Teaching Software Engineering for Artificial Intelligence: An Experience Report

Fabio Palomba[1 [0000−0001−9337−5116]], Gianmario Voria[1 [0009−0002−5394−8148]], Alessandra Parziale[1 [0009−0001−0758−3988]], Viviana Pentangelo[1 [0009−0003−1425−9398]], Antonio Della Porta[1 [0000−0003−1860−8404]], Vincenzo De Martino[1 [0000−0003−1485−4560]], Gilberto Recupito[1 [0000−0001−8088−1001]], and Giammaria Giordano[1 [0000−0003−2567−440X]]

Software Engineering Lab, Department of Computer Science — University of Salerno, Salerno, Italy {fpalomba,gvoria, alparziale, vpentangelo, adellaporta, vdemartino, grecupito, giagiordano}@unisa.it

Abstract. As Artificial Intelligence (AI) becomes integral to modern software systems, the software engineering (SE) research community has been actively developing methods, tools, and frameworks to address software quality assurance of AI-enabled systems across critical dimensions such as robustness, ethics, security, and sustainability. These contributions are designed to tackle the complexity of AI systems, such as their probabilistic nature, data dependencies, and societal impact, ensuring they meet the standards of modern software engineering. These advances have, in turn, inspired educators to introduce Software Engineering for Artificial Intelligence (SE4AI) courses aimed at preparing the next generation of software engineers, with notable success examples already reported in the literature. In this *experience report*, we contribute to the field of SE4AI education by sharing lessons learned in designing and teaching a course that addresses the unique characteristics of AI-enabled systems. Drawing on insights gathered over four iterations of the course, we discuss how students perceive and apply key software engineering concepts, the challenges they encounter with tools and techniques, and how project-based learning bridges the gap between theoretical knowledge and real-world application. Furthermore, we address the broader educational challenges, such as interdisciplinary barriers and the integration of rapidly evolving AI technologies, and provide recommendations to enhance SE4AI education. By reflecting on these experiences, we aim to offer insights and strategies for improving the teaching of SE4AI topics.

Keywords: Software Engineering for Artificial Intelligence · Experience Report · Software Engineering Education.

1 Introduction

The rapid growth of Artificial Intelligence (AI) has transformed industries and everyday applications, becoming integral to sectors like healthcare and finance by enhancing efficiency, decision-making, and innovation [41,38,29,27].

This widespread adoption of AI has introduced unique challenges for software engineering, prompting the Software Engineering (SE) research community to establish a dedicated field known as Software Engineering for Artificial Intelligence (SE4AI). SE4AI focuses on extending traditional SE practices to meet the demands of AI-driven systems. Bosch et al. [8] described AI engineering as integrating specialized technologies and processes essential for building AI-enabled systems, while Martínez-Fernández et al. [24] characterized these systems as architectures combining both traditional software components and AI-specific elements. Sculley et al. [34] noted that AI components often constitute only a small part of such systems, which are supported by conventional software that enables and manages AI functionalities.

AI-enabled systems face challenges beyond those of traditional software projects, particularly in ensuring quality across dimensions such as robustness, scalability, ethics, and risk management [8]. These challenges arise from the inherent characteristics of AI-enabled systems, including their probabilistic behavior, reliance on large-scale data, and significant societal impact [25]. Systematic processes and practices are essential for addressing these issues and for ensuring that software meets predefined quality standards. These standards cover both functional attributes, such as correctness, and non-functional ones, such as security, fairness, transparency, and environmental sustainability.

The growing need for advanced practices in AI-enabled systems has motivated educators to develop SE4AI courses that prepare the next generation of software engineers. These courses aim to equip students with the skills to develop, evaluate, and maintain systems integrating AI, while tackling the distinct challenges these systems present. Inspired by notable examples in the literature [20,22], this *experience report* contributes to the field of SE4AI education by sharing preliminary insights from designing and teaching a course focused on the attributes of AI-enabled systems. We discuss how students perceive and apply key concepts, the challenges they encounter with tools and techniques, and how project-based learning bridges the gap between theoretical understanding and real-world application. Furthermore, we address broader educational challenges, such as interdisciplinary barriers and the need to integrate rapidly evolving AI technologies. By reflecting on these experiences, we aim at offering recommendations for advancing SE4AI education and supporting future research and practice in preparing software engineers for the complexities of AI-enabled systems.

2 Related Works

For decades, the primary focus of software engineering research and education related to artificial intelligence has been on leveraging AI techniques to address SE challenges, often referred to as *Artificial Intelligence for Software Engineering* (AI4SE) [26]. This area includes, for instance, the use of AI approaches to predict and manage software defects [18,7], generate test cases [3,4,9], detect and refactor source code design flaws [2,5,23], or optimize software development processes [39,40,37]. Educational courses aligned with this focus are now widely

diffused, with many examples discussed in experience reports and educational articles describing how to emphasize the application of AI methods in traditional software engineering contexts[14,15,36].

