What Works in MDE Teaching? Insights from Students' Perspectives

Leydis Lamoth Borrero[1,*,†], Jenny Ruiz de la Peña[1,*,†] and Monique Snoeck[2,†]

[1] University of Holguin, XX Anniversary Ave., Holguin, Cuba
[2] KU Leuven, Naamsestraat 69, 3000, Leuven, Belgium

Abstract

This study examines the effectiveness of a teaching approach and supporting tools used in a Model-Driven Engineering (MDE) course. Drawing on students' perspectives, it investigates key factors such as ease of use, learnability, satisfaction, perceived complexity, effectiveness, and overall fitness for purpose. With respect to the toolset, the analysis focuses on usability, feedback, and the quality of the learning experience. The course design was informed by best practices in MDE education, combining the MERODE method with the Merlin tool and its prototyper to reduce complexity. A project-based learning strategy was adopted to reflect real-world development contexts. The study involved 36 students from Informatics Engineering and Computer Science programs at two Cuban universities. Data were collected through surveys and analyzed statistically. Results indicate generally positive perceptions of both the course and the tools, while also identifying areas for improvement, particularly in tool learnability and interface design. Beyond technical proficiency, the course aims to instill a deeper understanding of the role of models in software development. These findings may serve as a useful reference for educators seeking to design or enhance MDE courses.

Keywords

student perception, model-driven engineering, MERODE, software engineering education [1]

1. Introduction

The term "engineering" in Software engineering (SE) refers to the fact that building software should be seen as a disciplined engineering practice, using well-defined practices to ensure outcome quality [1]. A key practice in SE is "modelling": the creation of abstract representations of software artefacts. A fundamental element of SE education programs is therefore instructing effective modelling practices, to ensure the development of correct, maintainable, efficient, and usable software [2].

A variety of approaches have been used to train software engineers in using models. However, when students are training to use models only as a documentation artefact, they are at a disadvantage when entering the job market. There is a need to train students to turn their models into real executable systems [3]. Model-Driven Engineering (MDE) has been the focus of considerable research as a novel paradigm for software development, which has led to its growing prominence [4]. This paradigm relies on the use of models throughout the software development process [5] and the transformation of models into other models with varying levels of abstraction. These models offer a high-level perspective of the software and a means of abstracting platform complexity [6]. The application of MDE enables a more efficient SE process, resulting in higher-quality and reliable software [7]. Other authors propose its use to provide software engineers with the insights, techniques, and tools to mitigate the difficulty of designing software for complex systems [8], [9], [10], [11]. Therefore, the integration of MDE within the curricula of Computer Science (CS)/SE programs would be a recommended practice, as it equips software engineers with the necessary skills

ER2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France
* Corresponding author.
† These authors contributed equally.
llamothb@uho.edu.cu (L. Lamoth Borrero); jruizp@uho.edu.cu (J. Ruiz de la Peña); monique.snoeck@kuleuven.be (M. Snoeck)
0000-0002-6170-0989 (L. Lamoth Borrero); 0000-0002-1371-6353 (J. Ruiz de la Peña); 0000-0002-3824-3214 (M. Snoeck)

to navigate the intricacies of complex systems and to adapt to the rapid advancements in technology.

In order to introduce MDE into teaching practice, modelling competences must first be properly placed within the curricula. Despite of the potential advantages of integrating MDE into teaching practices, a review of the existing literature reveals a few studies that explicitly addressed its integration into CS/SE disciplines. MDE is considered a complex and challenging subject to teach due to its unique characteristics regarding the design of learning activities and the abundance of suitable available tools [11]. Further research into the pedagogy of MDE is key to address the gaps between its conceptualization and its practical implementation. This paper aims to contribute to MDE pedagogy by equipping professors and students with course design guidelines that foster a comprehensive understanding of MDE concepts and its appreciation in professional settings.

In the absence of compelling evidence supporting a singular approach to teaching MDE that has emerged as the most effective, this paper identifies best practices from the literature for designing a course for teaching MDE and evaluate it from a student's perspective. The newly designed course was offered to the bachelor's programs Informatics Engineering and Computer Science at the University of Holguin (UHo) and at the Central University from Las Villas (UCLV), Cuba, respectively. To evaluate the course, a survey was given to the 36 participants. The survey provides insights into the students' perceptions on ease of use and usefulness. Based on the results of this evaluation, we derived lessons learned for future enhancements to the tooling and for teaching MDE.

The remainder of this paper is organized as follows. Section 2 presents Related Works. Section 3 presents the methodology for course design and data collection. Section 4 presents and discusses the results of the evaluation of the course design and tooling. Section 5 concludes the paper.

2. Related Work

Several studies have explored the impact of MDE on engineering curricula, drawing from researchers' experiences, analyzing various teaching approaches, with some also considering the use of MDE in companies. This section reviews related work and concludes by identifying best practices from the literature for designing an MDE course and evaluating it from the students' perspective.

