AN AI ASSISTANT FOR EDUCATION IN AUTOMATION

Asmar Ali, Andreas Deuter

Technische Hochschule Ostwestfalen-Lippe, Campusallee 12, 32657, Lemgo, Germany

Abstract

One of the challenges in universities is to take into account the different competences of students throughout the process of imparting knowledge. It is unlikely that all students will grasp the course content in the same way, as they have different abilities. Similarly, it is unrealistic for teachers to address all the individual needs of each student at all times. In addition, maintaining learners' attention to a particular topic is also an essential part of teaching.

To overcome these challenges, we are developing AI learning software as part of a research project. The software generates learning scenarios for the students and takes their individual competences into account. The research project focuses on the field of automation technology, especially on programming, as programming skills are usually required in automation environments.

When solving the scenarios, students receive immediate feedback on how well they have solved the task. Immediate automated feedback not only meets the students' expectations of quick learning feedback, but it also relieves the teaching staff in the assessment process.

In addition, the software includes an interactive user interface that allows learners to see the results of their programming tasks in a simulated 3D environment. This approach aims to maintain learners' attention over a longer learning period.

This paper describes the above aspects that are being developed in the research project and previews the new learning opportunities that could arise from the use of AI.

Keywords: *AI Education, Automated Feedback, Automation Education, Educational Software, Programming*

1. INTRODUCTION

One of the major challenges encountered by universities is accommodating the varying competencies of students during the process of knowledge dissemination. Given that students possess different abilities, it is unlikely that they uniformly comprehend the course content. Furthermore, it is impractical for the teaching staff to cater to the specific needs of every student consistently. Also, maintaining the learners' attention and interest in a particular subject matter is an essential aspect of effective teaching. (Teng & Chong, 2014)

One example: Around 60 students are taking the subject "Fundamentals of Programming" at the Ostwestfalen-Lippe University of Applied Sciences and Art. Each student has to solve several programming tasks per week. If one imagines that each student works on 3 assignments per week and that each assessment takes at least one minute of time, the teachers have to spend 6 hours per week on one subject only. This is not possible with the resources currently available.

In recent years, there has been a growing interest in the use of artificial intelligence (AI) in education to enhance learning outcomes and improve teaching practices. AI-powered educational software has shown potential to personalize learning experiences and provide immediate feedback, which can lead to better engagement and improved learning outcomes.

Several studies have shown the benefits of using AI in education. For instance, a study by demonstrated that adaptive learning software can improve learning outcomes by adjusting the difficulty level of the learning material to match the student's skill level. (Yalçın, et al., 2021). Moreover, AI-powered educational software can enhance engagement and motivation in learners by providing personalized feedback and learning experiences. (Sottilare, et al., 2018).

In this paper, we present a research project that focuses on solving aforemantioned challenges by developing 3D AI-based learning software for teaching automation technology, premilinary programming skills in the automation context at university level. The software generates 3D learning scenarios that take into account the individual competences of the students and provides individualized feedback on their progress.

In the second section, we review the existing literature and software tools that are relevant to our research. Section 3 presents the conceptual framework of our approach. Section 4 explains the evaluation methodology we will use to assess the effectiveness of our approach. In the fifth section, concluding remarks on the study are made.

2. RELATED WORK

Several educational software programs have been developed to enhance teaching and learning outcomes in various fields. In the field of computer science and programming, several AI-powered educational software tools have been developed to improve students' programming skills and engagement.

For instance, Codecademy is an online learning platform that offers interactive coding lessons and exercises for various programming languages. The platform provides immediate feedback to learners and allows them to practice coding in a gamified environment (Gasteiger, 2015). Another example is Scratch, a visual programming language developed by MIT, which provides an interactive and engaging environment for students to learn programming (Resnick, et al., 2009). Codecademy has been studied extensively in research for its effectiveness in teaching programming languages. For example, (Martinez-Ortiz, et al., 2018) found that Codecademy improved students' performance in programming tasks and increased their motivation to learn. Additionally, (Magana, et al., 2014) found that Scratch improved students' computational thinking skills and problem-solving abilities.

