The Development of an Intelligent Tutorial System for System Development

D. Al-Jumeily, A. Hussain, M. Alghamdi and D. Lamb
Applied Computing Research Group
Liverpool John Moores University
Liverpool, UK
d.aljumeily@ljmu.ac.uk

Hani Hamdan
Ecole Superieure d'Electricite (Supelec)
Department of Signal Processing and Electronic Systems
France
Hani.hamadan@supelec.fr

***Abstract*—Educational software has frequently been criticized as it has not been explicitly planned to meet the demands of educational environment. Therefore, there is an increasing demand for an intelligent computer technology to become used in the environment of education. This paper proposes the development of an intelligent tutoring system, which will aid students to learn software development. The idea is to simplify the learning process of computer programming which is a difficult process for novice programmers. The proposed system will be designed to use for either individual learning or group based work. The characteristic of proposed system differentiates itself from other programming tutoring systems as this system will monitor student's progress for the purpose of interfering (interactive feedback) at the needs of student. The interactive feedback has been used as part of the learning process through the methodology of learning by assessment.**

***Keywords*—Intelligent Tutoring System; System Development; Artificial Intelligence**

I. INTRODUCTION

Teaching novice programmers the skills associated with software development is a challenging process [1]. This is due to the fact that teachers are required to individually assess their students and then according to their existing level of knowledge and preferred learning styles start teaching a number of tasks such as the technical aspects of programming, new ways of thinking to solve problems and so on. Moreover, programming is essentially a technically-rooted and practical set of skills. Therefore, novice programmers need to build their skills by entering code, building software, and then as necessary executing, debugging and correcting the software. In practical, lab-based sessions, this often needs one-on-one help from teaching staff. With large class sizes and demands on tutoring staff, weak students in particular may not have the opportunity to get the individual help they require [2].

This paper proposes the design of an intelligent tutoring system to support the learning process of software development. The rest of the paper is organised as follows. Section II will give an overview of intelligent systems and their applications. Intelligent tutoring system will be explained in Section III. Section IV presents the proposed framework in more details. Finally, the conclusion will be given in section V.

II. INTELLIGENT SYSTEMS

The term "*intelligence*" means the ability to understand, learn from experience and think in a logical way about things. There are also other definitions such as the ability to get knowledge, respond quickly and successfully to new conditions [1].

According to Negnevitsky [3], intelligence can be the ability to learn and understand, to solve problems and to make decisions. One of the definitions of Artificial Intelligence (AI) is a science to make machines that would require intelligence if done by humans (Boden, 1977 cited in [28]). However, there is some debate around the definition of AI. Therefore, Turing preferred not to give a definition of machine and thinking [28].

Russel and Norvig [4] reported that Artificial Intelligence (AI) consists of two main ways. One is humanistic AI (HAI) that studies machines which think and perform as humans. The other one is rationalistic AI (RAI) that looks at machines which can be built on the understanding of intelligent human behaviour.

To build intelligent machine, we have to capture, arrange and use human expert knowledge in some problem area [3]. Russell and Novig in their study [4] postulated that there are six disciplines that encompass the core aspects of Artificial Intelligence.

- Natural Language Processing.
- Knowledge Representation.
- Automated Reasoning.
- Machine Learning.
- Computer Vision.
- Robotics.

Knowledge can be a theoretical or practical understanding of particular subject (facts and rules). Those who have knowledge are called expert therefore they are the most powerful people in their companies. This is because those experts are capable to express their knowledge in the form of rules for problem solving. Machine Learning is an important discipline in AI. This is due to the fact that machine learning enables computer to learn from experience, learn by examples and learn by comparison. There are important methods to

978-1-4799-5739-2/14/$31.00 ©2014 IEEE

machine learning such as artificial neural networks and genetic algorithms [3]. There are a number of definitions for Intelligent System; one of the acceptable definitions is that an intelligent system is a system that copies some characteristics of intelligence presented by nature.

The intelligence of a system can be described by its flexibility, adaptability, memory, learning, reasoning, and the ability to manage uncertain information [30].

