An Intelligent Grading System for e-Learning Environments

Shehab A. Gamalel-Din

Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah, K.S.A.

Abstract. Many educators believe that among effective learning methods are learning-by-explaining-experience and learning-by-correcting-mistakes through one-on-one interactions with students. It is not surprising, then, that an effective way of teaching is to give students immediate feedbacks on exercises and problems that they have just solved. Unfortunately, such one-on-one teaching scenarios are becoming increasingly difficult to arrange in e-learning, especially web-based learning, where the number of participating students is continuously increasing. Computerized empowered intelligent graders that provide students with comprehensive explanations on their mistakes and what a correct answer would be are of urgent need to overcome such difficulties. This research have introduced the new concept of Intelligent Grading Systems (IGS), designed a generic framework, and implemented Smart Grader (SG) – a simple proof of concept prototype. SG is assumed to either work as a separate grading tool or as an integrated component to e-Learning systems for grading programming and mathematics problems.

1 Introduction

E-Learning technology offers various e-tools to support all types of education and training systems. This paper presents one of such tools, the Smart Grader, that supports one critical activity for all education systems, namely, student assessment. Due to the lack of enough human resources, full independence on a human grader was a target, which limits the types of questions to those that could be machine marked. In such types of questions, the final result is what matters, which means that the approach/strategy the student followed to reach a specific selection doesn't matter and is not assessed. This is just enough for the first three levels of Bloom's taxonomy, but doesn't generally suit the higher levels, e.g., application, synthesizing, analyzing, and evaluation, especially in those subjects, such as programming and mathematics, where the assessment of the problem-solving skill is what really matters.

Studies [1,2] have revealed that immediate feedback of tutors in problem-solving yields the most efficient learning model than those that do not provide such a feedback. Accordingly, these researches suggested that explicit guidance is required to improve the efficiency of learning. This observation had motivated the idea behind this research.

Therefore, this research tries to assess the above hypothesis by introducing a new concept that we analogously call it Intelligent Grading Systems (IGS). In contrast

A. Gamalel-Din S..
An Intelligent Grading System for e-Learning Environments.
DOI: 10.5220/0004087300030013
In *Proceedings of the 1st International Workshop on Interaction Design in Educational Environments* (IDEE-2012), pages 3-13
ISBN: 978-989-8565-17-4
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

with current grading systems, an IGS does not only evaluate students based on their final answers to quizzes and exercises but also on the steps they followed to reach those answers. A student might reach a correct problem solution; however, the grader might not accept the followed approach, as it doesn't match the taught concepts being tested; or on the other hand, the grader might suggest a better approach, which adds to the effectiveness of the learning process.

Unfortunately, current e-Learning systems do not usually interfere with the students during their exercise/test sessions. Therefore, integrating Interactive IGS with e-Learning environments would overcome such a weakness and improve its learning effectiveness. An interactive online IGS may interfere with the student in a testing/tutorial session for several reasons among them are correcting misconceptions, and highlighting better or more advanced approaches [3], which makes the assessment process an integral part of the learning process and which adds to its efficiency. Therefore, an interactive grader should be active, alert, and ignited all the time during problem solving sessions.

On the other hand, offline IGS would allow a totally new dimension of questions that had never been thought in distance learning tests. Such questions would usually require human graders, which in most cases is either practically impossible or very expensive, especially in case of international exams that are conducted worldwide simultaneously on the Internet with a huge number of students.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

In summary, the Intelligent Graders (IGS) are computerized empowered intelligent graders. This research introduced the new concept of Intelligent Grading Systems (IGS), developed a model of IGSs, designed a generic framework, and implemented Smart Grader (SG)—a prototype for grading Lisp programming problems. SG can work both offline or as integrated to Intelligent Tutoring Systems (ITS), to provide more effective learning through grading student tests, correcting mistakes, and providing advices on better ways of problem solving.