In the field of automation technology, several educational software programs have been developed to teach automation skills. For example, FANUC ROBOGUIDE is an industrial robot simulation software that allows users to program and simulate industrial robots in a virtual environment. This tool has been studied in research for its effectiveness in teaching industrial robot programming. For instance, a study conducted by (Chen, et al., 2019) found that using FANUC ROBOGUIDE improved students' learning outcomes and increased their interest in automation technology. In addition to the mentioned programs, there are other similar educational software tools that aim to teach automation skills, such as ABB RobotStudio. It has been studied for its effectiveness in teaching industrial robot programming (Ameri, et al., 2018).

Although many existing educational software programs have shown potential in enhancing learning outcomes and engagement, they may not fully consider the individual competencies of learners into account while generating learning scenarios. Our 3D AI-based learning software is specifically designed for teaching programming in the automation context and aims to address this challenge by generating individualized scenarios and feedback based on the learners' competencies.

3. CONCEPTUAL FRAMEWORK

3.1. Requirements Elicitation

The development of our software is a research project that requires careful planning and execution. In order to gather the expectations of project partners, video interviews were conducted and recorded. The transcripts of these interviews were then analyzed to extract the requirements for the software using the EARS approach. EARS, which stands for “Easy Approach to Requirements Syntax,” is a software engineering methodology that emphasizes the use of natural language to write software requirements. (Mavin, et al., 2009)

The EARS approach to software requirements writing has several advantages: It facilitates communication between stakeholders with different technical backgrounds, as it uses natural language

rather than technical jargon. It also allows for greater flexibility in defining requirements, as the requirements can be easily updated and refined as the project progresses. (Huang, et al., 2021) Using the EARS approach, the requirements were formulated, fully defined and recorded in a dedicated software called Polarion. Polarion is a commercial Application Lifecycle Management (ALM) software developed by Siemens (Anon., 2023). The software enables the team to manage the entire software development lifecycle, from requirements gathering to task management.

3.2. Competence Based Learning Scenarios

Personalized learning is one of the key approaches in pedagogy. It is an educational approach in which instruction and learning experiences are tailored to the individual needs, interests, and abilities of each learner. (Pane, et al., 2015) Research suggests that personalized learning can have a range of benefits for learners. For example, studies have found that personalized learning can improve student motivation, engagement, and achievement. (Pane, et al., 2015). Moreover, it can support the development of critical thinking, problem solving, and other 21st century skills that are increasingly important in today's workforce. (Dede, 2010). However, while personalized learning has the potential to improve educational outcomes for students, there are also challenges associated with its implementation.

The development of competence-based learning scenarios is a crucial aspect of our software, which serves as a complement to personalized learning. Upon a student's enrollment and initiation of learning, their initial level of competence is evaluated by assigning them a basic programming task. This baseline data is then recorded and stored in our database. Subsequent programming tasks are generated using an AI-based algorithm that takes into account the individual student's level of competence, thus providing a personalized learning experience.

Our approach is based on the principles of personalized learning, which tailors education to meet the unique needs and abilities of each individual learner. By utilizing a competence-based approach, we are able to design learning scenarios that are specifically targeted to the student's current level of knowledge and skill. This allows us to gradually increase the difficulty and complexity of the tasks as the student progresses, thereby maximizing their learning potential.

Fig.1 illustrates the competence-based learning scenarios generation process in our software. The sub-processes “Competence-Based Scenario using AI”, “Student Task Performance”, “Competence Update” illustrated in Fig.1 are the repeated processes, which means that after each scenario being solved by the student, the competence is updated, then based on the updated competence, a subsequent scenario is generated.

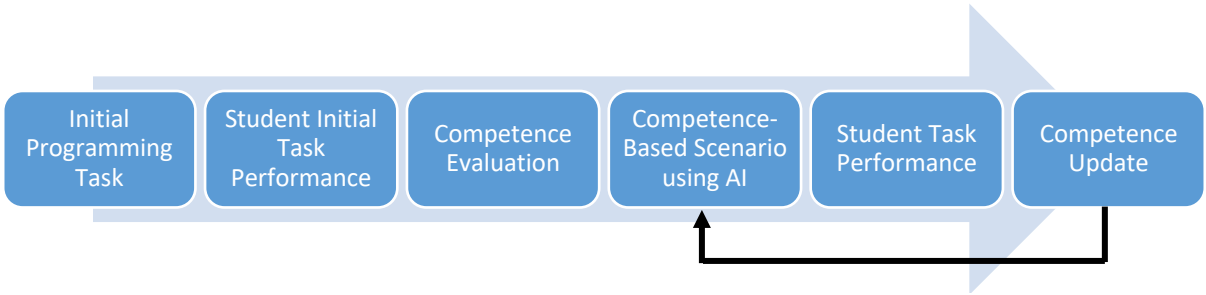

Fig. 1. Competence-based learning scenarios generation process.