In [12] the authors present a survey about integrating MDE into a software design class where the students used a modelling language to generate a communication system. The goal of the survey was to obtain empirical evidence on how MDE helps with understanding modelling concepts from the students' perspective. A key finding is the lack of good tools support: technical difficulties stand in the way of learning MDE. In [13], the authors report their experience with teaching generative MDE-based techniques for several years, combined with variability modelling in the context of software product line engineering. Also, these authors conclude that it is crucial to tame the complexity in order to successfully introduce MDE. In [14], the authors report on teaching MDE with a code generator-based approach developed by the students. Rather than teaching theory or how to manage a tool, this approach focused on teaching the underlying concepts. This way of working highlighted the importance of hands-on experience in MDE, leading to deeper understanding and skill development among both, students with some experience in MDE and novel students.

A course for undergraduate students is described in [15]. Model-Driven Development (MDD) was introduced as a tool for solving real problems from the perspective of programmers. This course explains techniques and principles and did not focus on supporting tools. Using good motivation, examples emphasizing industrial practice and the fact, that students could see the whole approach broken into small interconnected parts where positive elements, whereas technical difficulties were experienced as problematic. The state-of-the-practice of teaching MDD was analyzed in [16]. They performed a pilot study and provide suggestions to teach this approach: having external stakeholders participating in student projects, using motivating examples and case studies in tutorials or providing labs for training specific models before starting the work on a project, and using industrial projects.

In [17] the authors report university students' perceptions of software modelling. Through a multiple case study involving 5 courses across 3 universities, and analyzing interviews with students and instructors, the research identifies that although students recognize the value modelling, their

understanding is hindered by unclear assignment expectations, insufficient and irregular feedback, and limited familiarity with problem domains. The authors recommend enhancing education by providing more substantive feedback beyond syntactical issues and using problem domains familiar to students. The authors of [3] discuss the critical role of modelling in software engineering education and the need for effective tools to support student learning. Their research aims to assist both educators and students in selecting appropriate modelling tools by identifying the features most valued by students: students value tools that were easy to install and learn, supported key notations, and provided code generation capabilities, among others. Conversely, common criticisms included insufficient feedback, slow performance, difficulty in model creation, and overall complexity.

The authors of [18] surveyed 47 instructors teaching MDE about their course content, the modelling tools they use, and the factors that influence learning outcomes. According to the survey's results, the most important best practices are to make learners understand the benefits of MDE, using a small setting to start with, with a non-critical pilot project, where adoption efforts are useful, ensuring all participants in the project are able to read and understand the models.

Teaching MDD and traditional software development in a Master's program is compared in [19]. Considering that real project involvement is key for learning and applying MDE, enhancing adoption and success, they use a problem-based approach to evaluate attitudes towards MDD, knowledge gained, system quality, and developer satisfaction. Key lessons are that understanding the benefits of MDD happens through comparing with traditional software development, and students focus more on UI design than on functionality. For problem-based learning, teaching theoretical concepts and practical skills faced the general challenge of engaging students in the classroom [20].

Table 1
Overview of identified best practices

#	Best practice	Proposed by
BP1	Employ a modelling language to generate a system	[12]
BP2	Avoid technical difficulties through good tool support	[3], [12], [13]
BP3	Use a code generator	[3], [14]
BP4	Use real-life problems, motivating examples from the industry	[16], [15]
BP5	Explain in detail techniques and principles	[15]
BP6	Having external stakeholders participating in students' projects	[16]
BP7	Use case studies/labs for understanding and demonstrating MDE benefits	[16], [18], [19]
BP8	Use clear assignment	[17]
BP9	Give regular feedback	[3], [17]
BP10	Use problem domains familiar to the students	[17]
BP11	Use problem-based approach	[19], [20]

We identified 11 Best practices (BP) from the analysis of the related work (see Table 1). The analysis review of the related works investigating the impact of MDE in SE curricula, concludes that there is no particular approach that stands out as the best way to teach MDE. This makes it difficult to determine the most effective teaching method. However, key elements to consider are good tool support, avoiding technical difficulties, using motivating examples, highlighting MDE benefits, incorporating lab work for practical application, and using real-life projects. The approach presented in this paper is both similar to and distinct from prior works. The most significant difference is that the proposed course is grounded in best practices identified through an analysis of related literature. While this work shares similarities with several of those analyzed, as it applies the best practices they recommend, no study has implemented all of these practices in the manner proposed here.

3. Methodology

This study focused on designing an MDE teaching method and tooling, and then exploring students'

perceptions of the proposed MDE teaching method and tooling, while identifying its drawbacks. Through statistical analysis, we answered two research questions: How do students perceive the effectiveness and usefulness of the MDE teaching method? and How do students perceive the effectiveness and usefulness of the supporting tool? Figure 1 shows the research process.

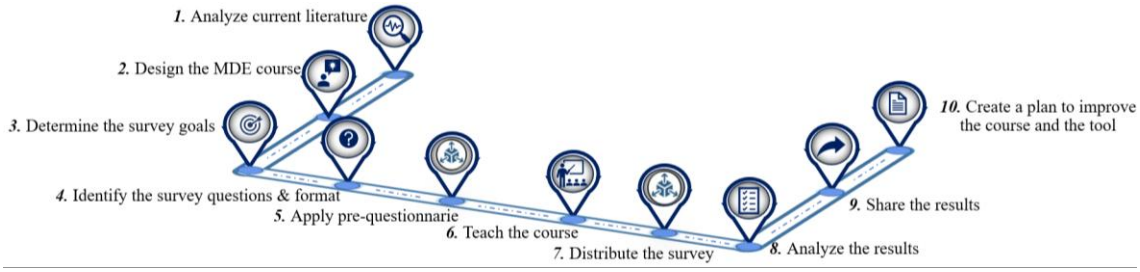

Figure 1*:* Overall survey process.