Artificial Intelligence (AI) is an important field of study for building intelligent systems such as fuzzy logic, neural networks and genetic algorithms [1]. There are a number of diverse Intelligent Systems (IS) which have been developed in the last 10 years. Table I compares some of the existing IS using a number of features such as the expert rules utilized, the problem area and the utilised programming languages.

TABLE I. COMPARSION BETWEEN VARIOUS EXISTING INTELLIGENT SYSTEMS

System-Name	Problem Area	Data	Rules	Platform	Language	Year
Nutrition Diagnosis	Nutrition	Dietetics experts	Rules	Windows	C#	2012
Breast Cancer Diagnosis	Medicine	Mammographic mass data	Fuzzy Rules	Windows	Visual C#	2011
Lubricating oil refining process	Petro-chemical Industries	Experienced experts and operators of Industrial plant	Rules	Windows	C++	2008

a. Intelligent Tutoring System (ITS)

There are numerous definitions available that can be applied to the term of Intelligent Tutoring Systems (ITS). The most basic description is that of computer software designed for use in education that demonstrates intelligence [6]. Nwana suggests that ITS can be an intelligent computer program which can improve the performance of teachers in the classroom; in factors such as how they teach, and how to enhance teaching for specific students [7].

Researchers have argued that a Tutoring System can be considered intelligent if its design satisfies one of the following: Curriculum Sequencing (CS), Intelligent Solution Analysis (ISA), and Problem Solving Support (PSS) [9]. First, Curriculum Sequencing (CS) means that providing the student with the most suitable individually-planned sequence of topics to learn [9]. Second, Intelligent Solution Analysis (ISA) involves an automated check of the student's solution and providing feedback on the work, whilst updating his/her student model. Lastly, Problem Solving Support (PSS) gives the student intelligent help on each step when he/she is working on exercises or solving problems. However, Brusilovsky et al conclude that this PSS technology is not as popular in Web-based systems due to implementation complexities [9].

Recent work has begun to develop the "intelligent" aspect of ITS; such that the learning plan is structured and adapted based on the student's need. However, such existing work, while focused on adaptive learning, does not yet identify the architecture or method for continual adaptation of a student's learning experience based on [12]. Moving on to focus work in the target domain – supporting tutoring software development – researchers have already investigated some of the significant challenges and produced limited prototypical systems.

Assessment-centric systems recognise the limitations associated with simply comparing a student submission with fixed "model code", and progress has been made into semantically analysing these "correct model" answers such that a student's work is compared meaningfully against a set answer code [2]. However, beyond showing an improvement in student performance, little work has been done on exploring the pedagogical impact of these systems and how they may be integrated as an assessment component in a structured learning environment; for example linking performance on desired learning outcomes to learning materials. Other work in automated program assessment focuses on assessing a student's performance in a set task and providing meaningful feedback [13]. However, this gives little consideration to modelling the curriculum and its assessment features in a consistent, editable taxonomy; facilitating an ontology-driven intelligent and adaptive relationship to learning outcomes and learning support. As such, at present, there is no programming-centric intelligent learning support system with curriculum sequencing, intelligent solution analysis or even fully-integrated problem solving support. Therefore, a key research aim is to solve this issue by investigating an intelligent learning support system, centred on an adaptive learning approach to guide programming students.

Several ITS applications have been developed to supplement "expert system knowledge"; used to provide or support corporate or military training in traditional expert-system fields such as medicine and engineering. Several example systems are shown in Table II, which compares them against the tutoring systems that we discussed earlier.

