Education and Information Technologies (2022) 27:6197–6209
https://doi.org/10.1007/s10639-021-10882-9

# Designing an intelligent tutoring system for computer programing in the Pacific

**Priynka Sharma**[1] · **Mayuri Harkishan**[1]

Received: 29 September 2021 / Accepted: 22 December 2021 / Published online: 5 January 2022


## Abstract

Intelligent Tutoring Systems (ITSs) are educational systems that reflect knowledge using artificial intelligence implements. In this paper, we give an outline of the Programming-Tutor architectural design with the core implements on user interaction. This pilot proposal is for designing a model domain of a subset in the computer programming language. The completed project would be adequate to show the idea of a completely developed computing Intelligent Tutoring System in online programming courses to offer benefits to students in the Pacific. This proposed concept would also provide students with an immersive learning experience in an online course to assist in a formative assessment to enhance student learning. A smart tutoring system can provide prompt input of high quality which not only conveys to students about the consistency of the solution but also provides them with information on the precision of the key concerning their existing solutions expertise. This Intelligent Tutoring System (ITS) is proposed to be designed using intelligent algorithms such as optimized ant colony to be able to support the online tutoring system that can initiate the complex learning principles in computing science courses. It is also hypothesized that, based on the performance of other Intelligent Tutoring Systems, students would be able to learn to program more easily in regional campuses and acquire experiences more rapidly and efficiently than students who are taught using conventional methods in an online mode.

**Keywords** Intelligent tutoring system · Education · Programming tutors · Software design · E-learning · Artificial intelligence

✉ Priynka Sharma
priynka.sharma@usp.ac.fj

Mayuri Harkishan
mayuri.harkishan@usp.ac.fj

[1] School of Information Technology, Engineering, Mathematics and Physics (STEMP), The University of the South Pacific Suva, Laucala Campus, Suva, Fiji

## 1 Introduction

The Pacific higher education institutions, are still confronted with the conventional mode of education (formal education) and are currently the most prevalent. Educators play an active role in this educational model, imparting their skills to students, who obtain information passively (Akyuz, 2020). With today's rapid technological advancements, computer learning is rapidly being combined with artificial intelligence techniques to create more personalized educational systems (Al-Shawwa & Alshawwa, 2020). ITS aren't some far-fetched futuristic idea. In a practical sense, they already exist Although unusual, they are capable of functioning without the presence of an instructor and may effectively challenge and help the learner through the use of various algorithms. The ever-increasing impact of technology is transforming how people learn and how institutions teach. People use the Internet to request online instruction from anywhere and at any time. This has resulted in the development of a plethora of online learning platforms that provide complete and effective educational solutions that are entirely online. Intelligent tutoring technologies that aid and guide students through the learning process, simulating the behavior of a human tutor, are available on these platforms.

The importance and contribution of e-learning systems, used as a supplement in the learning process, is unquestionable in normal situations, and it is more obvious in cases of lockdown scenarios, such as the one we are experiencing now (2021) due to the CORONAVIRUS health crisis (COVID-19 pandemic). In this situation, providing eLearning systems and promoting their use is one of the major challenges addressed by many educational institutions. Also, teaching programming is even more difficult when face-to-face classes are not conducted as it requires a lot of work and dedication from the students and teachers. Despite all the efforts of researchers, it seems to be difficult to find an effective method of teaching that is suitable to all students during this pandemic. An intelligent tutoring system is proposed in this paper to encourage students to learn through experimentation, proposing tasks on their initiative, which involves putting into use all the skills, abilities tools, and knowledge needed to successfully solve them. This system has been designed, developed, and applied for learning predictive parsing techniques and has been used by Computer Science students during four academic courses to evaluate its suitability for improving the student's learning process.

This paper is divided into two sections: first, an outline of the ITS structure followed by the implementation design of Intelligent Tutorial System for Computer Programing (ITSC).

