Automated Adaptive Support for Peer Tutoring

Erin Walker

October 2010
CMU-HCII-10-107

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee
Kenneth R. Koedinger, Carnegie Mellon University (Co-chair)
Nikol Rummel, University of Freiburg (Co-chair)
Carolyn Rose, Carnegie Mellon University
Robert Kraut, Carnegie Mellon University

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy.

This work was supported by the National Science Foundation grants SBE-0354420 and #SBE-0836012. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect those of the National Science Foundation.

Keywords

Adaptive collaborative learning support, intelligent collaborative learning support, intelligent tutoring systems, reciprocal peer tutoring, computer-supported collaborative learning, collaboration scripts, in vivo experimentation, mathematics learning, educational technology, human-computer interaction.

Abstract

Collaborative activities have been shown to be beneficial, provided that students exhibit certain positive behaviors. Unfortunately, these behaviors rarely occur spontaneously. Adaptive collaborative learning support (ACLS), where an intelligent system assesses student collaboration as it occurs and provides assistance when necessary, is a promising area of research that can help scaffold student collaboration. Little is known about how to build these adaptive systems and what effects they might have on collaboration and domain learning. In this dissertation research, I first augmented an existing intelligent tutoring system with a peer tutoring activity and then iteratively designed, built, and evaluated adaptive support for the activity.

This dissertation focuses on two broad research questions: (1) Where and how can intelligent tutoring approaches be applied to the development of ACLS, and (2) Are there benefits to using existing domain models developed as part of individual intelligent tutoring systems in ACLS? I began by implementing a learning environment for peer tutoring as an addition to a successful intelligent tutoring system, the Cognitive Tutor Algebra, and evaluating the benefits of peer tutoring without adaptive support (*Phase 1*). I then added adaptive support for peer tutors in giving tutees correct help, and discovered that while peer tutors benefit from reflecting on their partner's errors, they need additional support in giving tutees conceptual help (*Phase 2*). I designed and implemented adaptive help-giving support for peer tutors, and found positive effects of this support on interaction in a classroom study (*Phase 3*), and on learning in a lab study (*Phase 4*). In order to conduct these phases of research this dissertation has made two technical advances: the development of an architecture for extending intelligent tutors for collaboration (*Development 1*) and the improvement of automated assessment of peer tutor chat (*Development 3*). This dissertation has also explored potential designs for adaptive support that go beyond traditional intelligent tutoring paradigms (*Development 2*).

This work makes both technological and learning sciences contributions. The technological contributions involve demonstrating how individual intelligent tutoring approaches can be used to model collaboration, and what role intelligent tutoring components can play in collaborative models. For example, I have shown that the automated classification of peer tutor behaviors can be improved using problem-solving features, and that collaborative skills can be traced in the same way as problem-solving skills. This work makes learning sciences contributions by increasing understanding of the effects of adaptive support on student collaboration and learning. In two studies I have demonstrated that adaptive support, compared to fixed support controls, improves the quality of the help peer tutors give and improves their domain learning. As part of this work, I add to understanding of the cognitive and motivational mechanisms by which different types of adaptive support impact student collaboration. Overall, this dissertation demonstrates that adaptive collaborative learning support is a promising research direction for improving collaboration quality and domain learning.

Acknowledgments

The biggest influences on this work have been my two advisors, Ken Koedinger and Nikol Rummel. Both have provided me with hours of invaluable guidance and mentorship. I'd like to thank Ken for having a great perspective on the big picture issues, and Nikol for always having a very clear and insightful take on whatever obstacle we were facing. This work has also benefitted from my association with my committee members, Carolyn Rosé and Bob Kraut, who always made themselves very available to meet with me and offer constructive feedback. I thank Carolyn for her unending enthusiasm, and Bob for offering a strong external perspective on the work. I would also like to thank Bruce McLaren for his contributions to the early stages of this work. His involvement encouraged me to jump right into adaptive collaborative learning support research soon after I began grad school. Special thanks go out to Anind Dey, John Zimmerman, and Howard Seltman, who made time to meet with me to discuss various issues related to this dissertation. Also thanks to Vincent Aleven and Chris Jones, who provided me with valuable advice on other projects.

One of the best things about doing my dissertation has been the incredibly supportive community of graduate students. Thanks to Amy Ogan, for always having a opinion, Ruth Wylie, for having a fantastic ability to make things happen, Ian Li, for always making sure that my stuff looks good, Gary Hsieh, for being a great wingman, Ido Roll, for offering the voice of experience, and Dejana Diziol, for looking after me during the summer I spent in Germany. Thanks also to Moira Burke, Scott Davidoff, and Johnny Lee, who have made great travelling companions. Other students who offered guidance and friendship include Turadg Aleahmad, Lisa Anthony, Ryan Baker, Aruna Balakrishnan, Matt Easterday, Chris Harrison, Iris Howley, Gabi Marcu, Anne Meier, Stephen Oney, Martina Rau, Katharina Westermann, Jason Wiese, and the other PhD students.

Doing classroom research is a very difficult endeavor, and I have had a lot of support. Thanks to the people at Carnegie Learning that made everything possible, especially Steve Ritter, Jonathan Steinhart, and Dale Walters, who helped me refactor the Cognitive Tutor Algebra for collaboration. Thanks to Sean Walker for his enormous help with the implementation of the last study. Thanks to Frank Miller, Christy McGuire, Thomas Harris, and Tristan Nixon for their support in data collection in later stages of this project. There has also been a number of people involved with the Pittsburgh Science of Learning Center (PSLC) whose assistance has been invaluable, including Ido Jamar and Michael Bett. A special thanks to Gail Kusbit, who was a major help in organizing the final studies of this dissertation, and Jo Bodnar, who has been able to answer any administrative question I have run into. Thanks to Queenie Kravitz for her cheerful presence in the hallway. Thanks to all the teachers who have participated in my studies, but cannot be named due to anonymity concerns.

Finally, I have endless gratitude for the support of my parents, Joy and Bob Walker, and of my brothers, Adam and Sean Walker. Having all of you around to watch me defend was a fantastic feeling.