TABLE II. ITS SYSTEM COMPARISON / FEATURE MATRIX

System	Notes	Date	CS	ISA	PSS
Algebrain	Maths system, guides students through equation solution process.	1999	No	No	Yes
AutoLEP	Assesses code in C Programming; provides feedback. Limited scope for adaptation and limited objective evaluation presented.	2010	No	Yes	No
COMET	Clinical reasoning automated tutorial system. Can identify learning outcomes for further study.	2007	No	Yes	Yes
ZOSMAT	Maths tutoring system, dynamic content delivery based on limited "student model"	2009	Yes	Yes	No

"Algebrain" is the first example system considered, and is one of the older systems described in this paper. It increases the classroom learner experience, providing an environment for experimentation with algebraic equations. Beyond simply solving equations, the software guides the student through the process of solving the equation, providing hints and descriptions for each step of the solution, along with immediate feedback on the steps taken [14]. However, this system has not considered the concept of assessment for

learning, and also it has not been implemented due to implementation issues. Moreover, Brusilovsky reported that the developers of "Algebrain" could not execute their system as it was required for them to have some other development tools [9].

"COMET" is designed to help medical students develop and practice clinical reasoning skills. It is used by student tutor groups, and provides guided tutorial sessions working through clinical hypotheses. It identifies students who perform well in certain scenarios and those likely to do so in future scenarios. Required learning outcomes are identified when students are incapable of forming correct steps in a hypothesis based on a presented scenario. Work evaluating the performance of this system found that COMET provided, on average, tutorial hints in accordance with human tutors 74% of the time [11]. However, there are some drawbacks in this system. Some of these are as follows. This system does not measure the prior knowledge of the learner; "COMET" has no language processing capabilities (e.g. no communication between users), and it does not specify the learning style of the learners. The next system considered is AutoLEP, which is particularly interesting with regard to this work, as it is intended for novice programmers. It is primarily an automated assessment system, featuring both static and dynamic assessment. Static assessment provides feedback on syntactical and structural issues, similar to a compiler. The authors assert that dynamic assessment identifies functional and non-functional features in the program, even if the software does not compile; particularly useful for assessing novice programmers. However, the model supporting this dynamism is not explained. It is not clear whether this feature is bespoke for a given set of programming issues, or if it is entirely flexible; and in the future desirable features could in fact be specified by a tutor. Another issue on this system would be that there is no pre-assessment for the levels of novice programmers. Therefore; it is crucial for new programmers to get the opportunity to automatically learn through a system that considers their prior knowledge. Not only that, it also has to consider their preferred learning styles. This is because learning what to teach and how to teach are very significant in the education of students. Thus, our proposed technology will consider these two pedagogical aspects.

The final system described is ZOSMAT, a Mathematics tutoring system. This is the only system considered here that delivers learner materials to the student in a planned and personalized manner. The ZOSMAT architecture includes several components. These components are a Student Model, Expert Model, Question Bank, and a Planner / Advisor. At a high level, the Student Model records student-specific information; the Expert Model is a rule-base, populated by subject specialists, while the Planner/Advisor produces learning materials, utilizing the previous models [16]. The intelligence of the system is attributed to the student model permitting the system to generate relevant content based on the student's performance. However, the ZOSMAT team identified significant shortcomings in this model – a simple record of grades; at their admission, not sophisticated enough to improve student learning efficiency (No clarifications for the students about how much progress they have done from the learning objective/learning outcomes. Therefore, they have an issue with grading and students did not learn from their result. Also, any automated teacher should clearly mention the learning objectives/outcomes of each of its lesson. After that, it should do the comparison according to what the students have achieved from the learning objectives. Another shortcoming within this system is that there is no specification for the preferred learning style of the learners. Therefore, the designers of this system should consider the preferred way of teaching as many individuals have many different ways in gaining knowledge, and those learners need to be directed to their preferred ways of learning by a robust system. Therefore, our proposed work will examine how to address some of those preceding issues.

b. Teaching, Learning & Assessment

Bloom's taxonomy features in pedagogical science, as a classification of educational goals, which can help teachers and lecturers in structuring their approach to learning. This can be reflected in the classroom; such as how to prepare and deliver lectures to students, how to structure and write exam questions, how to assess students and how to encourage students to increase their attainment levels. In addition, Bloom has divided the educational goals into three domains: Cognitive, Affective and Psychomotor [15, 16]. In 2001, Krathwohl et al revised Bloom's taxonomy and made some changes in the cognitive domain; updating the six levels in the taxonomy based on feedback from teaching practitioners and their interactions with students, from lowest to highest as Remembering, Understanding, Applying, Analysing, Evaluating and Creating [16]. Thompson et al developed Bloom's taxonomy, citing the difficulties in applying the levels of cognition to software engineering and programming; the categories were thus explained, using examples specific to programming as shown in Table III [17]. Applying Bloom's Categories within a technological framework could tremendously benefit both students and instructors [18].

