Int J Artif Intell Educ (2014) 24:33–61
DOI 10.1007/s40593-013-0001-9

RESEARCH ARTICLE

Adaptive Intelligent Support to Improve Peer Tutoring in Algebra

Erin Walker · Nikol Rummel · Kenneth R. Koedinger

Published online: 18 October 2013

Abstract Adaptive collaborative learning support (ACLS) involves collaborative learning environments that adapt their characteristics, and sometimes provide intelligent hints and feedback, to improve individual students' collaborative interactions. ACLS often involves a system that can automatically assess student dialogue, model effective and ineffective collaboration, and provide relevant support. While there is evidence that ACLS can improve student learning, little is known about why systems that incorporate ACLS are effective. Does relevant support improve student interactions by providing just-in-time feedback, or do students who believe they are receiving relevant support feel more accountable for the collaboration, and thus more motivated to improve their interactions? In this paper, we describe an adaptive system we have developed to support help-giving during peer tutoring in high school algebra: the Adaptive Peer Tutoring Assistant (APTA). To validate our approach, we conducted a controlled study that demonstrated that our system provided students with more relevant support and was more effective at improving student learning than parallel nonadaptive conditions. Our contributions involve generalizable techniques for implementing ACLS that can function adaptively and effectively, and the finding that adaptive support does indeed improve student learning because of the relevance of the support.

Keywords Intelligent tutoring · Computer-supported collaborative learning · Adaptive collaborative learning support · Peer tutoring

E. Walker (✉)
School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, USA
e-mail: erin.a.walker@asu.edu

N. Rummel
Institute of Educational Research, Ruhr-Universität Bochum, Bochum, Germany
e-mail: nikol.rummel@rub.de

K. R. Koedinger
Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, USA
e-mail: koedinger@cmu.edu

Introduction

The ability to computationally model human problem-solving has had a profoundly positive impact on the advancement of effective educational technologies. In intelligent tutoring systems (ITSs), the system develops a model of student knowledge by comparing students' problem-solving steps to ideal performance, and then uses that model to give students individualized support such as hints, feedback, and problem selection (VanLehn 2006). These systems have been very successful at supporting individual learners in problem solving, often approaching the effectiveness of expert human tutoring (VanLehn 2011). For example, the Cognitive Tutor Algebra, an intelligent tutoring system for high school algebra, has been demonstrated to improve classroom learning by one standard deviation over traditional instruction (Koedinger et al. 1997). Recently there has been a movement in ITS research to take a more holistic view of the learning process and develop "tutors that care" (du Boulay et al. 2010), with a goal of modeling and supporting higher-order skills like metacognition, motivation, and social interaction (e.g., Roll et al. 2011; Muldner et al. 2010; Ogan et al. 2011).

Our focus within this movement is on modeling and supporting collaborative learning. An important aspect of learning is the social construction of knowledge, where students exchange ideas, reflect on their own misconceptions, and come to a shared understanding through dialogue with their peers (Schoenfeld 1992). Collaborative activities in the classroom can facilitate this kind of learning, but only if students are engaging in particular interactions, such as giving help when their partners need it (Johnson and Johnson 1990). As students do not engage in these interactions spontaneously, researchers and practitioners attempt to improve student collaboration through interventions that structure or script the collaboration, providing students with roles and activities to follow (Fischer et al. 2007; Kollar et al. 2006). For example, in King's reciprocal peer tutoring script, students take turns teaching a partner, and are prompted to ask their partner specific questions at increasing levels of depth (King et al. 1998). These collaboration scripts tend to be fixed, in that they do not adapt to individual or group needs. They may overconstrain collaboration for good collaborators, provide insufficient support for poor collaborators, and there is no guarantee that students will use them as designed (Kollar et al. 2005; Dillenbourg 2002; Lazonder et al. 2003).

In adaptive collaborative learning support (ACLS), collaborative learning environments adapt their characteristics, and sometimes provide intelligent hints and feedback, to improve students' collaborative interactions (Magnisalis et al. 2011). In theory, ACLS systems are more effective than nonadaptive collaborative learning systems because they can tailor the support they provide to students' needs and abilities (Rummel and Weinberger 2008). In this paper, we describe an ACLS system we developed to support peer tutoring in high school algebra: The Adaptive Peer Tutoring Assistant (APTA; Walker et al. 2011). APTA is based on the literal equation solving unit of the Cognitive Tutor Algebra (CTA), a successful individual intelligent tutoring system in high school algebra (Koedinger et al. 1997). In APTA, one student (the *tutee*) solves problems like "Solve for x; ", and a second student (the *peer tutor*) marks the problem steps right or wrong and discusses the problems with the tutee in a chat window. Our system provides adaptive support to peer tutors in order to help

them give more correct and more effective help. In this paper, we focus on the mechanisms we used to support the peer tutor in giving more effective help (a *help-giving tutor*). In constructing our system, we made a technological contribution by developing techniques to handle the uncertainty inherent in assessing, modeling, and supporting collaborative dialogue. We made a theoretical contribution by evaluating APTA to investigate the impact of adaptive support on learning from collaboration, improving understanding of when and why adaptive systems are effective.

Implementation of ACLS

Soller et al. (2005), in their review of systems that support collaboration, differentiate between mirroring systems that reflect the state of students' collaboration back to the students, metacognitive tools that provide information about the current and desired state of the interaction, and guiding systems that propose remedial actions in response to perceived collaborative weaknesses. Within guiding systems (what we are referring to as *ACLS*), Magnisalis, Demetriadis, and Karakostas (2011) make a further distinction between systems that focus on group formation, domain-specific support, or peer interaction support. For the purposes of this paper, our primary interest is in guiding systems that provide peer interaction support by analyzing student collaborative dialogue as it occurs and intervening when appropriate (e.g., Israel and Aiken 2007; Vieira et al. 2004).

Dialogue is a highly important element of learning from peer tutoring, and the target of the help-giving support implemented in APTA. The simplest way of supporting collaborative dialogue is to focus on metrics that do not rely on the content of the dialogue. Several systems count individuals' utterances as a way of generating a participation score, and then encourage those who are not participating to participate more (e.g., Vizcaíno et al. 2000; Rosatelli and Self 2004; Vieira et al. 2004; Kumar et al. 2007). For example, if a student does not participate in a chat window for more than 50 % of the estimated time the group will be working on a step, *LeCS* prompts the student to participate in the discussion by saying, "Would you like to participate in the discussion?" (Rosatelli and Self 2004). Another set of guiding systems facilitates dialogue between collaborating students by using sentence openers or classifiers. Students are asked to classify their own utterances, and then the sequence of labelled utterances is compared to a model of ideal dialogue in order to provide students with support (McManus and Aiken 1995; Baker and Lund 2003). For example, GroupLeader (Israel and Aiken 2007) asks students to select from one of 46 sentence starters in order to label their utterance. Based on the starter selected, the system identifies the dialog action students were taking (e.g., "Let's negotiate this…" represents the negotiation dialogue act), and students could then be encouraged to make more of a specific type of contribution (Barros and Verdejo 2000). Particular sequences of dialogue actions can also relate to student behaviors or beliefs that can serve as targets for support. For example, Tedesco (2003) uses sentence starters such as "I disagree" and other forms of highly constrained input to detect conflicts within a group and prompt group members to reflect and elaborate on their opinions. Unfortunately, students do not consistently select sentence starters that match the content of their statements, and therefore the inferences the system makes can be inaccurate (Israel and Aiken 2007; Lazonder et al. 2003). In recent years,

machine learning technology has been used to supplement sentence classifiers and content-free metrics by automatically classifying student contributions along a growing variety of dimensions (Rosé et al. 2008), and using features of the learning environment such as the way discussion contributions are linked to improve the impact of the machine learning (Mclaren et al. 2010). For example, Dragon, Florian, Woolf, and Murray (2010) have demonstrated that they can automatically classify the topic of student discussion. Recently, these classifiers have become effective at detecting relevant aspects of collaborative dialogue such as authority (Mayfield and Rosé 2011), with the goal of using these assessments as triggers for support.

Like an individual ITS, ACLS involves three phases: assessing student interactions, modeling ideal interactions, and using a comparison between the two to provide tailored support (Soller et al. 2005). Unlike many traditional ITSs, ACLS must deal with ambiguity at each phase, sharing commonalities with the growing body of work on ITSs in ill-defined domains (Mitrovic and Weerasinghe 2009). With regards to *assessment*, much of the benefits of collaborative learning come from the dialogue between collaborating students, but automatically assessing relevant aspects of student dialogue is not a solved problem. Regarding *modeling* (i.e., creating a model of ideal performance), while we know which collaborative interactions are broadly related to learning, we know less about under what specific circumstances these positive interactions should be employed. Therefore, when creating a model of collaboration it is difficult to specify what interactions should occur at what moments during the collaboration. Finally, in terms of *support*, little research has been done on how to provide adaptive support based on uncertain information delivered in the assessment and modeling phases. A single collaborative action or state is unlikely to be unilaterally correct or incorrect, and students have many options when choosing their next action. Traditional ITSs are ill-equipped to deal with this kind of ambiguity. It is difficult to build an adaptive system that can produce support relevant to student interaction, and thus ACLS systems are rarely sufficiently developed such that they are able to be deployed in a natural context.

Magnisalis and colleagues (2011) introduce several necessary research directions in their review of the state of the art of ACLS, including improving the assessment of student collaboration, deepening the interaction analysis of the collaboration, and understanding the design space for providing support. In our system, APTA, we advance ACLS by using multiple channels of information (automatic classifications, self-classifications, and domain information) to assess student dialogue. We then use a combination of production rule modeling, constraint-based modeling, and knowledge tracing to assess student peer tutoring skills. The feedback we give based on these assessments takes into account the uncertainty inherent in modeling collaboration, giving the peer tutor the benefit of the doubt with respect to many of their collaborative actions.

