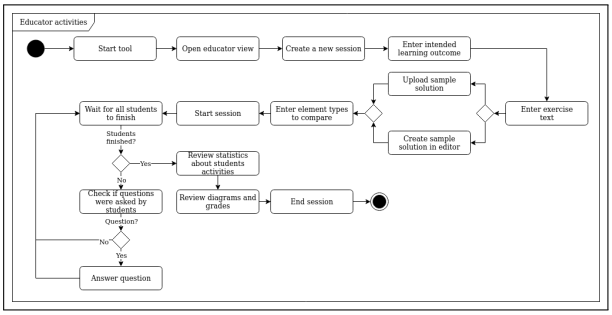


**Figure 1: Educators activities during the exercise**

the educator. When all students have finished modeling, statistics can be reviewed as well as the students' diagrams alongside the grades the system calculated.

## 4.2 Students' Activities

Activities a student could perform using such a tool during an exercise lesson, are visualized in Figure 2.

When the tool has been started, a student can login to a provided session. The information about the intended learning outcome and the requirements description can now be read. Afterwards, students can either choose to take an external modeling tool or use the provided editor. During the diagram generation process, they can use the described help functions (e.g. ask questions) or display the diagram using one of the visualization techniques. When the students have finished, they can submit their diagram and receive feedback alongside with suggestions, on which theoretical aspects to revise for improvement. If desired, the submitted solution can be redesigned or the exercise be exited.

# 5 CONCLUSION

Especially for novices in software engineering education, learning how to elicitate and convert requirements into an UML diagram is challenging. Therefore, educators and researches have made an effort to develop several supporting tools to improve their students

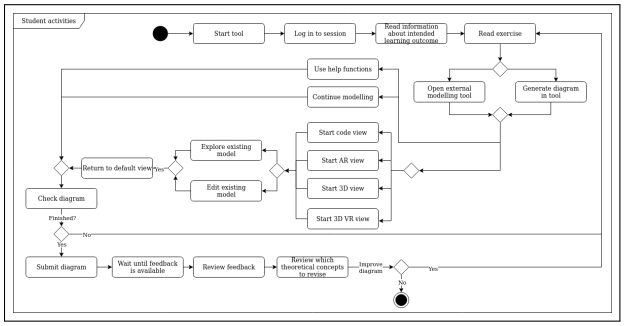


**Figure 2: Student activities during the exercise**

learning success. However, with these heterogenous solutions, it is unclear what functionality a tool needs to provide to address all difficulties and not just solve individual problems. This paper has extracted all the functional requirements from a selected set of publications (see [17]) and shows how they can be translated into activities during practice sessions. Since this is a summary from the literature, this list is not yet complete. It is important to emphasize that not every requirement presented needs to be met in an educational context in order to teach software engineering efficiently. In future work, students and teachers will be interviewed about further requirements.

# REFERENCES

[1] Sohail Alhazmi, Charles Thevathayan, and Margaret Hamilton. 2021. *Learning UML Sequence Diagrams with a New Constructivist Pedagogical Tool: SD4ED.* Association for Computing Machinery, New York, NY, USA, 893–899. https://doi.org/10.1145/3408877.3432521

[2] Norhayati Mohd Ali, Novia Admodisastro, Saádah Hassan, and Mohammed Sadeq Abdullah Saeed. 2018. UML DIAGRAM LEARNING TOOL. *UNIVERSITY CARNIVAL on e-LEARNING (IUCEL) 2018* (2018), 408.

[3] Weiyi Bian, Omar Alam, and Jörg Kienzle. 2019. Automated Grading of Class Diagrams. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C).* 700–709. https://doi.org/10.1109/MODELS-C.2019.00106

[4] Daria Bogdanova. 2019. Towards Personalized Feedback in a Smart Learning Environment For Teaching Conceptual Modelling. In *2019 13th International Conference on Research Challenges in Information Science (RCIS).* 1–5. https://doi.org/10.1109/RCIS.2019.8876983

[5] Younes Boubekeur, Gunter Mussbacher, and Shane McIntosh. 2020. *Automatic Assessment of Students' Software Models Using a Simple Heuristic and Machine Learning.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3417990.3418741

[6] Barbara Bracken. 2003. Progressing from Student to Professional: The Importance and Challenges of Teaching Software Engineering. *J. Comput. Sci. Coll.* 19, 2 (dec 2003), 358–368.

[7] Yuanfang Cai, Daniel Iannuzzi, and Sunny Wong. 2011. Leveraging design structure matrices in software design education. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T).* 179–188. https://doi.org/10.1109/CSEET.2011.5876085

[8] Yuanfang Cai, Rick Kazman, Ciera Jaspan, and Jonathan Aldrich. 2013. Introducing tool-supported architecture review into software design education. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T).* 70–79. https://doi.org/10.1109/CSEET.2013.6595238

[9] Stanislav Chren, Barbora Buhnova, Martin Macak, Lukas Daubner, and Bruno Rossi. 2019. Mistakes in UML Diagrams: Analysis of Student Projects in a Software Engineering Course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).* 100–109. https://doi.org/10.1109/ICSE-SEET.2019.00019

[10] Kleinner Farias and Bruno C. da Silva. 2020. *What's the Grade of Your Diagram? Towards a Streamlined Approach for Grading UML Diagrams.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3417990.3420052

[11] Carlo Ghezzi and Dino Mandrioli. 2006. The Challenges of Software Engineering Education. In *Software Engineering Education in the Modern Age*, Paola Inverardi and Mehdi Jazayeri (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 115–127.