In recent years, there has been a notable shift toward studying the application of software engineering principles and practices in the development of Artificial Intelligence-enabled systems, often termed *Software Engineering for Artificial Intelligence* (SE4AI) [24]. This paradigm has the opposite goal of AI4SE, namely that of addressing the engineering challenges posed by AI components, such as managing data and model quality, handling model evolution, and ensuring scalability and robustness in production systems. The SE4AI research community has grown rapidly, producing significant contributions to software quality assurance aspects of AI-enabled systems. A notable example is represented by the technical debt research field, where researchers attempted to study solutions to deal with AI debt, i.e., issues arising from the peculiar component and activities of AI-enabled systems, such as data dependencies, model versioning, and the maintenance of continuously evolving pipelines, making them harder to scale and manage over time. The seminal work by Sculley et al. [34] highlighted how these challenges differ from traditional software engineering debt, emphasizing risks like entanglement, undeclared consumers, and system-level anti-patterns that may impact other non-functional attributes of AI-enabled systems, including security and privacy [32]. Other rapidly growing research areas include ethics & fairness [13] and verification & validation [33]. In these areas, researchers have developed methodologies and tools to (i) detect and mitigate biases in AI models [12,31,17], (ii) support the robustness of AI-enabled systems to deal with non-deterministic nature of AI models [1,11,21].

The challenges of engineering AI-enabled systems have led to an even more pressing need to educate the next generation of software engineers with specialized knowledge and skills to address these complexities. In response to this need, many institutions have begun offering courses specifically focused on SE4AI—this was also fostered by the availability of books and teaching resources that emphasize the intersection of SE and AI engineering, like Smith's *"Machine Learning Systems"* [35], Hulten's *"Building Intelligent Systems"* [19], and Burkov's *"Machine Learning Engineering"* [10]. Among the most well-established courses on the matter, a notable case if the one of the *"Software Engineering for AI-enabled Systems"* course taught by Prof. Kästner at Carnegie Mellon University (CMU) [20]. The course combines theoretical foundations with practical assignments, leveraging real-world scenarios to teach students about AI system requirements, testing, deployment, and quality assurance practices. Similarly, Lanubile et al. [22] reported on teaching MLOps through project-based learning, emphasizing the importance of hands-on approaches to teach the complexities of AI system development and operations.

The course object of this experience report is inspired by the Kästner's course and indeed shares similar learning objectives. Specifically, it aims to (1) illustrate the engineering challenges in building production systems with machine learning (ML) components, beyond model creation; and (2) compare the roles,

goals, and challenges faced by software engineers and data scientists in developing AI-enabled systems. While grounded in the foundational structure of the CMU's course, our course varies some aspects to (i) address the rapidly changing landscape of AI and (ii) adapt itself to the structure of our Master's degree:

Teaching Material. The topics of the course have been evolved to reflect recent advances in AI engineering. For instance, the course includes topics such as the analysis of Large Language Models, advances in fairness engineering, and considerations for ML sustainability. These additions are informed by the latest research in SE4AI, including insights from our own research projects on these themes, and periodically updated to provide students with a fresh research perspectives on the fast-growing topics of the course.

Hands-on Education. While the CMU course emphasizes assignments to build hands-on experience, our course adopts a semester-long team project to stimulate practical learning. This divergence was mainly due to the need of adopting a similar educational approach as other courses available in our Master's degree—the recommended guidelines are to let students engage with real-world challenges which, in our case, implies the application of SE4AI methods across the lifecycle of a project, from conception to evaluation, thereby fostering collaboration and problem-solving skills.

Industry Integration. Guest lectures from partner companies provide students with practical insights into SE4AI challenges. These lectures are complemented by case studies and experience reports. For instance, one of the case studies discussed in the course is the one by Beede et al. [6], who evaluated the deployment of a deep learning system for diabetic retinopathy detection in clinical settings, showing issues and challenges that motivate the need for software engineering instruments in AI-enabled system development.

3 The Software Engineering for Artificial Intelligence Course

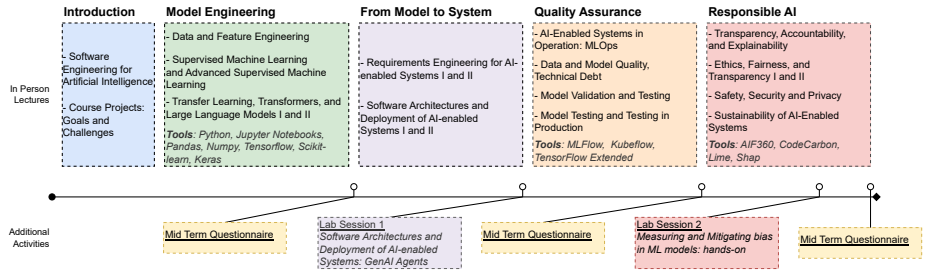

Fig. 1. Design and Timeline of the *"Software Engineering for Artificial Intelligence"* Course.

The course we base our experience on is entitled *"Software Engineering for Artificial Intelligence"* and is taught at the University of *Salerno*. It is designed for Master's students who have prior knowledge in functional and object-oriented programming, software engineering, and the fundamentals of artificial intelligence, obtained during their Bachelor's studies.

The course is part of the *"Software Engineering and IT Management"* curriculum within the Master's Degree in Computer Science, and provides 48 hours of instruction (6 ECTS). The course aims to introduce students to end-to-end ML engineering, encompassing key phases from requirements gathering to verification, validation, and deployment. Now, in its fourth edition, the course is taught in English by a main lecturer and supported by multiple teaching assistants.

