122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Paper ID #11861

Work in Progress: Intelligent Tutoring Systems in Computer Science and Software Engineering Education

Dr. John C Nesbit, Simon Fraser University

John Nesbit is a professor in the Faculty of Education at Simon Fraser University, Canada, where he conducts research on argumentation and learning, multimedia learning, self-regulated learning and the effectiveness of intelligent tutoring systems. He has collaborated with colleagues to publish over 60 peer-reviewed journal articles, book chapters and indexed conference papers.

Li Liu, Simon Fraser University

Li Liu is a second year doctoral student in Educational Technology and Learning Design at Simon Fraser University. With an interdisciplinary background in interaction design, media arts and education, her passion lies in exploring how innovative technologies can be harnessed to promote teaching and learning.

Qing Liu, Simon Fraser University

Qing Liu is a doctoral student in Educational Technology and Learning Design at Simon Fraser University. Her research focuses on conceptual change, the potential of learning by arguing, the role of need for cognition in learning, the effectiveness of intelligent tutoring systems, and meta-analysis of empirical studies.

Dr. Olusola O Adesope, Washington State University-Pullman

Dr. Olusola O. Adesope is an Assistant Professor of Educational Psychology at Washington State University, Pullman. His research is at the intersection of educational psychology, learning sciences, and instructional design and technology. His recent research focuses on the cognitive and pedagogical underpinnings of learning with computer-based multimedia resources; knowledge representation through interactive concept maps; meta-analysis of empirical research, and investigation of instructional principles and assessments in STEM.

Work in Progress: Intelligent Tutoring Systems in Computer Science and Software Engineering Education

John C. Nesbit, Arita Liu, Qing Liu
Simon Fraser University

Olusola O. Adesope
Washington State University

Abstract

Many research reports have been published over the last 30 years on the use of intelligent tutoring systems in computer science and software engineering education, but no previous systematic review has been conducted to describe and assess the field as a whole. This project (in progress) searched for publications meeting defined inclusion criteria and identified 280 eligible reports. We are currently coding these works using 28 variables that will allow us to describe the research field in aggregate. The results will tell us: What research questions are being asked? What are the types of student modeling being used? What subject domains have ITS been designed for? What issues or themes are most evident in recent research? What are the gaps in research on intelligent tutoring systems in computer science and software engineering education. Finally, what technological and pedagogical innovations are needed to advance research in this field?

Research on intelligent tutoring systems (ITS) has accelerated over the last decade, and scholarly interest in such systems has never been greater.[1] ITS have been developed for a wide range of subject domains (e.g., mathematics, physics, biology, medicine, reading, languages, and philosophy) and for students in primary, secondary and postsecondary levels of education. Although most ITS have been developed by researchers and never deployed outside the laboratory or the single university-level course for which they were designed, there are examples of mature systems that have been deployed more widely and extensively evaluated.[2, 3]

Like previous reviewers[1, 4, 5] we have adopted a definition of ITS that emphasizes student modeling as an essential characteristic. We identify an ITS as any computer system that performs teaching or tutoring functions (e.g., selecting assignments, asking questions, giving hints, evaluating responses, providing feedback, prompting reflection, providing comments that boost student interest) and adapts or personalizes those functions by modeling students' cognitive, motivational or emotional states. This definition distinguishes ITS from test-and-branch tutorial systems which individualize instruction by matching a student's most recent response against preprogrammed, question-specific targets. Complicating matters, there are sophisticated

computerized adaptive testing systems, not usually considered to be ITS, that use item response theory to model student ability as a single dimension.[6] To distinguish ITS from such systems we further specify that student modeling must be multidimensional.