[12] Robert W. Hasker. 2011. UMLGrader: An Automated Class Diagram Grader. *J. Comput. Sci. Coll.* 27, 1 (oct 2011), 47–54.

[13] Robert W. Hasker and Mike Rowe. 2011. UMLint: Identifying Defects in UML Diagrams. In *2011 ASEE Annual Conference & Exposition*. ASEE Conferences, Vancouver, BC. https://peer.asee.org/18929.

[14] Pavel Herout and Premek Brada. 2016. UML-Test Application for Automated Validation of Students' UML Class Diagram. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEE&T)*. 222–226. https://doi.org/10.1109/CSEET.2016.33

[15] Florian Huber and Georg Hagel. 2020. Work-in-Progress: Towards detection and syntactical analysis in UML class diagrams for software engineering education. In *2020 IEEE Global Engineering Education Conference (EDUCON)*. 3–7. https://doi.org/10.1109/EDUCON45650.2020.9125244

[16] Florian Huber and Georg Hagel. 2022. Semi-automatic generation of textual exercises for software engineering education. In *2022 IEEE Global Engineering Education Conference (EDUCON)*. 51–56. https://doi.org/10.1109/EDUCON52537.2022.9766802

[17] Florian Huber and Georg Hagel. 2022. Tool-supported teaching of UML diagrams in software engineering education - A systematic literature review. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*. 1404–1409. https://doi.org/10.23919/MIPRO55190.2022.9803560

[18] Florian Huber and Georg Hagel. 2022. Work-In-Progress: Converting textual software engineering class diagram exercises to UML models. In *2022 IEEE Global Engineering Education Conference (EDUCON)*. 1–3. https://doi.org/10.1109/EDUCON52537.2022.9766593

[19] Yuta Ichinohe, Hiroaki Hashiura, Takafumi Tanaka, Atsuo Hazeyama, and Hiroshi Takase. 2019. Effectiveness of Automated Grading Tool Utilizing Similarity for Conceptual Modeling. In *Knowledge-Based Software Engineering: 2018*, Maria Virvou, Fumihiro Kumeno, and Konstantinos Oikonomou (Eds.). Springer International Publishing, Cham, 117–126.

[20] Mantas Jurgelaitis, Lina Čeponienė, Jonas Čeponis, and Vaidotas Drungilas. 2019. Implementing gamification in a university-level UML modeling course: A case study. *Computer Applications in Engineering Education* 27, 2 (2019), 332–343. https://doi.org/10.1002/cae.22077 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cae.22077

[21] Jan Knobloch, Jonas Kaltenbach, and Bernd Bruegge. 2018. Increasing Student Engagement in Higher Education Using a Context-Aware Q&A Teaching Framework. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training* (Gothenburg, Sweden) *(ICSE-SEET '18)*. Association for Computing Machinery, New York, NY, USA, 136–145. https://doi.org/10.1145/3183377.3183389

[22] Gerald Kotonya and Ian Sommerville. 1998. *Requirements Engineering: Processes and Techniques* (1st ed.). Wiley Publishing.

[23] Stephan Krusche, Nadine von Frankenberg, Lara Marie Reimer, and Bernd Bruegge. 2020. An Interactive Learning Method to Engage Students in Modeling. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training* (Seoul, South Korea) *(ICSE-SEET '20)*. Association for Computing Machinery, New York, NY, USA, 12–22. https://doi.org/10.1145/3377814.3381701

[24] Stan Kurkovsky. 2015. Teaching Software Engineering with LEGO Serious Play. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15)*. Association for Computing Machinery, New York, NY, USA, 213–218. https://doi.org/10.1145/2729094.2742604

[25] Timothy C. Lethbridge, Gunter Mussbacher, Andrew Forward, and Omar Badreddin. 2011. Teaching UML using umple: Applying model-oriented programming in the classroom. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*. 421–428. https://doi.org/10.1109/CSEET.2011.5876118

[26] Salisu Modi, Hanan Abdulrahman Taher, and Hoger Mahmud. 2021. A Tool to Automate Student UML diagram Evaluation. *Academic Journal of Nawroz University* 10, 2 (Jun. 2021), 189–198. https://doi.org/10.25007/ajnu.v10n2a1035

[27] Juan Carlos Muñoz-Carpio, Michael Cowling, and James Birt. 2018. Framework to Enhance Teaching and Learning in System Analysis and Unified Modelling Language. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. 91–98. https://doi.org/10.1109/TALE.2018.8615284

[28] Dominique Py, Ludovic Auxepaules, and Mathilde Alonso. 2013. Diagram, a Learning Environment for Initiation to Object-Oriented Modeling with UML Class Diagrams. *Journal of Interactive Learning Research* 24, 4 (October 2013), 425–446. https://www.learntechlib.org/p/41241

[29] Tobias Reischmann and Herbert Kuchen. 2016. Towards an E-Assessment Tool for Advanced Software Engineering Skills. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '16)*. Association for Computing Machinery, New York, NY, USA, 81–90. https://doi.org/10.1145/2999541.2999550

[30] Rebecca Reuter, Florian Hauser, Daniel Muckelbauer, Theresa Stark, Erika Antoni, Jürgen Mottok, and Christian Wolff. 2019. Using Augmented Reality in Software Engineering Education? First insights to a comparative study of 2D and AR UML modeling. In *Proceedings of the 52nd Hawaii International Conference on System Sciences 2019*.