### 1.1 Theoretical understanding

Intelligent teaching systems, are focused on students' expertise derived from an interpretation of their contact with the subject matter. Based on this study, an artificially intelligent agent determines students' cognitive profiles and adapts them to the learning needs that fit these profiles (Binh & Trung, 2021). For instance, the pedagogical

material accessible to students is tailored to their learning needs to optimize. To infer the learner's cognitive states, such as his or her degree of competence or proficiency, an insightful tutoring device must be capable of diagnosing the learner's behavior (Kumar & Ahuja, 2020). This approach is well served by the use of the student model, which tracks the learners' cognitive states and is continuously modified by (the system's intelligence) (Al-Shawwa & Alshawwa, 2020; Kumar & Ahuja, 2020). As a result, the development of detailed student models with trace and thus corrected using computers has always been a strong focus in the field of intelligent tutoring systems research (Cao et al., 2021). A minimum of four modules is usually included in an ITS (Neagu et al., 2020). For instance, the Expert Module (EM), this module contains facts and rules in a specific domain to be conveyed to the student; the Tutoring Module (TM) deals with designs and regulates instructional interactions through students; the Student Module (SM), which is a dynamic representation of the student's current state of knowledge; and finally the User Interface (UI), a module that controls the interaction between the student and the computer (Neagu et al., 2020).

Many intelligent tutoring programs have been planned and built for educational purposes (Al-Shawwa & Alshawwa, 2020; Cao et al., 2021; Kumar & Ahuja, 2020; Neagu et al., 2020). Most of these ITS are devoted to teaching computer science for example English language (e.g., (Schez-Sobrino et al., 2020), and mathematics (e.g., (White & Legg, 2021)) and (Baneres & Saíz, 2016; Utterberg Modén et al., 2021) are two examples of English language and mathematics ITS. ITS for assisting Computer Science students in learning debugging skills (Schez-Sobrino et al., 2020; Sharma et al., 2021), ITS for assisting Computer.

Science students in learning computer theoretical concepts or even C++ programming (Schez-Sobrino et al., 2020; Ahn et al., 2018). In Java Programming(Crow et al., 2018), Java Expression Evaluation (Moreno-Marcos et al., 2020), and Linear Programming (Dermeval et al., 2018; Keuning et al., 2021), an agent-based ITS for parameter passing was developed.

ITSs have been implemented in a variety of ways, using tools such as LISP, CGI, and others (Ahn et al., 2018; González et al., 2013). Furthermore, keeping safety in mind due to technological advancement regular management and backup is mandatory. It goes without saying that as technology advances and digital media becomes more widely used, attackers become more sophisticated (Xhakaj et al., 2017; Sharma et al., 2021).

We will be presenting the application design of an ITS based on the Moodle Learning Management System in this study. Content, as well as evaluation development, delivery, and management, are all handled by Learning Management Systems (Romero et al., 2008). They make it easier to keep track of grades more reliably and effectively (Utterberg Modén et al., 2021).

## 2 Problem statement

The COVID-19 has presented a series of unique challenges to the education institution. In the face of the global pandemic, access to traditional learning methods is limited. In the Pacific, many regional students cannot travel to their campus or have

missed weeks of lessons while being in quarantine. As a result, the students have got behind in their studies. Especially those who take programming courses where they are missing out on practical lab sessions. However, the initial case-base was more grounded towards using the traditional methods in upholding the learning processes. Now, this problem has shifted to zoom due to COVID-19 pandemic, hence, it is not practical for a tutor to look at individual cases to work and give each student an instant feedback on whether the learner is doing their code correctly. These practical lab sessions are so important as it does not only give students hands on experience with coding but also a instant feedback and guidance from their tutors. This research paper will propose a design of an Intelligent Tutoring System to teach programming language that can be implemented and integrated by the educational institution in the Pacific Islands to assist regional students or even students who cannot travel to their respective campuses so that the students do not lag in their studies. The proposed system will facilitate students' learning via rapid feedback. The system will instantly tell students if they are doing it right and suggest hints. If the student wishes to view solution, there is an option to do that too.

### 2.1 Current challenges faced in the Pacific

- Accessibility

It is observed that not all students in the Pacific have a personal computer or a laptop. They rely on computer lab facilities provided on campus. The system however could be made available on mobile devices and students can practice to some extent [12, 13]. Nevertheless, the Intelligent Tutoring System may not give a hands-on programming experience for the real environment (Binh & Trung, 2021; Dermeval et al., 2018).

