Integrating AI and Machine Learning in Software Engineering Course for High School Students

Sperling, Ahuva
Leo Baeck Education Center
Haifa, Israel
97248300500
asperling@leobaeck.net

Lickerman, Dorit
Leo Baeck Education Center
Haifa, Israel
97248300500
dlickerman@gmail.com

ABSTRACT
This paper describes a unique software engineering curriculum for high-school students that includes subjects in artificial intelligence and machine learning. The students in the course deal with the implementation of solutions to riddles and games (complex algorithmic problems), use the DrRacket functional programming language as a tool that supports their comprehension and thorough understanding of blind search algorithms, informed search algorithms, search games trees and machine learning algorithms. During their studies, the students engage in self-learning, collaborative learning and peer teaching. At the end of the course, each student writes a final research project which integrates the main aspects that have been learned in the course.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education*, Curriculum.

General Terms
Algorithms, Human Factors, Languages.

Keywords
Computer science education, artificial intelligence, project based learning, machine learning, K-12.

1. INTRODUCTION
This paper describes a unique course in software engineering that was developed for high school students in the Leo Baeck Education Center (see Section 2). The course was developed as an advanced course for the functional programming unit of the Israeli high school CS curriculum (described in Sections 3 and 4). Section 5 describes the computer science curriculum in the Leo Baeck high school and Section 6 describes the software engineering course – the major contribution of this paper – which includes subjects in artificial intelligence (AI) and machine learning. In order to make the learning process more interesting and enjoyable, the students implement algorithms which enable them to write applications that solve riddles and play games. Apart from enjoying the learning process, this approach makes the learner active, encourages discussion between students, and stimulates the desire to continue studying and exploring the subjects being taught in class beyond the classroom hours [4]. Appendix A presents a letter that reinforces what has been said above.

ITiCSE'12, July 3–5, 2012, Haifa, Israel.

2. LEO BAECK EDUCATION CENTER
The Leo Baeck education center in Haifa was founded in 1938 for children who had survived the Holocaust. Its pluralistic approach is committed to democracy, egalitarianism and human rights, as well as to the teaching of the living values of progressive Judaism which inspires social change and imroving the world. The education center is one of Israel's finest institutions for academic excellence and it also promotes community outreach and social action.

Today, the Leo Baeck Education Center includes early childhood daycare, a middle school, a high school, an art school, a community center, a sport center, a Reform synagogue and the Lorry I. Lokey international center for Jewish studies.

In addition, the school runs many programs for the community, e.g., the World Youth Award[1], promotion of Jewish values, a district gifted class and a special education class for students with communication and other disabilities (PDD).

The Senior High School, in which we have been teaching during the last 20 years, is Leo Baeck's flagship, serving as one of Israel's premier high-school institutions. One hundred and fifty full and part-time teachers educate over 1,000 students from grades 10 through 12 (ages 16-18). The high school serves as an experimental school to develop leadership skills based on Jewish values. The school aims to educate its students to take personal responsibility for their actions and especially to fulfill their potential and to embark on an unrelenting climb to the summit of social, academic and ethical excellence. In addition, the school requires a personal contribution to the welfare of the community through extensive community service, develops an appreciation for the arts and culture, and instills a love of Israel.

The school encourages its students to innovation, initiative, independent thinking, originality and creativity, which are expressed in the class, in teaching and learning methods in various disciplines, in educational trips and in many additional community projects, where students improve and develop their leadership skills.

[1] This International Award is an exciting self-development program available to all 14 to 25 year olds. The program is intended to help young people and those concerned about their welfare. The program challenges youth, encouraging them toward personal achievement in sport, hobbies and contribution to the community. Young people, who achieve the Award, discover their potential, impact their community and develop a set of life skills.

3. COMPUTER SCIENCE IN ISRAELI HIGH SCHOOL

The high school computer science curriculum, as described in the article *A High-School Program in Computer Science* [5], aims to teach basic computer science concepts and principles of building computer systems. The program emphasizes the principles that stand the test of time [3] compared with technology-dependent concepts which quickly become outdated. The program provides learners with algorithmic reasoning ability, analysis and algorithmic problem solving skills, as well as basic concepts in the design of computerized systems.