Effects of ACLS

Once ACLS is implemented, it tends to be an effective way of supporting collaboration. COLLECT-UML, for example, adaptively supports students in collaborating on the design of UML class diagrams, and has been shown to lead to greater knowledge of collaboration over an individual learning system (Baghaei et al. 2007). A previous iteration of our system demonstrated that adaptive support improved

student peer tutoring in high school algebra over a static resource on how to collaborate effectively (Walker et al. 2011). There is also evidence that ACLS can improve domain learning. CycleTalk adaptively supports collaborative dialog in simulation-based learning environments using tutorial dialog agents, and has been shown to be better than fixed support at increasing domain learning (Kumar et al. 2007). Karakostas and Demetriadis (2011) provided adaptive domain support to collaborating students in the form of prompts relating to important concepts students had missed in their discussion. This adaptive support improved domain learning over a static resource that described all important domain concepts. Although the research involving actual adaptive systems and their effects on learning from collaboration is limited, there have additionally been Wizard of Oz studies where adaptive support within a collaborative learning system has been provided by a human acting as a computer, and benefits of the adaptive support have been demonstrated (Gweon et al. 2006; Tsovaltzi et al. 2010).

It is important to understand why ACLS might be effective, so we can better direct our efforts when building support. Research on collaborative learning has evolved from investigating whether collaborative learning is better than individual learning, to investigating when collaborative interactions are related to benefits of collaboration, to understanding how to support positive collaborative interactions to yield optimal outcomes (Dillenbourg et al. 1995). Now that it is possible to support collaborative interactions adaptively using technology, it becomes important to understand what kinds of adaptive support are effective for what kinds of collaboration. Most research on ACLS has assumed that it benefits students because when given relevant support in a timely manner, students might more easily apply it to their interactions. In fact, the success of ACLS systems is typically measured by looking at the validity of the collaborative model used (Suebnukarn and Haddawy 2006) or the applicability of the feedback messages given (Constantino-González et al. 2003). However, there is another explanation for why ACLS might be effective: students who believe they are receiving adaptive support may feel more motivated to engage with the support and take appropriate action, regardless of the actual relevance of the support. Accountability for one's partner's outcomes tends to be an important motivational force in collaboration (e.g., Slavin 1996), and in previous work we found results that suggested that if students believe the computer is monitoring and responding to their actions, they may feel an increased sense of accountability for those actions and become better collaborators (Walker et al. 2011). If support relevance is important, the uncertainty inherent in modeling student collaboration needs to be resolved, but if support relevance is not important, then designing engaging support becomes the most important goal. By implementing a truly adaptive system, we can begin to evaluate the impact more relevant help, delivered by an intelligent system, has on student learning from collaboration.

To demonstrate the effectiveness of our system, we examined whether and why providing more adaptive support improves learning from collaboration. We compared three conditions: one in which support is adaptive and students were told it is adaptive (*real adaptive*), one in which support is random but students were told it is adaptive (*told adaptive*), and one in which support is random and students were told it is random (*real nonadaptive*). If it is relevant support that matters, only students who actually receive adaptive support (i.e., those students in the *real adaptive* condition) should improve their collaboration quality and their domain learning. On the other hand, if accountability is most important, *told adaptive* students should also improve

their collaboration quality and domain learning. By distinguishing between these two explanations for the effectiveness of adaptive collaborative support, we can better evaluate the effectiveness of the specific adaptivity implemented in our system.

In the remainder of this paper, we first survey research on learning from peer tutoring, and describe they ways in which APTA, our collaborative learning environment, enables peer tutors and tutees to interact. We describe the implementation of the ACLS we developed for the peer tutor. We then present the results of an evaluation of APTA that explores why adaptive support might be effective in this context. For consistency, throughout this paper, the student being tutored is referred to as the *tutee*, the student doing the tutoring is referred to as the *peer tutor*, and the intelligent tutoring component that provides feedback to the peer tutor is referred to as the *help-giving tutor*.

Context: Peer Tutoring

We investigate adaptive support to collaborative dialogue within the context of a reciprocal peer tutoring script. In reciprocal peer tutoring, novices are put in pairs, and take turns tutoring each other. These scenarios have been shown to lead to learning in classroom environments (Fantuzzo et al. 1989). The core activity in reciprocal peer tutoring is help-giving, where one student helps another. Help-giving is a large component of many collaborative scenarios, and thus determining how to support students in giving better help might generalize to many different areas (Johnson & Johnson, 1990). Peer tutoring can have a positive impact on both the help-giver and help-receiver. Students of all abilities benefit from giving help (Ploetzner et al. 1999). When students know they will be tutoring another, they are more motivated to attend to the domain material. As their partner takes steps and makes errors, they reflect on the steps, noticing their own misconceptions. As they construct explanations to help their partner, they elaborate on their existing knowledge and construct new knowledge (Roscoe and Chi 2007). In contrast, for the help-receiver to benefit from reciprocal peer tutoring scenarios, many criteria need to be met: for example, help needs to target misconceptions, be conceptual, be elaborated, and, for the most part, be correct (Webb and Mastergeorge 2003). In addition, the actions help-receivers take can have a positive impact on their own learning, for example, by using peer tutor help constructively or by self-explaining their errors (Webb et al. 1995; Chi et al. 1994).

APTA, our peer tutoring system, is based on the literal equation solving unit of the Cognitive Tutor Algebra (*CTA*), a successful individual intelligent tutoring system in high school algebra (Koedinger et al. 1997). In literal equation solving, students are given a prompt like "Solve for x," and then given an equation like "" This domain is consistently identified by classroom teachers as one that is particularly difficult for students to master. In this unit, students use menus in an equation solver tool to manipulate the equation, selecting operations like "add x" or "combine like terms". The semantic label for the operation then appears on the right side of the screen. For certain problems, students have to type the result of the operation in addition to selecting it. As the students solve the problem, the *CTA* compares their actions to a model of correct and incorrect problem-solving behavior. If they make a mistake, they receive visual feedback in the interface, and often a message describing their misconception. At any point, students can request a hint on the next step of the

problem. The *CTA* monitors student skills, reflects them in a skill display, or skillometer, and selects problems based on student skill mastery. Students complete the unit when they have demonstrated mastery on problems at three levels of difficulty: 1) problems where all variable terms are on the same side, 2) problems where variable terms are on different sides of the equation, and 3) problems where the variable terms are in unusual positions (e.g., in the denominator of a fraction).

Our initial version of the peer tutoring script attempted to create interaction conditions conducive to the display of positive tutoring behaviors. During use of the script, students were in the same classroom, but tutors and tutees were seated on opposite sides of the room and asked to communicate with each other solely through a chat window on the computer. One component of the script involved encouraging the peer tutor to reflect on the correctness of tutee steps. The tutee solves problems using the equation solver interface in the *CTA*. The tutor sees the tutee's problem-solving steps and the results of her typed-in entries, but cannot solve the problem herself (see Fig. 1e). Instead, she can mark the tutee's actions right or wrong, providing feedback that the tutee sees and can use while continuing to solve the problem (Fig. 1f). Peer tutors did not have to mark every step before they could move to the next problem, but could mark whichever steps they feel necessary. Peer tutors receive feedback on these reflective actions; if they marked a step correct when it is actually incorrect, or marked a step incorrect when it is actually correct, the computer tutor highlights the answer in the interface, and presented the peer tutor with an error message, consisting of a randomly selected prompt to collaborate and the domain help the tutees would have received had they been solving the problem individually. The peer tutor could request a hint from the computer tutor at any time, and would receive multi-level hints involving domain help (including conceptual hints and feedback) and a prompt to collaborate. This feedback was designed to trigger reflective processes on the part of the peer tutor by encouraging them to mark steps more frequently. It also served to draw the peer tutors' attention to misconceptions by letting them know when they have made an error marking a problem step, further encouraging them to engage in reflection. In addition, the support is intended to lead peer tutors to provide tutees with more correct feedback on problem steps, which facilitates both students in building correct procedural knowledge in the domain. The hints that peer tutors receive are intended to further introduce more correct and conceptual content into the interaction. As students benefit from peer tutoring because they reflect on domain knowledge and their own misconceptions, and tutees benefit more from peer tutoring when they receive correct help, we felt that this intervention would improve the domain learning of both students involved.

A second component of the script involved natural language interaction between peer tutors and tutees using a chat tool, where, for example, tutees can ask questions and tutors can give hints and feedback (Fig. 1a). To facilitate the discussion in the chat window, we included a common form of fixed scaffolding: sentence classifiers. This form of fixed scaffolding is thought to be pedagogically beneficial by making positive collaborative actions explicit in the interface and encouraging students to consider the type of utterance they wish to make (Weinberger et al. 2005). We asked peer tutors to label their utterances using one of four classifiers: "ask why", "explain why wrong", "give hint", and "explain what next" (see Fig. 1d). Students had to select a classifier before they typed in an utterance, but they could also choose to click a neutral classifier ("other"). For example, if students wanted to give a hint, they could click "give hint" and then type "subtract x". Their utterance would appear as: "tutor hints: subtract

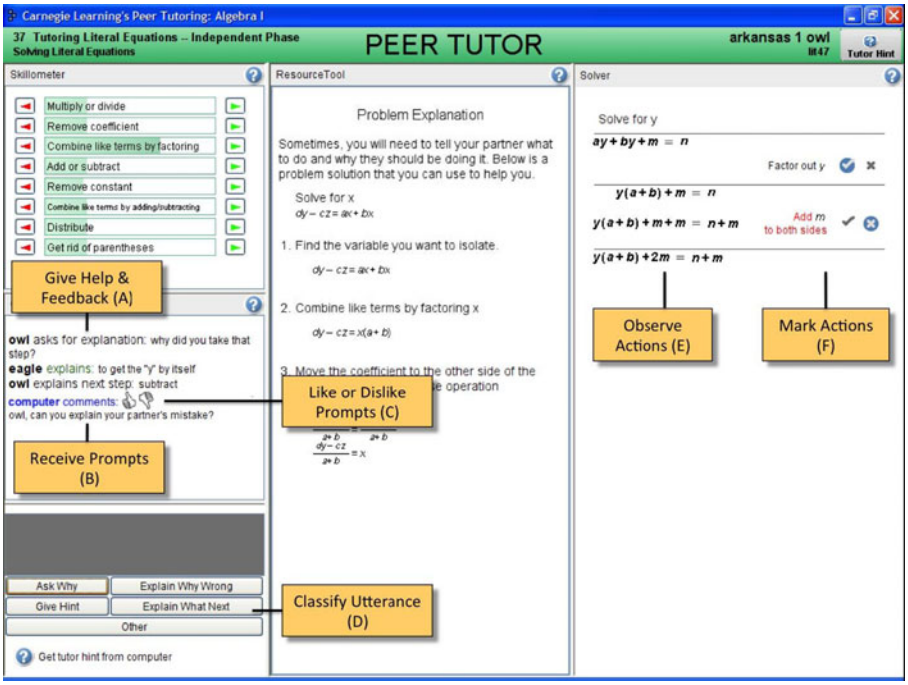

Fig. 1 Peer tutor's interface in APTA. The peer tutor watches the tutee take problem-solving actions (**e**), and marks the actions right or wrong (**f**). Students can talk in the chat window (**a**), where they classify their own utterances (**d**). They also receive prompts from the computer (**b**), and can choose to like them, dislike them, or ignore them (**c**)

x" to both students in the chat window. Tutees were also asked to self-classify each utterance as one of three categories: a "ask for help", "explain yourself", or "other". To further encourage students to chat appropriately, in the tutor's interface, we linked each sentence classifier to a brief description of how to use the classifier appropriately. This description appeared every time the student selected the sentence classifier. Dialogue is an important component of learning from peer tutoring; Peer tutors engage in elaborative and generative processes when they compose conceptual explanations and reflect on tutee questions; and tutees benefit by explaining their own actions and asking tutors specific questions (Webb and Mastergeorge 2003). Including a chat window gives students the opportunity to engage in these behaviors.