[31] 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)*. 592–600. https://doi.org/10.1109/EDUCON45650.2020.9125110

[32] Claudia Susie C. Rodrigues, Cláudia M. L. Werner, and Luiz Landau. 2016. VisAr3D: An Innovative 3D Visualization of UML Models. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 451–460.

[33] Chris Rupp, Stefan Queins, and SOPHISTen. 2014. *Requirements-Engineering und -Management*. Carl Hanser Verlag GmbH & Co. KG. 570 pages.

[34] Rijul Saini, Gunter Mussbacher, Jin L.C. Guo, and Jöerg Kienzle. 2019. Teaching Modelling Literacy: An Artificial Intelligence Approach. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 714–719. https://doi.org/10.1109/MODELS-C.2019.00108

[35] Rijul Saini, Gunter Mussbacher, Jin L.C. Guo, and Jöerg Kienzle. 2019. Teaching Modelling Literacy: An Artificial Intelligence Approach. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 714–719. https://doi.org/10.1109/MODELS-C.2019.00108

[36] Marcelo Schots, Claudia Susie C. Rodrigues, Claudia Werner, and Leonardo Murta. 2010. A Study on the Application of the PREViA Approach in Modeling Education. In *2010 XXIX International Conference of the Chilean Computer Science Society*. 96–101. https://doi.org/10.1109/SCCC.2010.29

[37] Gayane Sedrakyan and Monique Snoeck. 2012. Technology-Enhanced Support for Learning Conceptual Modeling. In *Enterprise, Business-Process and Information Systems Modeling*, Ilia Bider, Terry Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer, and Stanisław Wrycza (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 435–449.

[38] J. Soler, I. Boada, F. Prados, J. Poch, and R. Fabregat. 2010. A web-based e-learning tool for UML class diagrams. In *IEEE EDUCON 2010 Conference*. 973–979. https://doi.org/10.1109/EDUCON.2010.5492473

[39] D.R. Stikkolorum, Truong Ho-Quang, and M.R.V. Chaudron. 2015. Revealing Students' UML Class Diagram Modelling Strategies with WebUML and LogViz. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*. 275–279. https://doi.org/10.1109/SEAA.2015.77

[40] D. R. Stikkolorum, F. Gomes de Oliveira Neto, and M. R. V. Chaudron. 2018. Evaluating Didactic Approaches Used by Teaching Assistants for Software Analysis and Design Using UML. In *Proceedings of the 3rd European Conference of Software Engineering Education* (Seeon/ Bavaria, Germany) *(ECSEE'18)*. Association for Computing Machinery, New York, NY, USA, 122–131. https://doi.org/10.1145/3209087.3209107

[41] Dave R. Stikkolorum, Peter van der Putten, Caroline Sperandio, and Michel R. V. Chaudron. 2019. Towards Automated Grading of UML Class Diagrams with Machine Learning. In *BNAIC/BENELEARN*.

[42] Sylvia Stuurman, Harrie Passier, and Erik Barendsen. 2016. Analyzing Students' Software Redesign Strategies. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '16)*. Association for Computing Machinery, New York, NY, USA, 110–119. https://doi.org/10.1145/2999541.2999559

[43] Bingyang Wei, Harry S. Delugach, Eduardo Colmenares, and Catherine Stringfellow. 2016. A Conceptual Graphs Framework for Teaching UML Model-Based Requirements Acquisition. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEE&T)*. 71–75. https://doi.org/10.1109/CSEET.2016.35

[44] Christian Wolff. 2015. The Case for Teaching "Tool Science". Taking Software Engineering and Software Engineering Education beyond the Confinements of Traditional Software Development Contexts. In *2015 IEEE Global Engineering Education Conference (EDUCON)*. 932–938. https://doi.org/10.1109/EDUCON.2015.7096085

[45] Jeong Yang, Youlg Lee, and Kai H. Chang. 2017. Initial Evaluation of JaguarCode: A Web-Based Object-Oriented Programming Environment with Static and Dynamic Visualization. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. 152–161. https://doi.org/10.1109/CSEET.2017.32

[46] Jeong Yang, Young Lee, David Hicks, and Kai H. Chang. 2015. Enhancing object-oriented programming education using static and dynamic visualization. In *2015 IEEE Frontiers in Education Conference (FIE)*. 1–5. https://doi.org/10.1109/FIE.2015.7344152