Table of Contents

1 Motivation **1**

2 Background **5**
2.1 Introduction **5**
2.2 Potential Benefits of Adding Intelligent Support to Collaborative Learning **5**
2.3 Adaptive Collaborative Learning Support Systems **8**
2.3.1 Design 8
2.3.2 Technology 14
2.3.3 Learning Sciences 17
2.4 Context: Reciprocal Peer Tutoring **19**
2.4.1 Learning from Help-Giving in Peer Tutoring 19
2.4.2 Supporting Help-Giving Behaviors Using a Reciprocal Peer Tutoring Script 21
2.4.3 Promise of Adaptive Collaborative Learning Support in Reciprocal Peer Tutoring 22
2.5 Outlook & Discussion **23**

3 Phase 1: Peer Tutoring Learning Environment **25**
3.1 Introduction **25**
3.2 Design: Basic Peer Tutoring Script + Reflection Elements **25**
3.2.1 Interactions: Basic Script Design 25
3.2.2 Model: Expected Learning 27
3.2.3 Support: Problem Solutions & Reflective Elements 29
3.3 Implementation: Refactoring the Cognitive Tutor Algebra for Collaboration **30**
3.4 Evaluation: Study 1 **31**
3.4.1 Experimental Design 31
3.4.2 Method 31
3.4.3 Results 32
3.5 Outlook and Discussion **36**
3.5.1 Introduction 36
3.5.2 Design 36
3.5.3 Technology 36
3.5.4 Learning Sciences 37
3.5.5 Implications for Iteration 38

4 Development 1: The Collaborative Tutoring Research Lab **40**
4.1 Introduction **40**
4.2 Component Functionality **40**
4.3 Message Protocol **43**
4.4 Component Integration **46**
4.5 Summary of Technical Contribution **49**

5 Phase 2: Adaptive Correction Support **51**
5.1 Introduction **51**
5.2 Design: Adaptive Domain Hints & Feedback **52**
5.2.1 Interactions: Script Cohesiveness 52
5.2.2 Model: Correct Help-Giving 52
5.2.3 Support: Peer-Mediated Hints & Feedback 54
5.3 Implementation: Using CTA Models to Add Adaptive Correction Support **56**
5.3.1 New Tutor Component: Correction Tutor 56

5.3.2 Integration of Components 59
5.3.3 Comparison Conditions 60
5.4 Evaluation: Study 2 60
5.4.1 Experimental Design 60
5.4.2 Method 61
5.4.3 Results & Discussion 64
5.5 Outlook and Discussion 74
5.5.1 Introduction 74
5.5.2 Design 75
5.5.3 Technology 75
5.5.4 Learning Sciences 76
5.5.5 Implications for Iteration 76

6 Development 2: Student Needs & Design Space for Adaptive Support 78
6.1 Introduction 78
6.2 Ideation 78
6.2.1 Reflective Prompts 79
6.2.2 Peer-Mediated Feedback 80
6.2.3 Adaptive Opportunities 80
6.2.4 Adaptive Resources 81
6.3 Speed Dating Process 82
6.4 Results 84
6.4.1. Knowledge of Good Help-Giving Behaviors 84
6.4.2 Accountability and Control Design Principles 85
6.4.3 Relevance Design Principle 86
6.5 Summary of Design Implications 87

7 Phase 3: Adaptive Help-Giving Support 88
7.1 Introduction 88
7.2 Design: Help-Giving Support 88
7.2.1 Interactions: Discussion Scaffolding and Practicality 88
7.2.2 Model: Effective Help-Giving 91
7.2.3 Support: Hints on Demand, Conceptual Resources, and Reflective Prompts 94
7.3 Implementation: Adding Adaptive Help-Giving Support 95
7.3.1 New Tutor Component: Help-Giving Tutor 96
7.3.2 Integration of Components 102
7.3.3 Comparison Conditions 102
7.4 Evaluation: Study 3 104
7.4.1 Experimental Design 104
7.4.2 Method 104
7.4.3 Quantitative Results 106
7.4.4 Qualitative Results 110
7.5 Outlook and Discussion 114
7.5.1 Introduction 114
7.5.2 Design 114
7.5.3 Technology 115
7.5.4 Learning Sciences 115
7.5.5 Implications for Iteration 116

8 Development 3: Assessment of Help-Giving 117
8.1 Introduction 117
8.2 Context 117

8.3 Method **118**
8.3.1 Baseline Classification 118
8.3.2 Incorporating Domain Features 119
8.3.3 Adding Self-Classification 120
8.4 Results **121**
8.5 Summary of Technological Contribution **122**
9 Phase 4: Cognitive and Motivational Benefits of Adaptive Support **124**
9.1 Introduction **124**
9.2 Design: Moving to a Lab Setting **125**
9.2.1 Interactions: Lab Setting 125
9.2.2 Model 125
9.2.3 Support: Targeted Reflective Prompts 126
9.3 Implementation: Improving Adaptivity **126**
9.3.1 New Tutor Component: Help-Giving Tutor 126
9.3.2 Integration of Components 128
9.3.3 Comparison Conditions 128
9.4 Evaluation: Study 4 **128**
9.4.1 Experimental Design 128
9.4.2 Method 129
9.4.3 Results 132
9.5 Outlook & Discussion **136**
10 Outlook & General Discussion **138**
10.1 Introduction **138**
10.2 Design Implications **139**
10.2.1 ITS Approaches to Modeling and ACLS (Q1-D1) 139
10.2.2 ITS Approaches to Support and ACLS (Q1-D2) 140
10.2.3 The Role of Domain Context in Interaction Models and Support (Q2-D1) 141
10.3 Technological Contributions **142**
10.3.1 Adapting Intelligent Tutoring Methodology to Collaborative Activities (Q1-T1) 142
10.3.2 Integration of Existing and Custom Components (Q2-T1) 143
10.3.3 Using Domain Components to Improve Assessment (Q2-T2) 144
10.4 Empirical Results **145**
10.4.1 Benefits of Adaptive Support for Collaboration (Q1-L1, Q1-L2) 145
10.4.2 Intelligent Tutoring Components and Data Analysis (Q2-L1) 147
10.5 Final Thoughts **148**

List of Figures

Figure 1. Four stages of research on collaboration support 6
Figure 2. Context, support, interactions, and outcomes in *APTA* 24
Figure 3. Individual use of the Cognitive Tutor Algebra. 26
Figure 4. Peer tutor's interface.. 27
Figure 5. Abstracted model of learning from tutoring, for the peer tutor 28
Figure 6. Components used in *Phase 1*: *The Peer Tutoring Learning Enviornment*. 30
Figure 7. High level overview of *CTRL*.. 41
Figure 8. Message-passing between two tools, two tutors, and a translator 45
Figure 9. Logging format for student-tutor interaction. 48
Figure 10. "Ideal" model of basic peer tutoring 53
Figure 11. Adaptive correction support 55
Figure 12. Message passing logic in the mediator for *Phase 2*: *Adaptive Correction Support*. 59
Figure 13. Design space for adaptive collaborative learning support 79
Figure 14. Speed Dating scenario. 84
Figure 15. Tutee's problem-solving interface 89
Figure 16. Peer tutor's interface in *Phase 3: Adaptive Help-Giving Support* 90
Figure 17. Model of tutor and tutee helping behavior 92
Figure 18. Message passing in *Phase 3: Adaptive Help-Giving Support.* 102
Figure 19. Peer tutor's interface in fixed support condition. 103

