






Enhancing Object-Oriented Programming Pedagogy with an Adaptive Intelligent Tutoring System

Mthembe Dlamini  and Wai Sze Leung  

Academy of Computer Science and Software Engineering,
University of Johannesburg, Johannesburg, South Africa
methembedlamini@yahoo.com, wsleung@uj.ac.za

Abstract. Challenges to teaching programming include a lack of structured teaching methodologies that are tailored for programming subjects while the benefits of providing programming students with individual attention are not easily addressed due to high student-to-teacher ratios. This paper describes how adaptive intelligent tutoring systems may represent a potential solution assisting teachers in delivering individualized attention to their students while also helping them to discover effective ways of teaching a core programming concept such as object-oriented programming. This paper investigates how adaptability in traditional intelligent tutoring systems are achieved, presenting an adaptive pedagogical model that uses machine learning techniques to discover effective teaching strategies suitable for a particular student. The results of a prototype of the proposed model demonstrate the model's ability to classify the student models according to their learning style correctly. The knowledge obtained can be applied by educators to make better-informed choices in the formulation of lesson plans that are more appropriate to their students.

Keywords: Intelligent tutoring systems
Pedagogical decision-making · Adaptability · Artificial intelligence
Machine learning

1 Introduction

Aside from providing students with individualized attention, the deployment of Intelligent Tutoring Systems (ITSs) may also serve as a source of information that helps guide teachers in making better-informed pedagogical decisions.

Much merit exists in introducing programming at lower school levels, as world development is seen to rely on technology [32] and critical thinking skills relevant to surviving the information age are seen to be developed from learning to code [15]. Calls for programming to be taught at early academic stages are increasingly gaining traction [32], however, there are equally voices raising legitimate, realistic concerns: are teachers adequately equipped to *teach* coding [26]?

Unfortunately, the reality is that teachers struggle to come up with effective ways to teach programming [12, 15, 19, 24]. In some cases, teachers are simply not trained to be computer science teachers [26]. This paper, however, focuses on other factors, specifically limited time, limited resources, high student-to-teacher ratios, and student diversity [1, 12, 19, 24], all of which make it difficult for a teacher to devise lessons that cater for particular learning needs [11]. There is thus a need for tools that can assist teachers in monitoring individual students, gathering useful information to improve pedagogical decision-making.

ITSs have been used effectively to offer individual attention to students [8, 21, 23, 31]. Existing implementations, however, make use of predetermined rules to tailor content, thus lacking the ability to adapt and discover new knowledge about teaching. We propose introducing changes to the traditional ITS model, enabling it to autonomously discover effective teaching strategies and student preferences so that the knowledge generated from such ITSs may be used by human teachers to improve their teaching techniques and make informed lesson planning decisions. This paper presents an Adaptive Pedagogical Model (APM) that is capable of improving teaching strategies by means of machine learning in order to assess the learning preferences for different kinds of students.

The remainder of this paper is organized as follows: Sect. 2 reviews several research areas, specifically on traits that determine academic performance, teaching strategies, intelligent tutoring systems, and machine learning. Section 3 describes the proposed APM, the adaptation approach and its generic adaptation algorithm. Section 2.2 presents the details of the prototype that was implemented to test the APM, leading to an evaluation of the APM in Sect. 5. Finally, the paper concludes in Sect. 6 with a summary of the findings.

2 Related Work

2.1 Need for Adaptive Tutoring

Proponents of student-centered education theory have long highlighted the need for individualized learning mechanisms that allow a student to select their learning path [7, 22, 34]. The proliferation of eLearning platforms has opened up numerous opportunities to realize student-oriented learning where teachers merely guide and empower their students to take charge of their learning process [22].

The rationale for providing students with individualized learning paths is quite straight-forward: students are diverse—they have different backgrounds, have different learning objectives, and possess different learning styles [3, 7].

Given that research has demonstrated a very strong correlation between academic performance and personality traits [9, 27] and that students tend to respond better when offered tutorials that are dynamic, intelligent, and catering to their individual attention [1], the aforementioned individualized characteristics are clear indicators that the ‘one-size fits all’ approach can no longer be considered appropriate. Medical education, for example, believes that students should be allowed to pursue individualized learning while meeting standardized

outcomes as this would assist in developing the student's ability to self-regulate their own learning, something considered crucial to staying current in an ever-changing field [22]. Since programming can be closely linked to the rapid pace at which technology evolves, it would stand to reason that cultivating a practice of remaining abreast of developments in the discipline would be quite beneficial.