TABLE III. BLOOM'S TAXONOMY WITH SOFTWARE ENGINERRING DERIVATIVE

Bloom's Category	Software Engineering Derivative
Remember	Can the student remember the syntax of For Loop?
Understand	Can the student explain the syntax of For Loop?
Apply	Can the student implement the For Loop?
Analyse	Can the student differentiate between For Loop and Do While?
Evaluate	Can the student decide whether it is better to use For Loop or Do While in the given question?
Create	Can the student make novel software?

For example, using "Clickers" technology (student response systems that are small hand-held keypads, which allow students to provide their responses) in the classroom would lead to increase student performance and engagement for

learning and also enables instructors to automatically see the student's answers and observe whether they have understood the given outcome or not [19].

Kirsten Crossgrove et al found that non-majors biology students performed better on these three types of questions (knowledge, comprehension, and application/analysis), and retained knowledge from clicker-based exam questions compared with non-clicker-based exam questions. Whereas biology students found the clicker tool is not a helpful assessment method [19]. However, a Clicker is a good on-the-fly assessment technology, but it can only consider Multiple-choice questions and students can easily guess the answers. Therefore, we need a better technology that could consider some of the other types of questions; not only have multiple choice questions although this would prove to be challenging. Our proposed system will aim to solve this challenge by including some of the other various types of questions; for instance writing the syntax of a "for" loop, find the bug on the given code and so on. Furthermore, we intend to make our planned technology as an automated tool that can be used for teaching, learning and assessment for software development students. In addition, our envisioned technology will differ from the existing technologies by including assessment for learning (AfL) in the learning process and using some crowd-sourced-derived tools [20]. In our proposed system, we will design a computational grammar that allows a developer to specify education requirements, including curriculum specification, student learning needs and preferences, and assessment performance. However, before commencing that we will analyse the strengths and weaknesses of the present computer science specific learning taxonomies.

Previous studies have mentioned that including assessment for high-level categories of Bloom's taxonomy or any other recommended educational taxonomies within a technology would be a big challenge [18, 19]. This could be attributed that high-level learning outcomes (Analyse-Evaluate- Create) cannot be simply measured by a multiple choice question as lower-level learning outcomes [19]. Consequently, there is a real demand for a technology that assesses those high level of learning outcomes or at least some of them. Assessment will be discussed in depth in the next section.

C. Assessment

Assessment is an activity that teachers can use to evaluate the performance of their students, and allows the students to evaluate their own performance. However, it is important to establish, before setting or performing an assessment, the purpose of the assessment. This is vital because there are several types of assessment and teachers should know which assessment type is the most suitable to choose for their students and the learning outcomes being evaluated [21]. The rest of this section will briefly describe several assessment types considered particularly relevant to this work. They are: formative, summative, diagnostic, and finally continuous assessment.

