Artificial Intelligence Meets Software Engineering in the Classroom

Laura Diosan*
Simona Motogna*
lauras@cs.ubbcluj.ro
motogna@cs.ubbcluj.ro
Babeş Bolyai University
Cluj-Napoca, Romania

ABSTRACT

We aimed to assess the reliability of teaching Artificial Intelligence for Software Engineering master students. We propose a semi-interactive course where the students have to develop applications for solving real world problems by using various intelligent tools. We try to integrate these two disciplines, since both deal with modeling of the real case studies, sharing some common elements. We report on a study that we conducted on observing student teams as they develop AI-based applications. We validate the proposed semi-interactive course by using various criteria. In addition, we checked if some best practices from industrial teams are followed by our students.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Software engineering education, Software creation, theory and algorithms for application domain

ACM Reference Format:
Laura Diosan and Simona Motogna. 2019. 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 (EASEAI '19), August 26, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340435.3342718

1 INTRODUCTION

The content of a course must cover the fundamental aspects of the discipline, such that it will contribute to the overall formation of professionals in a specific domain. In computer science, the overall accepted guidelines are provided by ACM Curricula Recommendations [1], which offer relevant information about what a course should contain and what principles to follow in constructing an entire curricula. According to ACM, "Software engineering spans the entire software lifecycle - it involves creating high-quality, reliable programs in a systematic, controlled, and efficient manner using formal methods for specification, evaluation, analysis and design, implementation, testing and maintenance".

In the last decades Software Engineering (SE) curricula had focused on knowledge regarding these activities. However, the last decade has seen a significant increase in complexity and size of software products, as well as an unforeseen extension of application domains. Software has become more "intelligent", more "connected" and more "mobile". The advances obtained in Artificial Intelligence (AI) have made their way to be incorporated in modern applications, in order to solve more complicated problems, to provide better solutions (optimized) or to learn from their own behavior. These are the reasons why at this moment many software engineers will be required to design and implement AI based application solving real life problems (with a special attention to address societal problems and their challenges).

The curricula design should establish some course objectives and must take into consideration the amount of time dedicated to teaching and to study. Based on this, we then have to set evaluation criteria that should clearly reflect qualitative and quantitative measures of students' knowledge. The design should also take into account the level of knowledge and abilities a student who elects the course should have, so there should be a distinction between introductory courses (suitable for bachelor degree studies) and advanced topics (suitable for master degree studies). Bloom's taxonomy of learning [3] is mapped to course objectives based on this classification, due to the fact that the objectives are targeting tasks such as apply, analyze and evaluate.

The purpose of this paper is to report our experience of designing a course for software engineering master students, that integrates intelligent methods into real life applications. The main focus was to have a problem-oriented approach for the course, to challenge the students to gain project and team work experience. We begin by describing the principles in designing the course, then the course content. The projects and the development stages are presented in details, in order to highlight the specific software development stages for an AI solution, followed by project and course evaluation. In the end, we give some conclusions and consider some future improvements of the course.

*Both authors contributed equally to this research.

EASEAI '19, August 26, 2019, Tallinn, Estonia

ACM ISBN 978-1-4503-6852-0/19/08…$15.00
https://doi.org/10.1145/3340435.3342718

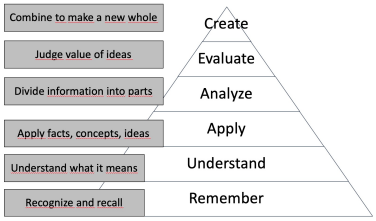

Figure 1: Bloom Taxonomy, adapted from [3]

2 COURSE STRUCTURE AND CONTENT

The course Intelligent Tools for Social Good is designed for Master students in Software Engineering. The premises from which we are building knowledge in this course are: the students have a solid background in Computer Science (they hold a B.Sc. in Computer Science), with good competences in software development (programming languages, IDE, software development processes and activities), and followed an introductory course in Artificial Intelligence (including basic Machine Learning algorithms). The design of the course followed three main principles that lead the construction of the content and the evaluation of the students:

(1) Interdisciplinarity: combine SE concepts with AI approaches
(2) Real-life challenging problems: not easy, when a brute force solution is not appropiate
(3) Soft skills: problem solving, team work, communication.