2 Intelligent Grading Systems—The Model

Active help systems (AHS) [4,5], intelligent assistants (IAS) [6,7], and intelligent coaching systems (ICS) [8,9] monitor their users, predict and evaluate their action plans, and give them advices on correct or more optimum directions. IGSs have many things in common with such systems; however, they also differ in many ways. The following discusses some of the major unique characteristics of an intelligent grader in contrast to help, assistance, and coaching systems. This discussion is aimed at highlighting the design directions of an IGS framework.

2.1 Action Plan Recognition in IGS

Generally, students during a problem solving session follow steps of a certain Action Plan leading to the final answer. Grading systems need to analyze such steps, evaluate them, and assess the student's grasping of the concepts being tested for assigning grades. Generally, those action plans have some unique characteristics that motivated this research:

- The goal is already known in advance; it is the solution to the given problem. Therefore, the grader is not puzzled with what the student is trying to achieve. Therefore, the space of solutions, and hence action plans, is limited and known.
- The student previous knowledge, with regard to those subjects under current training, will further narrow down the space to a subset of the solution space.
- In some cases, students may skip some steps to shorten their answer text. This will require the grader to try to infer how to relate the given high level steps to the detailed ones that are not explicitly stated in the answer and that lead to such results.
- Intelligent Graders are either synchronous (online) or asynchronous (offline) as opposed to the other systems that have to be active and interfering with the user actions all the times.

Because of these similarities and dissimilarities with AHS, IAS, and ICS systems, similar techniques are adopted with appropriate adaptations in this research to suit those IGSs' unique characteristics.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

2.2 IGS as Integrated to an e-Learning Environment

Fig. 1 depicts the different types of interactions of the Intelligent Grader with the different components of an e-learning environment. Based on whether the Grader is working under synchronous mode (online problem solving sessions simulating oral exams) or asynchronous mode (off/online assessment exams), IGS responses will differ.

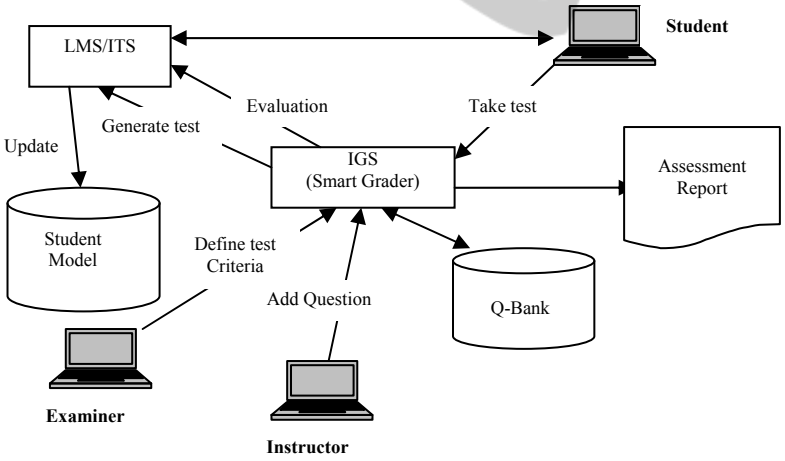

Fig. 1. IGS in the Context of e-Learning (A Use Case).

An Asynchronous Grading Scenario. The following is one possible scenario for Smart Grader interaction with the student:

1 A problem is drawn from a question bank and is given to the student to answer.

2 The student writes down the answer in a format that is acceptable and understandable by the Intelligent Grader. An intelligent editor for the problem domain is helpful here.
3 IG, would then, syntactically and semantically analyzes the answer trying to recognize the action plan behind it. The answer is compared to the knowledgebase of expert's solutions. Inconsistencies are analyzed and reasons for errors are identified by the Executable Action Plan Recognition Machine. Hence, comments and feedbacks are compiled for reporting back to the student with proper explanations and suggestions.
4 The Smart Grader, then, gives appropriate grade according to its assessment of how much the student have grasped the concepts being tested in the given question. This step will be guided by grading framework as described in the expert's knowledgebase.