In the above system as described, there is cognitive support that adapts to what students are doing, but not direct support for the quality of student collaboration (which we call *help-giving support*). If tutees take incorrect steps, peer tutors have access to hints and feedback on what the correct steps are, and the concepts behind them. However, peer tutors do not receive adaptive guidance on whether they are giving good help and how to improve. We hypothesized that introducing adaptive help-giving support into the system would significantly improve the learning of both the peer tutor and peer tutee. While it is likely that supporting the interactions of the tutee more directly would also improve the learning of both parties, as an initial attempt at introducing this ACLS we focused only on peer tutor actions.

Adaptive Help-Giving Support

Description

In improving peer tutor help-giving, our goal was to target four peer tutor skills that exemplify good help in our context:

1. *Timely help*. Giving help in response to tutee errors and help requests. While this skill is rarely discussed in research on peer tutoring, interacting with the tutee when they are struggling is a basic common-sense requirement of effective peer tutoring.
2. *Appropriate help*. Prompting the tutee to self-explain and giving error feedback when appropriate. As surveyed above, both prompting tutees to self-explain and explaining tutee misconceptions can be beneficial for tutee learning (Chi et al. 1994; Webb and Mastergeorge 2003). Reflecting on tutee misconceptions is also beneficial for peer tutor learning (Roscoe and Chi 2007).
3. *Conceptual help*. Explaining a problem-solving step using a domain concept. Building a conceptual explanation is an essential part of peer tutor knowledge construction (Roscoe and Chi 2007). Receiving these explanations can also be beneficial for the tutee (Webb and Mastergeorge 2003).
4. *Use of classifiers*. Using sentence classifiers to accurately label the content of peer tutor utterances. Sentence classifiers can be beneficial scaffolding on their own for collaborating students (Weinberger et al. 2005), by enabling students to reflect on the types of collaborative actions they want to take. In addition, we use the sentence classifiers to improve our understanding of what the peer tutor is doing. Ensuring that students use sentence classifiers correctly could potentially both improve peer tutor learning and the accuracy of our system.

Adaptive support, visible to both students, was provided in the chat window to help peer tutors give better help. For example, the peer tutor might give an instrumental hint like "then subtract" rather than a conceptual hint like "to get rid of *qcv*, you need to perform the inverse operation on that side of the equation." In that case, the computer uses an assessment of the peer tutor's help-giving skill to say in the chat window (visible to both students), "Tutor, why do you say that? Can you explain more?" (Fig. 1b). This utterance is designed to get both students reflecting on the domain concepts behind the next step, and to remind the peer tutor that she should be giving help that explains why in addition to what. However, the computer assistance is posed as a question and uses non-critical wording to avoid threatening the authority of the peer tutor. Students also received encouragement when they displayed a particular help-giving skill ("Good work! Explaining what your partner did wrong can help them not make the same mistake on future problems"). Only one reflective prompt was given at a time, and prompts (in this version of the system) were always addressed to the peer tutor. Parameters were tuned so that students received an average of one prompt in the chat window for every three peer tutor actions. There were several different prompts for any given situation, so students rarely received the same prompt twice. These prompts were intended to support peer tutors in engaging in the four help-giving skills described above, leading them to experience more reflective and generative elaborative processes while providing better help to their tutees.

A simplified architecture of the system is depicted in Fig. 2. The peer tutor and tutee interact with separate interfaces. Student actions are sent to a central control module, and then to the tutoring components of the system. The adaptive help-giving support consists of an assessment component, a modeling component (that does both model and knowledge tracing), and a support component. Once a pedagogical decision has been made, the support component sends the appropriate action back to the control module, which then directs the response to the student interfaces. In the following subsections we describe how the help-giving tutoring components were implemented in more detail.

Assessment

The first step in delivering adaptive prompts to the peer tutor was to assess the current quality of peer tutor help (see the Assessment Component of Fig. 2). The help-giving tutor used a combination of several inputs. First, it used the CTA assessment of tutee problem-solving steps. For each step tutees took, the CTA models classified the step as right or wrong, and our help-giving tutor had access to that data. Second, it used student self-classifications of chat actions, based on the sentence classifier selected (e.g., "give hint"). Third, a machine classifier of student help, constructed using Taghelper Tools (Rosé et al. 2008; currently called LightSIDE), could determine whether students gave help, what kind of help it was, and whether it was conceptual

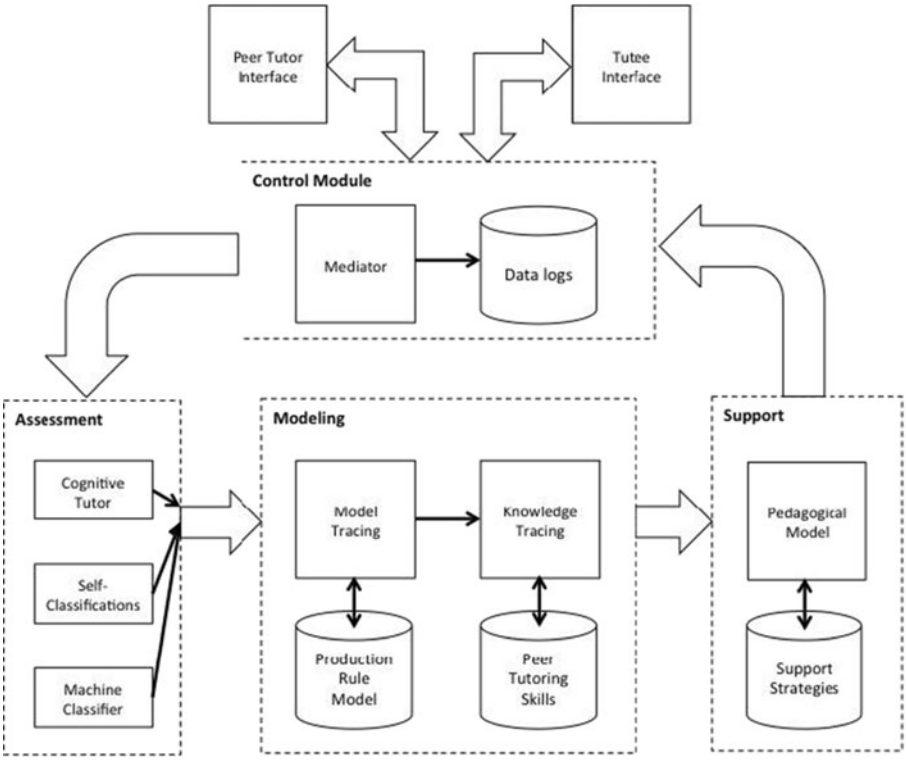

Fig. 2 Simplified architecture of APTA, depicting the help-giving tutoring components. First student actions are assessed, then we model trace to compare the actions to an ideal model and knowledge trace to update an assessment of tutor skills. Finally, an appropriate prompt is chosen by the support component

or not. The details of this classifier are described fully in Walker, Walker, Rummel, and Koedinger (2010); we summarize its functionality in the following paragraphs.

The system classified two aspects of peer tutor dialogue: help type and conceptual help. First, for *help type*, we looked at whether peer tutors were giving next-step help, previous-step help, both, or no help at all. Next-step help related to whether the utterance related to a future action on the problem (e.g., "How would you get rid of $2h$?") while previous-step help related to an action tutees had already taken (e.g., "No need to factor because there is only one g"). If the help segment contained both categories, its help type was labeled "both", and if it contained neither category (e.g., "on to the next problem"), its help type was labeled "none". Using the classified help type in conjunction with the problem-solving context (e.g., knowing whether the tutee has just made a correct step, incorrect step, or help request) can help APTA decide whether peer tutors are giving the appropriate kind of help. Second, for conceptual content, we looked at whether peer tutors gave help that explains concepts rather than simply stating what to do next (e.g., "add ax" was purely instrumental help, while "add ax to cancel out the $-ax$" was conceptual). Being able to identify this aspect lets APTA know whether peer tutors are providing enough conceptual help. We used a corpus drawn from a previous classroom study, where we compared adaptive and fixed support for peer tutoring (Walker et al. 2011). As part of the study, students participated in two supported peer tutoring sessions; one in which they acted as the tutor, and one in which they acted as the tutee. There were a total of 84 tutoring sessions from both conditions, consisting of an average of 21.77 tutor lines of dialogue per session (SD=10.25). Two raters coded tutor utterances for help type (kappa=0.83) and conceptual content (kappa=0.72). Interrater reliability was computed on 20 % of the data, and the remainder of the data was coded by one rater and checked by the second. All disagreements were resolved through discussion. The dialog was segmented by chat messages, creating a new segment every time students hit enter. In our dataset, 935 tutor instances were coded as "none", 764 were coded as "next-step help", 83 were coded as "previous-step help", and 47 were coded as "both"; 1654 instances were coded as non-conceptual help, and 165 were conceptual help.

We created baseline machine classifiers for help type and conceptual content using Taghelper Tools, state of the art text-classification technology designed for coding collaborative dialogue (Rosé et al. 2008). Taghelper automatically extracts several dialogue features for use in machine classification, including unigrams, bigrams, line length, and punctuation. The domain context of the interaction was used to provide additional features for a machine learning classifier. This context included information directly taken from the students' problem-solving behavior (e.g., a student has just taken a incorrect step in the problem), information about how student dialogue relates to the problem-solving context (e.g., a student has referred to another student's incorrect step), and information about the history of the interaction (e.g., a student referred to another student's incorrect steps 10 times over the course of the interaction). We used a chi-squared feature selection algorithm to rank the most predictive features, and 10-fold cross validation to train a support vector machine classifier for help type and conceptual content. On training data, the classifier had an accuracy of .81 for help type and .66 for conceptual content.