As we address, mainly, software engineering students, we evaluated the course objectives against the recommendations for such study programs [5]:

- "Focus on lasting principles rather than last-minute fashionable technologies and buzzwords": algorithms for solving classes of problems were presented at the course, discussing the usefulness of different solutions based on performance and efficiency; also, the project requirements asked to consider at least two possible solutions and to evaluate them;
- "Integrate class teaching with projects": the course has two teaching classes and one project class per week, followed by individual work, in order to combine the necessary theoretical knowledge with necessary development skills to implement them;
- "Try to make things easy and understandable": the theoretical part will not concentrate on the development of algorithms, but rather on their applicability;
- "Teach how to select and evaluate different methods and approaches rather than follow them like recipes": several algorithms are discussed with classes of problems; also, one of the main tasks of the project was the evaluation of the solution with a relevant data set.

2.1 Course Content

In order to tackle some challenges of today's society the course brings together several intelligent methods:

- optimization algorithms: classic algorithms (e.g. Greedy), but also modern ones such as nature-inspired (e.g. Evolutionary Algorithms [6])
- machine learning: supervised and unsupervised learning algorithms, traditional (e.g. Support Vector Machines [4], Decision Trees [12]) and modern ones (e.g. Deep Learning [2], Genetic Programming [10], Cellular Automata [14]).

Table 1: Body of Knowledge covered by the course

KA	Unit	Course	Project
SE	Software Project Management	0	2
SE	Tools and Environments	0	5
SE	Requirements Engineering	0	2
SE	Software Design	0	3
SE	Software Construction	0	3
AL	Advanced Data Structures Alg.	4	8
IM	Information Management Concepts	1	2
CN	Modelling and Simulation	4	9
IS	Advanced Represent. and Reasoning	5	10
IS	Advanced Machine Learning	5	10
IS	Perception and Computer Vision	5	10
SDF	Algorithms and Design	4	8

All these methods are deeply analysed and involved in solving social problems from various domains: medical/healthcare, safety/automotive, environmental sustainability. The final aim is to bridge the gaps between these areas, enabling interdisciplinary cooperations in a scientific and socially highly topical area through a shift in thinking towards new perspectives and solutions.

The content of the lectures are problem oriented, following a path from real life problem to model to applied algorithm, rather than from theory to practice. "Learning by doing"[5] is more effective in this case, since the course addresses master students, which already have a good theoretical background.

The course follows an active/interactive way of teaching, by offering examples and exercises and organizing a series of debates such that the students analyze and evaluate several solutions in order to find the optimal one.

A secondary aim of the course is to bring together all three academical components: knowledge generation by research, knowledge dissemination through teaching and knowledge usage by developing services to society (even if in a primary form— proof of concepts, not as commercial one — final products). The entire process of knowledge management is student-centric, which not only helps bring us convenience but also presents new challenges of processing and analyzing massive real data.

The content of the course and of the project have been designed according to the ACM Curricula recommendations [1], and cover different knowledge units from several knowledge areas, as shown in Table 1; the numbers associated to columns *Course* and *Project* represent the designed number of hours from the total allotment dedicated to the corresponding knowledge unit. More detailed information about the course can be found at https://softwareengubb.wordpress.com/page/. The focus on applicative part can be also noticed from this time allocation. For the aspects related to SE, we focused on practical aspects, that's why the allocation is on project time. However, this year experience showed us that some topics regarding these aspects should be included in the lectures.

2.2 Projects

The aim of project development was to *familiarize the students with specific software development stages for AI based solutions.* Software development phases are distinct when AI techniques are involved because: it is a more iterative process (depending on finding, integration and optimization of the model, which means several tries); it is more dependent on the application domain due to necessary estimations; and finally, it is more client-oriented since it requires clients' feedback for the model choice and model validation.

The students were asked to form teams, such that each individual student will learn teamwork skills, and each team will self organize and manage its' own project, as advised by experiential learning in teams [7]. 32 students formed 12 teams (the teams were decided based on students choices, so teams of 2 up to 4 students were formed), each choosing their a single project topic . These topics varied from real-life problems, such as automotive/driver assistance (pedestrian detection, traffic sign detection and recognition), medical (heart chamber segmentation, motion tracking for kinetic-therapy), to theoretical problems: computer vision (edge detection, image recognition), planning (scheduling), community detection in complex networks and multi-objective optimisation problems. Other SE topics such as project management, risk assessment are presented at other courses of the same program. The essential requirement was to use intelligent tools (AI algorithms from various libraries/frameworks: sklearn, Weka, Orange, RapidMinner, OpenCV).