Personality Models. Over the years, several personality models have been proposed to address the notion that individuals with differing characteristics are seen to learn and digest information differently [9, 30]. Because personality traits are used to make pedagogical decisions [16], a system or human tasked with selecting appropriate teaching strategies to deliver content may benefit from having a better understanding of these personality models.

One of the most adopted models is the Five Factor Model of Personality Traits (FFM). The FFM is a framework made up of the dimensions of Agreeableness (likability and friendliness), Conscientiousness (ability to be dependable and always have the zeal to achieve), Emotional Stability, Extraversion (level of socialization with other students and activity), and Openness (imaginativeness, broadmindedness, and artistic sensibility) [29]. For the purpose of this paper, we will be considering the FFM to model our students' behaviors.

Teaching Strategies. Teaching strategies are methods of presenting content to students and are used to personalize learning experiences based on the students' preferred learning style [1]. While students have different learning preferences, it is difficult to discover factors that affect each student's preferred learning style [17]. Among teaching methods, the most common assumption is that students learn better if the instruction is provided in a format that matches the preferences of the student (e.g., for a 'visual learner', the appropriate strategy would be to emphasize on the visual presentation of information) [17].

As with personality models, several theories on learning styles exist, the most common of these following the theory of multiple intelligences [6], and the Visual, Auditory, Read, and Kinesthetic (VARK) Model [17]. These models classify students using different measures and designations, with some classifying the student according to modalities of learning and perceptual styles while others refer to cognitive style, personality type, and aptitudes [5].

Controversy. Although research reveals the existence of learning styles, others argue that this hypothesis has not been adequately and properly tested, citing the lack of sufficient scientific reports to support their rigor [5]. Despite this reservation, researchers argue that there is value in identifying appropriate teaching strategies rather than treating all students the same way [1, 6].

As such, it would be beneficial to prepare content in different formats so that they can be presented to accommodate students with their various learning styles. The challenge then lies in ensuring that the most effective teaching strategy can be correctly identified. For ITSs, there will need to be some way in

which personality traits can be correctly classified, and an appropriate teaching strategy selected. The next section will look at the components that make up an ITS to establish how the aforementioned tasks can be achieved.

2.2 Intelligent Tutoring Systems

Intelligent Tutoring Systems (ITS) are computerized learning systems that have the ability to personalize the learning experience of students [21]. Figure 1 depicts the traditional ITS model which comprises four components, namely: user interface, pedagogical model, student model, and knowledge base [21]. The arrows in Fig. 1 represent communication flow between the different components. The ITS components work together during the tutoring process as follows:

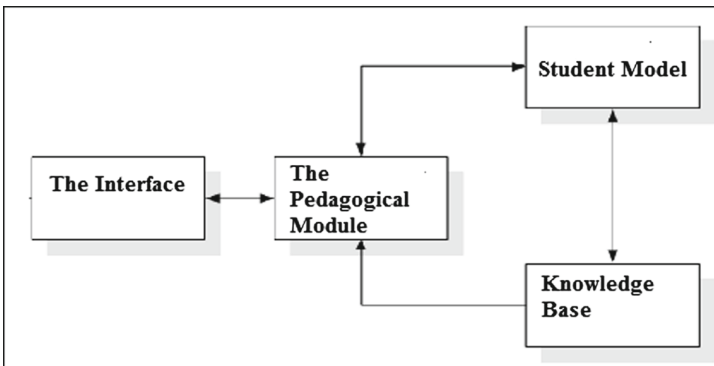


Fig. 1. Components of an ITS according to [10]

1. **User Interface:** The user interface is the main point of communication between the machine and the human user (in this case, the student). Interfaces may vary depending on the ITS implementation although most recent ITSs are conversational in nature and have dialog-based interfaces [25].
2. **Pedagogical Model (PM):** The PM is the reasoning component of the system, forming the decision-making component of the software. It is encoded with a rule interpreter to process and interpret rules [21]. Attempts to improve adaptability within an ITS will require that designers and developers focus on improving the decision-making strategies within this component.
3. **Knowledge Base (KB):** The KB contains the curriculum content and facts necessary for understanding, formulating and for solving problems [21].
4. **Student Model (SM):** The SM represents the student's emerging knowledge and skills. Information such as learning preferences, past learning experiences, and advancement may also be stored to help aid adaptability during the teaching process [21].

ITS implementations vary according to the relative level of intelligence of the components. For example, projects focusing on intelligence in the domain module may generate complex and novel problems while those with an emphasis on teaching strategies may concentrate on intelligence in student models, attempting to identify student characteristics, learning curves, and learning styles among other information.

Adaptability in ITSs generally occurs by tailoring feedback for each individual student to improve their learning experience [14]. This experience can be further enhanced by complementing the personalized feedback with adaptive pedagogical strategies. Such strategies will not only help students but will also ensure that knowledge accumulated by ITSs is accessible to educators.

