Teaching Engineering of AI-intensive Systems

Atif Mashkoor, Wesley Klewerton Guez Assuncao, Alexander Egyed

December 15, 2023

Abstract

With the increasing prevalence of AI, a key question is how to adequately prepare the next generation of software engineers to build AI-intensive systems. This article presents our teaching experience for the "Engineering of AI-intensive Systems" course to postgraduate students. This course is tailored for computer science students, bridging the disciplines of software engineering (SE) and artificial intelligence (AI). The primary goal is to equip participants with the knowledge and skills to adeptly engineer AI-intensive systems with a strong foundation in conventional SE principles. The article delves into the course's structure, teaching methods, and assessment techniques, underscoring the advantages inherent in this interdisciplinary educational approach.

1 Introduction

The use of AI-intensive systems has been rising for the last few years and AI-intensive systems, such as Uber[1] or Netflix[2], have taken the world by storm in today's fast-paced world [1]. Among other features, AI-intensive systems require high concurrency in data access (e.g., concurrent access to rides and bookings on the Uber app), fast-changing data streams (e.g., quick recommendation of the next available ride/booking), and fast analytics (e.g., Netflix's transcoding and encoding of a video according to the receiving device). However, these systems must be developed and governed by software engineering principles for reliability, maintenance, and compliance [2]. Integrating data analytics and software engineering is important as it offers unique opportunities for innovation, problem-solving, and decision-making processes.

AI aims to replicate or simulate human intelligence in machines and systems. It incorporates statistical analysis and other machine learning mechanisms to extract meaningful patterns, visualizations, and actionable insights from data. It consists of data collection, preprocessing, exploratory data analysis, predictive modeling, and data-driven decision-making through machine learning or deep learning [3]. On the other hand, SE lays the foundation of software modeling, design, analytics, and maintenance using modeling tools, programming languages,

[1]`https://www.uber.com/us/en/uberai/`
[2]`https://research.netflix.com/research-area/machine-learning`

and quality assurance mechanisms [4]. While AI engineers are responsible for finding patterns, trends, and insights in datasets using machine learning algorithms and making recommendations, software engineers make sure that the software governing that data and insights is functional and dependable. The interdisciplinary collaboration between the engineers of these two fields is necessary for enhanced decision-making, increased predictive capabilities, advanced analytics, low maintenance, etc.

Making AI-intensive systems that handle diverse data or provide useful insight for businesses (functional requirements) while meeting expected quality standards (non-functional requirements like response time, security, and maintainability) requires specific skills. However, AI-intensive systems and their engineering are still poorly understood at large [5]. This is made worse by AI and SE being seen as independent fields of knowledge, with different degree programs and sparsely overlapping courses. Thus, training the workforce with interdisciplinary knowledge to build software leveraging data analytics, AI, and machine learning following SE practices is challenging. A few initiatives have been created to bridge both AI and SE fields [6]. However, to meet the demands of an ever-increasing market, the education of students on how to build AI-intensive systems must be pervasive in every computer science education.

We initiated a novel course titled “Engineering of AI-intensive Systems” to promote collaboration between students from AI and SE. Although dedicated master courses target AI or SE at Johannes Kepler University, they have no direct connection. Hence, the course’s objectives were two-fold: firstly, introducing AI students to the fundamental principles of SE, and secondly, providing SE students with practical experience in tackling real-world AI challenges. The core concept was establishing mixed groups comprising AI and SE students. These groups collaborate on addressing realistic term projects derived from the AI domain while adhering to established SE practices, including modeling and comprehensive analysis involving verification and validation procedures. This course was introduced within the computer science postgraduate program at Johannes Kepler University during the summer semester 2023. The course design aimed to effectively translate the instructors’ extensive research expertise in AI and SE into a cohesive teaching approach. This article shares the insightful experience gained through the implementation of this course, shedding light on the successful merging of these two fields within an educational context.