The project was divided in several stages, corresponding to the specific methodology of developing AI based software application:

- Data management (collection, cleaning, labelling), with the following purposes:
  - transfer learning: from general or synthetic data to more specialized data — 83% (= 10 out of 12) of projects have successfully implemented transfer learning. Transfer learning refers to the adaptation and improvement of a model developed for a task in order to be used (reused or as starting point) on a new task (with more specialized data), by exploiting its generalization ability. Developing custom models, but general at the same time, requires both SE skills and ML knowledge.
  - feature engineering (data transformation/pre-processing) — only 50% (= 6 out of 12) of projects improved the performance of problem solving by explicit extraction of more informative features. Possible explanations: quality of data used strongly influences the identification of a good intelligent model (an accurate classifier or predictor). Without data, an ML algorithm does not work. Therefore, it is very important to do a good management of data: discovering the most appropriate and useful data, sourcing and storing the data, versioning and selecting some of data. Such a task requires competent specialists. In addition, the majority of problems were solved by using fully automatic learning algorithms (e.g. Convolutional Neural Networks that are able of performing simultaneously both feature learning and model learning). The final aim was to solve an important social problem by using an intelligent tool and not to develop a new one.
- Feedback loop for a good model — in 67% (= 8 out of 12) of projects the students have tried to tune the learnt model (identifying the best hyper-parameters and custom the solving method). The workflow of an AI-feature application is not linear [11], because it can contain one or more feedback loops. Model training and evaluation is a stage where the developers have to repeat various experiments, along more iterations, until the model converges to an optimal one. Integrating the ML workflow in the developing process [8, 9] was a real challenge for our students.
- Model improvement (AI/research perspective) – in 17% (= 2 out of 12) of projects, we have identified original contributions from a research perspective (by proposing an original intelligent solving algorithm or by enriching an existing method with new extensions); even if we addressed master students, the focus was on building software applications with intelligent features in order to solve a real-world problem (and not to develop new intelligent techniques).
- Statistical analysis of the results — we have asked that each problem to be solved by two or more different approaches (two different intelligent mechanisms) and thus, in order to evaluate the quality of problem solving, the students had to compare the results obtained by these AI-based methods. The comparison must be performed in terms of quality (error, accuracy, precision, etc.) and of complexity. The qualitative perspective involves the usage of some statistical tests in order to quantify the generalization ability of the learnt models (e.g. confidence intervals for ensuring that if we change the input data, the learnt model is able to deal with it). In addition, numerous AI-based problem solving technologies provide, to the end of the learning process, complex black-box models that suffer from interpretability; in this context, the programmers have to prepare either some visualizations of the trained models or to try to quantify and to reduce the model complexity. In our case, in 58% (= 7 out of 12) of projects this analysis stage was performed.

2.3 Evaluation

Teacher's perspective: One of the course requirements was to do a project; this project is an opportunity for students to explore an interesting machine learning problem of their choice, whether empirically or theoretically. In general, the students had to conduct a small experiment (pick a dataset and apply an appropriate machine learning algorithm) and prepare a scientific report. Both, the project and the report, were evaluated based on a few criteria: the clarity of the ideas and concepts, the technical quality of the application (with a special attention to the hidden technical debt in ML developing process [13] such as boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, configuration issues, changes in the external world, a variety of system-level anti-patterns), the extensiveness of the study and experiments, the analysis of the results, the writing style and the clarity of the written paper, the quality of the final oral presentation. All these criteria are mapped into experience points. Some penalties have been applied when the deadlines have passed.

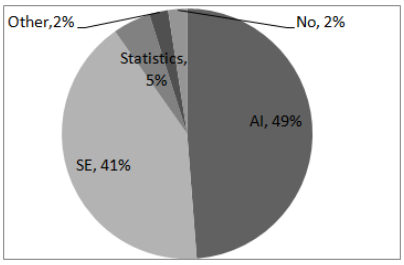

Figure 2: Other courses

In addition, a pro-active context was prepared for project dissemination: each question asked or discussion initiated by a student who is not part of the project team was granted with several experience points.