The ability to achieve autonomy and adaptive pedagogical strategies is often accomplished through the use of Machine Learning (ML) techniques. Given the extensive possibilities and alternatives when it comes to classifying the personalities of students, coupled with identifying an appropriate teaching strategy, it would be challenging, if not possible, to ensure that all outcomes are correctly implemented and catered for. ML thus plays a significant role in the construction of an adaptive ITS that will discover the necessary knowledge that would otherwise take a human a long time to figure out [18]. The following sub-section investigates ways how ML can be applied to achieve adaptive pedagogy in ITSs.

2.3 Machine Learning for Intelligent Tutoring Systems

In essence, ML eliminates the need for explicit domain modeling [2]. Instead, the tasks of engineering the knowledge are automated as ML enables the ITS to learn autonomously from educational datasets. Examples of such systems include AutoTutor and ActiveMath [33].

Although existing ITS implementations have used ML for student modeling, content sequencing, and tailoring feedback [4], little work has been done to use ML to improve pedagogical decision-making. As ITSs operate, they accumulate data about different students and their learning preferences. The data accumulated by the ITS may be used to update the decision-making strategies of the ITS continuously. The ability to continuously train ITSs may be achieved through the use of ML techniques capable of learning over time, these techniques are known as incremental machine learning (IML) [20].

Common IML techniques include Naïve Bayes Classifier (NBC), K-Nearest Neighbor (KNN), and Incremental Support Vector Machines (ISVM). A study comparing these IML techniques for implementation in ITSs revealed that ISVMs perform excellently but at the cost of requiring a large amount of training data. KNN had a tendency to introduce bias which in its selection due to the algorithm's nature of always selecting the most frequent class. Ultimately, NBC was not only generally simpler to implement, but also outperformed its peers. For these reasons, NBC was considered for the implementation of our adaptive ITS.

3 An Adaptive Pedagogical Model for ITSs

This paper proposes an adaptive pedagogical model which uses past tutoring experiences to adjust future pedagogical decision-making strategies. Figure 2 shows the architecture of an ITS that incorporates the proposed model. The proposed model is based on the traditional ITS architecture described in Fig. 1. All basic four components (User interface, Student model, and Knowledge Base are included, however, to make the ITS adaptive some components were added. The proposed model adds a central component to facilitate tutorial dialogs and the expert model for evaluation of student performance during and after tutorial sessions. Feedback from the expert model is used as learning input by the APM. The APM itself is divided into components which are explained in sub-sections that follow below.

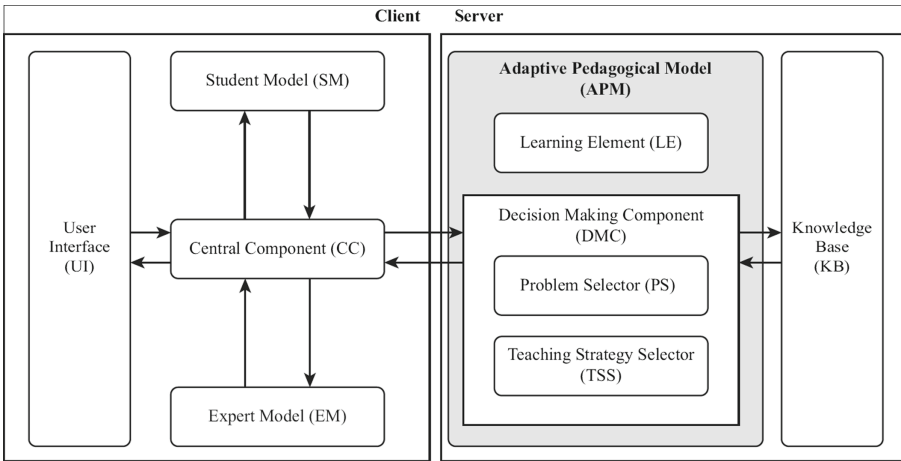


Fig. 2. The adaptive pedagogical model

The APM follows a horizontal approach rather than a longitudinal approach to achieve adaptability. This means that instead of tracking a single student’s interaction history, the model records and utilizes the tutorial history of all students using the ITS.

All ITS instances (hosted separately on client devices) interact with a central server. The model uses a machine learning algorithm to parse and discover patterns in the information that is obtained. These observed patterns then become future decision-making policies for the ITS. As the ITS continues to offer tutorial sessions, each upcoming interaction output is then used to update the decision-making policies continuously.