The program's fundamental principles are [5, p. 5-7];

- Computer science should be taught on par with other scientific subjects.
- The program concentrates on the key concepts and foundations of the field. The main notion to be emphasized throughout the program is that of an algorithmic problem.
- The program contains required units as well as electives.
- Conceptual and experimental issues are interwoven throughout the program. This two-track approach, which is dubbed the "zipper principle" ([5] page 6), is one of the salient points of the program, and is elaborated on in a sidebar.
- Two different programming paradigms are taught. The students learn a "mother tongue" first, and then, on a more humble scale, they learn another language, of radically different nature, that suggests alternative ways of algorithmic thinking.

3.1 Teaching methods

According to the official website of the Computer Science Teaching Supervision in the Ministry of Education [3], the recommended teaching method is the one in which, in the course of their studies and led by the teacher, students experience independent development of computer programs to solve problems, some of which are set by the teacher and some of which are selected by the student. Emphasis is placed on the development process, product quality and process documentation.

The studies take place mostly in the computer lab. Work in the laboratory allows the demonstration of educational content by using the computer and projection equipment, in addition to the implementation of algorithms which are executed on the computer.

3.2 Target population

The program is designed for high school students (in both regular high school and vocational school[2]).

3.3 Curriculum structure

The curriculum, as written in the article *A High-School Program in Computer Science* [5] comes in two versions: a 3-unit and a 5-unit program. The former consists of 270 hours of study, and the latter, 450. The programs are constructed, as explained below, from the following list of modules:

1. Fundamentals 1 and 2 (2 units; 180 hours): This module provides the foundation for the entire program. It introduces most of the aforementioned central concepts, and at the same time teaches how to apply them in an object oriented programming language[3].

2. Advanced programming (1 unit; 90 hours): This module is based on Fundamentals. It concentrates on data structures, introducing abstract data-types, and also takes a step beyond stand-alone algorithms, to discuss the design of complete systems.

3. Second paradigm (1 unit; 90 hours): This module introduces the students to a second programming paradigm, which is conceptually quite different from the object oriented approach adopted in Fundamentals 1 and 2. Currently, the alternatives are: logic programming, system-level programming, functional programming, web programming, computer graphics and information systems[4].

4. Theoretical computer science choices (1 unit; 90 hours): This module exposes student to selected topics in theoretical computer science. The alternatives are: computational models (mainly automata), object-oriented programming, operations research and computer systems and assembly.

The 5-unit program is based on all the above, while the 3-unit program consists on fundamentals 1 and 2 and a second paradigm.

4. SOFTWARE ENGINEERING CURRICULUM IN ISRAELI HIGH SCHOOLS

The Israeli software engineering course (5 units; 450 hours) is an advance course, beyond the 5 units program described above and in [5]. It trains its graduates for systematic thinking and application capability of algorithms and complex systems in the world of high technology. The subjects taught in the course reflect the development of computer science, programming languages and technological innovations in software engineering, as well as addresses AI and selected topics related to technologies of the new millennium. Classes are held in a computerized environment and the students use the most advanced programming languages and systems. Curricula are based on knowledge and use of mathematics and scientific principles, as well as on planning and programming sophisticated systems.

The program's conceptual approach integrates three types of complexities [3]:

Computational complexity: Search for algorithmic solutions to algorithmic problems or proving non-existing solutions to these problems and complexity analysis of algorithmic problems and search for more effective solutions.

System complexity: Analysis and design of large software systems that combine modularization and reactivity.

Cognitive complexity: Design of intelligent systems, combining different disciplines of research in human and machine behavior.

---

[2] Vocational school is an educational institution in which students derive the benefits of a combination of study and practical work. The professional preparation of the students for the social and professional world is carried out in the ninth and tenth grade. In these years the students learn general and vocational subjects together with practical experience in the school workshops. The eleventh and twelfth grades combine three or four days of study at school with two or three working days, as paid apprentices in the workplace.

[3] The article [5] refers to procedural languages. Languages taught today are object oriented languages.

[4] We updated the curriculum in accordance with the article.