2 Background

SE for AI, as well as AI for SE, are thriving fields. On the one hand, Shafiq et al. [7] present a study about using AI across various software development lifecycle stages. Additionally, they investigate the relationship between software development lifecycle stages and machine learning tools, techniques, and types. Similarly, Colanzi et al. [8] reviewed search-based software engineering, in which SE problems are modeled as search problems and solved by AI and machine learning techniques. On the other hand, Amershi et al. [9] present a study

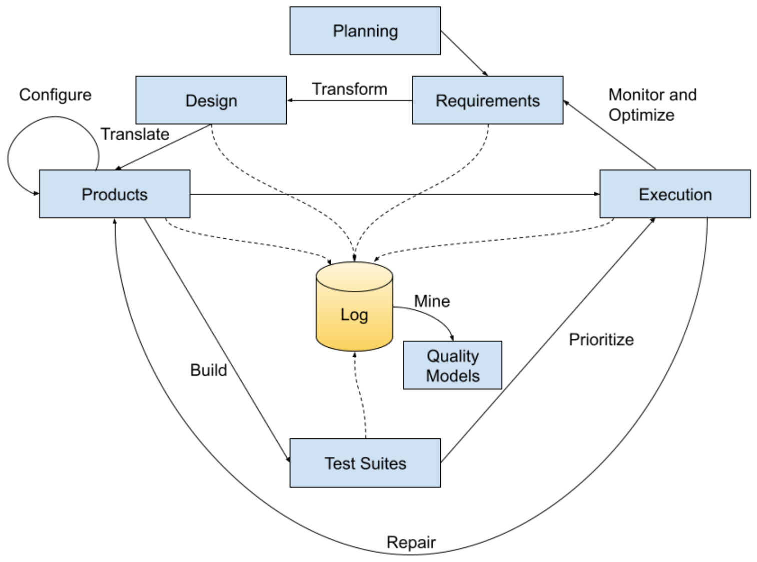

Figure 1: Lifecycle for engineering AI-intensive systems [2]

containing observations of SE practices for developing AI-based applications. To further advance the state of the art, Mashkoor et al. [2] presented a lifecycle for engineering AI-intensive systems as shown in Fig. 1, explaining how AI helps in various lifecycle stages for engineering software systems and how the knowledge of engineering software systems can be used to design AI applications.

As Fig. 1 suggests, a typical AI-intensive system engineering project starts with the planning phase. Here, AI can help sprint planning, for example, by suggesting the most critical issues that need to be fixed on a priority basis [10]. For the requirements phase, AI can help, for example, in task allocation [11]. For the design phase, AI helps, for example, identify and recognize design patterns in software through source code or user interface layout [12]. AI tools also help configure the software in the product development phase, for example, through automatic code generation [13]. Once configured and executed, AI tools can monitor and optimize the software through performance monitoring, anomaly detection, predictive maintenance, etc. AI tools can also help build test suites that reach all parts of a software system. However, large test suites can be slow and expensive to run. Hence, AI tools help reduce costs by prioritizing test cases likely to fail and giving developers faster feedback on problematic issues. Once systems are running and test case results are available, AI can support the software development process again by repairing the software product through automated bug detection and fault localization, automated code refactoring, automated patch generation, predictive bug analysis, etc. Finally, AI tools can study code repositories and logs to learn quality models that predict software development time, bug locations, how long issues will be resolved, anti-patterns in software development, etc. We discussed all these phases during the lectures, showing how AI-assisted technologies can support development. This course acts as a test-bed whether the lifecycle proposed for the engineering of AI-intensive systems, as shown in Fig. 1, is viable and serves as the basis for teaching the engineering of such systems.

From a more educational point of view, a study by Kästner and Kang [6] shares the experience of teaching SE for AI-enabled systems. This study is related to this present work. However, their course was designed to teach SE to students interested in AI. In a complementary way, our course's aim is much broader. We were teaching students of interdisciplinary fields to work in tandem to design real-life AI applications using standard SE principles.