- Lack of exposure to technology

It is observed that not all students in the Pacific have exposure to technology like a personal computer till they come to university. Nevertheless, these students do have the curiosity to learn about it and the courage to even take programming courses like C++, Java, Python (Palomino et al., 2014). While Pacific students with less exposure try to catch up and get hands-on experience using computers, they get behind in the course. Even though lecture notes, recordings, and lab activity solutions may be available, programming is best learned from practicing rather than watching, listening, or reading (Moreno-Marcos et al., 2020).

- Pandemic Restrictions

Before the pandemic, the students attended lab sessions where their tutor guided them. The students even had opportunities to meet their lecturers and tutors anytime and clarify their doubts. However, now, they can either email or consult when the

teaching personal is available on a scheduled video call. This isn't convenient or as effective as assisting face-to-face.

### 2.2 Added opportunities with ITS

- Cost-Efficient

Mostly in long term, having an Intelligent Tutoring System is far less costly than the traditional approach of completing the tutoring task.

The sole expense associated with ITS is the acquisition of the devices or applications needed to operate it thus is reusable. This is much less expensive than paying for an instructor's salary in a typical classroom. Because of its low cost, this technology is widely used in a variety of classes and jobs.

- Availability (24 h and 7 days per week)

Unlike an ordinary classroom, the Intelligent Tutoring System will be available to students for access any time and daily. So, based on student's convenience he or she can access to learn programming at any time.

- Time Productivity

The introduction of this sort of tool opens up a plethora of opportunities, one of which is the decrease of time spent in designing content and at the same time encouraging suitable knowledge.

## 3 Proposed methodology and operation

In this part, we present our proposed ITS system architecture with a planned core design. The ITS logical module, integrated with a distinct student modeling is implemented with dynamic hints generation capacities, that empowers ITS to fundamentally improve the programming performances of students (face-to-face or online registered students) (Xhakaj et al., 2017; Moreno-Marcos et al., 2020). As a result, the ITS authoring platform is designed to be extremely user-friendly and simple to incorporate problems of varying degrees of difficulty. When an instructor submits a programming problem, it is automatically made available to ITS and then to students. The authoring tool's user interfaces are as seen in Fig. 4. The authoring feature allows the instructor to see all of the issues in the session collection and can edit the selected ones if needed. To assess the required input to the user, ITS carefully reviews the student's submitted code based on the problem definition, specification, and problem-solution code. This allows each student to create their expertise to the fullest extent possible. Section 3.1 shows the detailing proposed architecture implements embedded with the Learning Management System (LMS) known as Moodle (Figs. 1 and 2) .

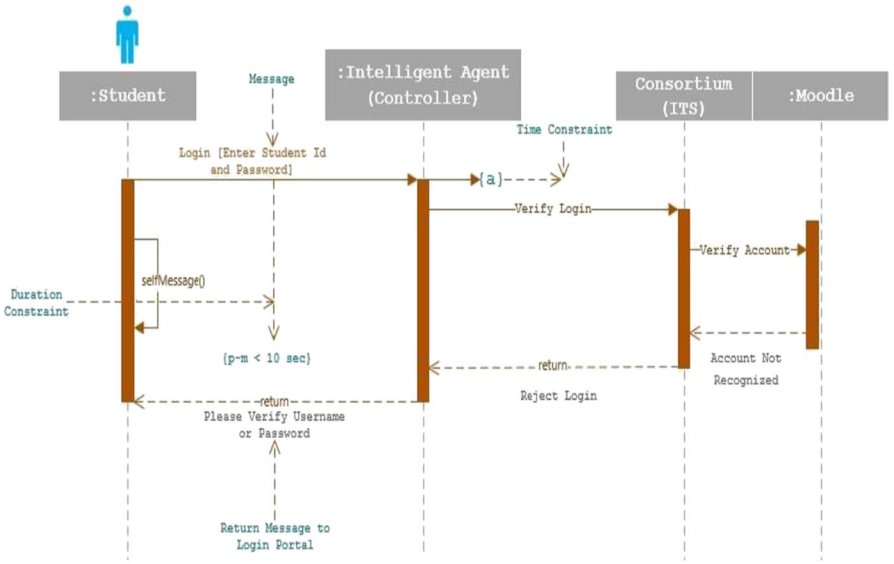


**Fig. 1** Shows the sequence diagram that describes a login interaction among a set of objects participated in for example (Student, Intelligent Agent, Consortium, and Moodle) that has a collaboration (or scenario on Intelligent tutoring), arranged in chronological order; it shows the objects participating in the interaction