The curriculum has six alternatives:
1. Planning and programming operating systems
2. Planning and programming graphical systems
3. Planning and programming information systems
4. Planning and programming expert systems
5. Planning and programming Web network and service systems
6. Planning and programming mobile systems

5. COMPUTER SCIENCE CURRICULA IN LEO BAECK HIGH SCHOOL

The students in our school are offered with two options to learn computer science:

1. The 5 units program with functional programming (using DrRacket language) as the second paradigm unit and with computational models as the theoretical computer science choice. The students participate in the program for two years in the eleventh and twelfth grades (see Section 3.3).

2. A software engineering course, in which the students learn computer science for three years. They start learning the 5 units in the tenth grade for two years (as described above), and in the twelfth grade, they specialize in AI and machine learning in a unique program that is taught only at our school (described in Section 6).

During the seven years, in which the functional programming was taught as the second paradigm unit, the added value that the unit gives students, both in understanding of recursion and in top down design, became apparent. This understanding was reinforced when graduate students (5 units program) of our school expressed their opinion that their algorithmic problem-solving skills, regardless of the programming language, were considerably improved after they had studied this unit (see students' responses in Appendix B). When the school decided to start a software engineering course of study, it was found suitable to study it with this second paradigm in the same computational workspace. At this time, there was no suitable specialty in software engineering curriculum, and we could not find a similar course for high school students elsewhere; therefore, we had to develop our own curriculum. Prof. Shaul Markovitch[5] from the Technion's Department of Computer Science helped us in its development process, and once it was ready, it was submitted to the Ministry of Education and received the formal approval.

6. THE AI - MACHINE LEARNING CURRICULUM

This section describes the curriculum of the software engineering course that we developed. The course is based on cooperative and interactive learning as part of the pedagogical conception that students should be active in their learning.

Table 1 gives an overview of the curriculum. The 450 hours of the course can be divided into 4 major categories: theoretical basis (170 hours, 6.1-6.6), introduction to machine learning (100 hours, 6.7), advanced topics (60 hours, 6.8-6.9), and a final individual project (6.10).

Table 1. Overview of the curriculum

	Topics	**Hours**
6.1	Introduction to Artificial Intelligence	10 hours
6.2	Extending the theoretical basis	40 hours
6.3	Problem solving by graph search	10 hours
6.4	Uninformed searches	40 hours
6.5	Heuristic searches (Informed searches)	30 hours
6.6	Non-cooperative games	40 hours
6.7	Introduction to Machine Learning	100 hours
6.8	Advanced topics in programming language I	15 hours
6.9	Advanced topics in programming language II	45 hours
6.10	Final personal project	120 hours

In what follows, we elaborate on each topic presented in Table 1.

6.1 Introduction to AI

The purpose of this introduction is to provide a theoretical background as the basis for the studying of the domain. The introduction is based on 6 parts as is shown in Table 2.

The teaching methods vary between classroom presentations and working in discussion groups on questions and answers for the Turing test.

Table 2. Introduction to AI: topics & subtopics

	Topic	Sub topics
1	What is artificial intelligence?	
2	Turing test	
3	Achievements in artificial intelligence	• Spacecraft managed by AI software • Computer program winning chess game Deep Blue • Automatic classification of documents • Medical diagnostics • Robotic assisted surgery • Computational Biology • Adaptive pets
4	Examples of unsolved tasks in artificial intelligence	• Understanding of natural language • Understanding a picture • Robot for housework
5	Intelligent agents	• Hardware agent • Software agent • Intelligent agents' structure (e.g. puzzle solving)
6	Research areas in artificial intelligence	• Multi-agents system with shared /conflicting goals • Computer vision • Understanding of natural language • Robotics • Learning

[5] Prof. Shaul Markovitch's website: http://www.cs.technion.ac.il/people/shaulm/index.html

6.3 Extending the Theoretical Basis

The purpose of this section is to create a theoretical basis for further learning. This part of the course is based on 2 topics as shown in Table 3. The teaching methods are lectures and class discussions.