The formative assessment is a positive type as it will help both teachers and students to see the shortcomings in/of submitted work; enabling teachers to provide helpful feedback in order that students can focus on their weak areas to make good progress in their studies. At the same time, it will help teachers to measure their own performance for a particular student cohort [22, 23]. On the other hand, the summative assessment is normally done at the end of the course and provides a quantitative measure of performance (such as a grade or mark) [22]. Conversely, a diagnostic assessment is performed at the start of a learning plan, and is mainly to identify the learner's current understanding and attainment levels, and in some cases, identify student's learning difficulties [21]. The other type of assessment is Continuous assessment, which may occur several times while a student is studying; it provides an on-going measure of student performance and can be used to direct or guide future learning [24]. Indeed, teachers could find assessing large numbers of students in classrooms challenging [25]. However, this issue could be solved by developing an automated assessment tool although this solution requires many issues to be solved. This could be due to a number of reasons. One of these is students have diverse capabilities in understanding and answering dissimilar types of questions. Therefore, building a system that assesses individuals various abilities is difficult and this is because high-level learning outcomes (Analyse-Evaluate-Create) cannot be simply measured by multiple-choice questions in the same way as lower-level learning outcomes [18, 19]. Christopher Douce reported another shortcoming, which is an-automated assessments would not be able to deal with more complex questions as they are not flexible enough as a human [26]. To illustrate this, a clicker tool could be one of the simplest automated assessment examples, which designed for a Multiple-Choice assessment. Clicker's strengths and weaknesses have been discussed in the previous section. Another example would be that an automated essay scoring which is computerized software designed to assign a grade for the given essay. Furthermore, this automated technology could lessen teachers' effort in marking their student's works [27]. Conversely, previous studies have mentioned several drawbacks about this automated tool, for instance it does not provide individualised feedback for a learner, while it is only provide a simple grade based on mathematical models built on organizational, syntactic, and mechanical aspects of writing [28]. In contrast, the next section will be looking upon the crowd-sources education in depth.

III. The Architecture of the Proposed System

As can be seen in Fig. 1, a student who uses the system for the first time, has to go for a comprehensive exam that can test his/her basic knowledge of programming such as If statement, looping and so on.

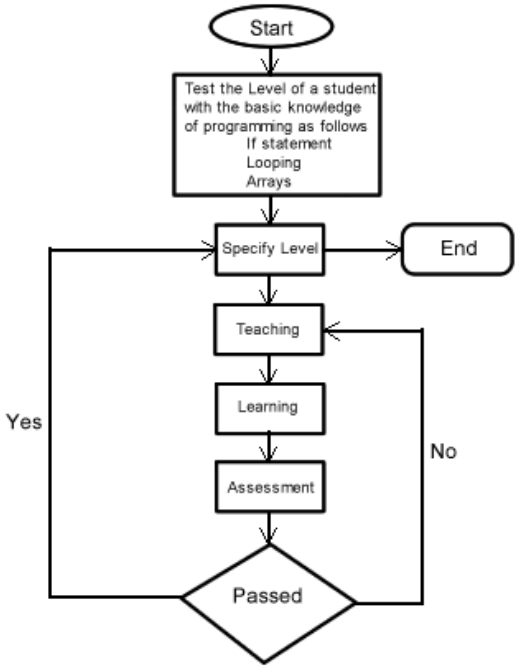

Fig. 1: Data flowchart of the proposed system

After that, the system will specify the current level (Beginner-Intermediate-Advanced) of this student and then direct him/her to the next step (Teaching), in this step he/she will take the most suitable individually planned sequence of topics to learn. Then, the system will take him/her to the "Learning" part as in this phase; he/she will start doing some more programming examples with intelligent help on each step of problem solving - from giving a hint to executing the next step. Following to that, the system will take him/her to the "Assessment" part as in this step he/she will have the most appropriate exam which can range from a simple question to a complex programming problem as well as the system will show his/her code error with providing error feedback for that. Lastly, in case this student passed the assessment part, the system will specify a new level for him/her (until he/she achieves the advanced programming level).

The main architecture of the proposed system consists of six components: Proposed manager, question bank, student model, content structure, expert model, and user interface (UI). In summary, the proposed manager coordinates the different components so they can work together smoothly. Conceptually, the relationship among the six components: proposed manager, content structure, student model, question bank, expert model and user interface (UI) can be viewed as in Fig. 2 and explained as in Table IV.