3.1. Course Design

The MDE course was offered as an elective subject, part of a Software Engineering and Computer Science majors during 2024 at two Cuban universities: UHo (3rd-year Informatics Engineering) and UCLV (4th-year Computer Science) for the first time. The overall goal of the course was to equip modelers with basic principles and an understanding of enterprise modelling techniques. In addition, students should acquire actionable capabilities to tackle the current challenges of enterprise engineering projects [21]. As summarized in Table 2, we incorporated each of the best practices in the course's design. Important best practices are to avoid technical complexities and their negative effect on students' learning and perception of MDE (BP2) and to make use of a code generator (BP3). However, none of the reviewed works pointed towards a specific code generator or combination of tools capable of avoiding technical difficulties. We therefore opted to base the course on the MERODE method as its tools are free and have been specifically geared towards didactic support for teaching MDE [22], [23]. To teach the MERODE theory, the teaching material of the corresponding KU Leuven course was used. It was however translated to Spanish first, and exercises, examples and cases were adapted to the Cuban context. To further address BP4, BP6, BP7, BP10 and BP11, the course adopted a Project-Based Learning approach and focused on real project involvement and lab work to support the development of practical competencies. Table 2 shows how the course implemented all the best practices.

The course's topics were organized into 56 hours (Lectures: 12, Practical Lesson: 16, Lab Session: 12, Partial evaluation: 8, Final evaluation: 6 and Q&A session: 2). The groups were divided into teams of 3 to 5 students. Each team submitted their work in three deliverables according to the MERODE method: (1) Existence Dependency Graph, (2) Object Event Table and (3) Set of Finite State Machines. Each deliverable consists of a written document and a class presentation. More details about the course description are available online (See Appendix 1). To create the models, the students used Merlin, the supporting tool of MERODE. They also used the code generator of MERODE to deliver a full working prototype and compliancy of the models with the requirements. In order to achieve the course goal, the teaching and learning processes adhered to the structure of Bloom's taxonomy [24]. This entails an instructional design with six levels. The first one is *remembering* tasks, which require recalling learned material such as definitions and recognizing notations. *Understanding* tasks involve interpreting and comparing prior knowledge. *Applying* tasks involve using learned information in new ways. *Analyzing, evaluating*, and *creating* tasks involve deconstructing material, making judgments based on criteria, and creating or improving structures.

The course was supported by the Moodle learning management system. Prior to each class, instructors prepared materials aligned with the 4C/ID instructional design model [25], and made them available through the platform. To encourage reflection, students received guiding questions on the assigned topics using smoothed grammar and word flow. For each topic, Moodle hosted practical exercises, discussion forums, videos, articles, and worked examples. The platform also

enabled assignment submission, facilitating grading and progress monitoring.

Table 2
Best practices and its use in the course design

#	Best practice	Its use in the course design
BP1	Use a modelling language to generate a system	The use of MERODE method as a proven didactic method to teach MDE.
BP2	Avoid technical difficulties through good tool support	The use of Merlin, the supporting tool for MERODE, a proven didactic tool to teach MDE.
BP3	Use a code generator	The use of MERODE´s code generator.
BP4	Use real-life problems, motivating examples from the industry	The design of the course focused on real-life problem involvement.
BP5	Explain in detail techniques and principles	Lectures and practical sessions to teach basic principles and understanding of enterprise modelling techniques.
BP6	Having external stakeholders participating in students' projects	The students have practical sessions in the enterprises where they do their internships.
BP7	Use case studies/ labs for understanding and demonstrating MDE benefits	The course proposes lab work to support the development of practical competencies and solved case studies.
BP8	Use clear assignments	The course proposes assignments that are clear and guarantee the scaffolding throughout the entire course.
BP9	Give regular feedback	Besides the feedback of the professor, the MERODE´s code generator generates a full working prototype with automatic feedback generation.
BP10	Use problem domains familiar to the students	Students choose their own project from the enterprises where they complete their pre-professional internship, that provides them the necessary domain knowledge.
BP11	Use a problem-based approach	The course adopted a project-based Learning approach.

This organization of materials and activities allowed students to engage with content outside the classroom, preparing them for in-class problem solving and discussion. As a result, students achieved greater mastery before face-to-face sessions, where they could address difficulties and consolidate understanding. The use of 4C/ID further supported the integration of knowledge, skills, and attitudes, fostering skill coordination and the transfer of learning to real-world contexts.

According to the learning paths defined in the bachelor programs, prior to taking this course, all students had previously completed an introductory course in software engineering and UML notation. The course itself combined theoretical instruction with practical assignments during lab sessions. Each team was tasked with developing a moderately complex software solution based on an industrial scenario. Teams selected projects from organizations where they undertook pre-professional internships, ensuring authentic domain knowledge, as noted in [17]. The domains selected were human resources management, patient appointment scheduling, academic records management, product sales, and multimedia catalog management.

Initially, only vague requirements were provided; groups were responsible for refining and extending these specifications with additional features. The students created both behavioral and structural models. The code generator used in the course provided the students with automated feedback on their models, which helped them to understand implications of design choices. Furthermore, during the practical session, discussions included peer evaluation. The final project must include a running prototype system consisting of the generated code.