Quantitative and Meta-Analytic Reviews on the Effectiveness of ITS

The first quantitative review which compared the instructional effectiveness of ITS to other types of instruction was an analysis published in 2011 by VanLehn that examined learning outcomes in STEM.[7] VanLehn was primarily interested in distinguishing between human tutoring and three types of computer-based tutoring systems: answer-based, step-based, and substep-based. Answer-based systems typically assign a problem and then provide feedback and further branching that depends on the student's answer. Conventional test-and-branch, computer-based instruction programs are usually classified as answer-based systems. Step-based and substep-based systems have finer-grained interfaces that provide instructional support as the student progresses through the solution of a problem. Most ITS that teach students procedures for solving problems in areas such as math, physics and chemistry would be classified as step-based or substep-based systems. VanLehn found very little difference in post-test performance between students learning from human tutoring and step-based or substep-based systems; and, based on a small number of primary studies, he found that answer-based systems tend to produce poorer posttest performance than step-based systems (.40 *SD*) and substep-based systems (.32 *SD*).

Steenbergen-Hu and Cooper published two methodologically rigorous meta-analyses on the effects of using ITS.[8, 9] In a 2013 review covering primary and secondary mathematics they found no significant difference between ITS and other modes of instruction when measured by standardized tests; but when measured by course-specific tests designed by teachers or researchers, there was a small, statistically significant effect (g = .19) favoring ITS. In 2014, they published a second meta-analysis that examined ITS learning outcomes in postsecondary education. They found that, overall, ITS significantly outperformed other modes of instruction (g = .35). They also found that human tutoring produced only slightly better results than ITS (g = -.25), and the difference was not statistically significant.

More recently, we co-authored the first comprehensive meta-analysis on the effectiveness of ITS.[1] It included all available studies prior to 2012 that compared ITS to other types of instruction. We analyzed 107 effect sizes comparing learning outcomes from ITS against other types of instruction and found a statistically significant, overall weighted mean effect size favoring ITS of approximately g = .40. Similar effect sizes were found when ITS were compared specifically to textbooks, large teacher-led classes, and non-ITS computer-based instruction. However, no significant differences in learning outcomes were found when ITS were compared to one-to-one tutoring and small group instruction. ITS were found to be significantly more effective than other types of instruction at all levels of schooling (elementary, secondary, and postsecondary) and in most subject domains, including computer science (approximately, g = .50).

To summarize, the recent quantitative reviews indicate that research has found ITS to be more effective than other types of instruction except one-to-one and small group instruction provided

by a human. For unknown reasons, this pattern of advantage for ITS was not found in the specific case of mathematics at the elementary and secondary levels.

Meta-Analysis of ITS in Computer Science and Software Engineering Education

We recently reported a meta-analysis of 22 effect sizes that compared ITS to other types of instruction in the subject domain of computer science and software engineering (CS/SE) education.[10] Although an overall effect size associated with ITS in the domain had been already established[1] our meta-analysis sought to examine how the effect of using ITS breaks out by moderator variables such as type of student modeling and whether the ITS modeled misconceptions.

The studies that met our inclusion criteria were published from 1998 to 2013. Although a few of the better-known student modeling techniques were represented in our sample, most of the ITS tested in the primary research used 'one-off' student model designs that appeared in only a single evaluative study. We found that learning outcomes were significantly higher for students using ITS than those learning in large teacher-led classes (g = .67) or from non-ITS, computer-based instruction (g = .89). ITS were associated with better learning outcomes when they were used as the principle means of instruction and also when they served an assistive or supplementary function. ITS were more effective than other types of instruction when they modeled misconceptions (g = .41) and when they did not (g = .68).

Purpose of the Systematic Review

Meta-analysis is appropriate for assessing an intervention that is identifiable *prior* to the searching and coding stages of the review process. Because meta-analysis focuses on comparison of pre-identified treatments it is not suitable for a broader examination of the whole body of research on a topic such as the use of ITS in CS/SE education. As it required quite specific inclusion criteria, our meta-analysis excluded all but 21 research publications, which we observed to be a minor fraction of all research on the topic. We are now conducting a systematic review to discover the significant features of research in the field. The review is addressing questions such as:

- What subject areas within CS/SE education (e.g., programming, database design) have been taught by ITS?
- What instructional functions (e.g., task assignment and sequencing, hints, feedback, motivational prompts) are adapted by the systems?
- What types of student models are being designed and used?
- What instructional strategies are the ITS based on?
- What interface features are being used in the ITS?