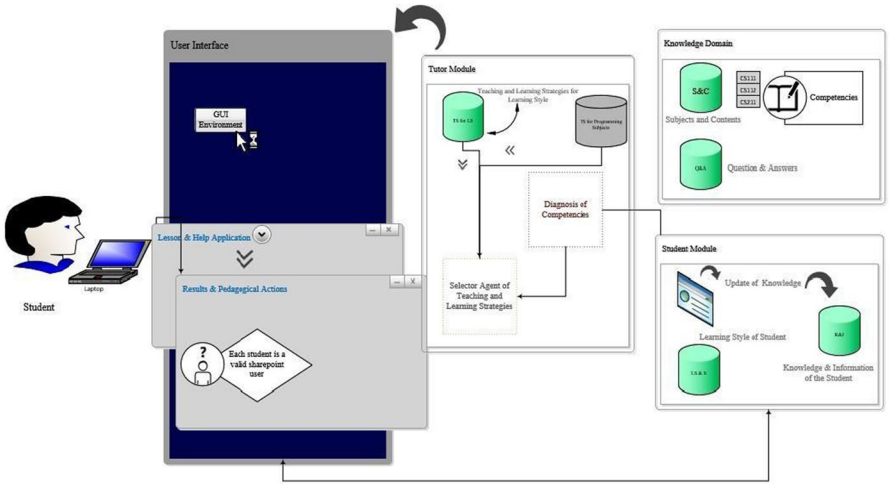


**Fig. 2** Shows the proposed architecture for ITS. Since the programming subjects have been chosen and customized, they will be made available to the student through the Moodle platform. The assessment will be applied after the student has revised a subject, and the database that stores the student's output will be updated with the results of the evaluation

### 3.1 Proposed ITS design and implements for the test-set

When the student's diagnosis process has historical information about him or her, the process would take the information about the student's performance (time of the activity, number of attempts, and assessments) to evaluate if the student has the competency and if not, the student will move on to the next topic and the selector agent will not execute. Otherwise, the selector agent will choose another resource to view, one that is more elaborated and uses a particular teaching approach that is more appropriate for the student's learning style (Fig. 3).

Students will be given a simple user guide through emails whereby they could follow to get started with this automatic system. The program will further have a getting started user/ student guide.

## 4 Discussion

The proposed system will be built on top of Moodle which is currently used by the universities in the Pacific. This means that students will not have to get used to a completely new system. To access the Intelligent Tutoring System for programming,