TABLE IV. ILLUSTRATES OF SIX COMPONENTS OF THE PROPOSED SYSTEM

Component	Description
Student knowledge model	Contains information about individual student's learning – ranging from materials studied, assessments taken, through to assessment results, extrapolated to identify performance against learning outcomes defined in the Curriculum model.
Curriculum model	Stores curriculum-related data; at its lowest level, specifications of the learning outcomes that make up a unit with differentiation levels. This model is likely to maintain appropriate learning materials and assessment templates for relevant outcomes. This will be populated by tutors and experts to specify courses and modules.
Curriculum assembly	Adapts curriculum model data to produce a series of materials for a given set of learning outcomes, tailored to a specific student model.
Assessment generation	Transforms curriculum model data into appropriate assessments for either a set of learning outcomes provided either directly by a tutor or inferred from a student model's outstanding learning outcomes.
Continual feedback	Produces feedback on assessment submissions, aligning student performance against learning outcomes in the curriculum models, using data from prior student attempts and ongoing tutor input.
Tutor, Learner and Expert interface	Provides a user interface and access control for the various roles. The tutor and expert interfaces will provide intuitive mechanisms for inputting curriculum materials and manually checking assignments and student performance, while the learner interfaces will provide rich lecture, tutorial and assignment user interfaces.

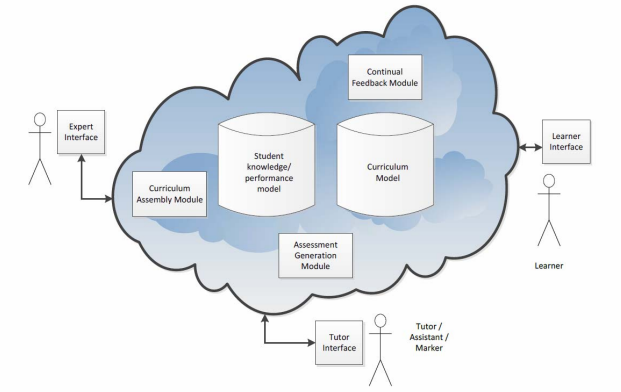

Fig. 2: The Proposed Software architecture

IV. Conclusion and Future Work

V. Currently, education provision is encountering new challenges, new inventions and emerging technologies. The main area where inventions are presented lies on instructional methodology. It becomes compulsory for the teachers to include new trends and methodologies in instructional methodology. One such feature that came into life is intelligent tutorial system. Consequently, we have developed a framework for a programming intelligent tutoring system as learning how to program is a difficult process for novice programmers. This paper provides an overview of intelligent systems and their applications and discussed how educational tutoring systems can be intelligent. The proposed system are

planned to be functional as part of this research work. The validity of the system will be examined in different universities in both the UK and in Middle East.

REFERENCES

[1] T. Wang, X. Su, P. Ma, Y. Wang, and K. Wang, "Ability-training-oriented automated assessment in introductory programming course," Computers & Education, vol. 56, pp. 220-226, 2011.

[2] I. J. Rudas and J. Fodor, "Intelligent Systems," Int. J. of Computers, Communications & Control, vol. III, pp. pp. 132-138, 2008.

[3] M. Negnevitsky, Artificial intelligence: a guide to intelligent systems, 3rd ed. Harlow: Addison-Wesley, 2011.

[4] S. Russell and P. Norvig, Artificial intelligence: a modern approach, 3rd ed. Upper Saddle River, NJ: Pearson, 2010.

[5] K. KrishnaKumar, "Intelligent Systems For Aerospace Engineering--An Overview," 2003.

[6] R. Freedman, "What is an intelligent tutoring system?" Intelligence, vol. 11, pp. 15-16, 2000.

[7] H. S. Nwana, "Intelligent tutoring systems: an overview," Artificial Intelligence Review, vol. 4, pp. 251-277, 1990.

[8] A. Keleş, R. Ocak, A. Keleş, and A. Gülcü, "ZOSMAT: Web-based intelligent tutoring system for teaching–learning process," Expert Systems with Applications, vol. 36, pp. 1229-1239, 2009.

[9] P. Brusilovsky, "Adaptive and Intelligent Web-based Educational Systems," International Journal of Artificial Intelligence in Education, vol. 13, pp. 156–169, 2003.