- What research questions are being investigated?
- What are the most recent research trends?
- In what ways might the research be advanced or improved?

The systematic review is critically evaluating research on the use of ITS in software engineering education and will make recommendations for improving the quality of methods and reporting in primary studies.

Method

We completed a search of five major bibliographic databases (IEEE, SpringerLink, Web of Science, PsycInfo, ERIC) using the following search expression:

> ("computer science education" OR "software engineering education" OR "computer literacy" OR "database design" OR "network security" OR "introductory computer" OR "introductory programming" OR "teach* programming" OR "learn* programming") AND ("intelligent tutor*" OR "adaptive tutor*" OR "cognitive tutor*")

The 1085 unique reports returned by this search were augmented by 72 reports obtained by consulting reference sections of review papers and conducting informal searches. We adopted the following inclusion criteria:

1. The paper must deal in a substantial way with an ITS.
2. The ITS must be intended for use or substantially evaluated in the curricular domain of software engineering education, computer science education, computer literacy, or cognate areas.
3. The paper must be a review, a system design report, a system evaluation report, or an empirical evaluation.

We read the titles and abstracts of the 1157 reports and excluded all but 325, which were downloaded so their full text could be examined. After examining the full texts, 280 reports were retained for the systematic review.

In developing an initial set of variables and codes, especially those which describe the characteristics of the ITS studied, we were guided by previous reviews and meta-analyses. For example, a recent review[11] attempted to classify ITS developed for learning programming into six tutoring approaches (example-based, simulation-based, dialogue-based, program-based, feedback-based, collaboration-based). These categories informed the development of the *instructional strategy* variable shown in Table 1.

A codebook was developed by a two-phase, formative coding process that was seeded with the initial set of 28 variables. There were two main goals for the formative coding process. The first

goal was to develop a final set of variables and codes that represented features of the studies matching our research questions and which could be efficiently and reliably applied to extract those features. The second goal was to train two researchers (Liu and Liu) to code the studies in a mutually consistent and accurate manner.

In the first phase, the two coders independently coded six randomly selected papers and then compared the results. The two coders met to resolve all differences that did not require changes in the tentative set of variables and codes. They subsequently met with a third researcher (Nesbit) to discuss coding problems and opportunities that indicated changes to the codebook. At that meeting decisions were taken to clarify, delete, add or revise variables and codes. The most common changes resulted from ambiguity in the descriptions of the initial variables and codes. Other changes were made so that the variables captured more of the relevant features of the ITS and research described in the primary studies. For example, the coders noticed the prevalence of pedagogical agents in the ITS described in the primary research and a decision was taken to add a pedagogical agent variable to the codebook. In the second phase of formative coding, another six papers were randomly selected and independently coded by the two researchers. A comparison found that the only disagreements between the coders centered on two variables: research validity and treatment fidelity. The discrepancies were mainly attributed to poor reporting of research methods in the primary studies, especially in brief conference proceedings. Most of the sampled reports provided insufficient information on the research methods to allow reliable and consistent coding of research validity and treatment fidelity. Consequently those variables were removed from the codebook. After adding and deleting several variables, the final codebook consisted of 28 variables and their associated codes.

The two phases of formative coding played an important role in (a) selecting, shaping and clarifying the variables and codes in the codebook, and (b) preparing the reviewers to independently code the primary research with a high degree of reliability. Table 1 shows a sample of the variables and codes in the final version of the codebook.