A Synchronous Grading Scenario. Another possible scenario might look like the asynchronous scenario except that the system is kept active during the answer session. Although the grader is kept silent with no interference with the student actions, it keeps analyzing his actions in the background and tries to recognize the strategy behind them through matching them with previously fed instructor's solution(s)' strategies. Interference is possible only upon a help request from student. IGS allows student to request a support: help, hint, or next answer step; the impact of each on marking will be weighed differently.

Assessment Reports and Evaluation. Different types of information could be reported:

- Annotation report in which explanations, suggestions, correct answers are annotated to each wrong step.
- The instructor's strategy is reported as a model solution's strategy.
- Overall assessment of the achievement level with reference to course objectives, knowledge, and other soft skills, e.g., good at solving problems, grasped and missed knowledge, … etc.

3 The Intelligent Grading Systems (IGS)—A Conceptual Framework

The Smart Grader uses the AI technique of *Action Plan Recognition* for its grading technique. The unique characteristics of action plans as mandated by IGS, simplifies the job of the automated IGS as it will not be puzzled with what the student is trying to achieve. This would also limit the space of solution strategies and action plans. Accordingly, the instructor can design few strategies and detailed action plans for possible solutions as model answers.

The following sections discuss how IGS utilizes action plan recognition through introducing what we called it the Executable Action Plan Machine (EAPM). The next section discusses the structure of APM in terms of the description of the Plan Hierarchy knowledge-base and how the different APMs might interact.

3.1 The Action Plan Hierarchy Knowledgebase

It should be noted that most problems might be solved by several correct ways. The student needs to follow only one single approach of them. Accordingly, the "Action Plan Hierarchy" (APH), a modified AND/OR tree, is a hierarchical structure in which the root node is the goal and all intermediate nodes are the sub-goals and the terminal leaves are the primitive actions or events. A sub-goal node can also be viewed as a root of a sub-tree for solving a reduced problem, the sub-goal.

IGS requires the instructor to define an APH for each problem. The root of the tree represents the problem's main plan. The next level below the root is either OR or AND branches but not both: AND branches indicate that this plan requires all the sub-plans to be done to achieve its goal, and OR branches indicate that there are several possible correct sub-plans. Similarly, all intermediate tree nodes are recursively defined. The leaves of the tree are the student's steps that are required to achieve the solution. To explain, let us consider the following example problem:

"The student is asked to write down a Lisp program that takes three lists A, B, C, and manipulates them to return a new list that is composed of the first element of A and the last two elements of C embedded with the reverse of B in between. In other words, Write a Lisp function that takes the lists (a b c), (d e f), and (g h i j) to return (a f e d i j)."

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

This problem can be solved in different ways. Two solutions with their strategies are presented in Fig. 2, which presents a portion of the *frame structure* used to represent the knowledge regarding the Action plans and Strategies for each problem. The instructor will be requested, with the aid of tools, to define one such a frame for each problem.

Noteworthy, the Action Plan Hierarchy is a top-down breakdown structure while student actions lie at the bottom of the hierarchy and hence, plan recognition goes bottom-up, which adds to the complexity of the recognition job.

3.2 The Executable Action Plan Machine

The plans of Fig 2 indicate that the solution strategies follow several interacting threads. Accordingly, students also might follow multiple threads during answering the question. Some of those threads might be in the correct direction and are actual sub-plans towards the final solution, while others have nothing to do with the solution and are only investigative trials in the answering process. Multiple threads are usually active concurrently and the student swaps from one thread to another. Therefore, this research investigated the use of Dataflow graphs (DFG) as an active pictorial means for representing such concurrency behavior. Action Plan Machine (APM) is an adaptation of DFG to get an executable effect for simulating and, hence, identifying the actual student behavior during the answering session. The graphical depiction of action plans by APM is automatically constructed from the APH representing the problem solutions. Fig 3 depicts the APM for the APH of Fig 2.