3 Course overview and learning outcomes

The course was offered at the postgraduate level to computer science students to teach them how to employ state-of-the-art SE practices to engineer AI-intensive systems. The course aimed to allow students to develop: (i) enhanced interdisciplinary problem-solving skills, (ii) improved understanding of interdisciplinary software development processes, and (iii) further career prospects with the latest trends in the market.

The following were the main learning outcomes (LO) of the course:

- LO1: Students become familiar with the lifecycle stages of systems and software engineering and how AI can assist in different stages.
- LO2: Students become familiar with statistical modeling, analysis, and data management.
- LO3: Students develop familiarity and/or expertise with relevant programming languages such as Python[3] or Dart[4], frameworks such as Langchain[5] or Flutter[6], and APIs such as FastAPI[7] or Monster API[8].
- LO4: Students can apply AI and SE concepts and methods to solve real-life problems and communicate these solutions effectively.

[3]`https://www.python.org/`
[4]`https://dart.dev/`
[5]`https://python.langchain.com/`
[6]`https://flutter.dev`
[7]`https://fastapi.tiangolo.com`
[8]`https://monsterapi.ai`

4 Teaching and assessment

This section presents an overview of the course organization, resulting projects developed by the students, and assessment.

4.1 Lecture-based instruction

The course was given in ten lectures of one and a half hours each over a semester. We introduced students to the necessity for AI-intensive systems during the introductory lecture. Then we briefly discussed the fundamentals of AI, machine learning, and statistical and data modeling. Then, the remaining lectures focused on the following topics:

- *Systems engineering using SysML.* This topic covers systems engineering fundamentals, including engineering and modeling processes, design choices, tradeoffs, etc. We also introduce SysML [14] because system artifacts are often modeled and designed using such visual modeling languages. We also discuss how AI could be a game changer in systems engineering and could significantly improve the processes of generating and specifying requirements, use case and user story writing, SysML model creation, automatic code generation, etc.

- *AI systems engineering lifecycle.* This topic covers the AI systems engineering lifecycle, shown in Fig. 1. We start with the planning phase and then proceed to the requirements, design, production, testing, and execution phases. At each stage, we discuss how AI-intensive systems differ from traditional systems and what practices can be adopted for their engineering.

- *Requirements engineering.* This topic covers the fundamentals of requirements engineering for AI-intensive systems. We particularly emphasize how the non-functional requirements for AI differ from traditional software systems and the new categories of non-functional requirements for AI-intensive systems, such as fairness, explainability, transparency, and ethics. Then we discuss different quality standards and modeling processes, particularly the IEEE standard model process for addressing ethical concerns during system design (IEEE 7000-2021)[9].

- *Design.* This topic covers how to design AI-intensive systems and what opportunities and challenges designers may encounter during this process. Naturally, the design needs to focus on the features the system and its AI components provide. However, considering human-centered AI aspects during the design stage is crucial to counter biased data sets, discrimination against minorities, or privacy threats. This increases the trustworthiness of the AI-intensive system. We also touch upon the unavailability of related standards, guidelines, and best practices for AI-intensive systems,

[9]`https://standards.ieee.org/ieee/7000/6781/`

as existing standards generally apply to systems and software. We also discuss the ethical aspects of designing AI-intensive systems, such as the onus of responsibility for potential mistakes.

4.2 Collaborative learning and group projects

One of the fundamental objectives of the course was to give students hands-on experience with the engineering of real-life AI-intensive systems using the proposed lifecycle and avoid using simple toy/classroom examples. To this end, we decided to allow students to build their groups and come up with their proposals for the term projects with two conditions: (i) The team should comprise both AI and SE students, and (ii) the project should represent a real-life problem and display the use of AI as a part of the solution.