Table 1. A Sample of the Variables and Codes used in the Systematic Review

Research Type
- Design proposal
- Empirical evaluation
- Review
- Other

Student Model Type
- Model tracing only
- Knowledge tracing
- Constraint-based modeling
- Bayesian network modeling
- Expectation and misconception tailoring
- Open learner modeling
- Other

Instructional Strategy
- Example-based
- Program visualization
- Program analysis
- Natural language dialogue
- Collaborative learning
- Problem-oriented

Adapted Instructional Functions
- Feedback
- Hint
- Task selection
- Task sequencing
- Dialogue summarization
- Agent facial expression
- Other

Results

Although few results are available now, we were able to complete four of the more easily coded variables. Figure 1 shows the distribution of publication dates for the included studies. It demonstrates continued growth of scholarly interest in the use of ITS in CS/SE education.

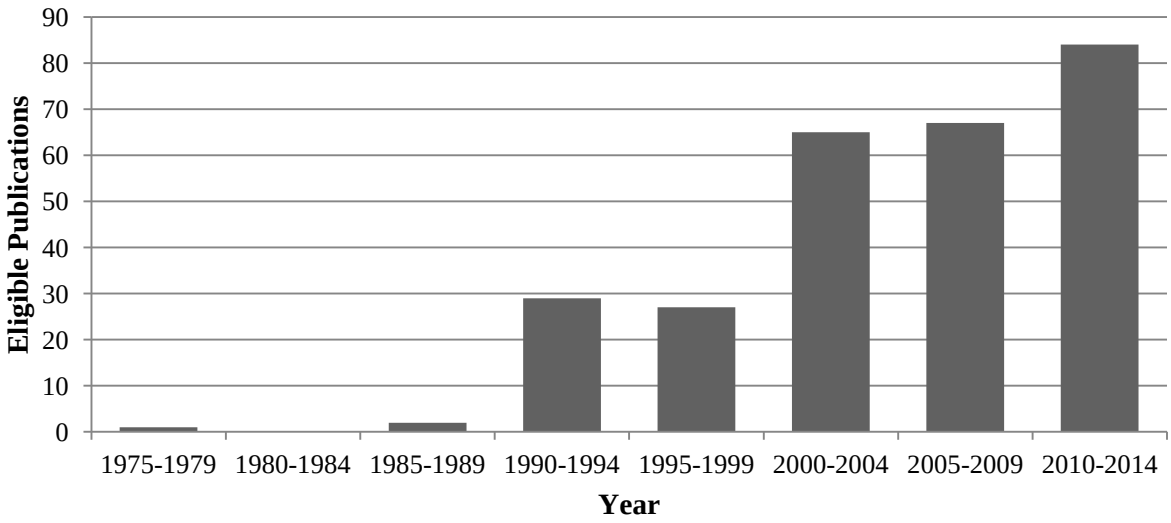

Figure 1. The number of published research reports on ITS in computer science and software engineering education (in 5-year periods).

Table 2 shows the number of research publications broken out by publication type, educational level and subject domain. We observed that ITS used for teaching programming dealt with a variety of programming languages including Java, PHP, Python, Ada, C, C++, LISP, and Prolog.

Note that under subject domain, the categories computer science and software engineering refer to general curricula in those subjects.

Table 2. Number of Publications by Publication Type, Education Level, and Subject Domain

Publication type	
Proceedings	198
Journal article	66
Book chapter	16
Dissertation	1
Education level	
Postsecondary	190
Postsec/Secondary	8
Postsec/Sec/Primary	1
Secondary	5
Not reported	72
Not applicable	1
Other	3
Subject domain	
Programming	175
Computer literacy	26
Database design	41
Computer science	15
Network security	5
Software engineering	6
Artificial intelligence	3
Other	9