The APM does not act independently but also depends on other ITS components such as the Knowledge Base and Student Model. In Figure 2 the arrows illustrate how information flows from component to computer during tutorial

sessions. In the following sub-sections, the components that make up the APM, namely the Problem Selector, Teaching Strategy Selector, and Learning Element, are discussed.

3.1 Problem Selector (PS)

The PS is responsible for deciding on a problem to present to the ITS's student for solving. Ideally, the problem selected is one that is perceived to be neither too challenging to frustrate, nor too simple to bore the student. This decision is based on the Student Model, Problems, and a Selection Policy. Information from the Student Model includes personality traits and information relating to the student's current progress.

As information from tutorial interactions with students continue to flow in, a machine learning algorithm produces a selection policy and continuously improves upon it. The algorithm described below shows the steps and decisions made during the selection of the optimal problem:

Inputs. An array of features extracted from the student model which include:

- Current topic.
- List of questions that the student has already solved successfully.
- Personality dimensions: may assume values 'low'/'medium'/'high'.

Processing. Problem selection involves the following steps:

```
ACCEPT <- list of unsolved problems for the current topic.

FOR EACH problem in problems DO
  Compute pass probability(pPass) of given student model;
  Compute fail probability(pFail) of given student model;
  Calculate the difference between pPass and pFail;
END FOR

RETURN <- Optimal problem (one with the minimum difference
                between the probability of a pass and fail.
```

Output. The optimal problem to be tackled (the problem where the probability of a failure is almost the same as the probability of a pass, hence the problem is deemed as neither too challenging nor too simple).

After problem selection, the next task is to choose the optimal teaching strategy to help the student come to a solution. This process is handled by the Teaching Strategy Selector.

3.2 Teaching Strategy Selector (TSS)

The TSS determines the best teaching strategy for the problem based on the Student Model, selected problem, and selection policy. The goal of the TSS is to identify the appropriate learning style that is proven to be effective in improving the learning gains of the student based on their learning style and the current content that is being delivered. To accomplish this, the TSS makes use of Naïve Bayes to discover patterns in order to predict outcomes of given strategies prior to the delivery of the tutorial.

In this paper we combine the theory of multiple intelligences and VARK to derive teaching strategies. Choosing from these strategies depends on various factors that include the student's cognitive style, personality type, and aptitude. Information on these factors is obtained from the student model for the strategy selector to establish the appropriate strategy.

Inputs. An array of features from the student model include:

- Selected problem (output of the problem selector).
- Personality dimensions: may assume values 'low'/'medium'/'high'.

Processing. Teaching strategy selection involves the following steps:

```
ACCEPT <- list of teaching strategies applicable to the problem(P)
           at hand.
FOR EACH strategy in strategies DO
  Compute pass probability(pPass) of the strategy for P;
  Compute fail probability(pFail) of the strategy for P;
  IF pPass >= pFail THEN
    Calculate difference between pPass and pFail;
    Add strategy to candidate list (item with pass-likelihood);
  ELSE
    Ignore the strategy;
  END IF
END FOR

RETURN <- Optimal teaching strategy (one with the greatest
           difference between pPass and pFail) from candidate list.
```

Output. The optimal teaching strategy for the problem that was selected by the problem selector. Options of the output are either: visual strategy, auditory strategy, kinesthetic strategy, or read and write strategy.

3.3 Learning Element (LE)

The LE basically receives, extract values that are to be used by the Naïve Bayes algorithm during the process of problem and teaching strategy selection

described above. Information is received after every tutorial session and the LE extracts the student's personality traits, the student's performance on the problem that was tackled, and the teaching strategy used to deliver content.

To achieve incremental machine learning through Naïve Bayes, the LE updates a frequency table that contains all statistical figures obtained from the LE overtime. During each update values are not converted to probabilities, conversion is done only when a decision has to be made (i.e., during problem and teaching strategy selection). Using such a strategy ensures that the Naïve Bayes always uses updated values as input and thus is adaptive.

In summary, the LE updates values that are used as inputs to the Naïve Bayes algorithms used by the PS and TSS, these updates are done after every tutorial session in order to achieve incremental learning.

4 A Conversational Adaptive Intelligent Tutoring System

To evaluate our proposed adaptive pedagogical model, we developed a prototype mobile conversational intelligent tutoring system (CITS) designed to introduce the programming concept of object-oriented programming (OOP) to Computer Science students.¹

4.1 Architecture

The ITS is implemented on a client-server architecture. Students are thus able to interact with the system via different client applications that are all connected to a single central server.

The two main functionalities of the server are to store knowledge (curriculum content) and to record previous tutorial experiences which can then be used internally by the ITS to self-improve, and externally by human teachers to derive pertinent student information as observed by the ITS.