The participants were further interviewed on difficulties faced during the entire course as well as preferences. This gave an indication about the potential increase of course delivering in practice. Also professors conducted an assessment as part of the course to determine the goals achievement.

3.2. Data Collection

We created a survey consisting of three main parts: (1) a pre-questionnaire to gather demographic and previous knowledge information, (2) a questionnaire to evaluate the method, and (3) a questionnaire to evaluate the supporting tool.

The first part of the pre-questionnaire consisted of four sections. The first section collects demographic information. Section 2 asks about previous knowledge related to modelling skills, UML notation, modelling languages, and MDE. Section 3 is about programming skills that the student acquired through previous courses, and finally, Section 4 is about general technological skills. This information is useful to check if the students are prepared for the course. The respondents were asked to answer the questions using a four or six-point Likert scale as presented in Appendix 2[2].

To design the method and tool questionnaires, we checked existing questionnaires and models: the System Usability Scale (SUS) [26], the Computer System Usability Questionnaire (CSUQ) [27], the Technology Acceptance Model (TAM) [28], the UMUX questionnaire [29] and the Game Experience Questionnaire [30]. The questionnaire to assess the method used in class consists of six sections: Ease of use/usability (EoU, based on TAM), Learnability (L, based on SUS), Perceived satisfaction/confidence/comfort (PS, based on SUS and CSUQ), Perceived difficulties/complexity (PD, based on TAM), Perceived Effectiveness/productiveness/efficiency (PE, based on SUS and UMUX) and Overall usefulness perception (Fit for purpose) (OUP, based on SUS, CSUQ, TAM and UMUX).

We decided to include three additional questions not directly based on existing questionnaires to gain additional insights about the students' perceptions and intentions of use. In Section 3, we added: "*This method motivated me to learn*" and "*I would recommend this method to other students*". In Section 6, we added: "*I would find this teaching method useful in my learning process.*" All questions use a 5-point Likert scale, from Strongly disagree to Strongly agree. We chose to use closed questions to process the survey more efficiently while reducing the load for respondents.

Likewise, the questionnaire to assess the supporting tool consists of five sections to gauge if the tool complements the method and was pleasant to use: System supporting information and feedback (SSI, based on CSUQ), System Interface/Interaction (SI, based on CSUQ) Game Experience (GE, based on Game Experience Questionnaire), Learning experience (LE, based on Game Experience Questionnaires), and Usability (U, based on SUS). It is based on the same sources as the previous questionnaire. The same 5-point Likert scale was used. A pilot study was carried out to identify ambiguities and areas for improvement. This involved both questionnaires to evaluate the method and the tool. The final questionnaire was modified to address the identified concerns. All questionnaires and results are available online. (See Appendix 3 and Appendix 4).

We started the course with a pre-questionnaire in the classroom. Then, we delivered the course. After the course finished, we distributed the rest of the survey.

4. Results and Discussion

The following section focused on response the research questions. Our findings are supported to frequency analysis of the survey's answers, means comparison tests, along with our interpretations. Subsection 4.1 reports the results obtained with the pre-questionnaire. Subsection 4.2 provides the results of the questionnaire related to the assessed course design (RQ1), and subsection 4.3 provides an assessment of the supporting tool (RQ2). Finally, subsection 4.4 declares the threats to validity.

4.1. Results of the Pre-Questionnaire

At the beginning of the course, we applied the pre-questionnaire to collect demographic information and previous knowledge related to modelling skills, UML notation, modelling languages, and MDE.

[2] The Appendices and data are available here: https://zenodo.org/records/17541096

4.1.1. Participants

This research was conducted during the academic year 2024 in two Cuban universities. The course involved 36 students (17 students from UHo and 19 students from UCLV). Study participation was voluntary, and students were informed that the data would be used for research purposes. Confidentiality was maintained throughout the research process, with all data anonymized to protect participants' identities. The demographic data of participants are presented in Table 3.

Table 3
Demographic Data

	Gender				Age Distribution				
	Male	Female	X	Prefer not to say	>18	>25	>30	>35	>40
UHo	71.00%	29.00%	.00%	0.00%	14	3	0	0	0
UCLV	68.24%	26.32%	.00%	5.26%	16	3	0	0	0
Overall	69.44%	27.78%	.00%	2.78%	30	6	0	0	0

4.1.2. Previous Knowledge

Regarding previous knowledge related to modelling skills, all students knew about UML reporting little or moderate knowledge. A few students (8,11%) reported no knowledge of conceptual modelling, while 91.66% declared little or moderate knowledge. About half (51.77%) declared they don't have previous knowledge about MDE, 33,33% declared little knowledge, and 13.89% moderate knowledge about using MDE before the class. Regarding programming and technological skills, students from UHo (64.71%) declared 1 or 2 years of programming experience, while students of UCLV declared 1 or 2 years (52.63%) or between 3 and 5 years of experience (42.11%). As participants of UCLV are students of the 4th year, it makes sense that most of the students have more years of experience. Not all students answered, so the percentage is below 100. Some report using computers for under an hour per day, while others' use exceeds 8 hours; most use them for 3 to 5 hours.