The teaching methods include: (1) *in-person lectures*, delivered by the main lecturer; (2) *laboratory sessions*, supervised by teaching assistants, focusing on hands-on practical work; (3) *individual study assignments*, where students independently explore specific topics and participate in discussions through a flipped-classroom approach; and (4) *classwork activities*, where students analyze case studies and reason about the application of SE practices introduced during lectures. Figure 1 provides a summary of the course design, including a timeline that illustrates how the course topics are delivered over time and how they interweave with the course's additional activities.

More specifically, the course is structured into five main parts: (1) *'Introduction to SE4AI'*; (2) *'Model Engineering'*, (3) *'From Model to System Engineering'*, (4) *'Quality Assurance'*, and (5) *'Responsible AI'*.

(1) Introduction to SE4AI. Students are introduced to the foundational concepts of SE4AI. Frontal lectures focus on comparing AI-enabled systems to traditional software systems. Additionally, this part provides an overview of key AI techniques underlying contemporary software systems, including data and feature engineering, search-based algorithms, and supervised learning.

(2) Model Engineering. The second part of the course focuses on building robust AI systems through model engineering. It begins with an overview of ML pipelines, emphasizing the critical roles of data preparation and feature engineering. Students then explore supervised ML, covering key concepts in model training, validation, and testing. The module progresses to advanced techniques such as deep learning and ensemble methods, followed by an in-depth look at transfer learning, transformers, and Large Language Models. Practical skills are developed using state-of-the-art tools including TensorFlow, Scikit-learn, and Keras.

(3) From Model to System. The third part of the course addresses the transition from ML models to full-fledged and engineered AI-enabled systems, exploring the diverse shapes of SE4AI practices. It begins with requirements engineering, focusing on when to adopt ML, how to define functional and non-functional requirements, and how to set meaningful measurement goals. Students also learn techniques for gathering requirements. The module then explores the architectural design of AI-enabled systems, discussing common patterns like client-server, multi-tier, service-oriented, microservices, and data-

flow architectures. A laboratory session on Generative AI agents further enriches this part by examining current trends and system integration challenges in modern AI applications.

(4) Quality Assurance The fourth part of the course centers on the quality of AI-enabled systems. Students are first introduced to MLOps practices, covering essential topics such as model versioning, pipeline management, and infrastructure deployment. Tools such as MLFlow, Kubeflow, and TensorFlow Extended are introduced to facilitate these practices. The module covers data and model quality, as well as challenges like concept and data drift. Topics also include managing technical debt, validating and testing models, and strategies for testing in production. A case study offers students hands-on insight into cutting-edge research and practices in production-level AI system testing.

(5) Responsible AI The final module focuses on the ethical, legal, and societal implications of AI. It begins with an overview of transparency, accountability, and regulatory considerations, concluding with methods for achieving explainability in AI models. Students are introduced to fairness in ML, including techniques for detecting and mitigating bias through pre-processing, in-processing, and post-processing approaches, through a laboratory session. The section draws on recent research and real-world applications. Topics also include safety, security, and privacy, highlighting threats like adversarial attacks and techniques for robust system design. The module concludes with a discussion on the sustainability of AI-enabled systems, covering environmental impact and long-term maintainability insights from current research. Students use tools such as AIF360, Lime, Shap, and CodeCarbon to assess, monitor, and improve these quality attributes over time.

The course includes three individual mid-term questionnaires designed to assess students' comprehension of the material covered up to that point, as illustrated in Figure 1. Each questionnaire consists of closed-ended questions focused on specific concepts discussed during the lectures. These assessments serve both as a tool to gauge students' understanding and as a component in determining their final evaluation.

In addition to the lectures and individual mid-term questionnaires, students are required to work on a team project, choosing between two project types. The first option involves developing a prototype of an AI-enabled system using the methods introduced in the course. The second option focuses on conducting a quality assurance analysis of an existing AI-enabled system, emphasizing specific properties such as fairness, robustness, or explainability. Student groups can range from 1 to 4 participants, depending on the scope of the project, which is preliminarily validated by the lecturer.

Finally, students are assessed through two components. At first, a project discussion, lasting 30 minutes: students present their team project (10 minutes) and engage in a discussion with the lecturer and teaching assistants (20 minutes) about the validity, limitations, and challenges encountered. The project discussion aims to evaluate students' abilities in engineering AI-enabled systems, as

well as their capacity to effectively communicate the key methodological approaches and results achieved in their projects. Secondly, an oral examination, typically lasting from 45 to 60 minutes: students are tested on the theoretical and practical topics covered during the course.

The course attracts approximately 50 students every year. Over the four editions, we collected data on participants' background knowledge across various topics in AI and SE. Participation in the questionnaire was voluntary, and up to today, we have collected 91 answers to the background survey. This data was used to inform and fine-tune the course content, addressing gaps in knowledge where needed or adapting to students' advanced understanding in specific areas. During these years, 56% of participants had a Software Engineering background, while 22% of students came from the Data Science and Machine Learning domain. A small number of participants came from the Security or Cloud Computing domain, while the others were enrolled in the course through other types of university programs (e.g., Ph.D. students). The survey's design, along with the anonymized aggregated responses, can be found in our online appendix [30].