Modeling

The modeling components of APTA consisted of both a model tracing and knowledge tracing algorithm (see the Modeling component of Fig. 2). The above inputs (machine

classification, self-classification, and CTA classification) were fed into a production rule model with 19 rules. Our model is a *metacognitive* model rather than a *cognitive* model, and thus, while our approach was inspired by model tracing, we made accommodations for the ill-defined nature of the collaborative domain. Instead of being labeled as either "correct" or "buggy," rules were divided into four types: effective, somewhat effective, somewhat ineffective, and ineffective behaviors. These four levels allowed us to represent ambiguity in the model. The identification of rules and division into types were conducted through a literature review, and leveraged previous data collected of how students tutor each other in this particular context. Table 1 presents the rules, their type, their associated skill, and what classification sources were used as inputs.

Effective behaviors, represented by the "++" in Table 1, are most analogous to correct behaviors in traditional ITSs. They are paths in our model of good peer tutoring that are considered to be beneficial for collaboration quality the majority of the time. For example, explaining a tutee error was considered to be an ideal behavior (rule 9 in Table 1). The tutee commission of the error was detected by the CTA, and then explanation of the error was detected using our automated classification algorithm (whenever an utterance was classified as previous-step help). There are four effective behaviors in the model.

Somewhat effective behaviors, represented by the "+" in the table, were considered to be probably beneficial for collaboration quality at any given time. An example of a somewhat effective behavior can be found at rule 2 in Table 1, where if the tutee makes an error, the peer tutor should give help. This rule is only somewhat effective because while there are many situations where peer tutor should give help after an error, it is not necessarily the best course of action in all cases; there may be many situations where tutees should repair their own error. There were six somewhat effective behaviors in the model.

Somewhat ineffective behaviors, represented by the "-"in the table, were considered to be probably detrimental to collaboration quality. An example of a somewhat ineffective behavior is rule 13 in the table, where peer tutors give next step help after an error. While this may be beneficial in cases where tutees are struggling, in many situations it is likely to be more beneficial if peer tutors address tutee misconceptions in their help. The model consisted of five somewhat ineffective behaviors.

Finally, ineffective behaviors, represented by the "–"in the table, were behaviors considered to be detrimental to collaboration quality in most cases (analogous to buggy rules in traditional ITSs). The model consisted of four ineffective behaviors. Two of the ineffective behaviors are related to the timely help skill: one where the tutee commits three errors in a row without a peer tutor response (rule 6), and another where the tutee makes three help requests without a peer tutor response (rule 4) These rules are two of the three rules in the model where the model firing is triggered on peer tutor *inaction*, rather than on peer tutor *action*, which is unusual in typical ITSs. They are indicators that the tutee is in trouble, and the peer tutor, possibly because of a gap in domain knowledge, is struggling to help.

Rules were represented in a fully configurable xml file, and then parsed as part of a model tracing engine in the java code. Through this representation, we were able to modify the rule set on the fly, without having to recompile the code. This implementation strategy is ideal for testing the collaborative model. During piloting, we easily iterated on the model by adjusting its parameters and the specifics of the rule definitions.

While our approach has much in common with traditional production rule modeling, there are a few key differences. As in much production rule modeling, our approach

Table 1 Production rules in APTA. Each rule has an associated skill, and is mapped to effective (++), somewhat effective (+), somewhat ineffective (−), or ineffective behaviors (—). The classification column describes how the elements of the rule are assessed, either by the peer tutor (self), the help-giving agent (machine), or the Cognitive Tutor Algebra (CTA)

#	Rule	Skill	Type	Classification
1	IF tutee makes a help request THEN peer tutor gives help	Timely	++	self machine
2	IF tutee makes an error THEN peer tutor gives help	Timely	+	CTA machine
3	IF tutee self-explains THEN peer tutor gives help	Timely	+	self machine
4	IF tutee makes 2 help requests in a row THEN tutee makes a 3rd help request	Timely	–	self self
5	IF tutee makes a help request THEN tutee makes an error	Timely	-	self CTA
6	IF tutee makes 2 errors in a row THEN tutee makes a third error	Timely	–	CTA CTA
7	IF tutee makes a correct stepAND peer tutor gives next-step help AND tutee takes another correct step THEN peer tutor gives next-step help	Timely	-	CTA machine CTA machine
8	IF tutee makes an error THEN prompt for explanation	Appropriate	++	CTA self
9	IF tutee makes an error THEN give previous-step help	Appropriate	++	CTA machine
10	IF tutee makes an error AND tutee makes a help request THEN prompt for explanation	Appropriate	+	CTA self self
11	IF tutee makes an error AND tutee makes a help request THEN give previous-step help	Appropriate	+	CTA self machine
12	IF tutee makes an error AND tutee makes a help request THEN give next-step help	Appropriate	-	CTA self machine
13	IF tutee makes an error THEN give next-step help	Appropriate	–	CTA machine
14	IF the peer tutor gives help THEN help is conceptual	Conceptual	+	self machine
15	IF the peer tutor gives next-step help THEN help is not conceptual	Conceptual	-	self machine
16	IF peer tutor labels help THEN give help	Classifiers	++	self machine
17	If peer tutor labels no help THEN don't give help	Classifiers	+	self machine

Table 1 (continued)

#	Rule	Skill	Type	Classification
18	If peer tutor labels no help THEN give help	Classifiers	–	self machine
19	IF peer tutor labels help THEN don't give help	Classifiers	-	self machine

focuses on student actions (in this case, peer tutor actions). Our model of peer tutor behaviors contains rules for both correct and incorrect behaviors, although we created four categories: 2 levels of correctness (effective and somewhat effective) and 2 levels of buggy-ness (somewhat ineffective and ineffective). The rules in our model form a self-contained peer tutor, in that they can reproduce a subset of peer tutor help-giving behaviors (although we cannot produce natural language dialogue). On the other hand, our approach is in many ways more congruent with constraint-based tutoring systems than with model tracing tutors (Mitrovic et al. 2003). As in constraint-based tutors, the model that we use is a subset of the domain, and anything not represented in our model is considered to be correct behavior, rather than incorrect behavior. Most importantly, a single peer tutor action can cause multiple rules to fire, which is not the case with traditional model-tracing tutors. These adaptations are acknowledgments to the ill-defined nature of supporting collaboration, where we give peer tutors the benefit of the doubt by not immediately assuming the actions we do not understand are wrong, and associate several rules for tutoring behaviors with a single collaborative action.

Each production rule contributed to an overall assessment of the degree to which students had mastered one of the four skills described above: timely help, appropriate help, conceptual help, and classifier use. Timely help, covered by model rules 1–7, represented whether the peer tutor gave help when tutees needed it. Appropriate help, covered by rules 8–13, represented whether peer tutors gave the type of help that tutees needed. For the purposes of our model, it can be divided into two subskills: whether peer tutors prompted tutees to self-explain after an error, and whether peer tutors provided error feedback after an error. Conceptual help, covered by rules 14–15, represented whether peer tutors gave help that included an explanation. Finally, use of classifiers, represented by rules 16–19, covered whether peer tutors used sentence classifiers appropriately.

We used an algorithm based on Bayesian knowledge tracing to update a running assessment of peer tutor mastery of these four skills (Corbett and Anderson 1994). Knowledge tracing computes the likelihood that students have mastered a skill for any particular opportunity to do so (called $p[L_n]$), based on four parameters: The probability that peer tutors had mastered the skill before the opportunity ($p[L_{n-1}]$), the probability that peer tutors will learn the skill at the next opportunity ($p[T]$), the probability that the peer tutor will exhibit an effective collaborative behaviour even if they have not mastered the skill ($p[G]$), and the probability that the peer tutor will exhibit an ineffective collaborative behaviour even if they have mastered the skill ($p[S]$). For each student step, the algorithm first calculates the probability that students had mastered the skill prior to taking the step, and then the algorithm calculates the probability that students currently know the skill.

While this type of knowledge tracing has been used in individual settings for the assessment of domain knowledge, it has not to our knowledge been used in collaborative settings. Thus we present a novel application of standard knowledge tracing algorithms. We made a few modifications to the basic knowledge tracing algorithm to make it more appropriate for collaborative settings. First, at the beginning of the tutorial session, we set $p(L_o)$ to 0.9 for the collaborative skills. The system assumes that students know how to collaborate effectively, unless they repeatedly provide evidence that they do not. This approach gives students the benefit of the doubt on initial interactions with each other, assuming that the students know more about collaboration than the system does until a pattern of interaction suggests that students do indeed need help. Next, according to the approach in Beck and Sison (2006), we inflated the values of $p(G)$ and $p(S)$, the probabilities that students behave effectively even if they do not know the skill and ineffectively even if they do know the skill. We also varied those probabilities based on the valence of the fired rule (e.g., $p(S)$ was larger for a "somewhat ineffective" rule than for an "ineffective" rule). This approach takes parameters that were initially meant to represent human error and incorporates system error as well. $p(G)$ now included the probability that the system characterizes student responses as effective even if they do not know the skill, and p(S) now included the probability that the system characterizes student responses as ineffective even if they have mastered the skill. Essentially, as in Beck and Sison (2006), we redefine $p(G)$ to be the broader probability of a "false positive", and $p(S)$ to be the broader probability of a "false negative". For example, for the conceptual help skill, we defined $p(G)$=0.20 and $p(S)$=0.30 for effective and ineffective rules. For somewhat effective and somewhat ineffective rules, $p(G)$=0.25 and $p(S)$=0.375. The higher values for $p(G)$ and $p(S)$ for somewhat effective and somewhat ineffective rules means that the skill assessment increases less after a positive interaction, and also decreases less after a negative interaction.

In Table 2, there is an example model and knowledge trace. The students were solving the problem "" for w, and the tutee had just subtracted 3n from both sides, which was incorrect (#1 in the Table 2). The peer tutor then said "factor out n" and labeled it as a "hint". The computer classified the chat as next-step help and nonconceptual help (#2 in the table), and recognized that it came immediately after an incorrect step. This action fires model rules 2, 12, 15, and 16, meaning that the knowledge tracing assessments for all four skills are updated (#3 in Table 2). The assessment of peer tutor mastery of timely help and use of classifiers increases, while the appropriate help and conceptual help skills decrease. The effective and ineffective rules fired lead to more fluctuation in the skills than the somewhat effective and somewhat ineffective rules.