4.2 Client Application

The client application serves as the interface between the user (student) and the ITS. The client has an internal database (SQLite) which is used to maintain a Student Model and also store content that is currently being tutored for easy retrieval.

The client application works in tandem with a server maintaining the central database which handles the curriculum information, pedagogical strategies, and previous tutorial data.

Tutorials are presented in dialog format. The ITS initiates a conversation by presenting a problem and engaging the student in a conversation that helps the student to construct a solution. The dialog is also used as a diagnostic tool to measure the student's understanding of a particular concept. The conversation is driven by a six-step cycle which is discussed in the subsection below.

¹ The software is available for researchers upon e-mail request: wsleung@uj.ac.za.

As shown in Fig. 3, the main tutorial screen has a dialog section and an input bar. The dialog section displays the chat history as speech bubbles which show messages passed between student and ITS.

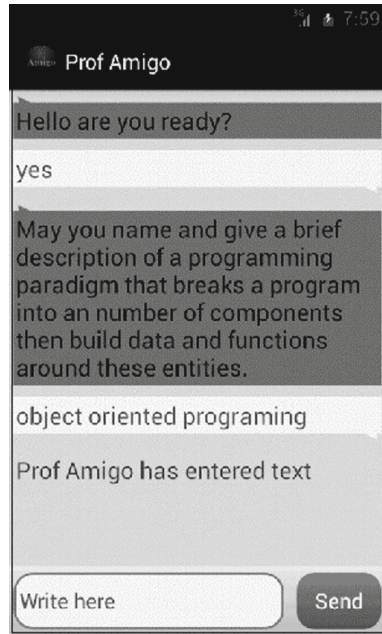


Fig. 3. Screenshot of tutorial conversation between student and ITS client

Since the conversation is in natural language, cosine similarity [28] is used to evaluate the student's inputs. The student's input text is represented as a vector which is compared with all possible responses known by the ITS. The text most similar to the student's input is then considered, ideally if the cosine of two given vectors is close to one then the two texts are considered similar while a value closer to zero denotes different texts.

As an example, the ITS contains a set of expected correct responses and wrong responses for each question. During a conversation, the student's response is compared with all expected responses and the response that is most similar to the student's input is considered. The subsection that follows discusses the speech constructs use to guide the conversation.

The Six-Step Tutorial Cycle. The ITS makes use of six speech constructs adopted from [13] to guide the conversation. The length of the conversation is dependent on the student's performance.

1. The Main Question is output by a Naïve Bayes algorithm and a decision tree. An optimal problem is selected based on previous tutorial experiences.

2. **A Hint** (short question) is presented to highlight a missing concept, given that a student's response is not satisfactory.
3. **A Pump** (question aimed at drawing more information from the student) is presented. For example: "*Do you want to add to your response?*"
4. **Prompts** (leading questions) are presented to the student to help to build an answer.
5. **Assertions** (detailed solutions) are given to the student after all conversation turns have revealed that the student is failing to solve the given problem.
6. **A Summary** (recapitulation of the problem and solution) might be presented by the ITS. In some cases the student provides the summary which is then used to evaluate the student's understanding.

The speech constructs of above form *conversation cycles* whereby each cycle begins with either the ITS's question or a question posed by the student. During the conversation, pumps are presented when there are missing concepts in the student's response. If the student fails to cover all essential concepts, an interior cycle of pumps, hints, prompts, and assertions is formed until the problem is fully solved before a summary is given.

After the ITS has given a summary of the tutorial, the student will be asked if there are any remaining questions about the problem that was just addressed. The cycle then starts again until the problem is solved. In cases where the ITS does not have a solution, an apology is displayed.

The APM uses a student's characteristics to make a prediction of performance to choose the optimal problem and teaching style to suit the characteristic set. These characteristics form part of the student model.

Student Model (SM). Student's characteristics that affect academic performance change as the student learns. In particular, factors that influence change include the frequency of interactions with the ITS, and the mood of the student.

SM variables include personality traits as derived from the FFM. The SM also keeps track of the student's progress, alongside the outcomes of each tutorial. Tracking progress eliminates the possibility of repeating problems. For decision making, the APM depends partly on information obtained from the SM. The SM is dynamically updated, based on feedback from the APM.

4.3 Central Server (CS)

The CS contains the APM and the Knowledge Base (KB). The KB contains the curriculum, teaching strategies, and historical information from previous tutorials. This information is used by the APM to make pedagogical decisions.

In the Knowledge Base, content is organized into topics. Each topic is made up of questions (problems) of varying levels of difficulty. Each question has one or more solutions and is made up of the concepts that the student needs to cover when answering the questions. For each question, there are hints, prompts, pumps, assertions and a summary. After each tutorial session, the student model

and results of the interaction are saved in the KB. These values are used to compute frequency tables used by a Naïve Bayes Classifier.