Table 3. Extending the theoretical basis: topics & subtopics

	Topic	Sub topics
1	Trees	• General trees • Knowledge representation using a tree • Directed trees • Minimum spanning tree
2	Algorithms and graph theory	• Graph characterization and description • Directed graph • Circular graph • Undirected graph • Shortest path algorithms • Knowledge representation using a graph

6.4 Problem Solving by Graph Search

The purpose of this section is to create a mathematical/theoretical basis for further learning, advancement from programming to theoretical/mathematical issues, and pupils' acquaintance with solving complex problems.

This part of the course is based on 3 topics as is shown in Table 4. The teacher formally defines a problem (for example, missionaries and cannibals), followed by students writing a formal definition of the other problems from the list; then, the class discusses the various definitions.

Table 4. Problem Solving by Graphs: topics & subtopics

	Topic	Sub topics
1	Formal definition of mathematical problems	• Missionaries and cannibals. • Puzzle problem 3×3. • Solution to the puzzle: "How to move a horse on all chessboards' slots without visiting a slot twice". • Solution to the puzzle: "How to fill 4 liters of water in a bucket with capacity of 5 liters using a bucket with capacity of 3 liters, without the option of measuring.
2	Formal definition of a complex problem	• Display problem • Set up a group of states in the graph illustrating the problem • Set a transition between two states in the graph depicting the problem
3	Solution of a complex problem by searching in the problem's states graph	• Develop a node • Introduction to search strategies with an emphasis on developing the next node, and "what is stored in the computer memory" as a distinction between the different algorithms • performance evaluation of search algorithms: o Memory Complexity o Runtime complexity o the nature of the solution: o Integrity o Optimal

6.5 Uninformed Searches

This part expands students' theoretical basis in algorithms as shown in Table 5, introduces the concepts of computational complexity of time and place, and emphasizes the finding of an optimal solution to complex tasks by utilizing integration programming in DrRacket computational workspace and theoretical content. The students implement:

- The breadth first algorithm to solve the problem of "missionaries and cannibals";
- The depth first search algorithm with depth limit to solve the problem of "leaping lizards";
- The backtracking search algorithm to solve the problem "How to place eight queens on a chess board".

The problems varied from one year to another.

Table 5. Uninformed Searches: topics & subtopics

	Topic	Sub topics
1	Breadth first search	Each algorithm is evaluated based on its completeness, optimality, time complexity and space complexity.
2	Depth first search	
3	Depth first search with depth limit	
4	Back tracking search	
5	Iterative depth first search	

6.6 Heuristic Searches (Informed Searches)

In this section, the students continue to acquire knowledge in computer science algorithms, as shown in Table 6, and deal with a complex task. They proceed from graph search to systematic search and finally heuristic search, finding the most significant factors leading to an optimal solution of a complex problem. The students implement and compare the solution paths (number of steps from a given state to the target state) of the following algorithms:

- The best first search algorithm to solve the 3×3 puzzle problem, using the evaluation function of Manhattan distance;
- The A* algorithm to solve the 3×3 puzzle problem, using the evaluation function of Manhattan distance.

Table 6. Heuristic Searches: topics and subtopics

	Topic	Sub topics
1	Differences between uninformed search and informed search	
2	Evaluation function	• Rule of thumb • Manhattan distance
3	Hill Climbing Search	Each algorithm is evaluated based on its completeness, optimality, time complexity and space complexity.
4	Best First Search	
5	Uniform cost search	
6	A* Search	

6.7 Non-Cooperative Games

In this part, a discussion on "How the computer can play and decide on its moves" takes place in the class.

The students are exposed to the idea that an opponent in a game chooses the best possible move s/he can make, and which is the worst move for the player in the algorithms presented in Table 7. The students implement the minmax algorithm in a given Tic Tack Toe game for two human players.

The game is prepared by the teacher in DrRacket computational workspace and the students replace one of the human players with a computer player. During implementation of the algorithm the students cope with changes in a complete program:

1. Understand the code written by someone else;

2. Write an alternative code fragment;

3. Make the adjustments necessary to incorporate the piece written instead of an existing code section.

Table 7. Non-cooperative games: topics & subtopics

	Topic	Sub topics
1	Game design as a system of two agents	
2	Tree game definition	
3	Game Tree Search	• Minmax algorithm • Minmax algorithm for games involving random (roll of the dice) • Alpha beta pruning.