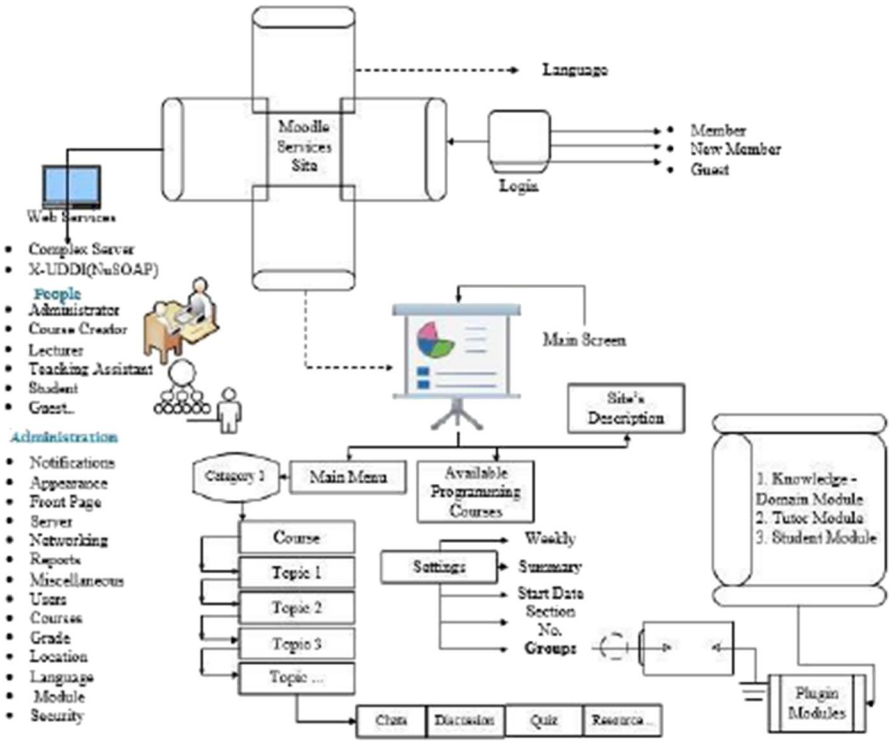


**Fig. 3** Shows the broad-spectrum architecture of Moodle and its functions and how the ITS system will be embedded into this LMS

the student will simply have to log into Moodle using their student id and password as illustrated in Fig. 1. Then the student needs to browse the course and open it. On their course page, a link to the Intelligent Tutoring System will be available for the student to utilize anytime. Clicking on the link will open an interface like the picture in Fig. 2. The system will solve the problem based on Case-Based Reasoning (CBR). The CBR will function in four simple steps; retrieve, reuse, revise and retain.

- Retrieve: Return related cases from the case base when given a new case.
- Reuse: Adapt the cases that were recovered to suit the current situation.
- Revise: Evaluate the approach and make the necessary changes depending on its effectiveness.
- Retain: Decide on whether or not to keep this new case in the case foundation.

If the conditions are simple, using the ***k-nearest neighbor's*** algorithm for any given number ***kn*** works well. The ***kn*** training examples with the input features nearest to the new example are used to predict the target value for the new programming problem when given a new problem. The prediction may be the mode, average, or any interpolation between the predictions of these ***kn*** training cases, with closer examples weighted more heavily than farther examples. A distance metric that calculates the distance between two examples is needed for this approach to operate. First, specify a metric for each feature's domain, in which the features' values are translated to a numerical scale that can be compared. Assume that $X_i(e)$ is a numerical representation of the value of $X_i$ for example e. The difference between examples $e_1$ and $e_2$ on the dimension described by function $X_i$ is then $(X_i(e_1) - Xi(e_2)$.

$$d(e_1, e_2) = \sqrt{\sum iWi * \left(X_i(e_1) - X_i(e_2)\right)^2} \tag{1}$$

The weights of the features may be used as feedback. These weights can be learned as well. The learning agent will strive to find weights that decrease the error in estimating the value of each element of the training set depending on the values of the other elements of the training set (Table 1).

The aim is to figure out which course the student enjoys learning and undertaking. In this case, the target function is Learning, and the aim is to find the definition as (Fig. 4):

$$\text{Learns}(e) \rightarrow C++(e)\Lambda\neg\text{Java}\ (e)\ \text{V}\ \text{Phyton}\neg\text{HTML}\ (e)) \tag{2}$$

**Table 1** Choices of the programming languages in a course

| Course ID | C++ | Java | Python | HTML | Learns |
|---|---|---|---|---|---|
| $c_1$ | *true* | *false* | *true* | *false* | *true* |
| $c_2$ | *false* | *false* | *false* | *true* | *false* |
| $c_3$ | *true* | *true* | *true* | *true* | *true* |
| $c_4$ | *true* | *true* | *false* | *false* | *false* |
| $c_5$ | *false* | *false* | *true* | *true* | *true* |