Support

We used a combination of the model tracing and knowledge tracing results to decide when to give students reflective prompts in the chat window, based on encoded support strategies and a pedagogical model (see the Support component of Fig. 2). The model tracing specified which skills students had exhibited or failed to exhibit with any particular action (firing particular production rules), and then the knowledge tracing recomputed the probability that students had mastered a skill. Each rule was linked to a set of feedback thresholds. If a rule fired, and the skill adjustment associated with the rule fell within one of the feedback thresholds linked to the rule, then the rule-threshold combination was added to a list specifying the possible

Table 2 Modeling and feedback example from Phase 3. The system uses the problem state to model student collaborative knowledge and select appropriate feedback

(1) Problem State				
Problem	Solve for	Last step	Last evaluation	State
-wn+3n=w	w	Subtract 3n	Incorrect	-wn=w-3n
(2) Assessment				
Tutor chat	Self labeling	Machine labeling	Machine labeling	Domain context
"Factor out n"	Hint	Next-step help	Non-conceptual	Incorrect step
(3) Model and Knowledge Tracing				
	Timely	Appropriate	Conceptual	Classifiers
$p(L_{n-1})$	0.903	0.911	0.903	0.794
Rule fired	2	13	15	16
Valence	+	–	-	++
$p(L_n)$	0.956	0.742	0.81	0.931
(4) Feedback Selection				
	Timely	Appropriate	Conceptual	Classifiers
Rule-threshold	None	[0.6,1]	[0.7,1]	None
Add to list	No	Yes	Yes	No
Priority	n/a	4	3	n/a
Chosen	No	Yes	No	No
(5) Message Choice				
Possible prompts	"Tutor, do you know if your partner has made a mistake?", "Tutor, can you explain your partner's mistake?", "Tutor, is there anything your partner doesn't understand right now about the problem?"			
Prompt chosen	"Tutor, is there anything your partner doesn't understand right now about the problem?"			

feedback to send. Each rule-threshold combination was assigned a particular priority, and once the list was complete, the rule-threshold with the highest priority was chosen to be the target rule for a reflective prompt. If there was a tie in priority, then the target rule-threshold was randomly chosen out of the tied candidates. Finally, each rule-threshold had a set of similar prompt messages associated with it, and one of the messages associated with the rule-threshold target was randomly chosen. The message was either sent to both students in the chat window or privately to the peer tutor, and this parameter was linked to the rule-threshold combination. This decision to make multiple messages for any given situation ensured that students rarely received the same message twice.

Expanding on the example in the previous section, although all the skills were adjusted, only the values of the appropriate and conceptual help fell within the feedback threshold, and were added to the list (#4 in Table 2). Because the rule associated with the targeted skill had the highest priority, it was selected to be delivered to both students in the chat window. Out of all the possible prompts that could be chosen (#5 in Table 2), the prompt "Tutor, is there anything your partner doesn't understand right now?" was sent to the students.

Both priorities and thresholds were assigned based on a combination of theoretical model of the relative importance from each rule, piloting, and data from previous studies. Priorities were assigned on a scale from 1 to 10. As an example, there were

two feedback thresholds based on rule #6: One was [0, 0.3) with a priority of 1, and one was [0.3, 7) with a priority of 2. Both thresholds were given extremely high priorities because the peer tutor likely needs immediate feedback if they are not responding to three tutor errors in a row. The messages associated with the higher threshold for this rule (e.g., "Tutor, is your partner still taking the right steps? Make sure both sides of the equation still equal each other") tended to have less urgency than the messages associated with the lower threshold (e.g., "Tutor, do you know what your partner should do? Try asking the computer for a hint"), as a lower skill assessment indicated the peer tutor needed more explicit support. As a second example, rule #15 had a single feedback threshold: [0,.7) with a priority of 7. If a student gives high level help, and they are not in the habit of doing so (i.e., their $p[L_n]$ was under .7), then giving them positive feedback to reinforce their behavior was a somewhat low priority, but still built into the system. In contrast, if they were already relatively proficient at the skill ($p[L_n]>= .7$), we did not give any positive feedback. We tuned these parameters in piloting so that peer tutors received one prompt for every three peer tutor actions.

As the primary use of the skill estimates was to trigger feedback, our knowledge tracing algorithm was designed and tuned more to accomplish this goal than to actually estimate student collaborative skills. We are aware our approach violates certain assumptions of knowledge tracing (Corbett and Anderson 1994): In our approach, a student can transfer from a learned to an unlearned state, and more than one skill maps to a given action. Future work will be to modify the core knowledge tracing algorithm to incorporate the assumptions we use as part of the peer tutoring model, evaluate the algorithm against a human-coded assessment of student skills, and improve the algorithm to be a more fine-grained representation of peer tutor skills. However, we think our approach is an effective first pass at using a knowledge tracing approach to mitigate the uncertainty inherent in collaborative scenarios, by using the (potentially flawed) skill estimates to trigger feedback rather than specific peer tutor behaviors.

Data Collection

Hypotheses & Conditions

We conducted a study in order to evaluate whether our adaptive system had a beneficial effect on peer tutor learning, and what features of the system may have contributed to its effectiveness. In the introduction, we presented two possible mechanisms that might link receiving adaptive support to benefitting more from a peer tutoring interaction. First, students who receive relevant support might more easily apply it to their interactions, improving the quality of their collaboration and learning. Second, students who *believe* they are receiving adaptive support may feel more motivated to engage with the support and take appropriate action, improving the quality of their collaboration and learning.

We tested these hypotheses and evaluated the effectiveness of our system by comparing the help-giving support in the adaptive version of our system (the *real adaptive* condition) to two conditions that received non-adaptive computer prompts in the chat window. In one of the nonadaptive conditions (the *real nonadaptive* condition), students were told that the prompts were nonadaptive. In the second

nonadaptive condition (the *told adaptive* condition), students were told that the prompts were adaptive, when they in fact were not. If it is relevant support that matters, only students who actually receive adaptive support (i.e., those students in the *real adaptive* condition) should improve their collaboration quality and their domain learning. On the other hand, if engagement is most important, and students in the *told adaptive* condition believe the system is adaptive, *told adaptive* students should also improve their collaboration quality and domain learning *real nonadaptive* students. By distinguishing between these two explanations for the effectiveness of adaptive collaborative support, we can better evaluate the effectiveness of the specific adaptivity implemented in our system.

Participants

Participants were 130 high-school students (49 males, 81 females) from one high school, ranging from 7th to 12th grade, and currently enrolled in Algebra 1 (46 students), Geometry (49 students), or Algebra 2 (35 students). While the literal equation solving unit was one that all students had theoretically received instruction on in Algebra 1, many students did not remember seeing the material before. The teacher we were working with identified this unit as challenging for the students. The study was run at the high school, either immediately after school or on Saturdays. Thus the study was somewhere in between a lab study and a classroom study: It was run in a school context and with several students at once, but it was not run during school hours as part of regular classes. All students were paid 30 dollars for their participation. Students participated in sessions of up to 9 students at a time (M group size=7.41, SD=1.35). Each session was randomly assigned to one of the three conditions, and then within each pair students were randomly assigned to the role of tutee or tutor. While APTA was designed to be used in a reciprocal scenario, for the purposes of this evaluation students retained the same role throughout the whole study; if a student was assigned the role of tutor, he or she tutored throughout the entire session.

Students came with partners that they had chosen, except in the case of 12 students who came to the study alone and were assigned to their partners by the researchers. The results of Ogan, Finkelstein, Walker, Carlson, and Cassell (2012) in their analysis of this dataset suggested that students who self-selected their partners and thus collaborated with friends interacted more effectively and learned more than students who have their partners selected by researchers. We thus excluded the 12 students with researcher-selected partners from our analysis, as they were small in number, may have been from a different population than those with self-selected partners, and had a substantially different interaction experience. Two dyads were excluded due to logging errors with the computer prompts. Further, for ease of scheduling, we sometimes assigned an extra student to a given session (in case somebody did not show up at the assigned time). There were 8 students who worked alone over the course of the session. Thus, a total of 108 students were included in the analysis. There were 45 same-gender pairs and 8 cross-gender pairs. We did a median split on pretest score to reclassify students as low-ability or high-ability, and then counted homogenous and heterogeneous pairs. 31 pairs were homogenous (two low-ability or two high-ability students) and 22 were heterogeneous (one low ability and one high ability student).

Procedure

During the study, students took a 20 min pretest. Next, students spent 20 min in a preparation phase, solving problems individually using the CTA. All students worked on easier problems in the literal equation solving unit, which consisted of factoring problems where the variable terms were on the same side of the equation. Students then spent 30 min in the tutoring phase, with the peer tutor helping their partner with factoring problems where the variable terms were on both sides of the equation. Students took up to 10 min to answer several survey questions on their motivational state, and then spent another 30 min in the tutoring phase. Students took a 20 min domain posttest.

In the tutoring phase, we varied whether students received adaptive support or not and whether they thought it was adaptive or not. The nonadaptive support was implemented as follows. We gave students pseudo-random prompts that ensured that the timing and content of the prompts did not depend on their behavior. Every time students would have received a reflective prompt were they in the adaptive condition, they never received a prompt in the fixed condition. However, we ensured that they received a prompt within the next three turns, essentially yoking the nonadaptive prompt to the adaptive prompt. We randomly choose the content of the prompt, but we never choose content that would have been relevant to the yoked adaptive prompt. All other support across conditions was parallel (i.e., all students received adaptive correction support).

We manipulated whether we told students that support was adaptive or nonadaptive prior to the tutoring phase. The adaptive instructions were as follows: "The computer will watch you tutor, and give you targeted advice when you need it based on how well you tutor. Both you and your partner will see the help in the chat." The nonadaptive instructions were as follows: "From time to time, the computer will give you a general tip chosen randomly from advice on good collaboration. Both you and your partner will see the help in the chat." As students began to use APTA, they were given further instruction, including directions to indicate how they felt about the reflective prompts using thumbs up and thumbs down widgets (Fig. 1c). To motivate the use of these widgets and reaffirm the experimental manipulation, students in the real and told adaptive conditions were told: "We will use that information to improve the computer's ability to track what you're doing and give you advice you can use." Students in the real nonadaptive condition were told: "We will use that information to describe which pieces of advice can go into the pool of advice we randomly select from."