The construction process of APM starts from the Action Plan Hierarchy. The constructs of the APM are only action nodes and directed links. Each action node represents one strategy step (or sub-strategy) of the problem solution. Those action

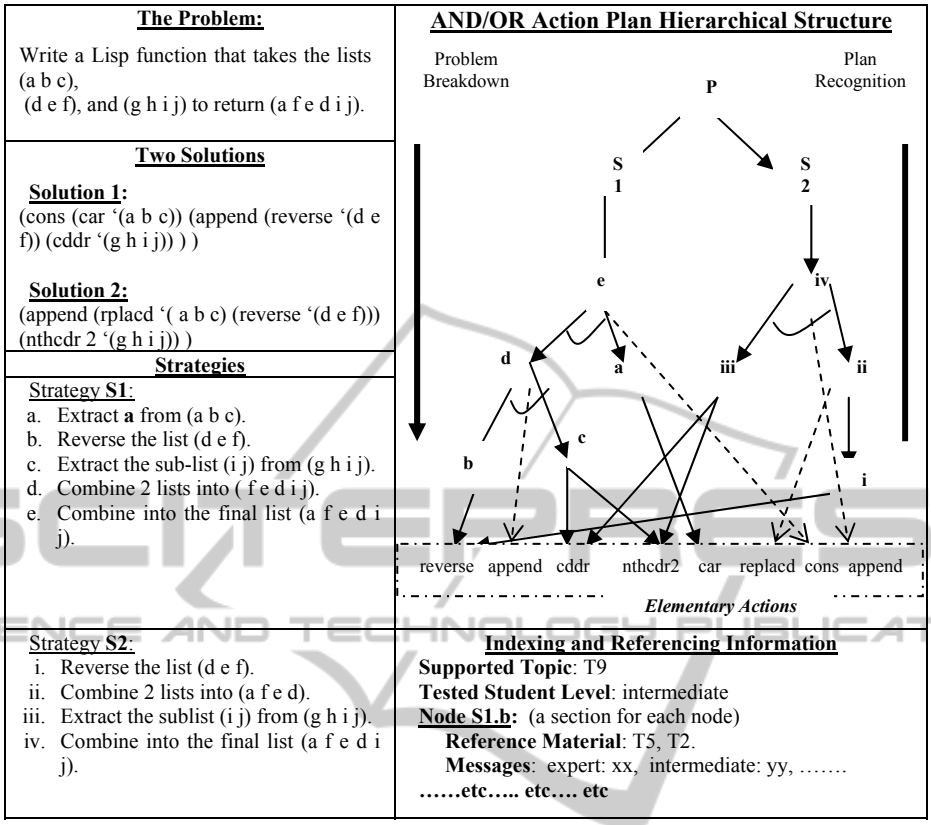

Fig. 2. A Problem Case-Frame for a Lisp Example.

nodes are connected together through the directed links to compose an Action Plan Network (APN). The action nodes in an APN are organized in such a way to reflect the order of execution for a correct plan. Therefore, both sequences and concurrencies are natively represented. The inputs to each action node are the parameters (or data) required for that sub-strategy to be achieved; hence, it might be an external input data or an intermediate result from another interlinked intermediate sub-strategy.

In fact, a node is an active actor that contains all necessary information for the grader to perform its job, e.g., messages, responses … etc. Therefore, each action node is an active mini-grader for the sub-goal it represents.

On the other hand, alternative strategies (OR branches in the Action Plan Hierarchy) are represented by separate nodes internal to the mother strategy node. The inputs to the mother strategy node are also inputs to each alternative sub-strategy, and so are the outputs. Fig. 4 depicts an Action node for the sub-strategy S1.c or S2.iii of the example of Fig. 3. Those internal alternative nodes are mutually exclusive.