**Alg. 1.** Case-based Reasoning for Intelligent Tutoring System (ITS)

**for each request until max-problem do**

**Step 1:** Determine the weights of variables from Problem Request

i. $wt = \{wt_1, wt_2, wt_3, wt_{(m-1)}, wt_m\}$

**if** (PR is true) and (C is Same) **then**

**Step 2:** Calculation of local similarities for each Old Problem

ii. $S(wt_N, wt_p) = 1 - \frac{|wt_N - wt_p|}{w_{max} - w_{min}}$

**else**

**Step 3:** Calculation of global similarities for each Old Problem

iii. $S(C_{N,} C_{Ps}) = \sum wt\ (s_i\ (C_{N,} C_{Ps}))$

**Step 4:** Best solution resulting global similarities

**Step 5:** Retrieve the solution

**Step 6:** Present solution

**end**

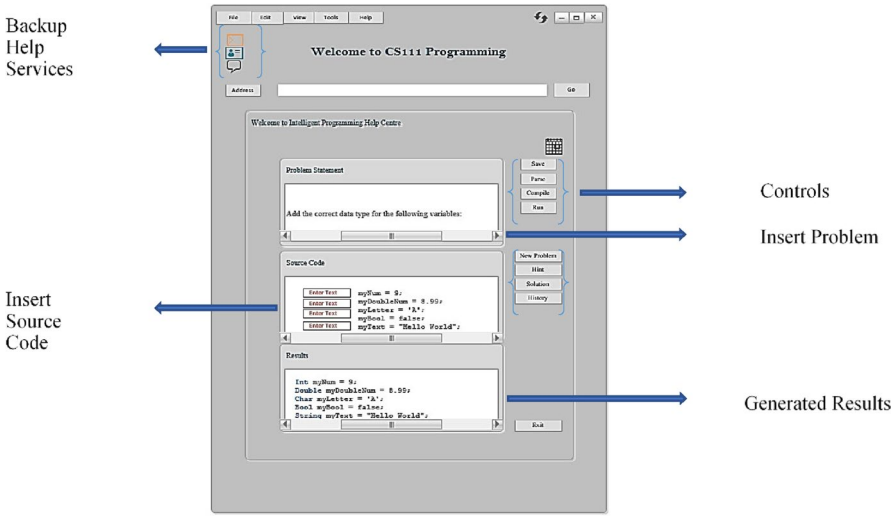


**Fig. 4** Shows the opening user-friendly proposed interface. This form enables the user to enter the details of the problem statement with prior source code respectively. The Graphical User Interface (GUI) for computer-based programming tutors is an important aspect that was taken into account during the design process. The user interface is built on a presentation style used by experienced programmers in many common Integrated Development Environments

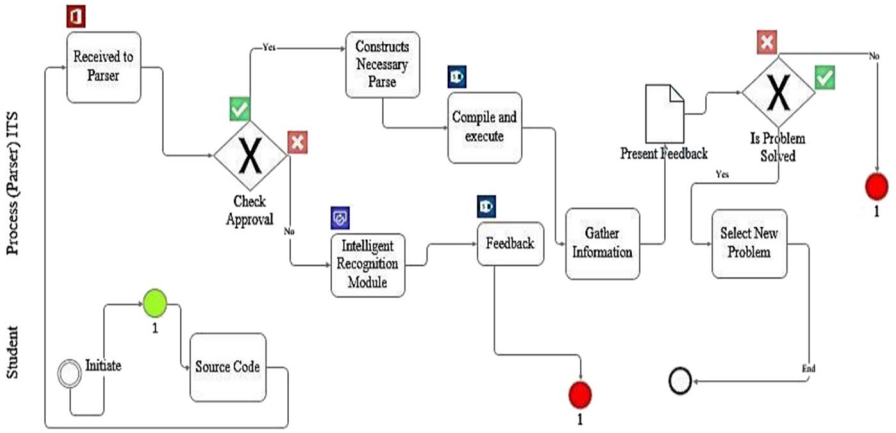


**Fig. 5** Shows the flow chart process model for the proposed ITS. Due to the complexities of semantic decoding, ITS can only mentor a limited subset of programming languages concurrently. Therefore, at some point, it necessitates making restrictions on a small subset

Furthermore, the Intelligent System will not only give programming activities to solve but will also provide an option to look at the hints to help the student to solve the problem. Also, instant feedback will be provided to the learner. If the learner wishes to look at the solution, the solution will also be made available instantly and a new activity will be given to the student to try and test his/her knowledge. Additionally, the student can view the history of his activities. This history will also be used by the system to determine the level of knowledge the student must determine the type of question he will get next. Moreover, the system will provide a compiler. This means that the student will not need to use a separate Integrated Development Environment to test the syntax or to view errors. The errors will be displayed in the results section of the user interface. A help tab is also provided for students to learn about how the user interface works. This will help the student to get familiarized and get started without needing any training. If the student has any question, there is an option to just click on the email icon and send the email to the course coordinator or the tutor or the peer mentor or even a friend with the activity question and his answer to get an even further explanation if the learner wishes too.