4.2. Perception of the Effectiveness and Usefulness of the MDE Teaching Method

This subsection addresses RQ1 on the effectiveness and usefulness of the MDE teaching method? Ease of use/usability (EoU) was positively rated, with over 75% finding the method easy to follow (EoU1) and well organized (EoU2), especially at UHo. Learnability (L) showed strong results, where clarity and understandability of the teaching method (L2) obtained the best score. As much as 86.1% of the total sample of respondents (strongly) agree, and 11.1% are neutral. Though, roughly 40% noted the need for some prior knowledge (L3) and teachers' support to start learning (L4). Perceived satisfaction/confidence/comfort (PS) was high. Around 75% or more expressed satisfaction with learning (PS1), confidence learning with this teaching method (PS2), motivation to learn (PS4), and willingness to recommend it (PS5). The highest-rated item was PS3 (comfortable learning) with 83,4% of the respondents (strongly) agreeing.

Perceived difficulties/complexity (PD) was measured with five items, some worded in a negative sense. For data processing inverted values were taken. Overall, perceived difficulties were low. The results shows that participants do not perceived the method as complex (PD1), inconsistent (PD2), or cumbersome (PD3). Also, they perceived the teaching method as a successful experience (PD4), although some of then spent extra time correcting errors (PD5).

The Perceived effectiveness/productiveness/efficiency (PE) was also favorable. Over 60% agreeing the method supported effective (PE1) and productive learning (PE3). Likewise, a same amount agree with a positive learning performance (PE2) and ease of learning (PE4).

The construct Overall Usefulness Perception (OUP) showed moderate to strong agreement, though a notable portion remained neutral on content (OUP2) and qualification improvement (OUP5). Figure 2 shows detailed information regarding perceptions of MDE teaching method per university.

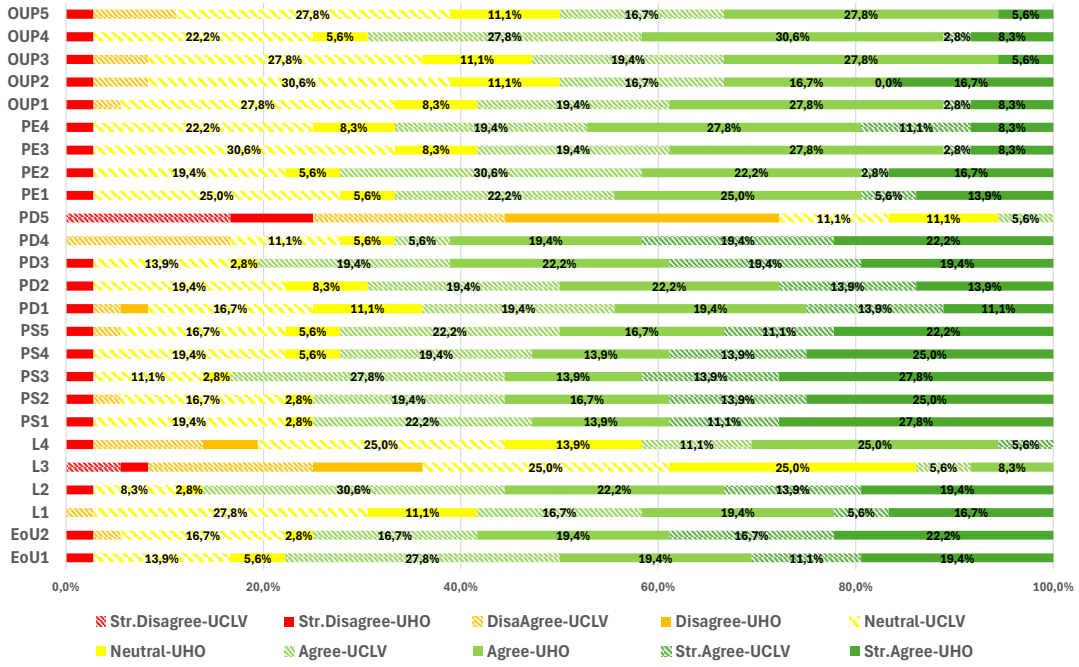

Figure 2. Perceptions of MDE teaching method per university

To ensure accurate conclusions on significance, we compared the means for each construct using the independent samples t-test, appropriate for small sample size and estimated effect size with Cohen's d [31] to assess the strength of differences. Significant differences were found in three items between UHo and UCLV: PD4 (frustrating experience) (p=.023) (d=.73), OUP2 (contains expected content) (p=.017) (d=.77), and OUP5 (easy to become more qualified) (p= .049) (d=.651).

According to Cohen's d values, the effect size is large for all but OUP5, which is medium, indicating meaningful practical differences. This reveals that the course offered at both universities do not contribute to student perception to a similar extent. This could highlight issues with the implementation of the methods in the context of either the teacher's role or access difficulties, as the same learning material was used for the course at both universities. It may be due to learning styles or motivation. It is necessary to review how the benefits of the method are communicated and strategies to improve it. This could consolidate the method's effectiveness and acceptance.

Overall, the evaluation per item ranks well above 3 out of 5, indicating a positive student opinion. There is a more pronounced positive perception of features such as clarity, ease of use, and satisfaction. Furthermore, the organization is considered good giving the rate of item "The method is well organized, so it is easy to find the necessary information". The results suggest a good instructional design, as evidenced by the item "This method is clear and understandable". The perception of teacher support points to the fact that some students do not feel fully autonomous.