We divided students into four groups, and the following are their projects:

Gesture control: This project proposes a gesture detection device that helps people suffering from tenosynovitis, i.e., inflammation and swelling in wrists caused by repetitive movements such as typing or mouse control. The project allows users to record gestures and map them to a particular task on the computer. The device then uses AI to detect when the user performs the gesture and simulates the action the user previously assigned to this gesture. The system's architecture comprises motion sensors, a microcontroller (Arduino[10]), and software. A short demo of this project is available at the following URL: `https://www.youtube.com/watch?v=SJex-WGscfM`.

Recipe finder: This project is about developing a recipe finder app that allows users to prepare tasty recipes based on the available ingredients. Additionally, it uses AI to generate matching images for the selected dishes. The app uses a simple client/server architecture, was trained on a dataset of 0.5 million recipes, and took approx. three seconds for dish image generation. The app uses the FastAPI framework at the back end and Material UI[11] at the front end. A short demo of this project is available at the following URL: `https://www.youtube.com/watch?v=4JKs9xmcOTY`.

Image generator: This project is about a mobile app providing users with AI-generated images related to nature based on the provided written description. The app is built using the Dart programming language[12] for the open-source Flutter framework[13] by Google. Additionally, it uses the Monster API[14] to generate relevant images. Users can either download the image as is or make further edits before downloading the image. A short demo of this project is available at the following URL: `https://www.youtube.com/watch?v=FWBHLQyeYig`.

[10]`https://www.arduino.cc/`
[11]`https://mui.com`
[12]`https://dart.dev`
[13]`https://flutter.dev`
[14]`https://monsterapi.ai`

Customer support: This project concerns an AI-powered sales support agent (chatbot) based on the Langchain framework. Langchain is a natural language understanding framework for developing data-aware and agentic applications using large language models. The chatbot is limited to processing human queries only in English. A short demo of this project is available at the following URL: `https://www.youtube.com/watch?v=EpgR59UoC3M`.

4.3 Assessment

As a part of the assessment, students had to work iteratively on the project. They first had to show their requirements documents (natural language requirements, requirement and use case diagrams, requirements specification, etc.) and later design documents (state machine diagram, activity diagram, etc.) to course instructors, who acted as relevant stakeholders. Students also had to show that they had documented requirement changes and resulting design decisions properly. Final project artifacts also contained requirements traceability links to code. At the end of the semester, students had to showcase their projects in front of the class and defend their requirements and choices. Students also had to demonstrate a grasp of concepts from both AI and SE fields through a written exam, mainly based on knowledge already imparted through lectures or gained while working on the project.

Ultimately, we were happy that all learning outcomes of the course were achieved successfully. In meetings with instructors and during presentations, student groups showed that they followed the recommended practices for different lifecycle stages of systems and software engineering, e.g., requirements management and architecture and design (LO1). During project development and its showcasing, students showed that they were familiar with AI and data science concepts (LO2), they were able to use different programming languages and frameworks effectively (LO3), and they could use SE principles for engineering AI applications (LO4). The approach of mixing students from interdisciplinary fields in a single group worked out well.

5 Challenges and lessons

Integrating diverse skill sets: AI and SE are two different fields requiring different skill sets. While AI requires familiarity with statistical modeling and machine learning models, SE mandates comprehension of software design principles and testing techniques. Integrating these skill sets in a collaborative environment can sometimes be challenging, such as differences in terminologies of both fields or differences between the nature of software, i.e., deterministic vs. probabilistic. However, grouping students from both fields worked well, and the students could learn from each other by working on collaborative projects. In the final assessment, all students showed a grasp of concepts from both fields.

Tooling and infrastructure: AI and SE students use different tools and platforms. For example, using TensorFlow[15], PyTorch[16], and Jupyter notebooks [15] is prevalent in AI. Whereas software engineers use tools like integrated development environments, version control systems, issue tracking and management platforms, and CI/CD environments. Students from both fields had to learn new tools and platforms for working collaboratively.