List of Tables

Table 1. Research questions explored in this dissertation 2
Table 2. Work described in the dissertation 4
Table 3. Facets of ACLS design which inform the review of background literature 9
Table 4. Tasks and assistance provided in several ACLS systems 10
Table 5. Facets of ACLS implementation which inform the review of background literature 13
Table 6. Evaluations of ACLS support 17
Table 7. Processes involved in learning from help-giving and help-receiving 20
Table 8. Study procedure in *Phase 1: The Peer Tutoring Learning Environment* 32
Table 9. Domain pretest and posttest scores in *Phase 1* 33
Table 10. Positive interaction in *Phase 1* 34
Table 11. Problem-solving interactions in *Phase 1*. 35
Table 12. Use of reflection exercises in collaboration + reflection condition in *Phase 1*. 36
Table 13. Messages passed between components. 44
Table 14. Model tracing in the correction tutor in *Phase 2: Adaptive Correction Support*. 57
Table 15. Assessment in the correction tutor in *Phase 2* 57
Table 16. Conditions for *Phase 2* study 61
Table 17. Study procedure in *Phase 2*. 62
Table 18. Coding scheme for tutor and tutee dialogue. 64
Table 19. Absolute scores on pretest, posttest, and delayed test. 65
Table 20. Frequencies of student progress variables and correlations with learning. 67
Table 21. Learning opportunity created by tutee feedback 68
Table 22. Percent good help given when needed and bad help given when not needed. 69
Table 23. Conceptual interaction about problem $ay + by + m = n$. 70
Table 24. Example of peer-mediated feedback 71
Table 25. Regression results used to predict student delayed learning in collaborative conditions 73
Table 26. Ideas presented to students as part of Speed Dating 83
Table 27. Three principles for designing adaptive support in a peer tutoring context. 85
Table 28. Assessment in the help-giving tutor in *Phase 3: Adaptive Help-Giving Support* 96
Table 29. Productions in the help-giving model 97
Table 30. Modeling and feedback example from *Phase 3* 100
Table 31. Study procedure in *Phase 3* 105
Table 32. Differences in peer tutoring context across conditions 107
Table 33. Help quality across conditions 107
Table 34. Positive effects of adaptive support on student interaction 111

Table 35. Problem with perceived relevance of adaptive support. 113
Table 36. Selected features created from particular tutor utterances. 120
Table 37. Kappas for the four classifiers 121
Table 38. The top ten ranked features in chi-squared feature selection 122
Table 39. Study conditions in *Phase 4: Cognitive and Motivational Benefits of Adaptive Support* 124
Table 40. Assessment in the help-giving tutor in *Phase 4* 126
Table 41. Production rules in *Phase 4* 127
Table 42. Study procedure in *Phase 4* 129
Table 43. Pretest and posttest scores in *Phase 4* 132
Table 44. Motivational effects in *Phase 4* 133
Table 45. Differences in peer tutoring context across conditions 134
Table 46. Relationship between student interaction and learning 135
Table 47. Differences in interaction variables between condition 135
Table 48. Student responses to reflective prompts across condition 136
Table 49. The findings of this dissertation 139

1 Motivation

Understanding robust learning and how to encourage it in students is a central objective of learning science research. The transfer of learned skills to new problems, the long-term retention of knowledge, and accelerated future learning are all indicators that students have acquired deep conceptual knowledge that might have a stronger effect on their achievement than shallow procedural skills (Bransford & Schwartz, 1999). In particular, to improve mathematical problem-solving skills, it is important to increase student conceptual understanding, which is crucial for the transfer of learned skills to new problems (e.g., Rittle-Johnson & Alibali, 1999). I combine two different technological interventions in an attempt to improve robust learning in mathematics: computer-supported collaboration and intelligent tutoring systems.

The combination of computer-supported collaborative learning and intelligent tutoring technologies can be termed adaptive collaborative learning support (ACLS), and is a considered to be a promising direction of research in both areas. Collaboration has been demonstrated to have a positive effect on individual and group learning outcomes (Lou, Abrami, d'Apollonia, 2001), and in particular to promote deep elaboration of the learning content (Teasley & Fischer, 2008). These effects do not emerge spontaneously, but require the careful structuring of the collaboration so that particular promotive interactions emerge (Johnson & Johnson, 1990). Traditional collaboration assistance has the disadvantage of being unable to adapt to student needs. Assistance that is unnecessary and demotivating for students who are skilled collaborators may provide insufficient guidance to poor collaborators, who without sufficient monitoring do not execute collaborative activities as planned (Kollar, Fischer, & Slotta, 2005; Dillenbourg, 2002; Ritter, Blessing, & Hadley, 2002). The use of intelligent tutoring technology to assess student collaboration as it occurs and provide feedback when it is most needed may therefore be an improvement over fixed techniques (Rummel & Weinberger, 2008).

The combination of collaboration and intelligent tutoring may also improve the effectiveness of the intelligent tutoring systems. Intelligent tutoring systems are systems that compare student actions to models of good and/or poor behavior in order to provide context-sensitive help and individualized problem selection (VanLehn, 2006). They have been successful at increasing learning in a classroom environment by as much as one standard deviation over traditional classroom instruction (Koedinger, Anderson, Hadley, & Mark, 1997). However, the impact of these systems still falls short of the effects achieved by expert human tutors (Bloom, 1984), possibly because the systems have historically focused on procedural learning. Extending intelligent tutoring systems to collaborative activities that encourage conceptual elaboration may further improve the robust learning of students (e.g., Aleven & Koedinger, 2002; Roll, Aleven, McLaren, & Koedinger, 2007). However, investigation into the design and construction of these ACLS systems is generally still at an early stage (see Soller, Martinez, Jermann, & Muehlenbrock, 2005, for a review), and few existing systems have been evaluated for their impact in the classroom. There are several open research areas with respect to how to design and implement ACLS, and how such support affects student interaction and learning (Diziol, Walker, Rummel, & Koedinger, 2009).

This dissertation investigated the combination of intelligent tutoring and collaborative learning by integrating an existing intelligent tutoring system, the *Cognitive Tutor Algebra* (*CTA*), with a reciprocal peer tutoring activity, and then iteratively adding intelligent support for peer tutors participating in the activity. Because little is known about ACLS, related research questions span all stages of an ACLS development cycle:

Design. What collaborative actions should these systems model and support?

Technology. What are the techniques for implementing these systems?

Learning Sciences. Does ACLS indeed improve collaboration and learning?

I investigate these three broad areas with a particular focus on two ways intelligent tutoring technology might facilitate the development of ACLS (see Table 1):

Q1. Where and how can intelligent tutoring *approaches* be applied to the development of ACLS?