4.3. Perception of the effectiveness and usefulness of the tool

Regarding RQ2, we proceed as in the previous section. Supporting Information and Feedback (SSI) received positive evaluations. Around 70% of respondents agree that the information provided was helpful (SSI1) and effective for completing learning tasks (SSI2), suggesting that the tool offers relevant information and fulfills its support function. Clarity of on-screen information (SSI3) and error messages (SSI4) received strong agreement, while ease of error recovery (SSI5) had slightly lower agreement but still positive, with a notable proportion of neutral responses.

System Interface/Interaction (SI) had moderate agreement, half of respondents found the interface pleasant (SI1) and enjoyable (SI2), though a considerable number remained neutral, especially at UCLV. The analysis of the items reveals room for improvement in clarity and comprehension of the tool's use. Game Experience (GE) items indicated that the tool was stimulating (GE1) and helped students maintain focus (GE3), with higher agreement at UHo. High scores were noted for students'

ability to achieve the tool's goals (GE2). Usability (U1) and Learning Experience (LE) also received strong positive rate, regarding the perceived learning value (LE3), clarity of goals (LE1), and feedback provision (LE2).

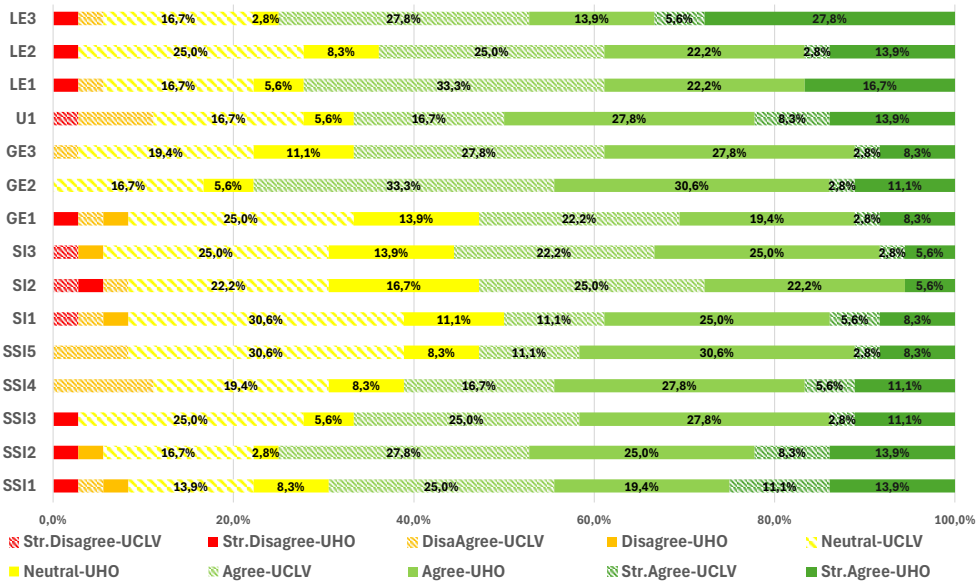

Figure 3. Perception of the tool per university

The independent samples t-test showed significant differences between UHo and UCLV in clarity of error messages and corrective suggestions (SSI4) (p=.010) (d=.830), ease of error recovery (SSI5) (p= .001) (d=1.041), tool's ease of use and organization (U1) (*p*= .011) (*d*=.810), and recognised learning value (LE3) (*p*= .034) (*d*=.699). The effect size was large for SSI4 , SSI5, and U1, and moderate for LE3.

The effect size shows differences between context. The item "I recognize the value as a tool for learning" scored highest (LE3), indicating that students value the tool as a useful learning resource. However, the effect size for SSI4 (clarity of error messages) and SSI5 (ease of error recovery) could point to misunderstanding of messages errors from the tool. The differences observed and their meaning in terms of the tool may also be related to the complexity of the approach, learning styles, or motivation. Further investigation is necessary.

The average rating per question is well above 3 out of 5, indicating a positive opinion. Some areas for improvement were identified, especially the interface. However, the results showed moderately high overall acceptance, which validates its relevance as a learning support resource.

By implementing best practices, we identified key lessons for improvement. The selected method and tool were positively received, with automatic feedback helping students recovering from errors. However, some students struggled with system variable configurations across different operating systems. The course includes lectures and practical sessions. We recommend more solved examples to enhance understanding of principles and techniques. Smooth communication and scaffolding are needed to foster greater student autonomy at the early stages, as some students require additional guidance when starting their learning. All this helps students gradually developing the needed skills and confidence to take control of their own learning, motivation, satisfaction, and academic achievement. Teamwork on real-life projects familiarizes students with professional settings, but motivation levels vary. Pairing motivated students with less prepared peers helps maintain project pace. Small teams of four to five are ideal for labs, as larger groups can be challenging for inexperienced students. Allowing students to propose their own project, in collaboration with the professor, boosts motivation and learning outcomes. We suggest using peer evaluation to encourage reflection on their own and others' projects, which requires building a strong community first.

4.4. Threats to Validity

Like in other empirical observations, the results are subject to certain threats to validity. To mitigate

these threats, we followed best practices for survey design [32].