Noteworthy, the constructed machine can be verified for consistency by verifying that all inputs to the modeled sub-strategy are consumed by the different action nodes. Note that some nodes exist in both machines expressing common sub-strategies, which adds another complexity dimension to the plan recognition process since such actions will confusingly activate both plans simultaneously.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

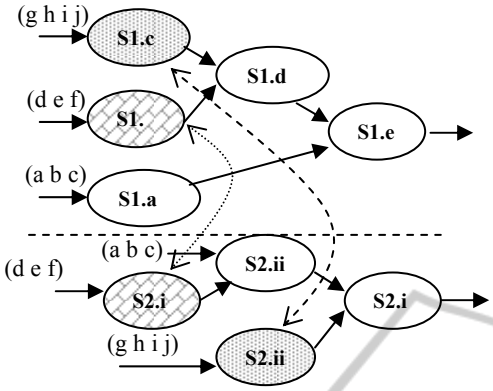

Fig. 3. Example APM.

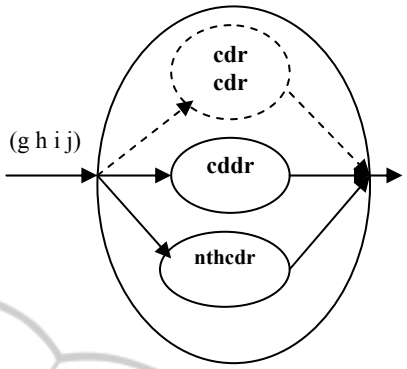

Fig. 4. Nested Alternative APM for Strategy S1.c and S2.iii.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

4 Prototype Implementation—Smart Grader

The Smart Grader (SG) is the prototype implemented to test the IGS framework. It is partially developed using the .NET C# development environment. The core of SG is the Smart Assessor responsible for implementing the IGS model.

4.1 The Implementation of the Smart Assessor

Fig. 5 simplifies the overall process followed by the Smart Assessor to fulfill its function. The Lisp machine is an APM whose nodes are active nodes corresponsing to Lisp constructs and hence can execute. Student's answer is first parsed to construct the Lisp APN. However, a Smart Lisp editor, through which the student writes down the answer, can also be used and hence, parsing is embedded and the machine would be implemented incrementally.

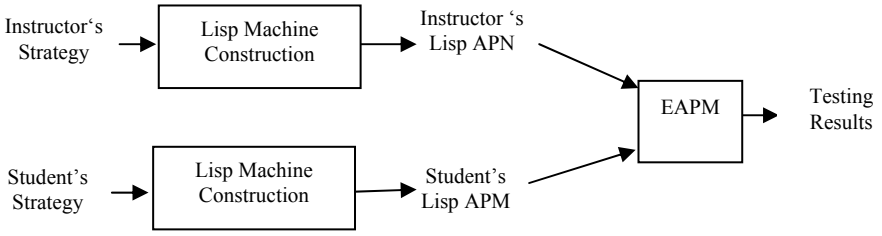

Fig 5. The Components of Smart Assessor.

Therefore, the parsed student's answer is first projected onto the APM structure. A time stamped token is generated to represent each terminal in the parse tree. The terminals of the parse tree are considered the input literals of the APM. Therefore, the algorithm matches those tokens with those literal inputs and, hence places them at the

appropriate places in the APM. The internal nodes of the parse tree represent the student actions corresponding to the APM's active nodes. In fact, the projection process uses existing nodes in the model APM, if they exist; otherwise it generates new ones and superimposes them on the same APM. In summary, the parse tree is projected onto the APM machine by inserting the appropriate time stamped tokens to the appropriate APM nodes.

Once the APM inputs are properly placed and the active nodes are properly colored, the APM is activated and tokens propagate through the APM network. Once the inputs of any node of the APM are complete and the node is colored, that node is activated to generate an output token that propagates through the corresponding output pipeline (link) to the next node. Token propagation indicates the plan followed by the student to answer the problem. If that plan followed those nodes of the original APM till the final goal node, then the answer is correct. Using newly inserted nodes that don't finally merge with the original APM would be reported and will require human evaluation.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

4.2 SG Interaction with the Student

Student interactions with SG may take many forms: solution actions and event-request actions. This is done via a set of interface buttons: Hint, Help, or even go back to ITS for reviewing certain material. SG working in a synchronous mode (e.g., simulating oral exams) detects when the students take longer than expected for the next move and accordingly issue a time-out event. In addition, the student can request a report on his/her performance to which the Grader reviews the Working APN and the token time stamps, and then reports a full detailed history of the student actions and events during solving a specific problem.