3.3. Automated Personalized Feedback

Giving personalized feedback to encourage student engagement is becoming more and more crucial in current education technology. Automated feedback has become a promising approach for personalizing the learning experience for students, as it caters to the individual strengths and weaknesses and provides guidance to learners.

Research has shown that receiving personalized feedback is a critical component to successful learning outcomes (Cai & Virvou, 2020). Personalized feedback encourages learners to be more engaged, can improve their motivation levels, and can promote deep and meaningful learning. According to a meta-analysis of over 600 separate experiments on feedback, personalized feedback is not only one of the most significant factors in driving student achievement, but it is also a primary factor in creating learning environments that foster responsive teaching. (Hattie & Timperley, 2007)

Our software represents a step forward in helping students learn by giving them personalized feedback based on their performance on programming tasks. After submitting programming code, students receive personalized feedback based on the parameters described in Table 1.

Parameter	Description
Programming Skills	Assess the overall proficiency of the student in writing code that is syntactically correct, follows best practices, and is easy to understand and maintain.
Algorithmic Skills	Evaluate the student's ability to design and implement efficient algorithms that solve complex problems.
Complexity Analysis	Measure the complexity of the code and identifies potential areas for optimization, such as reducing the time or space complexity of the algorithm.
Bug-Score	Identify potential bugs or errors in the code.

Table 1. Parameters to provide feedback on code quality.

By using the parameters indicated in Table 1, our software generates personalized feedback based on students' task performances, allowing them to receive specific feedback on their programming proficiency, encouraging them to be more self-aware, and empowering them to become independent learners.

3.3.1. Code Annotations

In addition to the assessment based on the parameters summarized in Table 1, our software also provides the code annotations, which are an important aspect of modern software development, as they serve as a mechanism for detecting potential code quality issues such as syntax errors, style violations, vulnerabilities, or critical issues (Parnin & Orso, 2011). Table 2 shows an example of code annotations generated by our software.

Type	Severity	Description	Line
Style	Info	Unused import numpy	1
Style	Info	Import time should be placed at the top of the code	2

Table 2. Code annotations provided to the student including the type, severity, description, and line number of identified issues.

According to a research study, using automated code annotations significantly improved code quality by identifying and resolving issues quickly, which led to a significant improvement in the overall software program (Ali, et al., 2019). Another study found that code annotations provided a valuable form of feedback, which help in improving coding skills and identify areas of improvement (Jung, et al., 2021). Incorporating code annotations feedback in the early stages of programming education can help students to develop good coding habits and reduce the prevalence of errors.

3.4. 3D User Interface (UI) for Students

To enhance student engagement and motivation in learning automation, we developed a user interface for the students using Unity 3D. The interface provides students with a simulation environment that visualizes manipulators, which are 3D models such as a conveyor belt and a robot that can be controlled and moved through user input. Students can write code to control the manipulators, such as specifying the speed and direction of the conveyor belt or the movement and actions of the robot, and see the results of their code in real-time. The manipulators are an integral part of the simulation environment, and the use of manipulators in the context of our project refers to the ability to interact with and control these 3D models in a simulated environment. The design of the user interface focusses on usability and ease-of-use. We aimed to make the interface intuitive and user-friendly to help students quickly become familiar with the simulation environment. The user interface includes clear instructions and visual aids to help students understand the simulation environment.

We hypothesized that the simulation environment would enhance student motivation and engagement in learning automation by providing a more immersive and interactive learning experience. Our approach is supported by previous research in the field of educational technology, which has found that providing learners with visual and interactive learning materials can enhance their motivation and engagement (Mayer, 2005) (O'Donnell, et al., 2017).

In particular, Mayer's multimedia learning theory proposes that incorporating visual aids such as images, diagrams, and animations can improve learning outcomes by reducing cognitive load and enhancing retention (Mayer, 2005). Similarly, (O'Donnell, et al., 2017) found that interactive visual simulations could improve students' attitudes towards science and their perceived learning outcomes.