A possible threat to *internal validity* is that the survey was distributed via a Google Form link and could therefore have been completed by any student having the link. To avoid this, the survey links was not made public but distributed exclusively via direct email to the students. Likewise, as the survey is taken as part of the course the students were informed that their responses would not affect their grading. The *reliability* of both instruments (to assess the method and the tool) was assessed for internal consistency using Cronbach's α. The obtained coefficients of .960 for the method´s questionnaire and .935 for the tool´s questionnaire indicate a high level of internal consistency [33]. Notice that the study does not have a control group. First, the size of the sample is limited by the size of the group that took the course in both universities, and does not allow splitting into two groups of reasonable size. Second, from an ethical perspective, it is not possible to deny part of the group access to a method and tool that could improve their learning. A possible threat to the *conclusion validity* is that the course was offered as an elective one; thus, students who choose the course are interested for MDE training. However, all students decided to enrolled in this course.

In terms of *external validity,* we do not claim that our conclusions are universally applicable, nor did we aim to define or target a representative sample of software engineering students. Instead, our findings are specific to the design and development of the course described in this paper, and are only representative of that course design. To achieve some level of generalization, we selected students from the Bachelor's program in Informatics Engineering at two universities—UCLV and UHo—as well as from the Bachelor's program in Computer Science at UCLV. These participants represent the broader population to the extent that generalization is possible. While they had similar academic backgrounds, the course was taught by different professors, introducing variability in instructional approaches. This diversity provides a broader perspective on the course, as students' perceptions were shaped by differences in how the course was taught. Additionally, our study ensures that the sample includes students with varying levels of experience acquired during the course, contributing to a more comprehensive understanding of their learning experiences.

To ensure *construct validity*, we used questions from proven questionnaires and models. Also, a pilot study with students of Informatics Engineering at UHo at the beginning of the course confirmed that adaptations maintained questions' clarity. The survey was administered consistently across both universities to reduce extraneous variability, enhancing the validity of the findings.

5. Conclusions

For software engineering students, the effective use of modelling remains a challenge. To address this issue, an MDE method and tool were incorporated into a software design course at two Cuban universities using best practices identified in the literature. This study incorporated innovative pedagogical approaches into the CS/SE curricula. Students gained collaborative skills, engaged in real-world problem solving, and bridged the gap between abstraction and practical application.

The research compiled 11 best practices for successfully training to model from current literature, followed by the design of an MDE course, that was then conducted at two Cuban universities. To evaluate students' perspectives on the method and tool, we administered a survey to 36 Cuban students from UHo and UCLV. Given the small sample, the results were presented using descriptive statistics, and demonstrate strong student acceptance and satisfaction. While the authors acknowledge that the small sample of students, and the single run of the course at two Cuban universities could limit the generalizability of the findings, this research could nevertheless inform other teachers in how to successfully introduce MDE at higher education institutions. This course, with its method and tools, can serve as a starting point for other teachers. The goal of teaching is not only to provide the ability to use an MDE tool, but also to convey the general principle of using models to develop software through the systematic approach, included in the training process.

Future research should address the limitations identified in this study and explore additional opportunities. Future evaluations of the course will consider open questions for a qualitative analysis of comments. Experimental research will allow a direct measurement of learning outcomes.

Declaration on Generative AI

During the preparation of this work, the authors used Deepl, ChatGPT and Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using these services, the authors reviewed and edited the content as needed. They take full responsibility for the publication's content.

Acknowledgements

The authors are grateful to VLIR_UOS for supporting the project "Better digital Services from Cuba through Model Driven Engineering", under grant CU2024TEA542A101.

References

[1] C. Y. Laporte and M. Munoz, 'Not teaching software engineering standards to future software engineers-malpractice?', *Computer*, vol. 54, no. 5, pp. 81–88, 2021.

[2] A. N. Kumar *et al.*, *Computer Science Curricula 2023*. New York, NY, USA: Association for Computing Machinery, 2024.

[3] L. T. W. Agner, T. C. Lethbridge, and I. W. Soares, 'Student experience with software modeling tools', *Software & Systems Modeling*, vol. 18, no. 5, pp. 3025–3047, Oct. 2019, doi: 10.1007/s10270-018-00709-6.

[4] R. Noel, J. I. Panach, and O. Pastor, 'Including business strategy in model-driven methods: an experiment.', *Requirements Engineering*, vol. 28, no. 3, pp. 411–440, doi: 10.1007/s00766-023-00400-3.

[5] J. Cabot, 'Positioning of the low-code movement within the field of model-driven engineering', presented at the Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 2020, pp. 1–3.

[6] A. Tsagkaropoulos, 'Model-driven adaptation of Function-as-a-Service applications', 2023.

[7] E. Felix, D. Lopes, and O. S. Jr, 'A Framework Based on Model Driven Engineering and Model Weaving to Support Data-Driven Interoperability for Smart Grid Applications', presented at the Proceedings of the 2020 European Symposium on Software Engineering, 2020, pp. 30–36.

[8] B. Selic, 'The Pragmatics of Model-Driven Development', *IEEE Software*, vol. 20, no. 5, pp. 19–25, 2003.

[9] R. France and B. Rumpe, 'Model-driven development of complex software: A research roadmap', presented at the Future of Software Engineering (FOSE'07), IEEE, 2007, pp. 37–54.