Balancing theory and practice: Balancing theory and practice is a general problem in any course. However, we addressed this challenge in our course by allowing hands-on exercises from real-life problems; project-based learning, i.e., the projects allowed students to apply theoretical knowledge into practical context; peer learning, i.e., students sharing their experiences and solutions among each other; constant feedback, i.e., students' assignments were reviewed periodically over the semester; and constructive assessments.

Limited applicability: We had a class of 30 students - maximum class size at Johannes Kepler University for such a course. The class was mainly balanced in terms of students from diverse background in computer science. As the course was not mandatory, most students just came to learn and not pass a course. Still, many students left the course in the middle due to obligations for mandatory courses. Although the initial response from students is fantastic, due to the limited sample size, the generalizability of the course results is limited.

Course design improvement: Primarily, the course was given by the SE department. However, all instructors had ample experience in designing and implementing AI-intensive systems. The current course setting was adequate but not ideal. We believe participation from both fields is necessary for teaching such as course. In the future, we intend to offer this course jointly from AI and SE departments requiring at least one instructor/subject expert from each field.

6 Conclusion

This research article details our experience of delivering the "Engineering of AI-intensive Systems" course to postgraduate students at Johannes Kepler University Linz, Austria. The course aimed to unite AI and SE students seamlessly by allowing them to employ fundamental SE principles to craft AI-intensive systems. The article covers this cross-disciplinary educational paradigm's framework, pedagogical approaches, and assessment methods.

[15]`https://www.tensorflow.org`
[16]`https://pytorch.org`

Acknowledgements

We gratefully acknowledge the LIT AI Lab (funded by the State of Upper Austria) and Austrian Science Fund (FWF) grants I4744 and P31989 for supporting this work.

References

[1] Y. Jiang, X. Li, H. Luo, S. Yin, and O. Kaynak, “Quo vadis artificial intelligence?,” *Discover Artificial Intelligence*, vol. 2, mar 2022.

[2] A. Mashkoor, T. Menzies, A. Egyed, and R. Ramler, “Artificial intelligence and software engineering: Are we ready?,” *Computer*, vol. 55, no. 3, pp. 24–28, 2022.

[3] A. V. Joshi, *Machine learning and artificial intelligence.* Springer, 2020.

[4] P. A. Laplante and M. Kassab, *What every engineer should know about software engineering.* CRC Press, 2022.

[5] A. Bewersdorff, X. Zhai, J. Roberts, and C. Nerdel, “Myths, mis- and preconceptions of artificial intelligence: A review of the literature,” *Computers and Education: Artificial Intelligence*, vol. 4, p. 100143, 2023.

[6] C. Kästner and E. Kang, “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.

[7] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, “A literature review of using machine learning in software development life cycle stages,” *IEEE Access*, vol. 9, pp. 140896–140920, 2021.

[8] T. E. Colanzi, W. K. Assunção, S. R. Vergilio, P. R. Farah, and G. Guizzo, “The symposium on search-based software engineering: Past, present and future,” *Information and Software Technology*, vol. 127, p. 106372, 2020.

[9] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, “Software engineering for machine learning: A case study,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 291–300, 2019.

[10] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, “Nlp4ip: Natural language processing-based recommendation approach for issues prioritization,” in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 99–108, 2021.

[11] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "Taskallocator: A recommendation approach for role-based tasks allocation in agile software development," in *2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE)*, pp. 39–49, 2021.

[12] H. Washizaki, F. Khomh, Y. Gueheneuc, H. Takeuchi, N. Natori, T. Doi, and S. Okuda, "Software-engineering design patterns for machine learning applications," *Computer*, vol. 55, pp. 30–39, mar 2022.

[13] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Chi conference on human factors in computing systems extended abstracts*, pp. 1–7, 2022.

[14] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language.* Morgan Kaufmann, 2014.

[15] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, *et al.*, "Jupyter notebooks-a publishing format for reproducible computational workflows.," *Elpub*, vol. 2016, pp. 87–90, 2016.