Measures

To assess students' individual learning we used counterbalanced pretests and post-tests, each containing 7 conceptual items (some with multiple parts), 5 procedural items, and 2 items that demanded a verbal explanation. Tests were approved by the coordinating classroom teacher, and were administered on paper. We scored answers on these tests by marking whether students were correct or incorrect on each item part, and then summing the item scores to get a total score.

As a manipulation check, we assessed perceived adaptivity with five items asking students how adaptive they thought the system was (e.g., for the peer tutor: "The computer gave advice at times when it was useful") and how positively they perceived

the system's effects (e.g., for the tutee: "The advice the computer gave improved how well my partner tutored me"). Items were rated on a 7-point likert scale.

All collaborative process variables were logged, including tutee problem-solving actions, sentence classifiers selected by both students, and chat actions made by both students. Along with the student actions, we logged computer tutor responses, which included both the system's evaluation of the action and the computer assistance students received.

We coded each instance of support delivered by the computer tutor for whether it was relevant to the current context, as defined by the tutee-tutor interactions spanning the last instance of tutee dialogue, peer tutor dialogue, and tutee problem step. To be relevant, negative feedback had to meet three criteria:

1. Not contradict the current situation. E.g., feedback that referred to an error contradicts the situation if tutees had not made an error.
2. Refer to something students were not currently doing. E.g., feedback that prompted for more conceptual help would only be relevant if students were not giving conceptual help.
3. If students were to follow the help, their interaction would be improved, based on the four skills. E.g., feedback that tells the peer tutor to give help would improve the interaction if the tutee had asked for help and not received it.

For positive feedback to be relevant, students had to be doing something to merit positive feedback, and then the advice given by the feedback had to meet the above criteria #1 and #3. To calculate inter-rater reliability, two raters independently coded 30 % of the data, with a kappa of 0.70. Conflicts were resolved through discussion.

Results

Domain Learning

Our first step was to determine whether students learned more from the real adaptive support condition than from the other two nonadaptive conditions. We conducted a two-way (condition x role) ANCOVA, controlling for pretest, with posttest as the dependent variable. Pretest score was significantly predictive of posttest score (F[1,99]=103.73, p<0.001; see Table 3). There was a significant effect of condition on posttest (F[2,99]=4.03, p=0.021, eta^2=0.075), indicating that the adaptivity of support had a positive effect on student posttest performance. A planned comparison of the effects of receiving real adaptive support revealed that it indeed had a significant effect (F[1,99]=7.73, p=0.006), while a planned comparison of the effects of receiving support that students were told was adaptive revealed that this manipulation did not have a significant effect (F[1,99]=0.990, p=0.322). These results suggest that the real adaptive support we implemented in APTA had a more beneficial effect than nonadaptive support. Telling students support was adaptive did not have a beneficial effect on learning compared to telling them support was not adaptive.

While the effect of role on posttest was not significant (F[1,99]=0.194, p=0.661), there was a significant interaction effect between condition and role (F[2,99]=3.87, p=0.024, eta^2=0.073). Applying the planned comparisons to the interaction effect revealed that the effects of real adaptivity had significantly differential effects on peer

Table 3 Mean pre and posttest results for tutors and tutees by condition. Standard deviations are in parentheses

Condition	Peer Tutor		Peer Tutee	
	Pretest	Postttest	Pretest	Postttest
Real Adaptive	0.27 (0.14)	0.42 (0.18)	0.28 (0.16)	0.37 (0.22)
Told Adaptive	0.25 (0.13)	0.29 (0.14)	0.27 (0.16)	0.29 (0.16)
Real Nonadaptive	0.30 (0.16)	0.29 (0.18)	0.24 (0.15)	0.35 (0.21)

tutors and tutees (F[1,99]=3.95, p=0.05), as did the effects of told adaptivity (F[1,99]=7.33, p=0.008). Inspecting student learning across role and condition (see Table 3), we see that while all students benefit from the real adaptive condition, peer tutors benefit more from the told adaptive condition than the real nonadpative condition, but tutees benefit more from the real nonadaptive condition than the told adaptive condition. The perception that the advice is relevant when it is not, as in the told adaptive condition, may impede the tutoring abilities of the peer tutor and thus may lead to less tutee learning.

Support Relevance

Given the encouraging results that students in the real adaptive condition learned more, we then verified that the support students received in the real adaptive condition was indeed more adaptive. Indeed, the total number of prompts each pair received from the adaptive system was not significantly different between conditions (F[2,50]=0.660, p=0.522; see Table 4). We conducted an ANCOVA with relevant prompts received as the dependent variable, condition as an independent variable, and total prompts received as a covariate. While the number of relevant prompts students received was not significantly different between conditions (F[2,47]=0.057, p=0.944; see Table 4), total prompts received was predictive of relevant support (F[1,47]=266.34, p<0.01). The interaction between condition and total support given was significantly related to relevant prompts received (F[2,47]=17.32; p<0.001). Inspecting Fig. 3, you can see that as total instances of feedback increase, the greater the difference in amount of relevant feedback between the real adaptive conditions and the other two conditions. Over a long period of time, with many instances of

Table 4 Mean support received and relevant support received per group by condition. Perceived adaptivity by condition and role. Standard deviations are in parentheses

	Support Characteristics		Perceived Adaptivity	
Condition	Total support	Relevant support	Peer tutor	Tutee
Real Adaptive	15.20 (12.17)	12.60 (11.34)	5.88 (1.82)	4.15 (0.85)
Told Adaptive	17.84 (9.67)	7.63 (5.63)	5.29 (1.40)	3.65 (1.23)
Real Nonadaptive	14.26 (11.86)	5.68 (4.44)	4.60 (1.34)	3.80 (1.01)

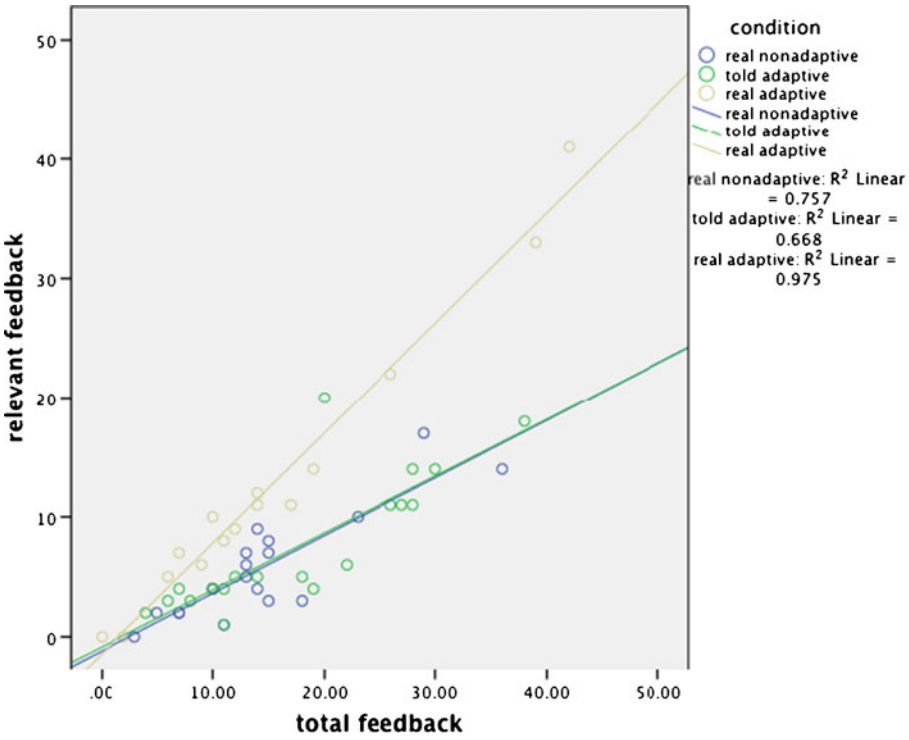

Fig. 3 Graph of relevant support compared to total support by condition. As total support instances increase, the adaptive condition has more relevant support than the other two conditions

feedback, the differences between adaptive and nonadaptive support become apparent.

Perceptions of Adaptive Support

We next examined whether student perceptions of adaptive support differed across conditions, as a check of our "told adaptive" manipulation. We had intended the told adaptive and real adaptive conditions to perceive support as more adaptive than the real nonadaptive condition. Thus, we had asked students questions intended to assess whether they perceived the support they received as adaptive. We conducted a two-way ANCOVA (including both peer tutors and tutees) with perceived adaptivity as the dependent variable, condition and role as independent variables, and relevant feedback received as a covariate. There was no significant differences across conditions on the perceived adaptivity measure ($F[2,80]=.330$, $p=0.72$). Student perceptions did not appear to be affected by the experimental manipulation of telling them support was adaptive. However, the amount of relevant support students received did predict the perceived adaptivity of the system ($F[1,80]=34.08$; $p<0.001$). In addition, there was a significant interaction between role and relevant support ($F[1,80]=9.99$, $p=0.002$). The scatterplot in Fig. 4 reveals that as the amount of relevant support increases, so do students perceptions of the adaptivity of support, and this effect is stronger for peer tutors than for tutees. Because peer tutors were the targets of the

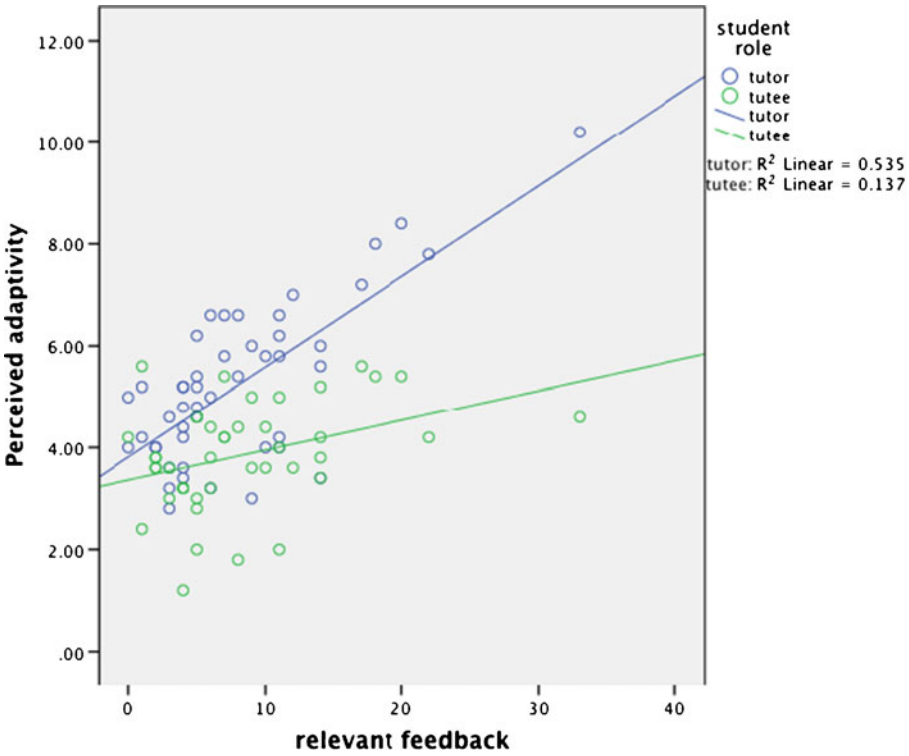

Fig. 4 Graph of relevant support and perceived adaptivity. As relevant support increases, perceived adaptivity increases. This effect is stronger for peer tutors than tutees

support, it seems logical that their perceptions of the support would be more affected by its relevance.