6.8 Introduction to Machine Learning

According to Smith, [10], learning is quantitative increase in knowledge, storing information that can be reproduced, acquiring facts, skills, and methods that can be retained and used for the interpreting and understanding of reality in a different way. In addition, learning involves relating parts of the subject matter to each other and to the real world and comprehending the world by reinterpreting knowledge.

In this part, the students are exposed to the idea that percepts should be used not only for acting, but also for improving ability to act in the future [9, p. 649]. The students get knowledge of inductive learning from observations through the algorithms presented in Table 8. They are referred to U.C.I[6]- machine learning repository website to collect data in order to implement the ID3 (decision tree) learning algorithm [8].

Table 8. Introduction to Machine Learning: topics

	Topic	Sub topics
1	What is learning?	
2	Learning classifier	• Learning decision trees - id3 algorithm • Nearest neighbor algorithm • K closest neighbors algorithm
3	L.M.S algorithm	
4	Genetic algorithms	

6.9 Advanced Topics in Programming Language I

In this section, students expand their practical basis in DrRacket programming language as shown in Table 9. Knowledge is acquired with the aid of worksheets that help students become familiar with the learning material through guided inquiry, followed by discussion and summaries of the material. This method, according to Brunner's teaching model [2], places the responsibility for learning on the learner who learns concepts and ideas independently with only limited involvement on the part of the teacher, thus developing an independent learner.

Table 9. Advanced Topics in Programming Language I

	Topic
1	Flat Lists
2	Progressive list processing
3	Higher-order functions map, filter, reduce
4	Recursion depth
5	Deepening the assembly functions

6.10 Advanced Topics in Programming Language II

The content in this part is acquired through peer teaching. As an example of the deliverables expected from the students later, the teacher presents the first subject (structures) and distributes worksheets on this subject. Students are then divided into groups of 2-3 students. Each group studies one subject from the list shown in Table 10, searches for material on the subject and subject-related activities under guidance from the teacher, presents the subject to the other students in the class and prepares the overall work assignments page on the subject being studied.

Learning from examples (given by the teacher) was selected as a teaching method to illustrate abstract principles, thorough analysis of the example, generalization and abstraction of the example's components, and learning from others' experience. Peer teaching and collaborative learning were selected also since it is highly effective: It develops leadership skills, self-learning, team work, ability to present to an audience and develop learning outcomes (work pages). Group work encourages the transfer of knowledge, and develops discussion skills, an ability to listen to others, including preparing students for the real world [1, 7].

Table 10. Advanced Topics in Programming Language II

	Topic
1	Structures
2	Vectors
3	Files
4	Strings
5	Graphic viewport
6	GUI
7	Streams
8	Lambda expressions
9	Threads

6.11 Final Project

The final project is done individually. The work includes:

1. Implementation of at least a search algorithm and a learning algorithm

2. Small scale research that examines the system performance depending on various parameters of the learning algorithm

3. Graphical interface

4. User interface

5. Project portfolio

While working on the final project, the students fully utilize their personal potential, get experience in decision making, get a thorough understanding of issues arising during the writing of the final project and become active learners. The small-scale research requires coping with how to define of a research question, how to

[6] http://archive.ics.uci.edu/ml/

propose a hypothesis to research question and check the validity of the hypothesis. Students write projects on a high level, characterized by originality and creativity. Developing the projects enables the students to develop themselves and learn topics outside of the academic program based on their interests. One reason that explains students' success in writing a final project at such a high level is the search algorithms and learning algorithms that have been written by the students throughout the course as mini projects, which prepared them for dealing with the final project. During the writing process, interesting phenomena occur in the class. For example:

- When a student has encountered a problem, s/he requests the teacher's guidance and, in addition, explains the difficulty to students in the class who try to locate the source of the problem and offer him or her ways to cope with it.
- After a student has overcome the difficulties arising during the writing of the project, there is a sense of "eureka – I found it".
- Students express great interest in each other's projects – even where no particularly difficulties are encountered. Private discussions take place on how to improve visual presentation, what can be added, how to improve performance, and so on.