[10] A. Lesgold, S. Katz, L. Greenberg, E. Hughes, and G. Eggan, "Intelligent coached apprenticeship systems: Experience from the Sherlock project," in Systems, Man, and Cybernetics, 1991. 'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference on, 1991, pp. 1725-1737 vol.3.

[11] S. Suebnukarn and P. Haddawy, "COMET: A Collaborative Tutoring System for Medical Problem-Based Learning," Intelligent Systems, IEEE vol. 22, 2007.

[12] R. Peredo, A. Canales, A. Menchaca, and I. Peredo, "Intelligent Web-based education system for adaptive learning," Expert Systems with Applications, vol. 38, no. 12, pp. 14690–14702, Nov. 2011.

[13] S. S. A. Naser, "Evaluating the Effectiveness of the CPP-Tutor , an Intelligent Tutoring System for Students Learning to Program in C ++," Journal of Applied Sciences Research, vol. 5, no. 1, pp. 109–114, 2009.

[14] S. R. Alpert, M. K. Singley, P. G. Fairweather, and I. B. M. T. J. Watson, "Deploying Intelligent Tutors on the Web□: An Architecture and an Example", 1999.

[15] M. Forehand, "Bloom's Taxonomy," Athens: The University of Georgia, 2010.

[16] D. R. Krathwohl, "A Revision of Bloom's Taxonomy: An Overview," Theory Into Practice, vol. 41, no. 4, pp. 212–218, Nov. 2002.

[17] E. Thompson, H. Grove, A. Luxton-reilly, J. L. Whalley, and P. Robbins, "Bloom ' s Taxonomy for CS Assessment," in Australian Computer Society, 2008, vol. 78, January, 2008.

[18] A. Crowe, C. Dirks, and M. P. Wenderoth, "Biology in Bloom□: Implementing Bloom ' s Taxonomy to Enhance Student Learning in Biology," vol. 7, 2008.

[19] K. Crossgrove and K. L. Curran, "Using Clickers in Nonmajors- and Majors-Level Biology Courses□: Student Opinion , Learning , and Long-Term Retention of Course Material," vol. 7, pp. 146–154, 2008.

[20] M. Alghamdi, D. Lamb, D. Al-Jumeily, and A. J. Hussain, "Assessing the Impact of Web-Based Technology on Learning Styles in Education," in DeSE□: Developments in eSystems Engineering, IEEE Society, ISBN 978-1-4799-5263-2, 2014.

[21] C. Kyriacou, Essential Teaching Skills, Third. London: Nelson Thornes, 2007.

[22] W. Harlen and M. James, "Assessment and Learning□: differences and relationships assessment Assessment and Learning□: differences and relationships between formative and summative assessment," Assessment in Education, February 2013, 1997.

[23] D. J. Nicol and D. Macfarlane Dick, "Formative assessment and self-regulated learning: a model and seven principles of good feedback practice," Studies in Higher Education, vol. 31, no. 2, pp. 199–218, Apr. 2006.

[24] C. Coll, M. J. Rochera, R. M. Mayordomo, and M. Naranjo, "Continuous assessment and support for learning□: an experience in educational innovation with ICT support in higher education."

[25] M. Feng, N. T. Heffernan, and K. R. Koedinger, "Addressing the testing challenge with a web-based e-assessment system that tutors as it assesses," Proceedings the 15th international conference on WWW, 2006.

[26] C. Douce, D. Livingstone, and J. Orwell, "Automatic test-based assessment of programming: A review," J. on Educational Resources in Computing, vol. 5, no. 3, 2005.

[27] J. Markoff, "Essay-Grading Software Offers Professors a Break," Science, The New York Times Issue No:266, 2014.

[28] P. Ware, "Computer-Generated Feedback on Student Writing," TESOL Quarterly, vol. 45, no. 4, Dec. 2011.

[29] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11, 2011.

[30] Y.Chen, C.-Y. Hsu, L. Liu, and S. Yang, "Constructing a nutrition diagnosis expert system," Expert Systems with Applications, vol. 39, no. 2, pp. 2132–2156, Feb. 2012.