Frequencies of Support Given

Finally, to get a better sense of what specific support students received based on our model, we looked at the *real adaptive* condition to determine how often each rule fired, how frequently support was given to students based on the rule, and how frequently that support was relevant. The means and standard deviations of this measure are presented in Table 5. Within our model, the most frequent peer tutor behavioural profile involves giving help after an error that focuses on the next step, is nonconceptual, and isn't appropriately labelled with a sentence classifier. Our system often gave students feedback on these behaviors. Our system also reinforced particular positive behaviors that students engaged in: giving help after a help request, giving conceptual help, and labelling utterances appropriately with sentence classifiers. The majority of support given in response to each rule was coded as relevant, except for the cases of prompting for self-explanation and using sentence classifiers when help is not being given. This result may have been the fault of poorly designed feedback messages or a failure of the automated classifier to detect help. There are also some rules in the model that did not fire at all (e.g., 4, 10, 11, and 12). These

Table 5 Production rules in APTA , along with how frequently they fired, how frequently they triggered feedback, and how many of those feedback instances were relevant. Each rule has a type representing whether the rule is mapped to effective (++), somewhat effective (+), somewhat ineffective (−), or ineffective behaviors (—)

#	Rule	Type	# Times Rule Fired M (SD)	Support Instances M (SD)	Relevant Support Instances M (SD)
1	IF tutee makes a help request THEN peer tutor gives help	++	1.40 (2.38)	0.67 (1.40)	0.47 (0.92)
2	IF tutee makes an error THEN peer tutor gives help	+	4.27 (2.69)	0 (0)	0 (0)
3	IF tutee self-explains THEN peer tutor gives help	+	0.20 (0.56)	0 (0)	0 (0)
4	IF tutee makes 2 help requests in a row THEN tutee makes a 3rd help request	–	0 (0)	0 (0)	0 (0)
5	IF tutee makes a help request THEN tutee makes an error	-	0.67 (1.40)	0.625 (1.36)	0.625 (1.36)
6	IF tutee makes 2 errors in a row THEN tutee makes a third error	–	1.33 (2.02)	1.33 (2.02)	1.27 (1.87)
7	IF tutee makes a correct stepAND peer tutor gives next-step help AND tutee takes another correct step THEN peer tutor gives next-step help	-	1.13 (2.07)	0.73 (1.71)	0.67 (1.63)
8	IF tutee makes an error THEN prompt for explanation	++	0.47 (1.30)	0.13 (0.35)	0.07 (0.26)
9	IF tutee makes an error THEN give previous-step help	++	0.33 (0.49)	0 (0)	0 (0)
10	IF tutee makes an error AND tutee makes a help request THEN prompt for explanation	+	0 (0)	0 (0)	0 (0)
11	IF tutee makes an error AND tutee makes a help request THEN give previous-step help	+	0 (0)	0 (0)	0 (0)
12	IF tutee makes an error AND tutee makes a help request THEN give next-step help	-	0 (0)	0 (0)	0 (0)
13	IF tutee makes an error THEN give next-step help	–	4.20 (2.68)	4.13 (2.61)	3.00 (2.48)
14	IF the peer tutor gives help THEN help is conceptual	+	2.27 (2.25)	1.00 (1.77)	0.93 (1.79)
15	IF the peer tutor gives next-step help THEN help is not conceptual	-	11.13 (10.44)	5.00 (4.52)	4.33 (4.67)
16	IF peer tutor labels help THEN give help	++	4.53 (5.34)	0 (0)	0 (0)
17	If peer tutor labels no help THEN don't give help	+	0 (0)	0 (0)	0 (0)
18	If peer tutor labels no help THEN give help	–	9.20 (12.16)	2.73 (5.46)	2.73 (5.46)
19	IF peer tutor labels help THEN don't give help	-	0.27 (0.59)	0.27 (0.59)	0.07 (0.26)

rules may have not fired because we relied on tutees to use sentence classifiers to label their help requests, and tutees may have failed to do so or done so inaccurately. Overall, however, the model behaved as expected and provided relevant support to tutees.

Discussion

In this paper, we discussed the assessment, modeling, and support provided in APTA, an intelligent tutoring system for peer tutoring. We demonstrated that APTA is indeed adaptive, in that it provides students with significantly more relevant support than non-adaptive control conditions, and also that APTA improves student learning over non-adaptive controls. We also found that student perceptions of the adaptivity of the system was directly linked to the actual adaptivity of support, making it difficult to convince students that support was adaptive when, in fact, it was not. Based on these results, our system was a successful implementation of an ACLS.

Our approach acknowledged the ambiguity inherent in assessing, modeling, and supporting collaboration. One challenge in supporting collaboration is the difficulty in automatically assessing student dialogue. APTA uses multiple sources of information to assess collaborative state: a combination of problem-solving information, student self-classifications of their own chat, and machine classifications of student chat. The incorporation of problem-solving information into our assessment would not have been possible without having built our help-giving tutor on top of the Cognitive Tutor Algebra, but we believe it was a main contributor to the success of the system. The multiple channels of information allowed us to understand better the context in which peer tutoring actions were executed, allowing us to interpret those actions more effectively.

Another challenge in supporting collaboration is the difficulty inherent in modelling collaborative behaviors by representing which interactions should be employed, and under what contexts. APTA maintains a production-rule style model of effective and ineffective peer tutor actions that it uses to compare the current collaborative state to an ideal model. The modeling in APTA draws both from model-tracing tutors and constraint-based tutors to incorporate the advantages of both approaches. Like model-tracing tutors, APTA focuses on peer tutoring actions and models both correct and incorrect behaviors. Like constraint-based tutors, APTA assumes that anything outside the model is correct rather than incorrect, and an action can fire multiple rules. By assuming that anything outside the model is correct, it gives the peer tutors the benefit of the doubt when they take unexpected tutorial actions. One innovation here is that APTA employs different levels of correctness, distinguishing, for example, between effective behaviors and somewhat effective behaviors. Using this technique, we can mitigate some of the ambiguity in judging the potential benefits of a particular peer tutor behavior by representing (albeit in a discrete way) the likelihood that the behaviour is effective.

A final challenge in supporting collaborative learning is understanding how to provide adaptive support based on the uncertain information delivered in the assessment and modeling phases. APTA uses Bayesian Knowledge Tracing to evaluate peer tutor skills and provide reflective prompts in a chat window. Our approach is used as

a trigger for feedback, and was not yet designed to be a fine-grained representation of student skills. However, it allows us to base feedback on the overall pattern of peer tutor behaviors rather than specific peer tutor actions, making us more certain that peer tutors will get positive feedback when they are truly excelling and negative feedback when they are truly struggling. Future work will be to iterate on the algorithm so it can more accurately assess collaborative skills.

We demonstrated that the techniques we used produced more relevant help than nonadaptive techniques, using a human coding of relevant help. Developing our coding scheme was a challenging and iterative process, in part because any given instance of support may appear to be relevant in multiple situations. Indeed, the nonadaptive technique we used still produced a large proportion of relevant help, even when we purposefully inhibited all responses congruent with our adaptive model of support. It is possible that with carefully designed feedback messages, nonadaptive techniques could produce similar amounts of relevant help as adaptive techniques. Nevertheless, the amount of relevant help affected student perceptions of the adaptivity of the system, suggesting that students do recognize and respond to adaptive support. Additionally, as relevant support increased, peer tutors perceived the system to be more adaptive than tutees. This suggests that in order to reap the full benefits of relevant support, it should be directed at all parties in the interaction. This link between relevant support and perceptions of adaptivity may have negated the effects of our second manipulation, where we told students support was adaptive when, in fact, it was not.

Our empirical results support some common-sense ideas about the benefits of adaptive support. We demonstrated that students in the adaptive support conditions learned more than students in the nonadaptive conditions. As the amount of support students received increased, the difference between the adaptive condition and the nonadaptive conditions became more apparent. While if students receive few instances of support it may not be necessary that the support be highly adaptive, as students use the system over a longer period of time they will be able to distinguish between adaptive and nonadaptive support. The techniques presented in this paper bring us closer to implementing an *ACLS* that can provide students with the long-term adaptive support they need to collaborate more effectively.

Acknowledgments This work was supported by the Pittsburgh Science of Learning Center, NSF Grant #SBE-0836012, and a Computing Innovations Fellowship, NSF Grant #1019343. Thanks to Ruth Wylie for her comments and Sean Walker for his work on the assessment algorithm.

References

Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting collaborative learning and problem solving in a constraint-based CSCL environment for UML class diagrams. *International Journal of Computer-Supported Collaborative Learning, 2*(2–3), 159–190.

Baker, M., & Lund, K. (2003). Promoting reflective interactions in a CSCL environment. *Journal of Computer Assisted Learning, 13*(3), 175–193.

Barros, B., & Verdejo, M. F. (2000). Analysing student interaction processes in order to improve collaboration. The DEGREE approach. *International Journal of Artificial Intelligence in Education, 11*(3), 221–241.