Student's perspective: In order to obtain a fair feedback from the students about this course, we have constructed a questionnaire divided in 3 parts: course content, project experience and evaluation methods. 25 students have responded to all questions. Most of the students considered the course challenging enough (52% voted for medium difficulty, none for easy, 1 for very difficult, 20% for low difficulty, and 24% for difficult), while most of them considered that incorporating such a tool in a software system is rather difficult (48% choose medium difficulty, and 40% choose difficult). For the project part, the challenges were to adapt the tool for the specificity of the problem (52% considered difficult, and 32% considered medium difficult) and to integrate the tool into the application (52% said it was medium difficulty, and 24% said it was difficult). From an experience point of view: almost half of the students declared that the tool influenced the choice of the programming language. 72% of them used traditional IDE, and only one student used a specific tool (namely, Jupyter notebook). The questionnaire also showns that most of the students (72%) managed to go through all development stages, while the other 28% didn't, mostly due to bad time management. Our assumption about interdisciplinarity was well founded, since students considered they used knowledge from other courses: AI (48%), SE (41%), but also Statistics (7%), as shown in Figure 2. The feedback shown that the students appreciated the evaluation, considering it fair and did not proposed any change.

The complete questionnaire and the students responses can be found at https://softwareengubb.wordpress.com/page/

3 CONCLUSIONS AND PERSPECTIVES

The current study presents the experience gained while introducing a course with two main purposes: i) use of AI methods and algorithms in solving real life problems and ii) integrate AI specific solutions in software systems. We explained the principle in designing the course content, with a special attention dedicated to the applicative part. We described in detail how the application stages interleaves AI tasks (model representation, training, statistical analysis) with SE tasks (requirements engineering, SE processes, project management). In the end, we give an overview of the course evaluation. The essential characteristics that differentiate the course from a generalist one, is dealing with real life industry data and specific software development phases for integrated AI projects.

Instead of comparing this approach with related work (which is very difficult considering the multitude of existing courses), we tried to analyze how the ACM recommendations are identified in the course and how much the good practices [5] are satisfied.

The course activities performed identified some challenges, that will be addressed in order to improve this course. First at all, the computing power of the students' laptops have raised some issues related to the data used and the selected algorithms; a solution can be to use a small cluster with enhanced computing capabilities. Secondly, both from a learning and teacher evaluation perspectives, an improved versioning discipline will be desirable; the students, in most of the cases, have kept versions from the development point of view (using GitHub), but failed to achieve this from project management. Lastly, the biggest threat to the successful completion of the course was time management; a defective time management of the student influenced evaluation of projects.

This analysis has highlight the improvements and future developments that can be brought to this course. In order to make the project more realistic, a domain expert from the application domain should be involved in different phases of the project: to establish the model, to analyze the results, and to validate the model integrated into the software solution. Another future extension, based on the students feedback will be to propose projects topics related to more familiar areas of interest from day-to-day life.

REFERENCES

[1] 2013. Computer Science Curricula 2013. Retrieved April 22, 2019 from https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf
[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.
[3] Engelhart M. D. Furst E. J. Hill W. H. Krathwohl D. R. A. Bloom, B. S. 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook 1: Cognitive Domain.* David McKay Co, New York.
[4] C. Cortes and V. Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20 (1995), 273.
[5] Carlo Ghezzi and Dino Mandrioli. 2005. The Challenges of Software Engineering Education. In *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*. 637–638. https://doi.org/10.1145/1062455.1062578
[6] D. E. Goldberg, K. Deb, and J. H. Clark. 1992. Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems* 6 (1992), 333–362.
[7] Anna B. Kayes, D. Christopher Kayes, and David A. Kolb. 2005. Experiential learning in teams. *Simulation & Gaming* 36, 3 (2005), 330–354. https://doi.org/10.1177/1046878105279012
[8] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The emerging role of data scientists on software development teams. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 96–107.
[9] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2018. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering* 44, 11 (2018), 1024–1038.
[10] J.R. Koza, A. David, F. H Bennett III, and M. Keane. 1999. *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufman.
[11] Brendan Murphy, Christian Bird, Thomas Zimmermann, Laurie Williams, Nachiappan Nagappan, and Andrew Begel. 2013. Have agile techniques been the silver bullet for software development at microsoft?. In *2013 ACM/IEEE international symposium on empirical software engineering and measurement*. IEEE, 75–84.
[12] J Ross Quinlan. 1983. Learning efficient classification procedures and their application to chess end games. In *Machine learning*. Springer, 463–482.
[13] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*. 2503–2511.
[14] Stephen Wolfram. 1983. Statistical mechanics of cellular automata. *Reviews of modern physics* 55, 3 (1983), 601.