Q2. Are there benefits to using existing *components* developed as part of individual intelligent tutoring systems in collaborative systems?

Table 1. Research questions explored in this dissertation. Questions span the design, implementation, and evaluation of adaptive collaborative learning support.

#	Research Question	Design	Technology	Learning Sciences
Q1	Where and how can ITS approaches be applied to the development of ACLS?	How do ITS approaches to modeling (*Q1-D1*) and support (*Q1-D2*) apply?	Can collaborative skills be knowledge traced (*Q1-T1*)?	What are the effects of ACLS on student collaborative interactions (*Q1-L1*) and learning (*Q1-L2*), compared to fixed forms of support?
Q2	Are there benefits to using existing domain models developed as part of individual intelligent tutoring systems in ACLS?	What role does domain information play in collaboration models and feedback (*Q2-D1*)?	How can existing and custom components be integrated (*Q2-T1*)? Can domain components improve assessment (*Q2-T2*)?	How can intelligent tutoring-style data logs augment the analysis of collaborative study data (*Q2-L1*)?

My work has unfolded in four major phases of design, implementation, and evaluation, with a development between each phase to make a research contribution necessary for the next phase to occur (Table 2). In *Phase 1,* I used theories of learning from tutoring to design a reciprocal peer tutoring activity, and then extended the *CTA* to implement the activity. I conducted a classroom study to determine how effectively students tutor each other in this environment, and found that peer tutors struggled to help their partner with the learning content. In *Development 1*, I developed a general technological platform for integrating existing domain components with custom-built collaborative ones, called the Collaborative

Tutoring Research Lab (*CTRL*). This platform supported the work in subsequent phases on adding tutoring to collaboration. In *Phase 2*, I developed a simple model of good peer tutoring, and designed feedback for the peer tutor based on the model. I implemented the model within my framework by leveraging the *CTA* individual models of problem-solving, and evaluated it in a classroom study, comparing the adaptive support to fixed support and individual learning. I found that peer tutors benefitted from reflecting on their partner's errors, but even with sufficient cognitive support, many peer tutors did not understand how to appropriately tutor their partners so that both parties can benefit. In *Development 2*, I used Human-Computer Interaction design methodologies to explore how students perceive adaptive support to prepare me for designing adaptive interaction support for students. In *Phase 3*, I designed a model of good help-giving that could serve as the basis for tutoring support which integrated both interaction information and student problem-solving progress as measured by the *CTA* individual models. After implementing the interaction tutor, I conducted a large-scale classroom study comparing adaptive support (both interaction and domain) to fixed support, and found that peer tutors in the adaptive condition improved the quality of their help as a result of using the system. However, it was clear that the system needed to be more effectively adaptive to truly test the effects of adaptive support, and thus in *Development 3* I improved the assessment of student chat, using problem-solving features. Finally, in *Phase 4*, I explored whether adaptive support is effective because students are more motivated to give better help, or because students are better supported in giving help. I found that a condition where students were given adaptive support and told it was adaptive improved their learning over conditions where students were given fixed support and either told it was adaptive or fixed.

This dissertation makes contributions to the design ACLS interactions, the construction of ACLS technology, and the understanding of whether and why students benefit from automated adaptive support in collaborative scenarios. See Table 1 for a list of the specific research questions surveyed in this dissertation. The remainder of this dissertation is organized as follows. In *Background* (Chapter 2), I survey the relevant research background, including related work in collaboration support, ACLS systems, and the specific reciprocal peer tutoring context. In *Phase 1: Peer Tutoring Learning Environment* (Chapter 3), I describe the learning environment I have built for reciprocal peer tutoring. In *Development 1: Collaborative Tutoring Research Lab* (Chapter 4), I describe an architecture for adding ACLS to collaborative learning environments. In *Phase 2: Adaptive Correction Support* (Chapter 5), I describe adaptive support provided to help peer tutors give more correct help and reflect more on problem-solving steps. In *Development 2: Principles for Adaptive Collaboration Support* (Chapter 6), I describe a design exploration of what collaborating students need from adaptive support and how they perceive adaptive support. In *Phase 3: Adaptive Help-Giving Support* (Chapter 7), I describe the support provided to improve the conceptual content of peer tutor help. In *Development 3: Assessment of Help-Giving* (Chapter 8), I describe the improvement of the automated classification of peer tutor help using problem-solving features. In *Phase 4: Cognitive and Motivational Benefits of Adaptive Support* (Chapter 9), I explore why adaptive

support is beneficial for student collaborative learning. I discuss the results in *Outlook and Discussion* (Chapter 10).

Table 2. Work described in the dissertation. Work occurred in four major phases, with a design, technology, and learning sciences component to each phase. Between each phase was a development, representing a research contribution that needed to be made for the work to continue. "C#" stands for chapter number.

Title	Design	Technology	Learning Sciences	C#
Phase 1: Peer Tutoring Learning Environment	Designed reciprocal peer tutoring script.	Extended *CTA* to support peer tutoring interactions.	Conducted pilot, discovered that peer tutors need domain support.	3
Development 1: Collaborative Tutoring Research Lab		Developed architecture for integrating domain and collaborative models.		4
Phase 2: Adaptive Correction Support	Modeled correction aspects of peer tutoring and designed support.	Combined *CTA* models with simple correction tutor.	Compared adaptive & fixed support. Peer tutors benefitted from reflection but help given was poor.	5
Development 2: Principles for Adaptive Collaboration Support	Generated principles for support design.			6
Phase 3: Adaptive Interaction Support	Modeled help-giving, designed multiple forms of interaction support.	Implemented help-giving model and knowledge tracing of collaboration.	Compared adaptive & fixed support. Adaptive support improved help quality.	7
Development 3: Assessment of help-giving		Used domain features to improve machine classification of student help.		8
Phase 4: Cognitive and Motivational Effects of Support		Incorporated machine learning models into system.	Compared actual adaptive support to told adaptive support and random support. Adaptivity improved learning.	9

2 Background

2.1 Introduction

This background has three content sections and a summary section. First, I discuss in more detail why there may be benefits to providing intelligent support to collaborative learning activities (*2.2*). Second, I survey the ACLS systems that already exist (*2.3*). Third, I describe the specific context for my system in more detail, including the Cognitive Tutor Algebra and the benefits of reciprocal peer tutoring activities (*2.4*). I conclude by summarizing how the surveyed research informs the work described in this dissertation (*2.5*).