The planned system would enable students to browse models depicting an individual knowledge level. The content and method of teaching subject displays can be adjusted to the student's capacities. Furthermore, the ITS is gathered on the implementation and delivery of artificially intelligent systems. Yet, the proposed system aims to enhance and improve the learning and teaching process in programming knowledge related to the student's abilities (Fig. 5).

## 5 Conclusion

Intelligent tutoring schemes have several benefits over traditional teaching methods. ITSs provide continuous guidance and assistance to quickly get students to mastery especially for the online regional students in the Pacific during this COVID-19 pandemic 2021. The device will adjust to provide customized assistance by continuously recording and retaining a picture of how the student is performing. We have designed and proposed an ITS that uses Moodle as an LMP because the ITS architecture is a complex operation. The proposed method reconstructs the basic characteristics of cognitive tutoring programs and employs CBR as a method for diagnosing a student's intelligence, adaptive problem generation, and knowledge acquisition. Through solved cases and prior experience with students, the designer can eventually create the desired solution with CBR. Agents are often used to automate the CBR steps, which is considered essential. However, this multi-agent scheme in our methodology distributes the situation base and the CBR period among many agents. These agents aid in dilemma formulation and dissolution, as well as the detection and retrieval of reusable cases. The modular architecture allows parts to be reused and the case base to be managed efficiently. As a result, the article's key contributions are a guide for using CBR in ITS architecture, as well as adapting learning material and teaching methodologies to the student profile. Second, multi-agent structures are being used to simplify the CBR loop, allowing for modularity and component reuse. We have also modified the algorithms used in the CBR cycle's processes for use by agents.

## References

Abu-Naser, S. S. (2008). Developing an intelligent tutoring system for students learning to program in C++.

Abu-Naser, S. S. (2009). Evaluating the effectiveness of the CPP-tutor, an intelligent tutoring system for students learning to program in C++.

Ahn, J.-W., Chang, M., Watson, P., Tejwani, R., Sundararajan, S., Abuelsaad, T., & Prabhu, S. (2018). Adaptive visual dialog for intelligent tutoring systems. Paper presented at the International Conference on Artificial Intelligence in Education.

Akyuz, Y. (2020). Effects of intelligent tutoring systems (ITS) on personalized learning (PL). *Creative Education, 11*(6), 953–978.

Al-Shawwa, M., & Alshawwa, I. (2020). ITS for learning java.

Baneres, D., & Saíz, J. (2016). Intelligent tutoring system for learning digital systems on MOOC environments. Paper presented at the 2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS).

Binh, H. T., & Trung, N. Q. (2021). Responsive student model in an intelligent tutoring system and its evaluation. *Education and Information Technologies*, 1–23.

Cao, J., Yang, T., Lai, I. K.-W., & Wu, J. (2021). Student acceptance of intelligent tutoring systems during COVID-19: The effect of political influence. *The International Journal of Electrical Engineering & Education* 00207209211003270.

Cheung, B., Hui, L., Zhang, J., & Yiu, S.-M. (2003). SmartTutor: An intelligent tutoring system in web-based adult education. *Journal of Systems and Software, 68*(1), 11–25.

Crow, T., Luxton-Reilly, A., & Wuensche, B. (2018). Intelligent tutoring systems for programming education: A systematic review. Paper presented at the Proceedings of the 20th Australasian Computing Education Conference.

Dermeval, D., Albuquerque, J., Bittencourt, I. I., Vassileva, J., Lemos, W., da Silva, A. P., & Paiva, R. (2018). Amplifying teachers intelligence in the design of gamified intelligent tutoring systems. Paper presented at the International Conference on Artificial Intelligence in Education.

Desmarais, M. C., & d Baker, R. S. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction, 22*(1), 9–38.

Erümit, A. K., & Çetin, İ. (2020). Design framework of adaptive intelligent tutoring systems. *Education and Information Technologies, 25*(5), 4477–4500.