What We Know So Far

Research has established that intelligent tutoring systems can be effective tools for learning that compare favorably with other types of instruction. There is evidence based on a few studies that for some learning goals they may be as effective as small-group human tutoring and nearly as effective as one-to-one human tutoring. ITS seem to be particularly effective in CS/SE education when compared with large-teacher-led classes and non-ITS computer-based instruction. We speculate that relative to some of the other subject domains in which ITS have been evaluated, CS/SE education has a higher proportion of procedural learning goals and a lower proportion of conceptual learning goals. Programming is largely a cognitive skill, and learning to program requires problem solving practice. Compared to large group instruction, ITS provide more individualized task assignment and feedback that may increase the opportunity to learn from practice. Compared with non-ITS computer-based instruction, ITS provide feedback at a more granular level – at the level of the problem solving step rather than the problem answer. This difference in level of interaction has been theorized to account for much of the ITS advantage in teaching procedural cognitive skills.[3]

Our systematic review has found that research on the use of ITS in CS/SE education has grown significantly over the last 30 years and, like ITS research generally, the field has never been more active. Most of the publications are brief proceedings papers, indicating that most of the research is being done by researchers from computer science and software engineering where proceedings are the most common publication format, rather than from education where journal articles are more common. There is far more research being conducted with curricula at the postsecondary level, and most of it is focused on teaching programming. It may be that researchers in computer science and engineering departments find it most convenient to develop ITS for courses they themselves teach, or perhaps they seek to develop systems that would have the most value within their own academic departments.

In the final version of this review we will report on the characteristics of the ITS themselves, including the types of instructional strategies, the types of student models, and the instructional functions that are individualized. We will also report on the research questions that are driving empirical and design-oriented studies in the field.

Recent Trends and Themes

An examination of the research published over the last two years suggests several emphasized themes. Although these themes are not necessarily new to the area, they are currently attracting considerable attention from researchers.

Some recently developed systems combine ITS and gaming.[13, 14, 15] In ITS designed to teach programming, gaming often takes the form of a simulation-based instructional strategy in which students are given puzzles to solve in a virtual environment. For example, in BOTS,[14] students program a robot to move blocks into specified positions in a virtual environment. The ITS acts as a coach by generating hints to help students improve their performance.

One current research trend involves tracking and modeling students' affective state to inform and individualize instructional interactions.[16, 17, 18] The interest in emotional modeling for CS/SE education reflects a wider trend in the ITS research community. Much of the research is still working out how to model emotion from one or more data sources and does not attempt to incorporate the model in an ITS. For example, Grafsgaard et al. tracked student posture, gesture and skin conductance during human-to-human, computer-mediated tutorial dialogues about Java programming. They found that students' shifts in posture and gesturing were associated with particular types of dialogical moves by the tutor (e.g., positive feedback).

Pedagogical agents are anthropomorphic characters in educational software that are usually represented by static or animated avatars and are used to deliver notifications, messages and tutorial dialogues. We found that systems combining ITS with pedagogical tutors, which first appeared about 17 years ago, continue to be improved and evaluated by researchers in CS/SE education.[12, 13, 19, 20] The accumulated evidence indicates that pedagogical agents are associated with small, positive effects on learning,[21] but that the content of instructional messages is far more important than whether the messages are presented by an anthropomorphic figure.[20] Researchers in CS/SE education are exploring whether virtual agents can help students interpret,

cognitively organize, and appropriately attend to the information ITS provide. For example, ITS that provide many types of messages or notifications may be able to improve pedagogical utility by assigning each broad category of message to a different pedagogical agent.[19]

Hints are strategic dialogical moves that assist learners in answering a question or solving a problem. The related problems of automatically generating hints and determining appropriate conditions under which to present them have received considerable attention from the ITS research community. We found substantial recent research on hints for ITS in CS/SE education.[14, 19, 22, 23] Interestingly, there is recognition that the conditions for providing hints include the motivational state of the student because de-motivated students are inclined to request hints immediately rather than expend effort in problem solving.[22] The problem of automatically generating hints is quite specific to ITS that coach beginning programmers. In generating hints for the BOTS environment, for example, researchers found that they were able to generate hints more easily by analyzing the state of the world resulting from the program that moved the blocks (i.e., the positions of blocks) than by analyzing the program itself.[14]

Recommendations

Even at this incomplete stage of data gathering and analysis, we have observed several inter-related barriers to advancement of ITS in CS/SE education. First, there may be insufficient attention given to coordination with and replication of work being conducted by other groups. We believe the field would benefit from greater effort to build on the work of others. Second, we believe the research could be accelerated by establishing a standard, open-source ITS platform for teaching introductory programming. When researchers conduct design research on a single aspect of an ITS, say a pedagogical agent, an open-source platform would allow lower development costs because the researchers would likely only need to develop the re-designed pedagogical agent component and not the entire system. Also, evaluation and comparison of alternate designs by different researchers would be more feasible because they would be variations on the same base system. Finally, we were able to locate very few review articles about ITS in CS/SE education. Review articles are crucial for (a) consolidating theories and methods that guide further primary research, (b) critically evaluating the state of research and recommending improvements, and (c) identifying neglected topics that require the attention of researchers. Our completed systematic review will contribute in each of these three areas.

Bibliography

1. Ma, W., Adesope, O. O., Nesbit, J. C., & Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analytic survey. *Journal of Educational Psychology, 106*, 901-918.

2. Sabo, K. E., Atkinson, R. K., Barrus, A. L., Joseph, S. S., & Perez, R. S. (2013). Searching for the two sigma advantage: Evaluating algebra intelligent tutors. *Computers in Human Behavior*, *29*, 1833-1840. doi:10.1016/j.chb.2013.03.001

3. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Tracy, D., Weinstein, A., & Wintersgill, M. (2005). The Andes Physics Tutoring System: Lessons Learned. *International Journal of Artificial Intelligence in Education, 15*, 147-204.

4. Shute, V. J., & Psotka. J. (1996). Intelligent tutoring systems: Past, present, and future. In D. Jonassen (Ed.), *Handbook of Research for Educational Communications and Technology* (pp. 570-600). New York, NY: Macmillan.

5. Sottilare, R., Graesser, A., Hu, X., & Holden, H. (Eds.) (2013). *Design Recommendations for Intelligent Tutoring Systems.* Orlando, FL: U.S. Army Research Laboratory.

6. Kuo, C., & Wu, H. (2013). Toward an integrated model for designing assessment systems: An analysis of the current status of computer-based assessments in science. *Computers & Education, 68*, 388-403. doi:10.1016/j.compedu.2013.06.002

7. VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist, 46*, 197-221.

8. Steenbergen-Hu, S., & Cooper, H. (2013, September 9). A meta-analysis of the effectiveness of Intelligent Tutoring Systems on K-12 students' mathematical learning. *Journal of Educational Psychology*. Advance online publication. doi:10.1037/a0032447

9. Steenbergen-Hu, S., & Cooper, H. (2014). A meta-analysis of the effectiveness of Intelligent Tutoring Systems (ITS) on college students' academic learning. *Journal of Educational Psychology, 106*, 331-347. doi:10.1037/a0034752

10. Nesbit, J. C., Adesope, O. O., Liu, Q., & Ma, W. (2014). How effective are intelligent tutoring systems in computer science education? IEEE 14th International Conference on Advanced Learning Technologies (ICALT) (pp. 99-103). Athens, Greece, July 7-10.

11. Le, N. T., Strickroth, S., Gross, S., & Pinkwart, N. (2013). A review of AI-supported tutoring approaches for learning programming. In N. T. Nguyen, T. van Do, and H. A. le Thi (Eds.), *Advanced Computational Methods for Knowledge Engineering: Studies in Computational Intelligence 479* (pp. 267-279). Heidelberg: Springer International Publishing

12. Graesser, A. C., Moreno, K., Marineau, J., Adcock, A., Olney, A., & Person, N. (2003). AutoTutor improves deep learning of computer literacy: Is it the dialog or the talking head? In U. Hoppe, F. Verdejo, and J. Kay (Eds.), *Proceedings of Artificial Intelligence in Education* (pp. 47-54). Amsterdam: IOS Press.

13. Ogar, O., Shabalina, O., Davtyan, A., & Kizim, A. (2014). Mastering programming skills with the use of adaptive learning games. In A. Kravets, M. Shcherbakov, M. Kultsova., and T. Iijima (Ed.), *Proceedings of the 11th Joint Conference on Knowledge-Based Software Engineering Communications in Computer and Information Science 466* (pp. 144-455). Volgograd, Russia: Springer International Publishing.

14. Hicks, A., Peddycord III, B., & Barnes, T. (2014). Building games to learn from their players: Generating hints in a serious game. In S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia (Ed.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems: Lecture Notes in Computer Science 8474* (pp.312-317). Honolulu, HI, USA: Springer International Publishing.

15. Xie, T., Tillmann, N., & de Halleux, J. (2013). Educational software engineering: Where software engineering, education and gaming meet. In *Proceedings of the 3rd International Workshop on Games and Software Engineering* (pp. 36-39), San Francisco, CA.

16. Bosch, N., Chen, Y., & D'Mello, S. (2014). It's written on your face: Detecting affective states from facial expressions while learning computer programming. In S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia (Ed.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems: Lecture Notes in Computer Science 8474* (pp.39-44). Honolulu, HI, USA: Springer International Publishing.

17. Grafsgaard, J. F., Wiggins, J. B., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2013). Embodied affect in tutorial dialogue: Student gesture and posture. In H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik (Ed.), *Proceedings of the 16th International Conference on Artificial Intelligence in Education: Lecture Notes in Computer Science 7926* (pp. 1-10). Memphis, TN, USA: Springer Berlin Heidelberg.

18. Guia, T. F. G., Rodrigo, M. M. T., Dagami, M. M. C., Sugay, J. O., Macam, F. J. P., & Metrovic, A. (2012). Modeling the affective states of students using SQL-Tutor. In S. A. Cerri, W. J. Clancey, G. Papadourakis, and K. Panourgia (Ed.), *Proceedings of the 11th International Conference on Intelligent Tutoring Systems: Lecture Notes in Computer Science 7315* (pp. 634-635). Chania, Crete, Greece: Springer Berlin Heidelberg.

19. Ivanović, M., Mitrović, D., Budimac, Z., Vesin, B., & Jerinić, L. (2014). Different roles of agents in personalized programming learning environment. In D. K. W. Chiu, M. Wang, E. Popescu, Q. Li, and R. Lau (Ed.), *New Horizons in Web Based Learning: Lecture Notes in Computer Science 7697* (pp. 161-170). Hong Kong, China: Springer Berlin Heidelberg.

20. Nye, B. D., Graesser, A. C., & Hu, X. (2014). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education, 24* (4), 427-469.

21. Schroeder, N. L., Adesope, O. O., & Gilbert, R. B. (2013). How effective are pedagogical agents for learning? A meta-analytic review. *Journal of Educational Computing Research, 49*, 1-39.

22. Verginis, I., Gouli, E., Gogoulou, A., & Grigoriadou, M. (2011). Guiding learners into reengagement through the SCALE environment: An empirical study. *IEEE Transactions on Learning Technologies, 4* (3), 275-290.

23. Gerdes, A., Heeren, B., & Jeuring, J. (2012). Teachers and students in charge: Using annotated model solutions in a functional programming tutor. In A. Ravenscroft, S. Lindstaedt, C. D. Kloos, and D. Hernández-Leo (Ed.), *Proceedings of the 7th European Conference of Technology Enhanced Learning: 21st Century Learning for 21st Century Skills, Lecture Notes in Computer Science 7563* (pp. 383-388). Saarbrücken, Germany: Springer Berlin Heidelberg.