2.2 Potential Benefits of Adding Intelligent Support to Collaborative Learning

Over the past 30 years there has been an evolution in research on how students learn by collaborating, depicted in Figure 1 (Dillenbourg, Baker, Blaye, & O'Malley, 1995). In the *context stage,* early work compared the effects of collaborative and individual activities, or looked at how the context of collaboration related to learning and attitudinal outcomes (see Slavin, 1996, for a review). However, to better understand the effects of collaboration, it is important to model collaborative interactions and relate them to outcomes (the *interactions stage*). By looking at student collaborative interactions and resulting cognitive processes, researchers have concluded that through participation in collaborative activities students socially construct knowledge (Schoenfeld, 1992). When students articulate their reasoning they can elaborate on their existing knowledge and build new knowledge (Ploetzner, Dillenbourg, Preier, & Traum, 1999); when they listen to other people's ideas, they integrate them with their own knowledge (Stahl, 2000), reflect on their own misconceptions, and work toward a shared understanding (Van den Bossche, Gijselaers, Segers, & Kirschner, 2006). However, for collaboration to be effective at engaging these processes, students need to display a variety of positive collaborative behaviors (Johnson & Johnson, 1990), ranging from providing each other with help, feedback, and relevant resources to challenging each other's conclusions to promoting and being motivated to strive for mutual benefit. Students do not generally exhibit these positive behaviors spontaneously (Lou et al., 2001).

Thus, it further became relevant to determine how to support collaboration in order to produce the desired interactions, which would then hopefully lead to the desired learning outcomes (Strijbos, Martens, & Jochems, 2004). Much current collaborative learning research is situated in this *fixed support stage* (see Figure 1), which focuses on the effects of giving students *fixed* assistance, including declarative instruction and training on how to collaborate (e.g., Prichard, Stratford, & Bizo, 2006; Saab, Van Joolingen, & Van Hout-Wolters, 2007), examples of good collaboration (e.g., Rummel & Spada, 2005), and collaboration scripts that provide students with designated roles and activities as they work together (e.g., Fischer, Kollar, Mandl, & Haake, 2007). In fixed assistance, students may not be capable of or motivated to follow the instructions given, and thus may not engage in collaborative activities as they were designed (Ritter, Blessing, & Hadley, 2002). In a face-to-face collaboration context, it is difficult for these techniques to

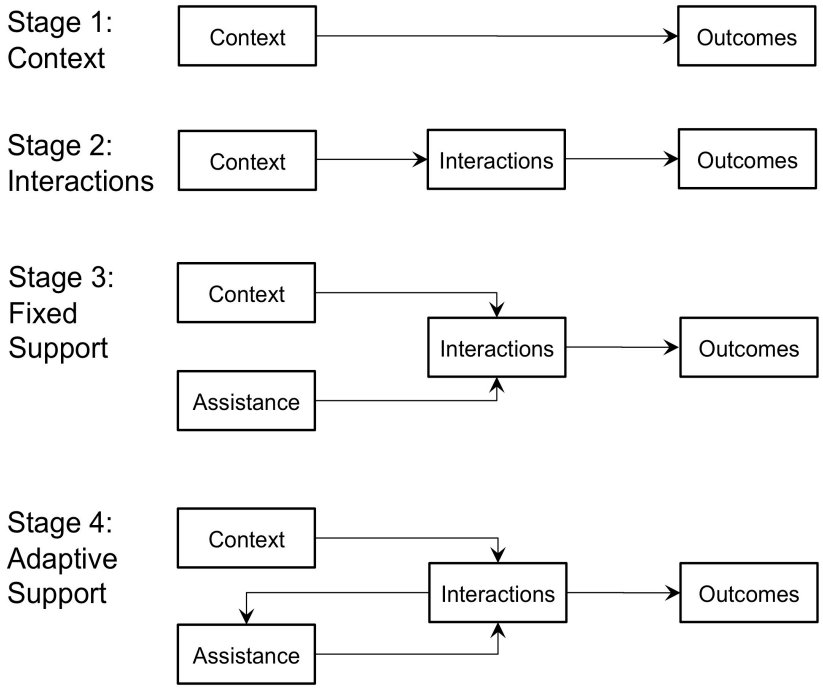

Figure 1. Four stages of research on collaboration support. Context and assistance are linked to collaborative interactions, which are linked to learning and motivational outcomes.

ensure that they do so. An increase in the presence of computer-mediated collaborative activities in the classroom has changed the way collaboration can be structured, as script elements can be embedded in the interface: Roles can manifest themselves through the types of collaborative actions students can perform using the system, phases can be strictly enforced, and prompts can take the form of sentence classifiers or starters, where students complete open-ended sentences such as "I agree, because…" However, this increase in support comes with a potential decrease in motivation, as this level of support can overstructure collaboration for students who already know how to collaborate (Kollar, Fischer, & Slotta, 2005). Further, students often fail to comply with script elements such as sentence starters (Lazonder, Wilhelm, & Ootes, 2003), perhaps because they do not know how to use them effectively or are not motivated to do so. For example, if students repeatedly use sentence classifiers to type something off-topic, such as "I agree because… I'm getting hungry," this is unlikely to contribute to a beneficial interaction.

There has been a movement toward developing *adaptive* assistance for collaboration, where collaborative interactions are modeled as they occur, and the results of the analysis determine the content of the assistance given (*adaptive support stage* in Figure 1). Automated adaptive support, as in the kind provided by intelligent tutoring systems, might be a better way of targeting the individual needs of students and increasing their benefits from collaboration (Soller, Martinez, Jermann, & Mühlenbrock, 2005; Kumar, Rosé, Wang, Joshi, & Robinson, 2007; Rummel & Weinberger, 2008). An adaptive implementation of a collaboration script would vary the script based on the needs of particular groups or individuals, ensuring

that students get support on improving their collaboration at the moments they need it. The intelligent system could also verify that students are indeed complying with the script and improving their collaborative interactions. This type of support is very similar to scenarios where humans facilitate collaborative interactions, which are very effective at improving the collaboration of groups (Hmelo-Silver, 2004; Michaels, O'Connor, & Resnick, 2008). However, these scenarios are resource intensive, as they require an expert facilitator to guide each group's discussion. Using an intelligent system to provide the adaptive support would be less resource intensive, but might have similar benefits. Studies comparing automated adaptive support to fixed support have indeed been promising (Baghaei, Mitrovic, & Irwin, 2007; Kumar et al., 2007), but research into ACLS is still at an early stage: Few ACLS systems have been developed, and only a small percent of the systems that have been developed have been evaluated for their effects on interaction and learning.