Our simulation environment provides students with a tangible connection between their code and the real-world applications of automation. This connection is important because it can enhance students' intrinsic motivation, which is the desire to engage in an activity for its own sake rather than for external rewards (Ryan & Deci, 2000).

We believe that the visualization of the manipulators and the real-time feedback on the results of their code helped students understand the practical applications of automation and provided a tangible connection between their code and the real-world applications of automation. Fig. 2 illustrates a screenshot from the student’s user interface.

The screenshot shows conveyor belt and the robot, along with a menu on the left side of the screen. The menu includes buttons for selecting the scenario based on the student's individual skill level, reading the task description and instructions for writing the code, and accessing the coding console located in the top-right corner of the screen.

With the coding console, students can write code and visualize the simulation of the manipulators in real-time based on their code. This provides a unique and interactive experience that allows students to see the impact of their code on the simulation. Furthermore, students can receive individualized feedback on their code by navigating the buttons present inside the coding console window.

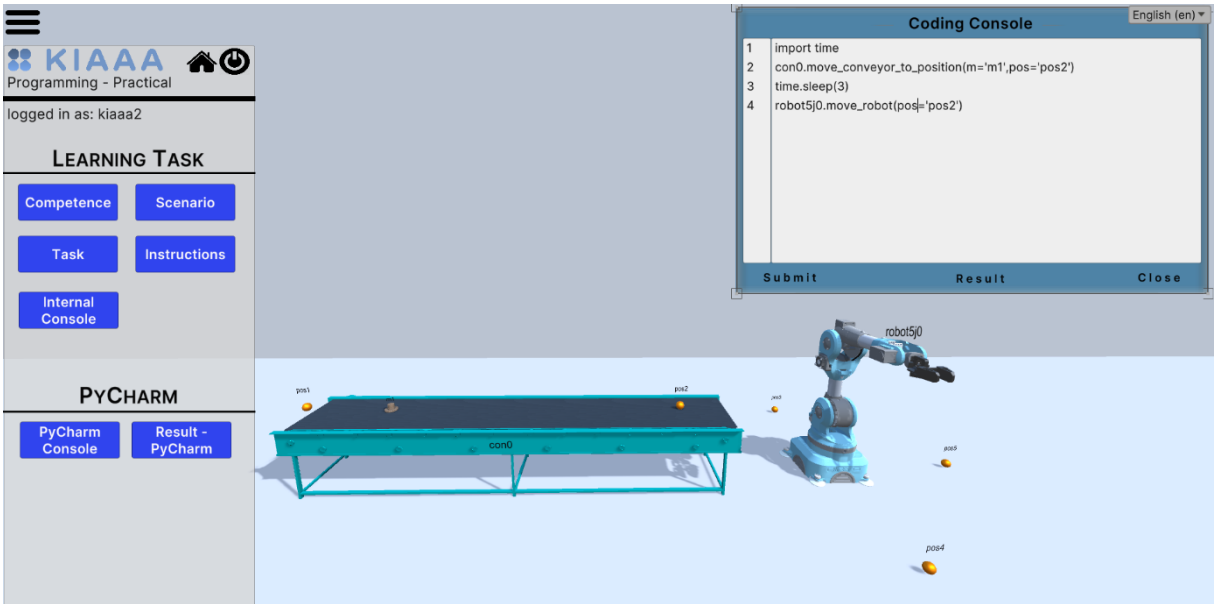

Fig. 2. Screenshot of student's user interface.

3.5. User Interface (UI) for Teachers

In addition to the user interface developed for students, we have also developed a user interface for teachers using a low code platform called Mendix. The teacher interface provides an easy-to-use dashboard that allows teachers to view updates on their students' performance, including progress on coding assignments, feedback on coding submissions, and overall coding skill level.

With this user interface, teachers can monitor the performance of individual students, track progress over time, and identify areas where students may need additional support or guidance. This provides teachers with valuable insights into their students' learning and helps them to tailor their instruction and support to meet the needs of each individual student.

Fig. 3 displays the user interface designed for teachers. The interface includes several buttons such as "Courses", "Students" and "Learning Scenarios" that allow teachers to add, edit and remove courses and students as well as view the learning scenarios created. Additionally, the interface allows the teachers to monitor the progress of the students. The user interface aims to provide a user-friendly and efficient platform for teachers to manage their courses and track their students' progress, ultimately enhancing the teaching and learning experience.