As indicated previously, the information generated by the APM is meant for both internal (the APM itself) and external use. Potentially, teachers seeking to improve their teaching strategies and lesson plans could very well take advantage of the findings established by the APM in their own pedagogical decision-making.

4.4 Pedagogical Analytics for Teachers

Information generated by the ITS during operation may be analyzed by human teachers to improve their pedagogical decision-making. Examples in which the ITS's acquired knowledge can assist are discussed below. Table 1 shows a sample display of information on the percentage of success in using different teaching strategies to teach selected topics. Such information may help human teachers in selecting the most effective teaching strategies to deliver particular content.

Table 1. Sample Data: different teaching strategies to teach different problems

Topic	1st Strat.	2nd Strat.	3rd Strat.
Top. 1	60%	10%	30%
Top. 2	25%	75%	0%
Top. 3	98%	1%	1%
Top. 4	5%	45%	50%

Table 2. Information about students using the ITS

Student	Problems tried per day	Passed	Failed
Stud. 1	20	17	3
Stud. 2	5	0	5
Stud. 3	50	10	40

For example, as shown in Table 1, the first teaching strategy yields better results when used to teach Topic 3, whereas poor performance is obtained when the same strategy is used on Topic 4. Thus, a teacher can easily make an informed choice on which strategy to use when presenting these topics.

Another example of useful information which may be used by human teachers is shown in Table 2: it summarizes the performance of each student using the ITS. This information gives the teacher insights on their students, thus identifying potential at-risk students that require intervention. In our example, the information shows that Student 2 only attempted five problems for the day and failed all attempts. The teacher could seek out the student in question in an attempt to provide them with more personal (and human) attention.

5 Evaluation of the Adaptive Pedagogical Model

The ability to make appropriate pedagogical decisions is at the center of the Adaptive Pedagogical Model. Appropriate in this context is measured by the model’s ability to recognize a teaching strategy that yields high academic performance when used to tutor a given problem to a specified student model.

Simulated student models were used during the testing phase where the preferences of each student model group (similar characteristics) were predefined. During simulation, the adaptive pedagogical model’s performance is measured by its ability to identify a student model and match it with the appropriate problem and teaching strategy. Analyses from two different perspectives follow.

5.1 Pedagogical Decision-Making Patterns

Instances in which the APM was able to match a student model with the appropriate problem and corresponding teaching strategy are expressed as a percentage of all APM decisions made at selected points in time.

The results reveal inconsistencies in decision-making initially, as shown by the sharp edges at the beginning of the graph in Fig. 4. However, as time progresses, the graph is smoothing. This trend shows that the APM is learning over time and is thus able to make consistent decisions as shown by the gradual smooth increase in elevation after time 56.

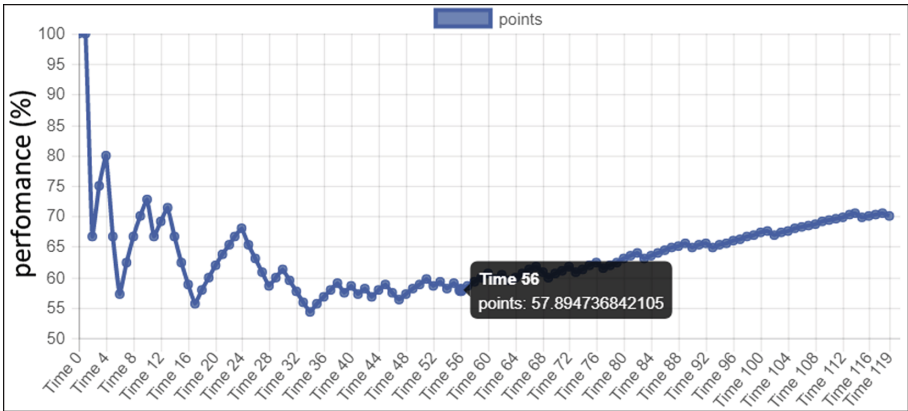


Fig. 4. Representation of past tutorial records based on prediction success statistics

Inconsistency in initial stages may be explained as due to the initial false data that the model was given to start the learning process. When exposed to the actual learning environment the model starts learning and pedagogical decisions made become consistent.

5.2 Teaching Strategy Selection Patterns

Monitoring the decisions made by the APM on choosing a strategy for a given problem for similar student models reveals that the APM is able to identify a strategy that performs well with each distinct group of student model.