In examining how to design and develop ACLS, it may be beneficial to leverage the extensive research on intelligent tutoring systems for individual learning (see VanLehn, 2006, for review). In intelligent tutoring, students typically interact one-on-one with an intelligent system. Students are given multiple tasks to accomplish, and for each task, are asked to perform several problem-solving steps. During this process, the tutoring system provides *corrective feedback* through indicating whether a step is correct, providing direct feedback on an incorrect step, or reviewing a student solution. Providing individualized feedback at the right time can help students to construct new knowledge based on their problem solving steps. The system can also take on an assistive role by providing a conceptual hint on the next step, thus helping students to overcome problem-solving impasses. Finally, the system provides a step-by-step assessment of the knowledge students hold – either at a course-grained level (e.g., problem by problem) or at a fine-grained level (e.g., skill by skill). This dissertation work draws heavily from the *cognitive tutor* approach to intelligent tutoring, as exemplified by the Cognitive Tutor Algebra. The Cognitive Tutor Algebra (*CTA*) is the intelligent tutoring system component of a complete high-school Algebra course that has been shown to increase student learning by approximately one standard deviation over traditional classroom instruction (Koedinger et al., 1997). It maintains a production-rule model of good and bad problem-solving steps, compares student behaviors to that model, and provides feedback and next-step instruction as appropriate. In a technique termed *model tracing*, the intelligent system generates the next correct steps for any given problem state, as well as a set of incorrect steps that students who hold particular misconceptions might take. The system then uses *knowledge tracing* to assess student skills and select problems tailored to individual student needs. The *CTA* is used in over 2600 classrooms across the United States (www.carnegielearning.com), making it an ideal platform for collecting large amounts of data and conducting externally generalizable research studies. In general, intelligent tutoring systems have evolved from acting as isolated interventions to serving as platforms for future research (Koedinger, Aleven, Roll, & Baker, 2009). For example, Project LISTEN's *Reading Tutor* supports the incremental addition and evaluation of features, and the collection of rich log data that can later be mined to provide insight into student learning processes (Beck, Mostow, & Bey, 2004).

As described in the introduction, this dissertation uses two broad approaches to draw from research on individual intelligent learning in order to design, develop, and evaluate ACLS. The first approach examines whether intelligent tutoring *techniques* might be appropriate for providing assistance to collaborating students. With respect to the above techniques, we examine whether model tracing, knowledge tracing, and corrective feedback, might apply to student collaboration at both the design and implementation phases. The second approach examines the role of intelligent tutoring *components* in assistance to collaborating students. We examine how components developed as part of existing intelligent tutoring systems can be used to model collaboration and provide support. In the following section, we review current work in ACLS, with a particular focus on how individual intelligent tutoring components and approaches have informed their development.

2.3 Adaptive Collaborative Learning Support Systems

This section surveys related work on adaptive collaborative learning support (ACLS). The types of systems of primary interest are *coaching systems,* as defined by Soller and colleagues (2005) in their review of collaboration support systems. Coaching systems help students who are engaged in computer-mediated collaboration by assessing the current state of student interaction, comparing the current state to a desired state, and then offering assistance to the students. Coaching systems have a lot in common with intelligent tutoring systems, which also support students using the three phases of assessment, comparison, and assistance, but focus on individual learning. As described in the previous section, cognitive tutoring systems can also assess student knowledge and adaptively tailor activities to their level of ability, functionality not currently present in coaching systems. I focus my review in this section on coaching systems for ACLS that have been implemented and evaluated. Further, I examine how individual intelligent tutoring approaches and components might contribute to the development of ALCS.

2.3.1 Design

The description of the design of ACLS systems encompasses the *interactions* students have with each other and with the system, the conceptual *model* of effective and ineffective student interaction, and the *support* provided to students. I use this structure throughout the dissertation (see Table 3).

Interactions. ACLS systems support both collaborative task actions and computer-mediated conversation (see Table 4 for a summary of interactions enabled by ACLS systems). Often, student interactions are structured either using micro-scripts, which operate on an action-by-action basis, or macro-scripts, which operate on the level of phases of activity (see Dillenbourg & Hong, 2008, for further discussion). In this work, of particular interest is micro-scripts, which structure interactions within a phase of collaborative activity. ACLS systems tend to include a shared workspace where students can work together toward a domain goal. Micro-scripts are often applied to these shared workspaces by giving students different roles in the workspace or by allowing them only to act at particular times. For example, as summarized in Table

Table 3. Facets of ACLS design which inform the review of background literature. In later chapters, I describe the design of various iterations of the system using these three facets.

Facet	General Description	Phase 1 (Chapter 3)	Phase 2 (Chapter 5)	Phase 3 (Chapter 7)	Phase 4 (Chapter 9)
Interactions	Student interactions with each other & systems	Basic script design	Script cohesiveness	Discussion scaffolding & practicality	Lab setting
Model	Effective & ineffective student behaviors	Expected learning	Correct help-giving	Effective help-giving	N/A
Support	Assistance by system to students	Problem solutions & reflection	Peer-mediated hints & feedback	Hints on demand, conceptual resources, & reflective prompts	Targeted reflective prompts

4, *COLER* contains a shared workspace where students can collaboratively construct entity-relationship diagrams by interacting with coupled nodes and edges (Constantino-González, Suthers, & Escamilla de los Santos, 2003). Students have to indicate their intention to draw in the workspace, and when one student is drawing the other students cannot. Learning systems that have a shared workspace also often include a private workspace that contains no coupled objects, so that students can do individual work. The other primary component of many implemented ACLS systems is a text-based tool that allows students to communicate with each other in natural language. Within these tools, micro-scripts are often applied through the use of sentence-starters that students select to begin their utterance (e.g., "I would like to explain that…") or classifiers that student select after typing their utterance (e.g., "Give an Explanation"). As described in Table 4, *Group Leader* currently has 46 sentence openers that represent 10 subskills students should be exhibiting while collaborating, such as "Task Leadership" (Israel & Aiken, 2007). Finally, interfaces may contain widgets such as buttons through which the students can get information from the intelligent system. For instance, students can request four different types of help from *HabiPro*: clues to the solution, a worked example of the current problem, a worked example for a different problem, and the solution to the problem (Vizcaíno, Contreras, Favela, & Prieto, 2000). Thus, the systems log collaborative task actions, verbal interactions, and meta-interactions that arise as a result of following micro- and macro-interaction scripts.

Table 4. Tasks and assistance provided in several ACLS systems. The goals described are information pooling (IP), reciprocal interaction (RI), dialogue management (DM), task orientation (TO), reaching consensus (RC), and domain learning (DL).