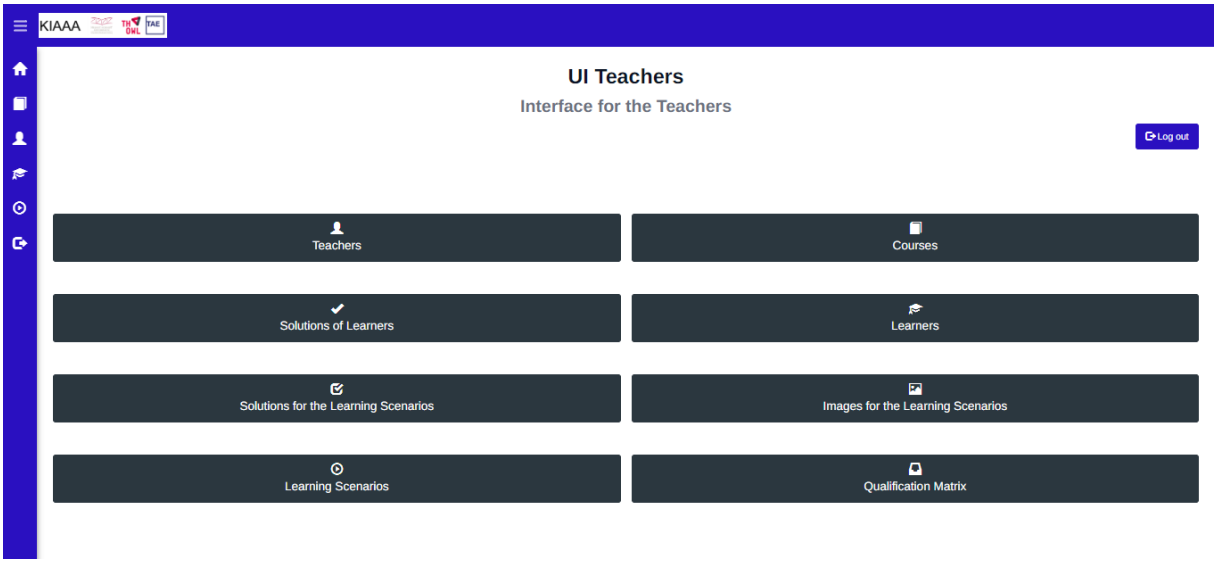

Fig. 3. Screenshot of teacher's user interface.

4. POTENTIAL EVALUATION METHOD

To evaluate the effectiveness of our approach in motivating students to learn programming, we plan to conduct a study with a group of 10 students. The study will be conducted in two phases. In the first phase, we will collect data on the students' baseline levels of motivation towards programming using pre-study surveys and interviews. In the second phase, we will provide the students with access to our software and competence-based learning scenarios for a period of time. We will collect data on the students' motivation levels during this period using surveys and interviews.

We will analyze the data using statistical methods to identify any significant changes in the students' motivation levels towards programming. Additionally, we will also consider the students' performance on the programming tasks generated by our software and their self-reported satisfaction with the learning experience.

Overall, this study aims to provide insights into the effectiveness of our approach in motivating students to learn programming. The results of the study will help refine our approach and improve the learning experience for future students.

5. CONCLUSIONS

Our research project focused on addressing the challenges of accommodating varying competencies among students by developing 3D AI-based learning software for teaching automation technology and preliminary programming skills in the automation context at the university level. Our software generates competency-based learning scenarios that take into account the student's current level of knowledge and skills, providing a personalized learning experience. The evaluation methodology that we plan to use will assess the effectiveness of our approach in improving student motivation, engagement, and achievement, as well as supporting the development of critical thinking, problem-solving skills.

Through our approach, we aim to not only improve educational outcomes for students but also reduce the workload of teaching staff by providing immediate feedback and automating the generation of competence-based learning scenarios. We believe this approach has the potential to modernize the way programming and automation technology is taught in higher education.

In conclusion, our research project demonstrates the potential of AI-powered educational software to enhance learning outcomes and improve teaching practices in higher education. We believe that our approach, which focuses on personalized learning through competence-based learning scenarios, has

significant potential to transform the way in which programming and automation technology are taught in universities. We hope that our research will inspire further innovation and exploration in this area and contribute to the ongoing evolution of educational technology.