7. CONCLUSION

To summarize, we are very satisfied with the material we teach in this advanced course. We have succeeded in adapting academic material to the needs of high-school students. Our students have dealt successfully with complicated algorithms that we think are even difficult for undergraduate students. We motivated our students to continue research in artificial intelligence and machine learning contents according to their ability.

Students' interest in the material and their will to develop projects at a higher level contributed to the improvement of the course. We plan to continue updating the contents learned and adjust it to our students.

Finally, we strongly believe that each student participating in this course improves his or her abilities.

8. ACKNOWLEDGMENTS

We would like to acknowledge the valuable contributions made by our advisor, Associate Professor Orit Hazzan. We wish also to thank Dr. Tami Lapidot, for her advice and support. Special thanks to Associate Professor Shaul Markovitch, for helping us recognize the domain and decide on the course content. We would like to thank also the Leo Baeck high school management: the CEO, Mr. Dan Fesler, former head of pedagogical department, Dr. Esti Schleyer, and former high school director, Mrs. Yona Katzir, for allowing us to carry out this unique advanced course, Finally, we would like to thank to Dr. Avi Cohen, Inspector-in-Chief, Computer Science and Information Technology, Ministry of Education, Israel, for giving us the formal permission to teach the course.

9. APPENDICES

9.1 Appendix A

This letter was written by a graduate student of the advanced course in Leo Baeck during his military service in one of the elite units of the intelligence corps. This unit recruits honors students in computer science from all over the country.

Dear Ahuva,
I would like to thank you, once again.

Only now, nearly two years after graduating from high school, I do understand how special the course you taught us was. While I was in high school, I didn't have contact with students from other schools so I didn't know how special the program you taught us was.

I couldn't wait to finish my military service and begin my studies at the Technion, where I had decided to learn computer science with specialization in artificial intelligence and Robotics.

Since graduating from high school, I have read many books on mathematics, statistics and game theory and I thank you again for revealing to me the fascinating area of artificial intelligence. In the near future I plan to rewrite the Checkers game I wrote in high school in Java language in order to improve efficiency.

These days, I am developing a minimax engine with some interesting add-ons, a generic engine that can be incorporated into zero-sum games, and hope to distribute it to a code-sharing site so that it can be developed by programmers from all over the world.

I hope to continue learning these subjects and do research by my own.

9.2 Appendix B

The following statements represent opinions of graduate students about the contribution of functional programming to their skills.

- I was exposed to a new way of thinking that was a little difficult at first. Over time I understood better this way of thinking. I enjoyed very much learning with the new programming language and I developed my ability to think.
- Project - I did it!!! I have learned to deal with things and not give up.
- I felt that I gained more from functional programming. ... I have developed my functional and logical thinking.
- I love this programming style because it makes me think, inquire, and improve my abstract logical thinking.

10. REFERENCES

[1] Brown, J. and Dobbie, G., (1999). Supporting and evaluating team dynamics in group projects. In the proceedings of the thirtieth *SIGCSE technical symposium on computer science education* ACM Press.

[2] Bruner, J. S., (1966). *Toward A Theory of Instruction*, W.W. Novton & company, NY.

[3] Computer Science and Information Technology Web site, Curricula (in Hebrew), http://www.csit.org.il/default.aspx?MenuShow=DOCS&Doc_ID=55

[4] David L. Parker (2003) Machine "learning" at all course levels, *Journal of Computing Sciences in Colleges* **18**(3).

[5] Gal-Ezer, J., Beeri, C., Harel, D. and Yehudai, A. (1995). A high-school program in Computer Science, *Computer* **28**(10), pp. 73-80.

[6] Leo Baeck Education Center Web site http://www.leobaeck.org.il/

[7] Koppelman, H., van Dijk, E.M.A.G., van der Mast, C.P.A.G., and van der Veer, G.C. (2000). Team projects in distance education: A case in HCI design. Proc. *ITiCSE 2000*. 97-100.

[8] Mitchel, Tom (1997). Machine Learning. McGraw Hill

[9] Russell, S. and Norvig, P. (1995). *Artificial Intelligence A Modern Approach.* Prentice Hall.

[10] Smith, M. K. (2003) 'Learning theory', the encyclopedia of informal education, www.infed.org/biblio/b-learn.htm