5 Model Validation and Evaluation

To validate the mode of IGS, other examples were tried. A Java programming problem for calculating *Factorial(n)* is used. Recursive and non-recursive strategies with many alternative sub-strategies are considered. Fig. 6 depicts the alternative hierarchical solution strategies, which are used as the basis for building the APH Frame.

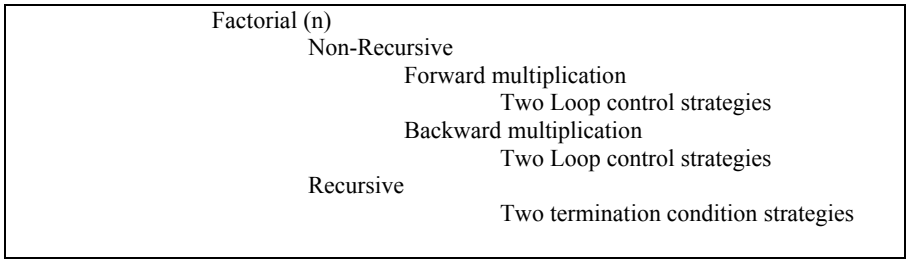

Fig. 6. Hierarchical strategic plans for *Factorial(n)* Java Programming Problem.

6 Related Work

Gamalel-Din [9] presented the Smart Coach (SC). He introduced an approach for action plan recognition, which is more suitable for coaching systems. This approach is based on architecture of an executable Action Plan Machine (APM) that arranges planned actions into a DFD network of active nodes. This architecture made the plan recognition process as well as student advising possibly automated. SC intrudes with the student work to offer more appropriate solution strategies. SC reports graphical depictions of both expert's and student's solution strategies for more powerful and effective coaching. SG research is the most inspiring for this IGS research.

On the other hand, many researches focused on supporting the grading process by providing human graders with supportive tools. For instance, Eftimie et al. [10] had designed Korect that is a solution for automatic test generation, processing and grading, with inline answer marks, and answer sheet processing using auto-feed scanners and OCR detection with an objective of optimizing the process workflow. Wu et al [11] designed a computer-aided grading support system for spoken English tests. Answer data are divided into numerous independent data cells. Human graders are then automatically allocated by the system to do grading jobs and independently report grading results to the system. Rules are defined to obtain a final result. Bloomfield [12] designed a system that allows digital grading, by a human, of traditional paper-based exams or homework assignments, which is intended to reduce the grading time due to the automation of: flipping to the correct page, summing up the pages, recording the grades, returning the exams, etc.

Furthermore, other researchers worked on semi-automating the grading. For instance, Juedes [13] presented a semi-automated Web-based grading system for data structure courses in which the student is requested to write Java codes that are then assessed partially automatic through running the code with predefined test cases and with providing aids for the human grader for evaluating the program design and documentation.

On the other hand, other researchers focused on student assessment rather than test grading. For instance, Zhang [14] designed an indicator system for evaluating college student performance based on student characteristics. They used the analytic hierarchy process (AHP) and case based reasoning (CBR). Antal and Koncz [15] discussed the need for computer-based test systems according to the student model, and hence, proposed a method for the graphical representation of student knowledge.

7 Conclusions and Future Work

This research introduced the new concept of Intelligent Grading Systems (IGS). IGS monitors students, analyzes their steps in solving problems, predicts and evaluates their action plans, and gives them advices on correct or more optimum solutions. In contrast with ITS systems, IGS does not only evaluate students based on their final responses to quizzes and exercises but also on the plans they had considered in reaching those responses. IGS's feedback considers course objectives, e.g., a student might reach a problem solution correctly; however, the grader might not accept

his/her followed approach, as it doesn't match the taught concepts that were the subject of the current training being tested.