ACKNOWLEDGMENTS

We would like to express our gratitude to the Federal Ministry of Education and Research, Germany (BMBF) for funding this research project. We also want to thank our project partners, Technische Hochschule Ostwestfalen-Lippe (TH OWL), Helmut-Schmidt-Universität - Universität der Bundeswehr Hamburg (HSU), and Technische Akademie Esslingen (TAE), for their valuable contributions and support.

REFERENCES

Ali, S., Nassif, A., Sayed, S. & Al-Qutayri, M., 2019. Quality Analysis Tools for Measuring Open Source Code Quality: A Survey. *IEEE Access*, pp. 7, 9410-9429.

Ameri, F., Moosavi, S. & Davari, A., 2018. The effectiveness of simulation in learning of industrial robot programming using ABB RobotStudio. *Technology and Engineering Education*, pp. 8(2), 49-62.

Anon., 2023. *Siemens.* [Online] Available at: https://polarion.plm.automation.siemens.com/products/polarion-alm

Cai, Y. & Virvou, M., 2020. Personalized e-learning systems: a review of the architecture and applications. *Computers*, pp. 9(1),6.

Chen, J., Chen, W. & Chen, T., 2019. Teaching and learning robot programming using FANUC ROBOGUIDE. *Intelligent & Robotic Systems*, pp. 95(3), 767-778.

Dede, C., 2010. Comparing Frameworks for "21st Century Skills". In J. Bellanca & R. Brandt (Eds.), 21st Century Skills: Rethinking How Students Learn. *Solution Tree Press,* pp. 51-76.

Gasteiger, H., 2015. Codecademy: A Platform for Gamified Programming Education. *n Proceedings of the 8th International Conference on Computer Supported Education* , pp. 375-380.

Hattie, J. & Timperley, H., 2007. The power of feedback. Review of educational research. pp. 77(1), 81-112.

Huang, J., Wang, L. & Lin, T., 2021. *An Exploratory Study on the Adoption of EARS in Software Industry.* s.l., Proceedings of the 19th International Conference on Advanced Information Management and Service.

Jung, H., Bae, D., Oh, H. & Joo, Y., 2021. What motivates developers to fix code smells?. *Information and Software Technology*, pp. 132, 106530.

Magana, A., Kumar, V. & Tang, J., 2014. Scratch and computational thinking in the CS classroom. *ACM Transactions on Computing Education*, pp. 14(4), 25:1-25:16.

Martinez-Ortiz, C., Camacho-Collados, J. & Martín-Valdivia, M., 2018. Codecademy and peer learning: Programming education at the turn of the century. *Computers & Education*, pp. 122, 28-42.

Mavin, A., Wilkinson, P., Harwood, A. & Novak, M., 2009. Easy Approach to Requirements Syntax (EARS). *17th IEEE International Requirements Engineering Conference*, pp. 317-322.

Mayer, R., 2005. Cognitive theory of multimedia learning. *The Cambridge Handbook of Multimedia Learning*, pp. 31-48.

O'Donnell, A., Dansereau, D. & Hall, R., 2017. Knowledge maps and interactive visual simulations as tools for enhancing science learning. *Journal of Educational Psychology*, pp. 109(7), 958-975.

Pane, J. et al., 2015. *Continued Progress: Promising Evidence on Personalized Learning.* s.l.:RAND Corporation.

Parnin, C. & Orso, A., 2011. *Are automated debugging techniques actually helping programmers?*. s.l., Proceedings of the 2011 International Symposium on Software Testing and Analysis.

Resnick, M. et al., 2009. Scratch: Programming for All. *Communications of the ACM*, pp. 52(11), 60-67.

Ryan, R. & Deci, E., 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, pp. 25(1), 54-67.

Sottilare, R. et al., 2018. *Developing adaptive learning technologies for adult learners in cybersecurity.* s.l., IEEE Frontiers in Education Conference (FIE).

Teng, M. & Chong, M., 2014. An integrative review on factors influencing student engagement in higher education. *Journal of Educational Evaluation for Health Professions*.

Yalçın, H., Sahin, C. & Bayar, U., 2021. An intelligent tutoring system for software engineering courses. *Applications in Engineering Education*, Issue 542-556.