4 Experience Design and Report

This section presents the lessons learned from our four-year experience delivering the *"Software Engineering for Artificial Intelligence"* course at the University of *Salerno, Italy*. We outline the primary focus areas of our analysis and provide an account of the insights gained over the years.

4.1 Primary Focus Areas of Our Experience Report

This experience report is centered on the aspects of AI-enabled systems, as taught and applied in our course. The following focus areas highlight the key dimensions of our report, emphasizing the lessons learned in addressing specific challenges of AI-enabled systems.

F1. Students' Perception of Course Topics. We analyze students' perceptions of the topics covered in the course, focusing on their relevance to real-world AI engineering tasks. In particular, we concentrate on how students value the introduction of concepts such as technical debt, security, fairness, explainability, and verification & validation in the context of AI-enabled systems and whether they find these concepts practically useful in their projects. Through the analysis of feedback and project outcomes, we identify areas where students feel confident and areas where additional support or alternative teaching methods may be required.

F2. Challenges in Understanding Concepts and Tools. As further discussed in this paper, certain topics and tools introduced during the course posed significant challenges for students. These include advanced testing techniques for AI-enabled systems, methods to identify and mitigate AI debt, and tools for monitoring fairness and robustness. This focus area examines these difficulties

and discusses approaches to help students overcome them. This focus area also explores the role of hands-on activities, case studies, and real-world examples in helping students bridge the gap between theoretical knowledge and practical application.

F3. Challenges in SE4AI Projects. The project component of the course serves as a testing ground for applying the methods taught to AI-enabled systems. We discuss the obstacles students encountered, such as integrating best practices into iterative ML development cycles, managing trade-offs between quality attributes (e.g., accuracy vs fairness), and using tools for monitoring and maintaining model attributes over time. In addition, we explore the socio-technical dynamics that arise during team-based projects, including how students collaborate to address conflicting priorities between software quality attributes and AI-specific goals.

F4. Educational Challenges in Teaching AI-enabled Systems. According to our experience, teaching on aspects as quality, robustness, ethics, security, and sustainability for AI-enabled systems introduces unique challenges, particularly given the rapidly evolving nature of AI technologies. This section explores issues such as keeping course content aligned with state-of-the-art practices, providing practical examples for abstract concepts, and ensuring students understand how traditional software practices apply in the context of AI.

To address these four focus areas, we systematically collected and analyzed student data across the four years the course has been offered. Specifically, we gathered demographic and academic background information for each cohort. In addition, at the end of each course edition, we conducted a survey to capture students' perceptions of the course, with a particular emphasis on the practical applicability of the topics, tools, and methods covered. The survey included both closed-ended and open-ended questions, and participation was not mandatory. Closed-ended questions use Likert-scale responses [28] to assess various aspects, such as (1) the complexity of the course, (2) the usefulness of the tools and methods taught, (3) the satisfaction with specific themes, (4) the usefulness of the hands-on and case study activities, and (5) the perceived practical applicability of the tools and methods studied in the course. Open-ended questions provide students the opportunity to elaborate on areas for improvement and share detailed reflections on their learning experiences, challenges encountered, and overall perception of the practicality of the course. The survey structure is accessible in our online appendix [30].

We analyzed the collected data using established research methods, treating our experience similarly to other survey-based studies conducted in the field [16]. Likert-scale responses from closed-ended questions were analyzed through descriptive statistics to identify patterns and trends in students' feedback. Responses to open-ended questions were examined using content analysis research methods, enabling the identification of recurring themes and insights into students' experiences. The most significant open-ended student sentences were extracted and represented through the symbol 💬 . In total, we collected 59 survey participants over the years. For the sake of space limitations, our experience

report treats the four course's editions in an aggregated manner (as opposed to reporting year-by-year trends); nonetheless, we plan to further elaborate on the evolution of the course as part of our future research agenda. The results of these data analysis procedures informed our discussion of focus areas **F1**, **F2**, and **F3**. As for **F4**, insights were derived from periodic retrospective meetings between the main lecturer and the teaching assistants, complemented by direct feedback collected from students who previously completed the course.