Accordingly, this research have studied and identified the characteristics of IGS as opposed to other active support systems. Achieved are models for testing, questions, and grading, which led to the design of a framework for IGS systems. Finally a prototype (Smart Grader—SG) was then implemented. SG is supposed to integrate to ITS systems to provide more effective learning through grading student tests, correcting mistakes, and providing, on the spot, advices on better solution strategies. A List programming problem was used for experimentation.

In general, the results achieved by the Smart Grader research project were promising and encouraging. However, further investigations are still required along several directions: first, assessing SG's practical viability through measuring the cost and time efficiency of the whole process of creating a question bank, preparing a test, and holding a testing session, and hence, improving the framework design; second, adding accumulative experience and machine learning component to the framework so as to detect new correct answers and hence enlarging the knowledgebase; third, adding dynamic adaptation of the Grader's behavior; finally, Implementing a more comprehensive environment with practical considerations, e.g., using the mobile and cooperative agent technology to give more flexibility and power.

SCIENCE AND TECHNOLOGY PUBLICATIONS

References

1. Corbett and H. Trask, "Instructional Interventions in Computer-Based Tutoring: Differential Impact on Learning Time and Accuracy", in the ACM CHI Letter, the Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems, Vol. 2, issue 1, April 2000.
2. Corbett, J. Anderson, "Locus of Feedback Control on Computer-Based Tutoring: Impact on Learning Rate, Achievement, and Attitudes", Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2001.
3. Y. Shang, H. Shi and S. Chen, "An Intelligent Distributed Environment for Active Learning", Journal of Education Resources in Computing, ACM, Vol. 1, No. 2, Summer 2001.
4. W. Hefley, "Helping Users Help Themselves", IEEE Software magazine, Vol. 11, No. 2, March/April 1995.
5. G. Knight, D. Kilis, and P. Cheng, "An Architecture for an integrated Active Help System", in the Proceedings of the 1997 ACM Symposium on Applied Computing, 1997.
6. C. Gutierrez and J. Hidalgo, "Suggesting What to Do Next", in the Proceedings of the 1988 ACM SIGSMALL/PC Symposium on ACTES, 1988.
7. J. Davies, A. Gertner, N. Lesh, C. Rich, C. Sidner, and J. Rickel, "Incorporating Tutorial Strategies Into an Intelligent Assistant", in the Proceedings of the 6th International Conference on Intelligent User Interfaces, 2001.
8. H. Shah and A. Kumar, "A Tutoring System for Parameter Passing in Programming Languages", in the Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education, 2002.
9. Shehab Gamalel-Din, "Intelligent Active Coaching – An Executable Plan Approach", AUEJ, Vol. 5, No. 4, Sept. 2002.
10. Eftimie, M. Bardac, R. Rughinis, "Optimizing the workflow for automatic quiz evaluation", in the Proceedings of the 10th Roedunet International Conference (RoEduNet), 2011, pp.1-4.

11. Yijian Wu; Wenyun Zhao; Xin Peng; Yunjiao Xue, “A Computer Aided Grading System for Subjective Tests”, In the proceedings of the International Conference on Internet and Web Applications and Services/Advanced, AICT-ICIW '06, 2006, pp.2.
12. A. Bloomfield, “Evolution of a digital paper exam grading system”, Frontiers in Education Conference (FIE), IEEE, 2010, pp. T1G-1 - T1G-6.
13. D.W. Juedes, “Experiences in Web-based grading”, 33rd Annual Frontiers in Education, IEEE, 2003, Vol. 3, pp. S3F: 27-32.
14. Yucai Zhang, “Research on intelligent college student evaluation system based on CBR and AHP”, In the Proceedings of IEEE International Conference on Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008, Vol. 1, pp. 39 - 43
15. Margit Antal, Szabolcs Koncz, “Student modeling for a web-based self-assessment system”, Expert Systems with Applications, Vol. 38, Issue 6, June 2011, Pages 6492-6497.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS