The Diagnosing Behaviour of Intelligent Tutoring Systems

Renate van der Bent[1], Johan Jeuring[1,2][0000−0001−5645−7681], and Bastiaan Heeren[2][0000−0001−6647−6130]

[1] Department of Information and Computing Sciences, Universiteit Utrecht
[2] Faculty of Management, Science & Technology, Open University of the Netherlands
J.T.Jeuring@uu.nl Bastiaan.Heeren@ou.nl

Abstract. Intelligent Tutoring Systems (ITSs) determine the quality of student responses by means of a diagnostic process, and use this information for providing feedback and determining a student's progress. This paper studies how ITSs diagnose student responses. In a systematic literature review we compare the diagnostic processes of 40 ITSs in various domains. We investigate what kinds of diagnoses are performed and how they are obtained, and how the processes compare across domains. The analysis identifies eight aspects that ITSs diagnose: correctness, difference, redundancy, type of error, common error, order, preference, and time. All ITSs diagnose correctness of a step. Mathematics tutors diagnose common errors more often than programming tutors, and programming tutors diagnose type of error more often than mathematics tutors. We discuss a general model for representing diagnostic processes.

Keywords: Intelligent Tutoring Systems · Diagnosis · Feedback.

1 Introduction

More than a decade ago, VanLehn published his paper on the behaviour of Intelligent Tutoring Systems (ITSs) [60]. An ITS consists of an outer loop, which serves tasks to a student matching her progress, and an inner loop, which gives a student feedback and hints about steps she takes towards solving a task. Completing a task in an ITS often requires multiple steps, where "a step is a user interface action that the student takes in order to achieve a task" [60]. An important responsibility of the inner loop is what VanLehn calls step analysis.

Diagnosing student steps is essential for determining progress, and for giving feedback and hints. Feedback and hints are important factors supporting learning [28]. How do different ITSs diagnose a student step? We perform a systematic literature review of available step-based ITSs to classify the diagnostic processes of these systems. We determine the various components that play a role in diagnosing student steps, and study how these components are combined to perform a full diagnosis. Furthermore, we compare the diagnoses of ITSs from different domains (such as mathematics, programming, and physics), and ITSs using different approaches (such as constraint-based tutoring [50], model tracing [5], example tracing [43], and intention-based tutoring [39]). The results of our study inform the design of ITSs, and might in the future be combined with results from effectiveness studies [61] to get a better understanding of what kind of diagnostic processes are likely to be more effective.

The research question we address in this paper is: How do ITSs determine the quality of student responses? To answer this question, we will look at the aspects that can be distinguished in the diagnosis of a student response, how these aspects are combined in various ITSs, and if there are patterns or perhaps even a general scheme that can be identified in the diagnostic processes of the different ITSs. The contributions of this paper are:

- we distinguish eight aspects that are used in various tutors in diagnosing a student step;
- we describe patterns in combining these aspects;
- we compare how diagnosing differs between domains and tutoring approaches.

This paper is organised as follows. Section 2 discusses related work. Section 3 describes the research method, and the resulting diagnostic aspects and processes are presented in Section 4. Section 5 concludes.

2 Related work

We are not aware of research on comparing diagnostic processes of ITSs across various domains and using different tutoring approaches. In the 1970s and later, research focused on diagnosing a particular aspect of students' work, namely misconceptions [14]. Diagnosing misconceptions requires collecting and checking for buggy rules, which sometimes leads to overwhelming and impractical numbers of buggy rules, even for simple domains such as fractions [31]. Modern approaches, such as algorithmic debugging [68], automatically distinguish buggy rules. Heeren and Jeuring present an advanced diagnose service, which is used in ITSs for mathematics, logic, and programming [29].

Diagnosis of student steps has been studied extensively in ITSs and assessment systems for mathematics, such as Stack [54] and ActiveMath [24]. El-Kechaï et al. [21] evaluate the diagnosing behaviour of PépiMep, a diagnosis system for algebra that is part of a web-based mathematics platform. This system can distinguish 13 different patterns in student responses. Chanier et al. [16] review how errors are analysed in several ITSs for second-language learning. More related work on diagnosing student steps is described later in this paper.

3 Research method

For our review we selected papers describing an ITS that is capable of providing feedback at the level of individual steps and that has been used in classrooms, or tested on data from real students. These inclusion criteria ensure that the ITS has an inner loop with a step analysis, and ensure ecological validity, i.e. that the ITS makes realistic diagnoses.

We searched for relevant papers in two ways. First, we considered systems discussed in three relevant reviews. Keuning et al. [41] classify the types of feedback given in programming tutors. Specifically, we included papers describing systems that are labelled as providing feedback on task-processing steps, because these papers are assumed to meet the first two criteria. VanLehn's review on the effectiveness of tutoring systems [61] classifies systems as answer-based, step-based, or substep-based. The step-based and substep-based systems satisfy our inclusion criteria. Finally, Cheung and Slavin's review [17] discusses the effectiveness of educational software in mathematics. From these reviews, we included 14, 17, and 0 papers, respectively (i.e. 31 papers in total). The papers in Cheung and Slavin's review [17] did not meet the inclusion criteria, or lacked a description of the system's working.

Second, we searched for papers using a literature search. A preliminary search in several search engines (Google Scholar, Scopus, and ERIC) revealed that Scopus produces the most relevant search results. See Van der Bent's thesis [12] for different search terms and the resulting number of papers. We judged relevance of papers by reading the abstract and, when necessary, by skimming through the article. The search term that produced the most relevant results was

```
intelligent AND (tutoring OR tutor) AND systems
  AND ((step AND based) OR stepwise)
```

in Scopus, giving 195 papers. Using the same terms in ERIC resulted in fewer papers, largely a subset of documents found in Scopus. Searching in Google Scholar resulted in many more, but less relevant, papers. The papers found in Scopus were also found in Google Scholar. Hence, we used the 195 documents found in Scopus. Note that using the search term `(step AND based) OR stepwise` may have resulted in finding fewer papers from less-structured domains.

Next, we checked this initial selection of papers for the inclusion criteria. The first author read the abstracts. If the information in the abstract was insufficient to determine whether a system meets all criteria, she read the full paper. If this did not result in a decision, the second author read the paper, and discussed the paper's relevance with the first author. The literature search resulted in 16 more papers that meet the inclusion criteria.

We categorized the ITSs described in the selected papers by their tutoring approach (model tracing, example tracing, constraint-based, or intention-based: Aleven et al. [2] explain the differences between the first three of these paradigms) and by domain. Then, starting with a small subset of papers (around 10), we iteratively designed a system for labelling the diagnostic processes and diagnosed aspects. With this labelling system we categorized the rest of the selected ITSs.

After labelling the diagnostic processes, we checked whether there are any noticeable differences between approaches or domains, by comparing the frequency at which aspects are diagnosed per approach and per domain. We also described the diagnostic processes in diagrams and tried to abstract a general model from the labelling system.

4 Diagnostic aspects and processes

We found 47 papers on 40 ITSs that satisfy our inclusion criteria. Table 1 gives an overview of the ITSs, including references, domain, and tutoring approach. We found 26 model tracing tutors, 8 example tracing tutors, 11 constraint-based tutors, and 1 intention-based tutor. An ITS can make use of multiple approaches, for example, Andes [63] and Mathtutor [1] use constraints in combination with the example tracing approach. We could not determine the approach used by the Technical Troubleshooting tutor [38].

Subsection 4.1 describes the diagnostic aspects we found based on a small sample of papers, which we used to label the rest of the ITSs. Subsection 4.2 describes the frequency of aspects per approach and domain. Subsection 4.3 discusses models representing the diagnostic processes of some tutoring systems, followed by a general model for diagnostic processes in Subsection 4.4.

4.1 Diagnostic aspects

We found that ITSs use the following aspects to diagnose a student step: correctness, difference, redundancy, type of error, common error, order, preference, and time. We explain and illustrate these aspects below. Whenever relevant, the running example will be the following algebra problem: "Solve for x: $5x+6 = 7x$".

Correctness (C) determines whether or not a student step matches an expected step, or does not violate any constraint. Possible outcomes are *correct* and *incorrect*. For instance, if a student submits $2x+6 = 0$, this step is diagnosed as incorrect because it does not match the expected next step $5x-7x+6 = 0$. The equation $5x+6-7x = 0$ is considered correct because it is semantically equivalent to the expected answer.

Difference (D) is similar to correctness, in that it determines whether or not a step matches an expected step. The result is a measure such as a number or percentage that indicates the edit distance between the student step and an expected step. When the difference is zero, the step is correct. For example, if we use the edit distance, the above incorrect response results in a difference value of 1, since it requires one edit operation (replace "$+$" by "$-$") to change the incorrect step into the expected step.

Redundancy (R) refers to a superfluous step: this includes steps that are too small to be recognized as a meaningful step. Possible outcomes are *redundant*, *not redundant*, and *unknown*. For example, the rewrite step from $5x-7x+6 = 0$ into $-7x+5x+6 = 0$ can be considered redundant.

Type of Error (ToE) refers to a classification of errors. Possible outcomes differ per problem domain or ITS. For example, $5x-(7x+6 = 0$ can be classified as a syntax error.

Common Errors (CE) or buggy rules are misconceptions that a student may have. Possible outcomes differ per problem domain or ITS. An example of a buggy rule is forgetting to change the sign when moving an expression from one side of the equation to the other side, for instance, rewriting an expression of the form $5x+6 = 7x$ into $5x+6+7x = 0$.

ITS	**Domain and approach**		C	D	R	ToE	CE	O	P	T
(Why2-)Atlas [62]	Qualitative physics	mt	●			●	●			
(Why2-)Autotutor [26, 25]	Physics & Computer literacy	mt	●	●			●			
ACT Programming Tutor [18]	Programming	mt	●				●			
AITS [27]	Search algorithms	ex	●	●	●	●				
Andes [63]	Physics	mt,cb	●		●	●			●	
ANGLE [44]	Geometry	mt	●				●			
APROPOS2 [49]	Prolog programming	ex	●	●	●	●	●	●	●	
Ask-Elle [34]	Haskell programming	mt,cb	●			●	●		●	
Assistment [51]	Mathematics	mt	●				●			
AzAR 3.0 [20]	Foreign language pronunciation	ex	●	●		●	●			
CIMEL ITS [13]	OO design and programming	mt	●			●				
CIRCSIM-TUTOR [42, 23]	Circulatory physiology	mt	●			●		●		
C-Tutor [57]	C programming	ib	●			●				
Design-A-Plant [46, 47]	Botany	cb	●							
Dragoon [66]	Dynamic systems	ex	●		●	●				
ELM-ART [64]	LISP programming	mt	●			●				
Geometry Explanation [4, 3]	Geometry	mt	●			●	●			
Geomtery Tutor [6]	Geometry	mt	●				●			
HBPS [9, 10]	Algebra word problems	mt	●			●	●			
Hong04 [32]	Prolog programming	mt	●			●				
iList [22]	Computer Science	cb	●			●	●			
ITAP [52]	Python programming	ex	●			●				
Jin12 [35]	Programming	ex	●			●				
Jin14 [36]	Programming	ex	●			●				
JITS [59]	Java programming	mt	●			●				
KERMIT [58]	Database design	cb	●			●				
Keuning14 [40]	Imperative programming	mt	●		●	●			●	
Mathesis [55, 56]	Algebra	mt	●			●	●	●		
Mathtutor [1]	Mathematics	ex,cb	●				●	●		
Ms. Lindquist [30]	Algebra word problems	mt	●			●	●			
Newton's Pen [45]	Statics	mt,cb	●			●				
PAT2Math [33]	Algebra	mt	●				●			
PHP ITS [65]	PHP programming	cb	●		●	●				
PLATO [15]	Arithmetic	cb	●			●	●		●	
Quantum Accounting [37]	Accounting	mt	●			●				
RMT [11]	Psychology research methods	mt	●	●						
Technical Troubleshooting [38]	Aircraft engineering	mt,cb?	●							
The Invention Lab [53]	Scientific inquiry	mt,cb	●			●	●			
The LISP Tutor [19, 7]	LISP programming	mt	●				●			
Zatarain-Cabada13 [67]	Arithmetic	mt	●							●

Table 1. Overview of the 40 systems with their domain, tutoring approach (mt: model tracing, ex: example tracing, cb: constraint-based, ib: intention-based), and diagnosed aspects; the eight aspects are correctness (C), difference (D), redundancy (R), type of error (ToE), common errors (CE), order (O), preference (P), and time (T)

Order (O) refers to the order in which a student takes steps. Possible outcomes are *correct order*, *incorrect order*, and *unknown*. Note that this is a diagnosis over multiple steps.

Preference (P): some solutions may be preferable over others. Possible outcomes are *preferred*, *not preferred*, and *unknown*. For instance, in a programming tutor, a particular algorithm may produce the correct result, but be less efficient than the preferred algorithm. A teacher can express a preference for pedagogical reasons, if she wants students to use a particular approach rather than another.

Time (T) refers to the time a student takes to submit a step or solve a problem, measured in (milli)seconds. This aspect was only labelled when time was used for diagnostic purposes. While many systems measure time, only few use it for diagnostic purposes.

Table 1 gives an overview of the diagnosed aspects per ITS.

Of the eight aspects that ITSs diagnose, correctness is the most common aspect, and is used in all systems. Most other aspects depend on its outcome. For example, type of error relies on correctness, because errors can only be found in steps that are known to be incorrect. Likewise, preference also depends on correctness, because it can only determine preference between correct steps. Aside from correctness, the most commonly diagnosed aspects are the type of error and common errors.

Only one ITS [67] diagnoses time with the assumption that the time it takes to answer a question reflects the difficulty of the question. Why are other ITSs not diagnosing time? Most ITSs can be accessed at home, without supervision. This makes it difficult to monitor how much time is actually spent on answering a question. For example, a student might take a long time to answer because she is taking a break or doing something else. Perhaps this is why most ITSs do not use time for diagnosis.

4.2 Diagnostic aspects per approach and domain

We distinguish four ITS approaches: model tracing (mt), example tracing (ex), constraint-based (cb), and intention-based (ib). There is some overlap between these categories: five ITSs combine model tracing and the constraint-based approach, and one ITS (Mathtutor) uses example tracing and the constraint-based approach. Only one ITS uses the intention-based approach. Table 2 (left-hand side) shows the frequency of the occurrence of aspects in the various ITS approaches. The results do not show very different patterns for the approaches.

The ITSs we study deal with tasks in a large variety of problem domains. At an abstract level, we can group them into four domains: mathematics, programming, physics, and other domains. Mathematics includes topics such as algebra, arithmetic, and geometry. Programming includes programming in specific languages, and more general topics such as object-oriented design and data structures. Physics includes qualitative physics and statics. The remaining ITSs involve topics such as botany, foreign language pronunciation, database design,

	approach				domain			
Aspect	mt	ex	cb	ib	math	progr	physics	other
Correctness	26	8	11	1	11	15	4	12
Type of Error	16	7	8	1	5	13	3	8
Common Errors	14	3	5		10	5	2	4
Preference	3	1	3		1	3	1	
Difference	2	3	1			1	1	4
Order	2	2	1		2	1		1
Redundancy	1	3	1			3		2
Time	1				1			

Table 2. Frequency of diagnostic aspects per tutoring approach and problem domain, both in absolute numbers and their relative frequency of occurrence (as bars)

and aircraft engineering. The domains partially overlap. (Why2-)Autotutor is in both the physics and 'other domains' category, because it teaches both physics and computer literacy. iList is in both the programming and 'other domains' category, because it teaches students about lists, which is an important data structure in programming, but not programming per se. Table 2 (right-hand side) also shows the frequency of the occurrence of aspects in the various ITS domains.

Table 2 shows that ITSs in the domain of mathematics more often diagnose common errors than ITSs in the other domains: 91% of the math tutors diagnose common errors, compared to only 33% of programming tutors, 50% of physics tutors, and 33% of the tutors in other domains. In mathematics, problems typically have a single correct solution, and there are only a few ways to reach that solution. Many errors in student steps can be explained by buggy rules, also because the solution space is relatively small. This partially explains why common errors are relatively often diagnosed in ITSs for mathematics.

In the domain of programming, ITSs diagnose the type of error more often than in other domains: 87% of the programming tutors diagnose the type of error, compared to 46% of the mathematics tutors, 75% of the physics tutors, and 67% of the tutors in other domains. This is perhaps due to the solution space in the domains. In programming tutors, the solution space is usually very large, which makes diagnosing common errors infeasible. Programs may have errors on different levels: syntax, dependency, typing, semantics, and more. This makes type of error a more informative diagnosis than in situations where only syntax and semantics play a role, as is usual in mathematics.

Redundancy is diagnosed in three programming tutors and two other domain tutors, but not in any mathematics or physics tutor. Because of the small sample size, we did not perform a statistical test to determine the significance of these results. The rest of the aspects seem to be diagnosed at a lower frequency across domains.

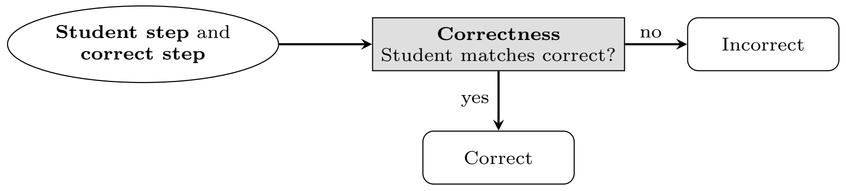

Fig. 1. Diagnostic process of Assistment, Design-a-Plant, and Quantum Accounting

4.3 Diagnostic processes

Most ITSs use multiple aspects for diagnosing student responses. How are these aspects combined in a diagnosis? We discuss how the different aspects are combined by the different ITSs to arrive at a diagnosis, and what the commonalities are between these systems. Not all ITSs are covered here, because some papers do not provide enough detail to extract the precise diagnosing process.

Figure 1 shows the most basic diagnostic process. Ovals represent input, grey nodes represent diagnostic ITS components, and rounded rectangles represent a diagnosis. This diagram represents the diagnostic processes in Assistment, Design-a-Plant, and Quantum Accounting. A student step is checked against a single good step. If it matches, the response is correct; if not, the response is incorrect. Although Assistment and Quantum Accounting have an additional diagnostic aspect, namely type of error, this is not shown in the diagram, because it is unclear where the type of error is determined. RMT's diagnostic process is very similar, except that it uses cosine similarity to check whether a step matches an expected step.

The basic diagram in Figure 1 can be extended in several ways. The diagnostic processes of the ACT Programming Tutor, LISP Tutor, Geometry Tutor, and PAT2Math add a second diagnostic component (i.e. a grey block) after correctness has determined that the student step does not match a good step. In this second component, common errors are searched for by using a set of buggy rules. Dragoon, on the other hand, extends the diagram with a diagnostic component that determines redundancy before checking correctness.

We give a single example of a more involved diagnostic process, and refer the reader to Van der Bent's thesis [12] for many more diagrammatic representations of diagnostic processes that were found in ITSs.

The diagnostic process of AITS is illustrated in Figure 2. AITS calculates the difference using edit distance. This information is used to infer correctness. If the edit distance is zero, the node sequence is correct. Otherwise, AITS checks the number of nodes and the content of the nodes in the submitted answer, and uses this to determine redundancy and type of error: AITS treats redundancy as one type of error. The *complete* and *accurate* diagnoses are labelled as types of errors. In AITS, a type of error is a combination of completeness and accuracy, so a step can be *complete but inaccurate*, *incomplete but accurate*, or *incomplete and inaccurate*. The diagnosis *complete and accurate* never occurs since then the edit distance would be zero, and the step would have been diagnosed as correct.

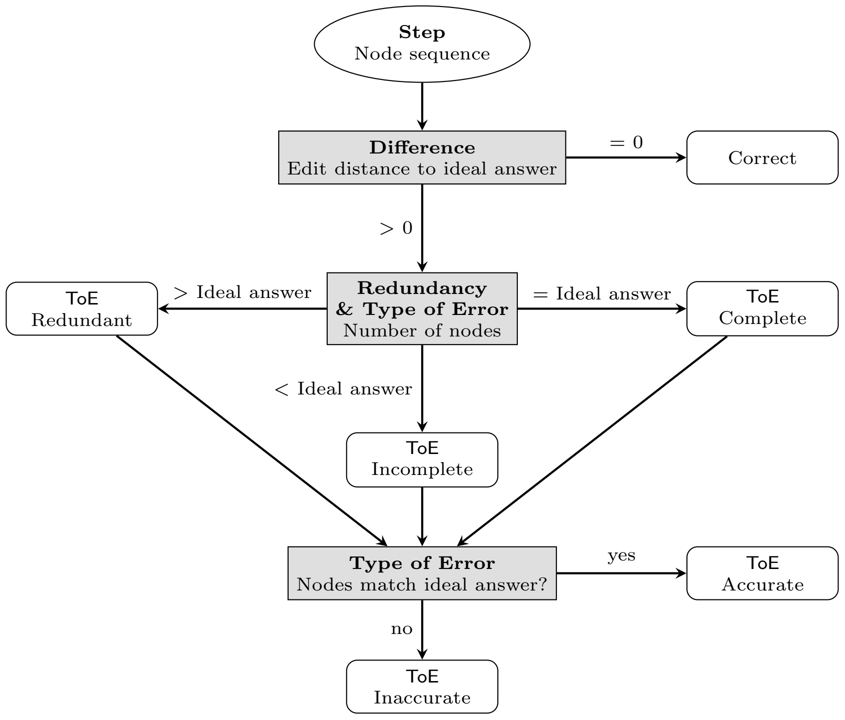

Fig. 2. Diagnostic process of AITS

4.4 Patterns in diagnostic processes

Figure 3 illustrates the general diagnostic process. A dashed border indicates that the components are optional. All tutors check whether a step is correct using correctness or difference. Before this is done, however, some tutors check the order of steps or how much time was taken to submit a step. After it has been determined that a step is correct, some tutors check whether the correct step is also a preferred step. Some tutors also check whether a correct step is redundant. For incorrect steps, some tutors check whether the step contains common errors, and what type of error was made. Lastly, some tutors check whether an incorrect step is redundant. Note that, as was mentioned before, some tutors consider redundancy as an error, while others treat it as correct.

Some ITSs make more fine-grained diagnoses than the ones discussed in this study [8, 48]. For example, in Arends' ITS [8] expressions can be semantically equivalent after an incorrect step. To signal such a step, the system can diagnose expressions that are semantically equivalent while also following a buggy rule, or expressions that are expected by a strategy despite not being semantically equivalent. Since these types of diagnoses only appear in this particular ITS, and seem to be very particular to the domain, we did not include them in our research.

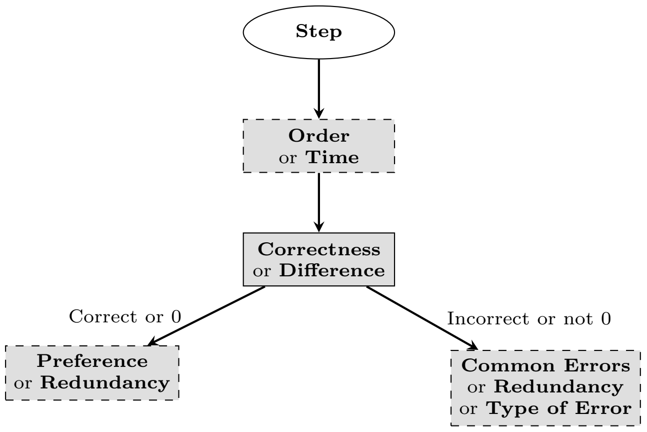

Fig. 3. General diagnostic process

5 Conclusion

As an answer to our research question, we found eight diagnostic aspects of student responses in Intelligent Tutoring Systems: correctness, difference, redundancy, type of error, common error, order, preference, and time. The diagnostic aspects are combined in various ways in the full diagnoses of the ITSs. Although these processes vary widely between systems, we distilled a general, abstract process that is used in all ITSs. All ITSs diagnose correctness, and although there are differences between domains, common errors and the type of error are also often diagnosed. The main difference between domains is that common errors is the second most frequently diagnosed aspect in mathematics tutors, whereas type of error is the second most frequently diagnosed aspect in programming tutors. Our analysis found no difference between four common tutoring approaches.

A limitation of our work is that the analysis of diagnostic processes is based on the information given in the papers written about the ITSs, rather than on the source code of the ITSs. Not all papers provide an in-depth description of how student steps are diagnosed, which made it impossible to describe the diagnostic processes of some systems. Sometimes we had to interpret the text to determine the diagnostic process.

Our analysis of diagnostic processes in ITSs contributes to a better understanding of the diagnosing behaviour of ITSs. For future research, the results of this study could be combined with results from evaluations of the effectiveness of tutoring systems [61]. This would give insight into which diagnostic processes are most effective at improving learning. This insight could then inform the design and development of tutoring systems in the future.

Acknowledgements. The authors would like to thank the anonymous reviewers and the members of the Utrecht reading club on educational technology for their helpful suggestions.

References

1. Aleven, V., McLaren, B.M., Sewall, J.: Scaling up programming by demonstration for intelligent tutoring systems development: An open-access web site for middle school mathematics learning. IEEE Transactions on Learning Technologies **2**(2), 64–78 (2009)
2. Aleven, V., Mclaren, B.M., Sewall, J., Koedinger, K.R.: A new paradigm for intelligent tutoring systems: Example-tracing tutors. International Journal on Artificial Intelligence in Education **19**(2), 105–154 (2009)
3. Aleven, V., Popescu, O., Koedinger, K.: Pilot-testing a tutorial dialogue system that supports self-explanation. In: International Conference on Intelligent Tutoring Systems. pp. 344–354. Springer (2002)
4. Aleven, V., Popescu, O., Koedinger, K.R.: Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. In: Proceedings of Artificial Intelligence in Education. pp. 246–255. Citeseer (2001)
5. Anderson, J.R., Boyle, C.F., Reiser, B.J.: Intelligent tutoring systems. Science(Washington) **228**(4698), 456–462 (1985)
6. Anderson, J.R., Boyle, C.F., Yost, G.: The geometry tutor. In: IJCAI. pp. 1–7 (1985)
7. Anderson, J.R., Reiser, B.J.: The LISP tutor. Byte **10**, 159–175 (1985)
8. Arends, H., Keuning, H., Heeren, B., Jeuring, J.: An intelligent tutor to learn the evaluation of microcontroller I/O programming expressions. In: Proceedings of the 17th Koli Calling Conference on Computing Education Research. pp. 2–9. ACM (2017)
9. Arevalillo-Herráez, M., Arnau, D., Marco-Giménez, L.: Domain-specific knowledge representation and inference engine for an intelligent tutoring system. Knowledge-Based Systems **49**, 97–105 (2013)
10. Arnau, D., Arevalillo-Herráez, M., Puig, L., González-Calero, J.A.: Fundamentals of the design and the operation of an intelligent tutoring system for the learning of the arithmetical and algebraic way of solving word problems. Computers & Education **63**, 119–130 (2013)
11. Arnott, E., Hastings, P., Allbritton, D.: Research methods tutor: Evaluation of a dialogue-based tutoring system in the classroom. Behavior Research Methods **40**(3), 694–698 (2008)
12. van der Bent, R.: The Diagnosing Behaviour of Intelligent Tutoring Systems. Master's thesis, Universiteit Utrecht (2018)
13. Blank, G., Parvez, S., Wei, F., Moritz, S.: A web-based ITS for OO design. In: Proceedings of Workshop on Adaptive Systems for Web-based Education at 12th International Conference on Artificial Intelligence in Education (AIED'2005). Amsterdam, the Netherlands. pp. 59–64 (2005)
14. Brown, J.S., Burton, R.R.: Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science **2**(2), 155–192 (1978)
15. Burton, R.R., Brown, J.S.: A tutoring and student modelling paradigm for gaming environments. ACM SIGCUE Outlook **10**(SI), 236–246 (1976)
16. Chanier, T., Pengelly, M., Twidale, M., Self, J.: Conceptual modelling in error analysis in computer-assisted language learning systems. In: Intelligent tutoring systems for foreign language learning, pp. 125–150. Springer (1992)
17. Cheung, A.C., Slavin, R.E.: The effectiveness of educational technology applications for enhancing mathematics achievement in k-12 classrooms: A meta-analysis. Educational research review **9**, 88–113 (2013)

18. Corbett, A.T., Anderson, J.R.: Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 245–252. ACM (2001)
19. Corbett, A.T., Anderson, J.R., Patterson, E.G.: Student modeling and tutoring flexibility in the Lisp intelligent tutoring system. Intelligent tutoring systems: At the crossroads of artificial intelligence and education pp. 83–106 (1990)
20. Demenko, G., Wagner, A., Cylwik, N.: The use of speech technology in foreign language pronunciation training. Archives of Acoustics **35**(3), 309–329 (2010)
21. El-Kechaï, N., Delozanne, É., Prévit, D., Grugeon, B., Chenevotot, F.: Evaluating the performance of a diagnosis system in school algebra. In: International Conference on Web-Based Learning. pp. 263–272. Springer (2011)
22. Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, C., Chen, L.: Data driven automatic feedback generation in the ilist intelligent tutoring system. Technology, Instruction, Cognition and Learning **10**(1), 5–26 (2015)
23. Glass, M.: Some phenomena handled by the circsim-tutor version 3 input understander. In: Proceedings of the Tenth Florida Artificial Intelligence Research Symposium, Daytona Beach. pp. 21–25 (1997)
24. Goguadze, G., Melis, E.: Combining evaluative and generative diagnosis in activemath. In: Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling. pp. 668–670 (2009)
25. Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H.H., Ventura, M., Olney, A., Louwerse, M.M.: Autotutor: A tutor with dialogue in natural language. Behavior Research Methods, Instruments, and Computers **36**(2), 180–192 (2004)
26. Graesser, A.C., Wiemer-Hastings, P., Wiemer-Hastings, K., Harter, D., Tutoring Research Group, T.R.G., Person, N.: Using latent semantic analysis to evaluate the contributions of students in autotutor. Interactive learning environments **8**(2), 129–147 (2000)
27. Grivokostopoulou, F., Perikos, I., Hatzilygeroudis, I.: An educational system for learning search algorithms and automatically assessing student performance. International Journal of Artificial Intelligence in Education **27**(1), 207–240 (2017)
28. Hattie, J., Timperley, H.: The power of feedback. Review of Educational Research **77**(1), 81–112 (2007)
29. Heeren, B., Jeuring, J.: Feedback services for stepwise exercises. Science of Computer Programming **88**, 110–129 (2014)
30. Heffernan, N.T., Koedinger, K.R.: An intelligent tutoring system incorporating a model of an experienced human tutor. In: International Conference on Intelligent Tutoring Systems. pp. 596–608. Springer (2002)
31. Hennecke, M.: Online Diagnose in intelligenten mathematischen Lehr-Lern-Systemen (in German). Ph.D. thesis, Hildesheim University (1999)
32. Hong, J.: Guided programming and automated error analysis in an intelligent prolog tutor. International Journal of Human-Computer Studies **61**(4), 505–534 (2004)
33. Jaques, P.A., Seffrin, H., Rubi, G., de Morais, F., Ghilardi, C., Bittencourt, I.I., Isotani, S.: Rule-based expert systems to support step-by-step guidance in algebraic problem solving: The case of the tutor pat2math. Expert Systems with Applications **40**(14), 5456–5465 (2013)
34. Jeuring, J., Gerdes, A., Heeren, B.: A programming tutor for Haskell. In: Central European Functional Programming School, pp. 1–45. Springer (2012)

35. Jin, W., Barnes, T., Stamper, J., Eagle, M.J., Johnson, M.W., Lehmann, L.: Program representation for automatic hint generation for a data-driven novice programming tutor. In: International Conference on Intelligent Tutoring Systems. pp. 304–309. Springer (2012)
36. Jin, W., Corbett, A., Lloyd, W., Baumstark, L., Rolka, C.: Evaluation of guided-planning and assisted-coding with task relevant dynamic hinting. In: International Conference on Intelligent Tutoring Systems. pp. 318–328. Springer (2014)
37. Johnson, B.G., Phillips, F., Chase, L.G.: An intelligent tutoring system for the accounting cycle: Enhancing textbook homework with artificial intelligence. Journal of Accounting Education **27**(1), 30–39 (2009)
38. Johnson, S.D., et al.: Application of cognitive theory to the design, development, and implementation of a computer-based troubleshooting tutor. (1992)
39. Johnson, W.L.: Intention-based diagnosis of novice programming errors. Morgan Kaufmann (1986)
40. Keuning, H., Heeren, B., Jeuring, J.: Strategy-based feedback in a programming tutor. In: Proceedings of the Computer Science Education Research Conference. pp. 43–54. ACM (2014)
41. Keuning, H., Jeuring, J., Heeren, B.: A systematic literature review of automated feedback generation for programming exercises. ACM Trans. Comput. Educ. **19**(1), 3:1–3:43 (Sep 2018)
42. Kim, N., Evens, M., Michael, J.A., Rovick, A.A.: Circsim-tutor: An intelligent tutoring system for circulatory physiology. In: International Conference on Computer Assisted Learning. pp. 254–266. Springer (1989)
43. Koedinger, K.R., Aleven, V., Heffernan, N., McLaren, B., Hockenberry, M.: Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In: International Conference on Intelligent Tutoring Systems. pp. 162–174. Springer (2004)
44. Koedinger, K.R., Anderson, J.R.: Reifying implicit planning in geometry: Guidelines for model-based intelligent. Computers as cognitive tools p. 15 (2013)
45. Lee, W., de Silva, R., Peterson, E.J., Calfee, R.C., Stahovich, T.F.: Newton's pen: A pen-based tutoring system for statics. Computers & Graphics **32**(5), 511–524 (2008)
46. Lester, J.C., Stone, B.A., O'Leary, M.A., Stevenson, R.B.: Focusing problem solving in design-centered learning environments. In: International Conference on Intelligent Tutoring Systems. pp. 475–483. Springer (1996)
47. Lester, J.C., Stone, B.A., Stelling, G.D.: Lifelike pedagogical agents for mixed-initiative problem solving in constructivist learning environments. User modeling and user-adapted interaction **9**(1-2), 1–44 (1999)
48. Lodder, J., Heeren, B., Jeuring, J.: A domain reasoner for propositional logic. Journal of Universal Computer Science **22**(8), 1097–1122 (2016)
49. Looi, C.K.: Automatic debugging of prolog programs in a prolog intelligent tutoring system. Instructional Science **20**(2-3), 215–263 (1991)
50. Mitrovic, A., Suraweera, P., Martin, B.: Intelligent tutors for all: The constraint-based approach. IEEE Intelligent Systems **22**(4), 38–45 (2007)
51. Razzaq, L.M., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Koedinger, K.R., Junker, B., Ritter, S., Knight, A., Mercado, E., Turner, T.E., et al.: Blending assessment and instructional assisting. In: AIED. pp. 555–562 (2005)
52. Rivers, K., Koedinger, K.R.: Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. International Journal of Artificial Intelligence in Education **27**(1), 37–64 (2017)

53. Roll, I., Aleven, V., Koedinger, K.R.: The invention lab: Using a hybrid of model tracing and constraint-based modeling to offer intelligent support in inquiry environments. In: International Conference on Intelligent Tutoring Systems. pp. 115–124. Springer (2010)
54. Sangwin, C.: Computer Aided Assessment of Mathematics. Oxford University Press (2013)
55. Sklavakis, D., Refanidis, I.: An individualized web-based algebra tutor based on dynamic deep model tracing. In: Hellenic Conference on Artificial Intelligence. pp. 389–394. Springer (2008)
56. Sklavakis, D., Refanidis, I.: Mathesis: An intelligent web-based algebra tutoring school. International Journal of Artificial Intelligence in Education **22**(4), 191–218 (2013)
57. Song, J., Hahn, S., Tak, K., Kim, J.: An intelligent tutoring system for introductory C language course. Computers & Education **28**(2), 93–102 (1997)
58. Suraweera, P., Mitrovic, A.: Kermit: A constraint-based tutor for database modeling. In: International Conference on Intelligent Tutoring Systems. pp. 377–387. Springer (2002)
59. Sykes, E.R.: Design, development and evaluation of the Java intelligent tutoring system. Technology, Instruction, Cognition & Learning **8**(1) (2010)
60. VanLehn, K.: The behavior of tutoring systems. International journal of artificial intelligence in education **16**(3), 227–265 (2006)
61. VanLehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist **46**(4), 197–221 (2011)
62. VanLehn, K., Jordan, P.W., Rose, C.P., Bhembe, D., Böttner, M., Gaydos, A., Makatchev, M., Pappuswamy, U., Ringenberg, M., Roque, A., Siler, S., Srivastava, R.: The architecture of why2-atlas: A coach for qualitative physics essay writing. In: International Conference on Intelligent Tutoring Systems. pp. 158–167. Springer (2002)
63. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M.: The Andes physics tutoring system: Lessons learned. International Journal of Artificial Intelligence in Education **15**(3), 147–204 (2005)
64. Weber, G., Brusilovsky, P.: Elm-art: An adaptive versatile system for web-based instruction. International Journal of Artificial Intelligence in Education (IJAIED) **12**, 351–384 (2001)
65. Weragama, D., Reye, J.: Analysing student programs in the php intelligent tutoring system. International Journal of Artificial Intelligence in Education **24**(2), 162–188 (2014)
66. Wetzel, J., VanLehn, K., Butler, D., Chaudhari, P., Desai, A., Feng, J., Grover, S., Joiner, R., Kong-Sivert, M., Patade, V., et al.: The design and development of the dragoon intelligent tutoring system for model construction: lessons learned. Interactive Learning Environments **25**(3), 361–381 (2017)
67. Zatarain-Cabada, R., Barrón-Estrada, M.L., Pérez, Y.H., Reyes-García, C.A.: Designing and implementing affective and intelligent tutoring systems in a learning social network. In: Mexican International Conference on Artificial Intelligence. pp. 444–455. Springer (2012)
68. Zinn, C.: Algorithmic debugging to support cognitive diagnosis in tutoring systems. In: Proceedings KI 2011: Advances in Artificial Intelligence. LNCS, vol. 7006, pp. 357–368 (2011)