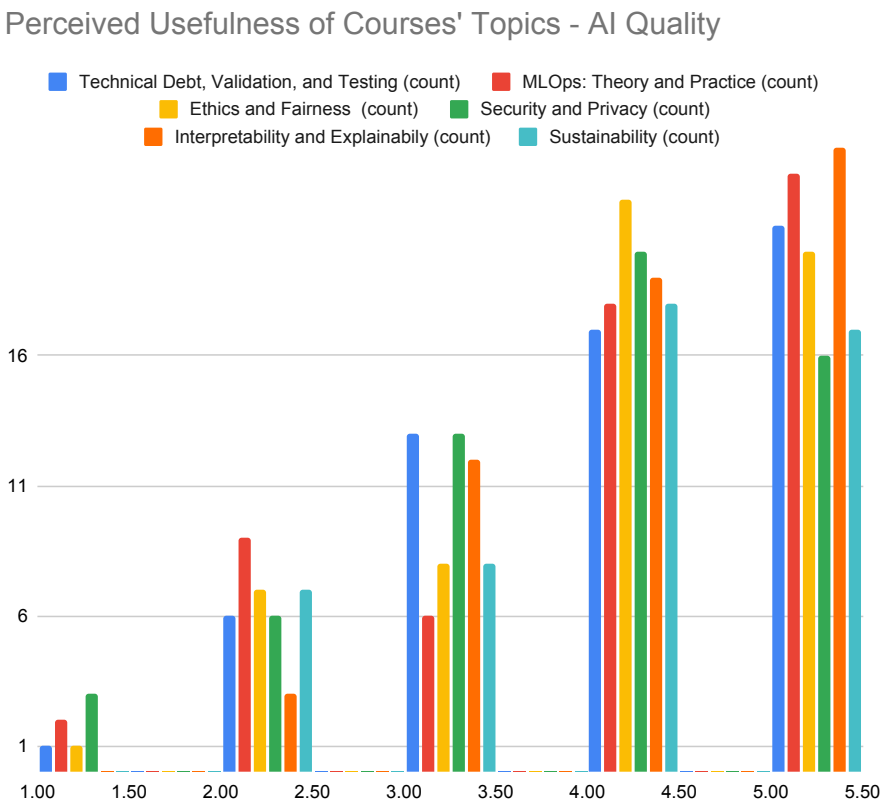

Fig. 2. Students' perceived usefulness of topics during the course.

4.2 Insights and Implications from Our Experience

According to the data acquired over the four editions of the course and our own retrospective, this section discusses the main lessons learned for each of the primary focus area.

F1 - Students' Perception of the topics. Figure 2 shows students' perceived usefulness of topics discussed during the course, rated on a scale from 1 (least useful) to 5 (most useful). Many participants valued the introduction of non-functional quality aspects such as fairness and explainability while also pushing for greater concreteness and connections to real-world applications. For instance, one participant emphasized the importance of approaching the subject matter from a highly practical perspective. Several students highlighted that a sense of concreteness in how the subject is presented is essential for fully understanding its concepts and usefulness. This feedback suggests a potential shift in teaching topics, moving toward in-person lectures that focus less on standalone theoretical definitions and more on integrating these definitions into concrete, practical examples.

F1.1 - Lesson Learned

Non-functional requirements such as fairness, explainability, and sustainability are essential in AI-enabled system development but often challenging for students to grasp in isolation. Embedding these topics in practical, real-world scenarios—through case studies or applied assignments—can improve student engagement and conceptual retention across diverse SE4AI contexts.

Concerning less appreciated topics, our analysis revealed that *technical debt* and *MLOps* emerged as areas with significant room for improvement, with MLOps being particularly variable in students' perceptions. This variability did not derive from a lack of interest; on the contrary, MLOps was among the most frequently discussed topics in open-ended questions. The feedback indicated that students considered it one of the *most interesting and promising* areas but felt that more time is needed to understand its relevance and applicability fully. In particular, advanced topics like technical debt and MLOps require an understanding of the long-term evolution of AI-enabled systems, which may not be fully conveyed within the constraints of a two-month course. This likely highlights the need for alternative teaching strategies, e.g., longitudinal case studies or follow-up courses, to provide deeper engagement with these advanced topics.

F1.2 - Lesson Learned

Topics like MLOps and technical debt demand systems-level thinking and familiarity with long-term software evolution. Educators should consider spreading these concepts across multiple modules or courses, or supporting them with longitudinal case studies, to reinforce their practical relevance and complexity.

F2 - Bridging Conceptual Gaps with Practical Tools.

Students emphasized difficulties with some advanced topics and tools introduced during the course. They specifically highlighted the need for a more

gradual introduction and additional focus on complex concepts or tools, such as generative adversarial networks (GANs), MLOps, or TensorFlow. This may stem from the diverse range of technologies introduced, which often assume a technical machine learning background that should not be taken for granted. As one student commented: 💬 *"...furthermore, I would dedicate more time and effort to recent topics such as MLOps, GANs, etc., as these are interesting but need more laboratory sessions..."*, which confirms the importance of hands-on practice in the teaching process and highlights the potential for implementing active learning approaches, such as flipped classrooms and lab sessions, to address the more challenging aspects and enhance students' understanding and application of these concepts.

F2 - Lesson Learned

Advanced tools common in AI engineering—such as TensorFlow or SHAP—can pose a steep learning curve, especially for students without strong ML backgrounds. Gradually introducing tools in problem-driven contexts, supported by hands-on labs, helps reduce overload and builds practical confidence.

F3 - Challenges in SE4AI Projects. One of the challenges revealed was the course's project-based nature. The feedback analysis revealed that integrating specific practices into a software project can be challenging and not easily intuitive in interactive machine learning development cycles. This may indicate that there is a need to give particular space to the project activity so that each student can learn and transpose the concepts to what they are working on, whose contexts were disparate, from medical applications to smart agriculture models. Concerning technical tools and methods, students' comments align with discussions in the other focus areas, as these tools are frequently revealed to be challenging for students to understand and apply in a short time. Despite this, students perceived their tutors as highly actively supporting them during the project and did not deem it too complex. This indicates that certain technical tools pose specific challenges, compensated by easier challenges in other SE4AI areas.