[10] I. Khriss, A. Jakimi, and H. Abdelmalek, 'Towards an Effective Implementation of a Model-Driven Engineering Approach for Software Development', presented at the 2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), IEEE, 2020, pp. 1–6.

[11] W. Barnett, S. Zschaler, A. Boronat, A. Garcia-Dominguez, and D. Kolovos, 'An Online Education Platform for Teaching MDE', presented at the Proceedings - 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELSC 2023. 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 114–121. doi: 10.1109/MODELS-C59198.2023.00035.

[12] P. J. Clarke, Y. Wu, A. A. Allen, and T. M. King, 'Experiences of teaching model-driven engineering in a software design course', presented at the Online Proceedings of the 5th Educators' Symposium of the MODELS Conference, 2009, pp. 6–14.

[13] P. Collet, S. Mosser, S. Urli, M. Blay-Fornarino, and P. Lahire, 'Experiences in teaching variability modeling and model-driven generative techniques', presented at the Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools-Volume 2, 2014, pp. 26–29.

[14] A. Schmidt, D. Kimmig, K. Bittner, and M. Dickerhof, 'Teaching model-driven software development: Revealing the" great miracle" of code generation to students', presented at the Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148, 2014, pp. 97–104.

[15] J. Porubaen, M. Bačíková, S. Chodarev, and M. Nosal, 'Teaching pragmatic model-driven software development', *Computer Science and Information Systems*, vol. 12, no. 2, pp. 683–705, 2015.
[16] L. Kuzniarz and L. E. G. Martins, 'Teaching model-driven software development: a pilot study', in *Proceedings of the 2016 ITiCSE Working Group Reports*, 2016, pp. 45–56.
[17] S. Chakraborty and G. Liebel, 'We do not understand what it says – studying student perceptions of software modelling', *Empirical Software Engineering*, vol. 28, no. 6, p. 149, Nov. 2023, doi: 10.1007/s10664-023-10404-w.
[18] F. Ciccozzi *et al.*, 'How do we teach modelling and model-driven engineering? a survey', presented at the Proceedings of the 21st ACM/IEEE international conference on model driven engineering languages and systems: Companion proceedings, 2018, pp. 122–129.
[19] J. I. Panach and Ó. Pastor, 'A Practical Experience of How to Teach Model-Driven Development to Manual Programming Students', *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 18, pp. 1–6, 2023.
[20] S. Ouhbi and N. Pombo, 'Software engineering education: Challenges and perspectives', in *IEEE Global Eng. Edu. Conf., EDUCON*, Cardoso A., Alves G.R., and Restivo T., Eds, IEEE Computer Society, 2020, pp. 202–209. doi: 10.1109/EDUCON45650.2020.9125353.
[21] M. Snoeck, *Enterprise Information Systems Engineering. The MERODE Approach*. Belgium: Springer, 2014.
[22] S. Strecker, U. Baumöl, D. Karagiannis, A. Koschmider, M. Snoeck, and R. Zarnekow, 'Five Inspiring Course (Re-)Designs', *Business & Information Systems Engineering*, vol. 61, no. 2, pp. 241–252, Apr. 2019, doi: 10.1007/s12599-019-00584-5.
[23] G. Sedrakyan and M. Snoeck, 'Cognitive Feedback and Behavioral Feedforward Automation Perspectives for Modeling and Validation in a Learning Context', in *Model-Driven Engineering and Software Development*, S. Hammoudi, L. F. Pires, B. Selic, and P. Desfray, Eds, Cham: Springer International Publishing, 2017, pp. 70–92.
[24] D. R. Krathwohl, 'A revision of Bloom's taxonomy: An overview', *Theory into practice*, vol. 41, no. 4, pp. 212–218, 2002.
[25] J. J. G. K. Van Merrienboer P. A., *Ten steps to complex learning*, 3rd edn. New York: Routledge, 2018.
[26] J. Brooke, 'SUS-A quick and dirty usability scale', *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.
[27] J. R. Lewis, 'IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use', *International Journal of Human-Computer Interaction*, vol. 7, no. 1, pp. 57–78, 1995.
[28] F. D. Davis, 'Technology acceptance model: TAM', *Al-Suqri, MN, Al-Aufi, AS: Information Seeking Behavior and Technology Adoption*, vol. 205, pp. 219–219, 1989.
[29] K. Finstad, 'The usability metric for user experience.', *Interacting with Computers*, no. 22, pp. 323–327, 2010, doi: 10.1016/j.intcom.2010.04.004.
[30] W. A. IJsselsteijn, Y. A. W. de Kort, and K. Poels, *The Game Experience Questionnaire*. Eindhoven: Technische Universiteit Eindhoven, 2013.
[31] J. Cohen, 'CHAPTER 2 - The t Test for Means', in *Statistical Power Analysis for the Behavioral Sciences*, 2nd edn, Library of Congress Catahloging-in- Publication Data: Lawrence Erlbaum Associates, 1977, pp. 19–74. doi: 10.1016/B978-0-12-179060-8.50007-4.
[32] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*, 2nd edn, vol. 236. Springer, 2012.
[33] P. Mallery, 'SPSS for Windows step by step: A simple guide and reference', *Wind. Step by step a simple guid. Ref. Needham Height. MA Allyn Bacon*, 1999.

A. Online Resources

The appendices and the data have been made available online here:
https://zenodo.org/records/17541096