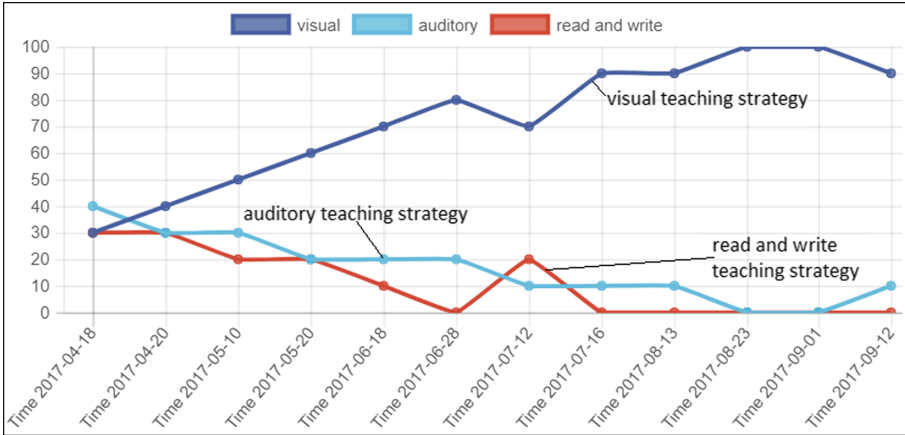


Fig. 5. Graphical representation of teaching strategy selection statistics over time

On identifying an optimal strategy, the APM continues to use the same strategy on students with similar characteristics. This trend can be seen in Fig. 5. As soon as the APM learns the best strategy, the graph of that strategy continues to rise. This rise is due to increases in the selection frequency of that strategy as compared to other strategies.

The two results reveal that patterns are emerging from the pedagogical decision-making data gathered. The results did reveal inconsistencies in decision-making initially. However, as time progressed, the graph smoothed, showing that the APM has managed to learn over time. On the other hand, focusing on the teaching strategy selection choices reveals that the APM can identify a strategy that performs well on a given student model as the APM then continues to use the same strategy on students with similar characteristics.

6 Conclusions

This paper presented the APM which is capable of identifying and adapting its teaching strategies to best suit their student. As proof of concept, an implementation of the APM in the form of a prototype conversation ITS that focused on teaching object-oriented programming principles was deployed for testing. By making use of the Naïve Bayes algorithm, the prototype successfully demonstrated its ability to classify the learning style model of the ‘students’ it interacted with, coming up with teaching strategies deemed appropriate for the identified learning style.

Such a system can be a useful tool in the programming classroom in the following ways: first, it can serve as a personalized tutoring companion to students, offering them one-on-one revision. Second, through its interactions with students, the ITS can provide human teachers with the necessary information regarding the learning styles of their students, enabling the teachers to make better-informed pedagogical decisions. Lastly, less-experienced programming teachers may also learn from the ITS's successes and failures.

References

1. Baine, D., Mwamwenda, T.: Education in southern Africa: current conditions and future directions. *Int. Rev. Educ.* **40**(2), 113–134 (1994)
2. Beck, J., Woolf, B.P., Beal, C.R.: ADVISOR: a machine learning architecture for intelligent tutor construction. In: Joint Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence, pp. 552–557 (2000)
3. Caputi, V., Garrido, A.: Student-oriented planning of e-learning contents for Moodle. *J. Netw. Comput. Appl.* **53**, 115–127 (2015)
4. Chrysafiadi, K., Virvou, M.: Student modeling approaches: a literature review for the last decade. *Expert Syst. Appl.* **40**(11), 4715–4729 (2013)
5. Cuevas, J.: Is learning styles-based instruction effective? a comprehensive analysis of recent research on learning styles. *Theor. Res. Educ.* **13**(3), 308–333 (2015)
6. Davis, K., Christodoulou, J., Seider, S., Gardner, H.: The theory of multiple intelligences. In: *Cambridge Handbook of Intelligence*, pp. 485–503 (2011)
7. Dorça, F.A., Lima, L.V., Fernandes, M.A., Lopes, C.R.: comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: an experimental analysis. *Expert Syst. Appl.* **40**(6), 2092–2101 (2013)
8. Evens, M.W., et al.: CIRCSIM-Tutor: an intelligent tutoring system using natural language dialogue. In: *Proceedings of 12th Midwest AI and Cognition Science Conference*, pp. 16–23 (2001)
9. Felder, R.M., Silverman, L.K.: Learning and teaching styles in engineering education. *Eng. Educ.* **78**(7), 674–681 (1988)
10. Freedman, R.: What is an intelligent tutoring system? *Intelligence* **11**(3), 15–16 (2000)
11. Ghadirli, H.M., Rastgarpour, M.: A web-based adaptive and intelligent tutor by expert systems. In: Meghanathan, N., Nagamalai, D., Chaki, N. (eds.) *Advances in Computing and Information Technology. Advances in Intelligent Systems and Computing*, vol. 117, pp. 87–95. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-31552-7_10
12. Gomes, A., Mendes, A.J.: Learning to program – difficulties and solutions. In: *ICEE 2007 Proceedings of the International Conference on Engineering Education*, pp. 283–287 (2007)
13. Graesser, A.C.: Conversations with autotutor help students learn. *Int. J. Artif. Intell. Educ.* **26**(1), 124–132 (2016)
14. Gross, S., Mokbel, B., Hammer, B., Pinkwart, N.: Learning feedback in intelligent tutoring systems. *Künstliche Intelligenz* **29**(4), 413–418 (2015)