F3.1 - Lesson Learned

Integrating SE practices into AI-enabled systems is not always intuitive for students, particularly in iterative ML workflows. SE4AI educators should allocate dedicated space within project work for students to contextualize concepts and explore quality trade-offs relevant to their domain.

As highlighted in Section 3, the SE4AI course brings together participants from diverse knowledge areas, such as Software Engineering, Security, or Data Science and Machine Learning. This diversity has revealed broader socio-technical challenges, including differing team dynamics stemming from varied perspectives on project needs. These insights highlight the importance of providing both

technical and organizational support to help students navigate the complexities of collaborative activities in AI engineering. More importantly, it seems that project-based education may effectively simulate real-world conditions, offering students a valuable experience that prepares them for interdisciplinary collaboration in professional environments.

F3.2 - Lesson Learned

SE4AI projects often bring together students from diverse backgrounds (e.g., software engineering, ML, cloud). These perspectives can lead to valuable, real-world-style team dynamics, but may also cause misalignments. Educators should actively support interdisciplinary collaboration with coaching or reflective checkpoints.

F4 - Educational Challenges in Teaching SE4AI. Upon collection of the students' feedback and our retrospective, we could identify several challenges for educators related to the teaching of engineering practices for AI-enabled systems.

Keeping the Course Up to Date. One of the primary challenges encountered in our experience is maintaining course content that reflects state-of-the-art practices. The rapid advances in AI, especially given by the rise of Large Language Models and Foundation Models, require continuous updates to the course design. While regular updates are a common requirement for most courses, this challenge is especially pronounced in the context of SE4AI. This applies not only to the topics themselves but also to the training required for teaching novel tools and frameworks effectively.

In our case, we implemented a collaborative approach by creating a shared channel where the lecturer and teaching assistants could exchange research articles, tutorials, and other resources. These materials are then analyzed during multiple ad-hoc meetings held prior to the course's start. The objective of these sessions is to evaluate how disruptive techniques and tools can be seamlessly integrated into the course. While this strategy has been sometimes effective, it also highlights a significant challenge: *finding the right balance between breadth and depth in course content*. Covering too many topics superficially risks overwhelming students and diluting their understanding, while focusing too deeply on a few areas can leave critical knowledge gaps.

Our experience suggests that the core lectures should focus on well-established and widely applicable methods, avoiding the frequent introduction of topics that are still rapidly evolving. Simultaneously, the exploration of the latest technologies can be reserved for classwork sessions or industry talks, where students can engage with cutting-edge topics through practical case studies. This approach allows students to cultivate curiosity and deepen their knowledge independently, without disrupting the overall course structure.

At the same time, we really see the potential added value of *collaborative efforts among educators* in sharing teaching materials and best practices. Such collaboration could significantly reduce the burden on individual instructors

to independently track every development in this fast-evolving field. We hope this experience report may open a broader discussion among SE4AI educators, encouraging the development of shared resources to support the community.

F4.1 - Lesson Learned

SE4AI content evolves rapidly, making it difficult to maintain both breadth and depth. A recommended strategy is to focus core lectures on foundational methods, while reserving rapidly evolving topics for exploratory sessions (e.g., case studies, guest lectures, or flipped classrooms).

Accounting for Domain Specificity. The diversity of domains where AI is applied introduces additional complexities, as the quality assurance requirements of systems like medical diagnostics, recommendation engines, and autonomous vehicles vary significantly. Understanding these differences is essential for students to grasp how engineering practices must be adapted to meet domain-specific challenges. In our case, we incorporated diverse case studies and industry talks to provide students with real-world examples of how such practices are applied in different contexts. For instance, classwork sessions might explore fairness in medical AI applications or robustness in autonomous systems. While generally satisfactory, the results of these sessions revealed a key limitation: *students often understood the general idea of tailoring practices based on context but struggled to provide concrete, tailored solutions for engineering AI-enabled systems*. This gap suggests a need for more immersive and interactive learning methods to bridge the divide between theoretical understanding and real-world application.

In this respect, we are considering enhancing learning by integrating *game-based strategies*. For instance, by designing interactive scenarios or simulations, students could actively experiment with applying SE practices across various domains, exploring trade-offs, and crafting solutions themselves. Such an approach could not only foster active learning and critical thinking but also allow students to engage directly with diverse challenges in a controlled yet dynamic environment. We see this as an opportunity for further research and innovation in SE4AI education. The design and evaluation of effective game-based learning tools tailored to SE4AI could significantly enhance students' ability to connect theory to practice.

F4.2 - Lesson Learned

Quality assurance challenges vary significantly across AI domains (e.g., healthcare vs. recommendation systems). Using varied case studies is helpful, but immersive techniques such as domain simulations or game-based learning can deepen student understanding of domain-specific trade-offs.