Beck, J. E., & Sison, J. (2006). Using knowledge tracing in a noisy environment to measure student reading proficiencies. *International Journal of Artificial Intelligence in Education, 16*, 129–143.

Chi, M. T., De Leeuw, N., Chiu, M. H., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18*(3), 439–477.

Constantino-González, M. A., Suthers, D., & Escamilla de los Santos, J. (2003). Coaching web-based collaborative learning based on problem solution differences and participation. *IJAIED, 13*, 263–299.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction, 4*(4), 253–278.

Dillenbourg, P. (2002). Over-scripting CSCL: The risk of blending collaborative learning with instructional design. In Kirschner, P. A. (Ed.), *Three worlds of CSCL: Can we support CSCL?* 61–91.

Dillenbourg, P., Baker, M. J., Blaye, A., & O'Malley, C. (1995). The evolution of research on collaborative learning. *Learning in Humans and Machine: Towards an interdisciplinary learning science*. 189–211.

Dragon, T., Floryan, M., Woolf, B., & Murray, T. (2010). *Recognizing dialogue content in student collaborative conversation. In Intelligent Tutoring Systems* (pp. 113–122). Berlin: Springer.

du Boulay, B., Avramides, K., Luckin, R., Martinez-Miron, E., Rebolledo-Mendez, G., & Carr, A. (2010). Towards systems that care: A conceptual framework based on motivation, metacognition and affect. *International Journal of Artificial Intelligence in Education, 20*(3), 197–229.

Fantuzzo, J. W., Riggio, R. E., Connelly, S., & Dimeff, L. A. (1989). Effects of reciprocal peer tutoring on academic achievement and psychological adjustment: A component analysis. *Journal of Educational Psychology, 81*(2), 173–177.

Fischer, F., Mandl, H., Haake, J., & Kollar, I. (2007). *Scripting computer-supported collaborative learning—cognitive, computational, and educational perspectives. Computer-supported collaborative learning series*. New York: Springer.

Gweon, G., Rose, C., Carey, R., & Zaiss, Z. (2006, April). Providing support for adaptive scripting in an on-line collaborative learning environment. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 251–260). ACM.

Israel, J., & Aiken, R. (2007). Supporting collaborative learning with an intelligent web-based system. *International Journal of Artificial Intelligence and Education, 17*(1), 3–40.

Johnson, D. W., & Johnson, R. T. (1990). Cooperative learning and achievement. In S. Sharan (Ed.), *Cooperative learning: Theory and research* (pp. 23–37). NY: Praeger.

Karakostas, A., & Demetriadis, S. (2011). Enhancing collaborative learning through dynamic forms of support: the impact of an adaptive domain-specific support strategy. *Journal of Computer Assisted Learning, 27*(3), 243–258.

King, A., Staffieri, A., & Adelgais, A. (1998). Mutual peer tutoring: Effects of structuring tutorial interaction to scaffold peer learning. *Journal of Educational Psychology, 90*, 134–152.

Koedinger, K. R., Anderson, J., Hadley, W., & Mark, M. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education, 8*, 30–43.

Kollar, I., Fischer, F., & Slotta, J. D. (2005). Internal and external collaboration scripts in web-based science learning at schools. In T. Koschmann, D. Suthers, & T.-W. Chan (Eds.), *Proceedings of the International Conference on Computer Support for Collaborative Learning 2005* (pp. 331–340). Mahwah: Lawrence Erlbaum Associates.

Kollar, I., Fischer, F., & Hesse, F. W. (2006). Collaboration scripts—A conceptual analysis. *Educational Psychology Review, 18*(2), 159–185.

Kumar, R., Rosé, C. P., Wang, Y. C., Joshi, M., & Robinson, A. (2007). Tutorial dialogue as adaptive collaborative learning support. In R. Luckin, K. R. Koedinger, & Greer J. (Eds.), *Proceedings of Artificial Intelligence in Education* (pp. 383–390). IOS Press.

Lazonder, A. W., Wilhelm, P., & Ootes, S. A. W. (2003). Using sentence openers to foster student interaction in computer-mediated learning environments. *Computers and Education, 41*, 291–308.

Magnisalis, I., Demetriadis, S., & Karakostas, A. (2011). Adaptive and intelligent systems for collaborative learning support: A review of the field. *IEEE Transactions on Learning Technologies, 4*(1), 5–20.

Mayfield, E., & Rosé, C. P. (2011, June). Recognizing authority in dialogue with an integer linear programming constrained model. In Proceedings of Association for Computational Linguistics.

7Mclaren, B. M., Scheuer, O., & Mikšátko, J. (2010). Supporting collaborative learning and e-discussions using artificial intelligence techniques. *International Journal of Artificial Intelligence in Education, 20*(1), 1–46.

McManus, M. M., & Aiken, R. M. (1995). Monitoring computer-based colaborative problem solving. *Journal of Artificial Intelligence in Education, 6*(4), 307–336.

Mitrovic, A., Weerasinghe, A. (2009). Revisiting ill-definedness and the consequences for ITSs. In: The 14th Conference on Artificial Intelligence in Education, pp. 375–382. IOS Press, Marina Del Ray

Mitrovic, A., Koedinger, K. R., & Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modeling. In P. Brusilovsky, A. Corbett, & F. D. Rosis (Eds.), *Proceedings of the Ninth International Conference on User Modeling, UM 2003 (Vol. LNAI 2702)* (pp. 313–322). Berlin: Springer.

Muldner, K., Burleson, B., VanLehn, K. (2010). "Yes!": Using tutor and sensor data to predict moments of delight during instructional activities. In *Proceedings of the International Conference on User Modeling and Adaptive Presentation*, 159–170.

Ogan, A., Aleven, V., Kim, J., & Jones, C. (2011). Persistent Effects of Social Instructional Dialog in a Virtual Learning Environment. In *Proc. 15th International Conference on AIED*, pp.238-246.

Ogan, A., Finkelstein, S., Walker, E., Carlson, R., & Cassell, J. (2012). *Rudeness and Rapport: Insults and Learning Gains in Peer Tutoring. In Proceedings of the 11th International Conference on Intelligent Tutoring Systems. ITS '12* (pp. 11–21). Berlin: Springer.

Ploetzner, R., Dillenbourg, P., Preier, M., & Traum, D. (1999). Learning by explaining to oneself and to others. In P. Dillenbourg (Ed.), *Collaborative learning: cognitive and computational approaches* (pp. 103–121). UK: Elsevier Science Publishers.

Roll, I., Aleven, V., McLaren, B. M., & Koedinger, K. R. (2011). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction, 21*, 267–280.

Rosatelli, M., & Self, J. (2004). A collaborative case study system for distance learning. *International Journal of Artificial Intelligence in Education, 14*(1), 97–125.

Roscoe, R. D., & Chi, M. (2007). Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of Educational Research., 77*(4), 534–574.

Rosé, C., Wang, Y. C., Cui, Y., Arguello, J., Stegmann, K., Weinberger, A., et al. (2008). Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in computer-supported collaborative learning. *International Journal of Computer-Supported Collaborative Learning, 3*(3), 237–271.

Rummel, N., & Weinberger, A. (2008). New challenges in CSCL: Towards adaptive script support. In G. Kanselaar, Jonker, V., Kirschner, P.A., & Prins, F. (Eds.), *Proceedings of the Eighth International Conference of the Learning Sciences (ICLS 2008)*, Vol 3 (pp. 338–345). International Society of the Learning Sciences.

Schoenfeld, A. H. (1992). Learning to think mathematically: Problem-solving, metacognition, and sense making in mathematics. In D. Grouws (Ed.), *Handbook for research on mathematics teaching and learning* (pp. 334–370). New York: Macmillan.

Slavin, R. E. (1996). Research on cooperative learning and achievement: What we know, what we need to know. *Contemporary Educational Psychology, 21*, 43–69.

Soller, A., Jermann, P., Mühlenbrock, M., & Martinez, A. (2005). From mirroring to guiding: A review of state of the art technology for supporting collaborative learning. *International Journal of Artificial Intelligence in Education, 15*(4), 261–290.

Suebnukarn, S., & Haddawy, P. (2006). Modeling individual and collaborative problem-solving in medical problem-based learning. *User Modeling and User-Adapted Interaction, 16*(3–4), 211–248.

Tedesco, P. (2003). MArCo: Building an artificial conflict mediator to support group planning interactions. *International Journal of Artificial Intelligence in Education, 13*(1), 117–155.

Tsovaltzi, D., Rummel, N., McLaren, B. M., Pinkwart, N., Scheuer, O., Harrer, A., et al. (2010). Extending a virtual chemistry laboratory with a collaboration script to promote conceptual learning. *International Journal of Technology Enhanced Learning, 2*(1), 91–110.

VanLehn, K. (2006). The behavior of tutoring systems. *IJAIED, 16*(3), 227–265.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist, 46*(4), 197–221.

Vieira, A. C., Teixeira, L., Timóteo, A., Tedesco, P., Barros, F. A., Lester, J. C., et al. (2004). Analyzing on-line collaborative dialogues: The OXEnTCHÊ-Chat. In F. Paraguaçu (Ed.), *Proceedings of the 7th International Conference on Intelligent Tutoring Systems* (pp. 315–324). Germany: Springer.

Vizcaíno, A., Contreras, J., Favela, J., & Prieto, M. (2000). An adaptive collaborative environment to develop good habits in programming. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *5th International Conference on Intelligent Tutoring Systems, ITS'2000* (pp. 262–271). Berlin: Springer.

Walker, E., Walker, S., Rummel, N., & Koedinger, K. (2010). Using problem-solving context to assess help quality in computer-mediated peer tutoring. In V. Aleven, J. Kay, & J. Mostow (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 145–155). Berlin: Springer.

Walker, E., Rummel, N., & Koedinger, K. R. (2011). Designing automated adaptive support to improve student helping behaviors in a peer tutoring activity. *International Journal of Computer-Supported Collaborative Learning, 6*(2), 279–306.

Webb, N. M., & Mastergeorge, A. (2003). Promoting effective helping behavior in peer-directed groups. *International Journal of Education Research, 39*, 73–97.

Webb, N. M., Troper, J. D., & Fall, R. (1995). Constructive activity and learning in collaborative small groups. *Journal of Educational Psychology, 87*(3), 406.

Weinberger, A., Ertl, B., Fischer, F., & Mandl, H. (2005). Epistemic and social scripts in computer–supported collaborative learning. *Instructional Science, 33*(1), 1–30.