15. Kalelioğlu, F., Gülbahar, Y.: The effects of teaching programming via scratch on problem solving skills: a discussion from learners' perspective. *Inf. Educ.* **13**(1), 33–50 (2014)
16. Kim, J., Lee, A., Ryu, H.: Personality and its effects on learning performance: design guidelines for an adaptive e-learning system based on a user model. *Int. J. Ind. Ergonomics* **43**(5), 450–461 (2013)
17. Klement, M.: How do my students study? an analysis of students' of educational disciplines favorite learning styles according to VARK classification. *Procedia Soc. Behav. Sci.* **132**, 384–390 (2014)
18. Knight, W.: AI's language problem (2016). <https://tinyurl.com/y7r9haju>
19. Koorse, M., Cilliers, C., Calitz, A.: Programming assistance tools to support the learning of IT programming in South African secondary schools. *Comput. Educ.* **82**, 162–178 (2015)
20. Kulkarni, P., Ade, R.: Prediction of student's performance based on incremental learning. *Int. J. Comput. Appl.* **99**(14), 10–16 (2014)
21. Latham, A.M., Crockett, K.A., McLean, D.A., Edmonds, B., O'Shea, K.: Oscar: An intelligent conversational agent tutor to estimate learning styles. In: *FUZZ 2010 Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 1–8 (2010)
22. Lockspeiser, T.M., Kaul, P.: Using individualized learning plans to facilitate learner-centered teaching. *J. Pediatr. Adolesc. Gynecol.* **29**(3), 214–217 (2016)
23. Melis, E., Siekmann, J.: ACTIVE MATH: an intelligent tutoring system for mathematics. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) *ICAISC 2004. LNCS (LNAI)*, vol. 3070, pp. 91–101. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24844-6_12
24. Milne, I., Rowe, G.: Difficulties in learning and teaching programming – views of students and tutors. *Educ. Inf. Technol.* **7**(1), 55–66 (2002)
25. Padayachee, I.: Intelligent tutoring systems: architecture and characteristics. In: *SACLA 2002 Proceedings of 32nd Annual Conference of the Southern African Computer Lecturers' Association* (2002)
26. Partovi, H.: Should Computer Science Be A Mandatory Class In U.S. High Schools? (2017). <https://tinyurl.com/yddfe2n7>
27. Pashler, H., McDaniel, M., Rohrer, D., Bjork, R.: Learning styles: concepts and evidence. *Psychol. Sci. Public Interest* **9**(3), 106–119 (2008)
28. Perone, C.S.: Machine Learning: Cosine Similarity for Vector Space Models (Part III). Technical report (2013). <http://blog.christianperone.com/2013/09/>
29. Poropat, A.E.: A meta-analysis of the five-factor model of personality and academic performance. *Psychol. Bull.* **135**(2), 322–338 (2009)
30. Saucier, G., Goldberg, L.R.: The language of personality: lexical perspectives on the five-factor model. In: *The Five-Factor Model of Personality: Theoretical Perspectives*, pp. 21–50 (1996)
31. Schulze, K.G., Shelby, R.N., Treacy, D.J., Wintersgill, M.C., VanLehn, K.: Andes: an active learning, intelligent tutoring system for newtonian physics. *Themes Educ.* **1**(2), 115–136 (2000)
32. Sterling, L.: An education for the 21st century means teaching coding in schools (2015). <https://tinyurl.com/ybuqoh56>
33. Susarla, S.C., Adcock, A.B., van Eck, R.N., Moreno, K.N., Graesser, A.: Development and evaluation of a lesson authoring tool for AutoTutor. In: *AIED 2003 Supplemental Proceedings*, pp. 378–387, Sydney (2003)
34. Wan, S., Niu, Z.: A learner-oriented learning recommendation approach based on mixed concept mapping and immune algorithm. *Knowl.-Based Syst.* **103**(3), 28–40 (2016)