Cultural Barriers and Educational Boundaries. Teaching SE4AI presents unique challenges due to its interdisciplinary nature, the cognitive shift required

for students, and the incorporation of ethical and societal considerations. A key difficulty lies in the inherent interdisciplinarity: many topics, such as MLOps, rely on foundational knowledge from areas like software architecture, cloud infrastructure, and data engineering. A single course cannot comprehensively cover these domains, requiring a *coordinated educational effort across multiple courses and disciplines*. For example, a software architecture course might address containerization and microservices, while a data engineering course could cover pipeline design and data quality, both crucial for MLOps. Viewing SE4AI education as part of a broader learning network allows each course to contribute to building the necessary skills. In our case, coordination across different research groups posed a challenge, though improved collaboration among SE4AI educators may help address this issue.

Another major barrier involves guiding students through the shift from traditional software engineering principles to those needed for AI-enabled systems. Unlike conventional systems, AI-enabled systems are probabilistic, data-driven, and less deterministic, which often creates a *cognitive gap* for students. Bridging this gap requires emphasizing AI-specific characteristics while showing how traditional practices, such as modularity and monitoring, remain relevant. Explicitly contrasting these paradigms and using hands-on activities can help ease the transition.

Finally, integrating ethical and societal dimensions adds further complexity. Topics like fairness, transparency, and environmental impact push students *to think beyond technical implementation and consider broader consequences*. Real-world failures and their repercussions can make these themes more relatable and impactful. We have observed increasing interest among students in pursuing ethical SE practices in their thesis work, reflecting the engaging nature of this educational approach.

F4.3 - Lesson Learned

The interdisciplinary nature of SE4AI—spanning software engineering, ethics, ML, and systems design—creates both pedagogical and curricular challenges. Coordination across courses, combined with contrastive examples and societal case studies, can help students bridge conceptual gaps and recognize the broader impact of AI-enabled systems.

5 Conclusion

This experience report presents preliminary insights from our four-year Master's course, *"Software Engineering for Artificial Intelligence"*. The course introduces students to end-to-end ML engineering, covering key phases from requirements to verification, validation, and deployment, with a focus on the challenges and practices relevant to AI-enabled systems.

We discuss lessons learned across four focus areas, highlighting key challenges in teaching software engineering for AI. These initial findings offer insights for

future research and educational efforts, which we plan to further refine and extend to better support the next generation of SE/AI engineers.

Acknowledgment

We acknowledge the use of ChatGPT-4 to ensure linguistic accuracy and enhance the readability of this article.

We acknowledge the support of the European Union - NextGenerationEU through the Italian Ministry of University and Research, Project PRIN 2022 PNRR "FRINGE: context-aware FaiRness engineerING in complex software systEms" (grant n. P2022553SL, CUP: D53D23017340001).

References

1. Abdessalem, R.B., Nejati, S., Briand, L.C., Stifter, T.: Testing vision-based control systems using learnable evolutionary algorithms. In: Proceedings of the 40th International Conference on Software Engineering. pp. 1016–1026 (2018)
2. Albuquerque, D., Guimarães, E., Tonin, G., Rodríguezs, P., Perkusich, M., Almeida, H., Perkusich, A., Chagas, F.: Managing technical debt using intelligent techniques-a systematic mapping study. IEEE Transactions on Software Engineering **49**(4), 2202–2220 (2022)
3. Ali, S., Briand, L.C., Hemmati, H., Panesar-Walawege, R.K.: A systematic review of the application and empirical investigation of search-based test case generation. IEEE Transactions on Software Engineering **36**(6), 742–762 (2009)
4. Anand, S., Burke, E.K., Chen, T.Y., Clark, J., Cohen, M.B., Grieskamp, W., Harman, M., Harrold, M.J., McMinn, P., Bertolino, A., et al.: An orchestrated survey of methodologies for automated software test case generation. Journal of systems and software **86**(8), 1978–2001 (2013)
5. Azeem, M.I., Palomba, F., Shi, L., Wang, Q.: Machine learning techniques for code smell detection: A systematic literature review and meta-analysis. Information and Software Technology **108**, 115–138 (2019)
6. Beede, E., Baylor, E., Hersch, F., Iurchenko, A., Wilcox, L., Ruamviboonsuk, P., Vardoulakis, L.M.: A human-centered evaluation of a deep learning system deployed in clinics for the detection of diabetic retinopathy. In: Proceedings of the 2020 CHI conference on human factors in computing systems. pp. 1–12 (2020)
7. Bocu, R., Baicoianu, A., Kerestely, A.: An extended survey concerning the significance of artificial intelligence and machine learning techniques for bug triage and management. IEEE Access (2023)
8. Bosch, J., Olsson, H.H., Crnkovic, I.: Engineering ai systems: A research agenda. Artificial Intelligence Paradigms for Smart Cyber-Physical Systems pp. 1–19 (2021)
9. Brunetto, M., Denaro, G., Mariani, L., Pezzè, M.: On introducing automatic test case generation in practice: A success story and lessons learned. Journal of Systems and Software **176**, 110933 (2021)
10. Burkov, A.: Machine learning engineering, vol. 1. True Positive Incorporated Montreal, QC, Canada (2020)
11. Byun, T., Sharma, V., Vijayakumar, A., Rayadurgam, S., Cofer, D.: Input prioritization for testing neural networks. In: 2019 IEEE International Conference On Artificial Intelligence Testing (AITest). pp. 63–70. IEEE (2019)