System	Interactions	Modeling Goals	Assessment Method
COLER (Constantino-González et al., 2003)	Modeling, shared & private workspace, chat (classifiers)	IP, RI	Solution structure, individual contributions
COLLECT-UML (Baghaei et al. 2007)	Modeling, shared & private workspace (phases), chat (classifiers)	IP, RI, DM, DL	Solution structure, individual contributions, solution quality
COMET (Suebnukarn & Haddawy, 2004)	Medical problem-based learning in shared workspace, chat (unstructured)	IP, RI	Action counts, action sequences, student expertise, solution quality
CycleTalk (Kumar et al., 2007)	Shared workspace (different phases), unstructured chat	RI, TO, DL	Chat counts, keywords in chat, parsing of chat
Group Leader (Israel & Aiken, 2007)	Programming with chat (sentence openers)	RI, DM, RC	Count dialogue acts, keywords, sequences of disagreement
HabiPro (Vizcaíno et al., 2000)	Editing computer programs using chat, shared workspace	IP, TO, RI, DL	Solution quality, individual expertise, help type requested, chat counts, keywords
LeCS (Rosatelli & Self, 2004)	Case study (phases) in chat (sentence openers), shared text editor, solution representation	RI, DL	Length of time to complete a step, chat counts, solution
MArCo (Tedesco, 2003)	Graphical planning in shared workspace, chat (dialogue games)	RC	Logical conflict between student utterances
OXEnTCHÊ (Vieira et al., 2004)	Chat (sentence starters)	TO, RI	Chat counts, keywords

Modeling. Like intelligent tutoring systems, current ACLS systems assess collaboration based on targeted aspects of student interactions, compare the assessment to ideal collaborative qualities, and then provide feedback based on the comparison (see Table 4 for an overview). ACLS systems have broad commonalities with respect to collaborative skills targeted and how the skills are assessed in the context of the system. In fact, the types of support provided by ACLS can be described using a collaboration analysis scheme developed by Meier and colleagues (Meier, Spada, & Rummel, 2007), where student interaction is rated on 9 dimensions. Some systems attempt to improve student interaction on Meier and colleagues' dimension of *information pooling* (IP), i.e. how much students share their knowledge with their groupmates

(Constantino-González et al., 2003; Baghaei et al., 2007). As represented in Table 4, assessment on this dimension is drawn from workspace actions: Student actions in a public workspace are compared to their actions in a private workspace in order to evaluate how much of their individual actions they are sharing with the group. Some systems instead support Meier and colleagues' dimension of *dialogue management* (DM), or how students execute conversational acts. Assessment in this area is based on chat actions; sentence classifiers are used to count utterances of particular types or even create a model of student dialogue acts and compare it to a sequence of ideal dialogue acts. Then, drawing from earlier analysis systems such as *EPSILON* (Soller, 2004), the ACLS system can give feedback to students based on their contributions (e.g., Israel & Aiken, 2007). Some of the systems described in Table 4 help students in *reaching consensus* (RC; encouraging students to engage in productive conflict) by detecting and responding to loops of disagreement. There is also a growing trend toward using machine learning to classify student utterances instead of (or in addition to) sentence starters, with some success (e.g., Kumar et al., 2007). These efforts have mostly focused on *task orientation* (TO), making sure students discuss particular topics with the goal of increasing the conceptual depth of the discussion. Up until now, we have discussed supporting either workspace actions or chat actions, but not both. Even systems that use metrics of assessment that might apply to both types of interactions often focus their analysis on either one. For example, a common dimension targeted for assistance is *reciprocal interaction* (RI), or whether everyone in a collaborative group is participating. Systems track actions in the shared workspace (e.g., Constantino-González et al., 2003), chat contributions (e.g., Vieira, Teixeira, Timóteo, Tedesco, & Barros, 2004), or the length of time since students have contributed last (Rosatelli & Self, 2004) in order to assess this dimension. However, systems do not generally use all three metrics at once. In addition to providing collaboration feedback on aspects of student interaction, some systems also provide task-related feedback that targets *domain learning* (DL). This feedback is generally provided in a manner similar to individual learning systems. For example, *CycleTalk* (Kumar et al., 2007) engages collaborating students in tutorial dialogues that are identical to those used for individual learners.

There are two areas where intelligent tutoring could be further applied to extend research on modeling in ACLS. First, one common methodology in intelligent tutoring systems is to model a sequence of problem-solving steps, where students have particular actions that they should take when particular conditions are met. This production-style modeling is used in cognitive tutoring systems. However, most of the current modeling in ACLS overlap more with another intelligent tutoring technique: A constraint-based approach where a problem-solving state is checked against particular constraints. It has been suggested these approaches are complementary (Mitrovic, Koedinger, & Martin, 2003), but a constraint-based approach might be more appropriate for domains that are ill-defined, like collaboration (Mitrovic & Weerasinghe, 2009). However, little has been done to investigate whether (and which aspects of) collaboration can be modeled as a sequence of steps, and whether there are benefits to this approach. *Q1-D1* ("How do ITS approaches to modeling apply to ACLS"; see Table 1) investigates whether this approach appears to be useful, and where modeling assumptions have to be relaxed to make this approach

viable. Next, intelligent tutoring domain models are rarely used to augment collaboration models in ACLS, even when it would make sense to do so. *COLLECT-UML* (Baghaei et al., 2007) provides students with both task-related feedback on the quality of their group solution and prompts to contribute elements from their individual solutions to their group solution. However, the system does not provide information on whether the elements students have *not* shared with the group are correct or incorrect. This knowledge would augment the system's capabilities to provide relevant feedback: The system could suggest that students only share the correct elements with their group, or even suggest that students ask their group why an element in their individual solution is incorrect. One system that does integrate domain and collaboration information is *COMET* (Suebnukarn & Haddawy, 2006), where the next participant in a collaborative dialogue is selected based in part on which student has the domain expertise to make a contribution. The effect of this assistance on users has not been explored. Research on cognitive tutors (and intelligent tutoring systems more generally) has recently begun to explore the integration of task-related modeling with metacognitive modeling (Koedinger et al., 2009), and this type of integration could certainly be beneficial in collaborative scenarios as well. I explore this line of research in *Q2-D1* ("What role does domain information play in collaboration models and feedback").

Support. In designing ACLS researchers have mainly adapted individual learning paradigms to providing support, such as providing explicit feedback directly to the unproductive collaborator (see Soller, Martinez, Jermann, & Mühlenbrock, 2005, for review). As in individual intelligent tutoring systems, the timing of feedback tends to vary; some feedback is triggered by user actions (Tedesco, 2003), some is triggered by user inaction (Constantino-González et al., 2003), some is provided on demand (Vizcaíno et al., 2000), and some is only provided when a user submits a solution (Baghaei, Mitrovic, & Irwin, 2007). However, the presentation and target of feedback remains fairly constant; feedback tells ineffective collaborators explicitly which aspects of their collaboration are ineffective, and how they should correct it. For example, the system *COLLECT-UML* responds to a lack of elaboration by saying: "You seem to just agree and/or disagree with other members. You may wish to challenge others' ideas and ask for explanation and justification" (Baghaei et al., 2007). This form of feedback has been demonstrated to be successful in individual learning (e.g., Koedinger & Aleven, 2007), as students can immediately reflect on how the feedback applies it to their current activity and make appropriate changes to their behavior. Additionally, ACLS systems tend to keep different types of feedback separate by design, with each type appearing to students at different times during the collaboration. For example, *GroupLeader* (Israel & Aiken, 2007) has three types of feedback: get back on topic, incorporate a single idea per post, or re-evaluate a conflict. In the system, there is never a case where it is appropriate for the different types of feedback to be combined or given at the same time, avoiding the issue of how to decide between multiple feedback options. Even systems that have access to both collaboration and domain feedback like COLLECT-UML keep their feedback separate.