Galindo, J., Galindo, P., & Corral, J. M. R. (2019). Multimedia system for self-learning C/C++ programming language. Paper presented at the International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning.

González, C., Burguillo, J. C., Llamas, M., & Laza, R. (2013). Designing intelligent tutoring systems: A personalization strategy using case-based reasoning and multi-agent systems. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, 2*(1), 41–54.

Keuning, H., Heeren, B., & Jeuring, J. (2021). A tutoring system to learn code refactoring. Paper presented at the Proceedings of the 52nd ACM Technical Symposium on Computer Science Education.

Khazanchi, R., & Khazanchi, P. (2021). Artificial intelligence in education: A closer look into intelligent tutoring systems. In *Handbook of research on critical issues in special education for school rehabilitation practices* (pp. 256–277). IGI Global.

Kumar, A., & Ahuja, N. J. (2020). An adaptive framework of learner model using learner characteristics for intelligent tutoring systems. In *Intelligent communication, control and devices* (pp. 425–433). Springer.

Lee, C., & Baba, M. S. (2005). The intelligent web-based tutoring system using the C++ standard template library. *Malaysian Online Journal of Instructional Technology, 2*(3), 34–42.

Mishra, K., & Mishra, R. (2010). An intelligent tutoring system for c++. Paper presented at the 2010 International Conference on Electronics and Information Engineering.

Moreno-Marcos, P. M., de la Torre, D. M., Castro, G. G., Muñoz-Merino, P. J., & Kloos, C. D. (2020). Should we consider efficiency and Constancy for adaptation in intelligent tutoring systems? Paper presented at the International Conference on Intelligent Tutoring Systems.

Naghizadeh, M., & Moradi, H. (2015). A model for motivation assessment in intelligent tutoring systems. Paper presented at the 2015 7th Conference on Information and Knowledge Technology (IKT).

Neagu, L.-M., Rigaud, E., Travadel, S., Dascalu, M., & Rughinis, R.-V. (2020). Intelligent tutoring systems for psychomotor training–a systematic literature review. Paper presented at the International Conference on Intelligent Tutoring Systems.

Palomino, C. E. G., Silveira, R. A., & Nakayama, M. K. (2014). An intelligent LMS model based on intelligent tutoring systems. Paper presented at the International Conference on Intelligent Tutoring Systems.

Romero, C., Ventura, S., & García, E. (2008). Data mining in course management systems: Moodle case study and tutorial. *Computers & Education, 51*(1), 368–384.

Schez-Sobrino, S., Gmez-Portes, C., Vallejo, D., Glez-Morcillo, C., & Redondo, M. A. (2020). An intelligent tutoring system to facilitate the learning of programming through the usage of dynamic graphic visualizations. *Applied Sciences, 10*(4), 1518.

Sharma, P., & Chaudhary, K. (2020). Evaluating the malware threat cases in Fiji 2020. Paper presented at the 2020 IEEE Asia-Pacific conference on computer science and data engineering (CSDE).

Sharma, P., Chaudhary, K., Khan, M. G., & Wagner, M. (2019). Ransomware noise identification and eviction through machine learning fundamental filters. Paper presented at the 2019 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE).

Sharma, P., Chaudhary, K., & Khan, M. (2021). The art-of-hyper-parameter optimization with desirable feature selection. Paper presented at the International Conference on Medical Imaging and Computer-Aided Diagnosis.

Utterberg Modén, M., Tallvid, M., Lundin, J., & Lindström, B. (2021). Intelligent tutoring systems: Why teachers abandoned a technology aimed at automating teaching processes. Paper presented at the Proceedings of the 54th Hawaii International Conference on System Sciences.

Weitekamp, D., Harpstead, E., & Koedinger, K. R. (2020). An interaction design for machine teaching to develop AI tutors. Paper presented at the Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.

White, J., & Legg, P. (2021). Unsupervised one-class learning for anomaly detection on home IoT network devices. Paper presented at the 2021 International Conference on Cyber Situational Awareness, Analytics and Assessment (CyberSA).

Xhakaj, F., Aleven, V., & McLaren, B. M. (2017). Effects of a teacher dashboard for an intelligent tutoring system on teacher knowledge, lesson planning, lessons and student learning. Paper presented at the European conference on technology enhanced learning.