12. Chakraborty, J., Majumder, S., Menzies, T.: Bias in machine learning software: Why? how? what to do? In: Proceedings of the 29th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering. pp. 429–440 (2021)
13. Chen, Z., Zhang, J.M., Sarro, F., Harman, M.: A comprehensive empirical study of bias mitigation methods for machine learning classifiers. ACM Transactions on Software Engineering and Methodology **32**(4), 1–30 (2023)
14. Chenoweth, S., Linos, P.K.: Teaching machine learning as part of agile software engineering. IEEE Transactions on Education (2023)
15. Diosan, L., Motogna, S.: Artificial intelligence meets software engineering in the classroom. In: Proceedings of the 1st ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence. pp. 35–38 (2019)
16. Fioravanti, M.L., Sena, B., Paschoal, L.N., Silva, L.R., Allian, A.P., Nakagawa, E.Y., Souza, S.R., Isotani, S., Barbosa, E.F.: Integrating project based learning and project management for software engineering teaching: An experience report. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. p. 806–811. SIGCSE '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3159450.3159599, https://doi.org/10.1145/3159450.3159599
17. Galhotra, S., Brun, Y., Meliou, A.: Fairness testing: testing software for discrimination. In: Proceedings of the 2017 11th Joint meeting on foundations of software engineering. pp. 498–510 (2017)
18. Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S.: A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering **38**(6), 1276–1304 (2011)
19. Hulten, G.: Building intelligent systems: a guide to machine learning engineering. Apress (2018)
20. Kästner, C., Kang, E.: Teaching software engineering for ai-enabled systems. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training. pp. 45–48 (2020)
21. Kim, J., Feldt, R., Yoo, S.: Guiding deep learning system testing using surprise adequacy. In: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). pp. 1039–1049. IEEE (2019)
22. Lanubile, F., Martínez-Fernández, S., Quaranta, L.: Teaching mlops in higher education through project-based learning. In: 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). pp. 95–100. IEEE (2023)
23. Mariani, T., Vergilio, S.R.: A systematic review on search-based refactoring. Information and Software Technology **83**, 14–34 (2017)
24. Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A.M., Wagner, S.: Software engineering for ai-based systems: a survey. ACM Transactions on Software Engineering and Methodology (TOSEM) **31**(2), 1–59 (2022)
25. Menzies, T.: The five laws of se for ai. IEEE Software **37**(1), 81–85 (2020). https://doi.org/10.1109/MS.2019.2954841
26. Menzies, T., Zimmermann, T.: Software analytics: What's next? IEEE Software **35**(5), 64–70 (2018)
27. Miller, C.C.: Can an algorithm hire better than a human. The New York Times **25** (2015)

28. Nemoto, T., Beglar, D.: Likert-scale questionnaires. In: JALT 2013 conference proceedings. pp. 1–8 (2014)
29. Ni, J., Chen, Y., Chen, Y., Zhu, J., Ali, D., Cao, W.: A survey on theories and applications for self-driving cars based on deep learning methods. Applied Sciences **10**(8), 2749 (2020)
30. Palomba, F., Voria, G., Parziale, A., Pentangelo, V., Della Porta, A., De Martino, V., Recupito, G., Giammaria, G.: Online appendix - teaching software quality assurance for artificial intelligence: An experience report `https://figshare.com/s/c0797c1cd4569ce3d285`
31. Peng, K., Chakraborty, J., Menzies, T.: Fairmask: Better fairness via model-based rebalancing of protected attributes. IEEE Transactions on Software Engineering **49**(4), 2426–2439 (2022)
32. Recupito, G., Pecorelli, F., Catolino, G., Lenarduzzi, V., Taibi, D., Di Nucci, D., Palomba, F.: Technical debt in ai-enabled systems: On the prevalence, severity, impact, and management strategies for code and architecture. J. Syst. Softw. (Oct 2024). `https://doi.org/10.1016/j.jss.2024.112151`
33. Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., Tonella, P.: Testing machine learning based systems: a systematic mapping. Empirical Software Engineering **25** (2020)
34. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F., Dennison, D.: Hidden technical debt in machine learning systems. Advances in neural information processing systems **28** (2015)
35. Smith, J.: Machine Learning Systems: Designs that Scale. Simon and Schuster (2018)
36. Sperling, A., Lickerman, D.: Integrating ai and machine learning in software engineering course for high school students. In: Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education. pp. 244–249 (2012)
37. Wan, Z., Xia, X., Lo, D., Murphy, G.C.: How does machine learning change software development practices? IEEE Transactions on Software Engineering **47**(9), 1857–1871 (2019)
38. Wang, P., Fan, E., Wang, P.: Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. Pattern recognition letters **141**, 61–67 (2021)
39. Yang, Y., Xia, X., Lo, D., Bi, T., Grundy, J., Yang, X.: Predictive models in software engineering: Challenges and opportunities. ACM Transactions on Software Engineering and Methodology (TOSEM) **31**(3), 1–72 (2022)
40. Yang, Y., Xia, X., Lo, D., Grundy, J.: A survey on deep learning for software engineering. ACM Computing Surveys (CSUR) **54**(10s), 1–73 (2022)
41. Zhou, J., Chen, F.: Human and Machine Learning. Springer (2018)