Thus, we see two places to explore the relationship between intelligent tutoring and support design in ACLS. First, while feedback in ACLS does borrow heavily from intelligent tutoring techniques, these principles may not in fact be appropriate for collaborative scenarios. In fact, Kumar and colleagues (2007) found that students tended to ignore adaptive prompts while collaborating. It might be that students ignore adaptive feedback because it appears irrelevant to their task, or violates other Gricean maxims of the conversation (Bernsen, Dybkjær, & Dybkjær, 1997). If the feedback is perceived as intrusive and critical, it might also threaten their sense of control (Nicol & Macfarlane-Dick, 2006), or disrupt their belief that interpersonal risk taking is safe in a collaborative context, an important contributor to effective team learning behaviors (Van den Bossche, et al., 2006). In fact, two recent studies have found that including socially sensitive features of adaptive support are indeed important for getting positive outcomes from the adaptive support (Chaudhuri, Kumar, Howley, & Rosé, 2009; Kumar, Ai, Beuth, & Rosé, 2009). In general, intelligent tutoring paradigms are perhaps over-used in collaborative learning support, and it may be that new principles of support need to be established (exploring *Q1-D2*; "How do ITS approaches to support apply to ACLS?"; see Table 1). Second, like in modeling, it may be beneficial to integrate domain and collaboration feedback, so that students can see how the collaboration support they receive applies to

Table 5. Facets of ACLS implementation which inform the review of background literature (similar to Table 4). In later sections, I describe the implementation of the system using these facets.

Facet	General Description	Phase 1 (Chapter 3)	Phase 2 (Chapter 5)	Phase 3 (Chapter 7)	Phase 4 (Chapter 9)
Assessment	Collection & aggregation of behaviors	N/A	Tutee actions and CTA evaluation	CTA evaluation, self-labeling, machine labeling	CTA evaluation, self-labeling, machine labeling
Model Tracing	Comparison of current & desired state	N/A	8 rule production model of correction	16 rule production model of help-giving	20 rule production model of help-giving
Knowledge Tracing	Assessment of collaboration skills	N/A	N/A	Trace 4 skills: timely, targeted, elaborated, & classifier use	Trace 5 skills: timely, prompts, error feedback, conceptual, & classifier use
Support Construction	Support based on model & knowledge tracing	N/A	Combined CTA help & collaborative prompt	Prompts given when skills fall within defined thresholds	Prompts given when skills fall within defined thresholds.
Component Integration	Integration of tutoring components with system	Refactored CTA for peer tutoring	Used CTA as input to correction tutor	Used CTA & text-classification as input to help-giving tutor	Used CTA & text-classification as input to help-giving tutor

the problem that they are working on (*Q2-D1*; "What role does domain information play in collaboration models and feedback?"). In this dissertation, I explore the application of intelligent tutoring principles to feedback and the use of domain information in feedback.

2.3.2 Technology

When discussing the implementation of ACLS, I am most interested in the implementation of the *tutor components* that provide support to the collaborating students. The implementation of these components encompasses the *assessment* of collaborative behaviors, and then the *model tracing* (models of collaborative problem-solving) and *knowledge tracing* (tracking of collaborative skills) that use those assessments to provide support to students. Here, I make a distinction between the idealized models of student behavior found in the *design* section and the instantiation of those models in the actual ACLS system. This *implementation* is what provides an assessment of the effectiveness of student behavior based on the desirability of the current problem state or interaction sequence. In addition to discussing the tutor components, I also discuss from a technical standpoint how they are integrated into a larger system for facilitating collaboration. This structure is maintained throughout the paper (see Table 5).

Tutor Components. One way of assessing the quality of student interactions is by tracking student dialogue patterns, commonly accomplished by asking students to indicate the type of contribution that they are making before they compose it. For example, students may select a sentence starter like "We need to work together on this…" to begin their utterance. Based on the starters that students select, the system can make inferences about what students are saying, and use these inferences to provide feedback (Tedesco, 2003). However, students do not consistently select sentence starters or classifiers that match the content of their utterances, and therefore the inferences that the system makes based on those labels can be inaccurate (Lazonder, Wilhelm, & Ootes, 2003). Thus, automated dialogue assessment solutions are beginning to be developed (Israel & Aiken, 2007). So far this technology has only been used successfully in limited ways, such as for classifying the topic of conversation (Kumar et al., 2007), or for assessing student accuracy when they use sentence starters (Israel & Aiken, 2007). Some researchers try to circumvent the problems of assessing dialogue by relying on simple metrics like participation to trigger feedback. For instance, these systems evaluate the amount or length of contributions collaborators make to a shared workspace or to a dialogue and support the interaction by directly encouraging the non-contributors to participate more (Constantino-González et al., 2003). Unfortunately, the same assessment metrics cannot be used to give students feedback on how to participate, which may ultimately be more valuable. These assessments are then used as input to computational models of student collaboration. The representation of ideal student performance varies between systems, ranging from finite state machines (Israel & Aiken, 2007) to decision trees (Constantino-González et al., 2003) to constraints (Baghaei et al., 2007). In fact, a large proportion of research in ACLS has been in this area of appropriate representations for student collaboration. Despite this

focus, it is work on model tracing that has evolved, and to our knowledge there has been no knowledge tracing of student collaborative skills.

Here in the implementation of tutoring components, there are again avenues for future research in terms of applying intelligent tutoring techniques and components to collaborative systems. Specifically, it may be beneficial for researchers to explore the knowledge tracing of collaborative skills in addition to feedback on collaborative steps, and I make this innovation in my research (*Q1-T1*; "Can collaborative skills be knowledge traced?"; see Table 1). Second, it might be that domain components can be used to improve the machine classification of student help (*Q2-T2*; "Can domain components improve assessment?"). This leveraging of domain components has been applied successfully in asynchronous collaborative contexts (Wang et al., 2007), and domain features have been successfully used to enhance the ability of automatic classifiers in other fields (e.g., Dybowski, Laskey, Myers, & Parsons, 2003).

Component Integration. In addition to implementing tutoring components to assist students in collaborating, a critical part of ACLS systems involves the integration of different components to create the complex systems. Many coaching systems (Soller et al., 2005) use a component-based architecture, which can enable the easy modification of an existing system and the reuse of system modules in novel configurations. In component-based architectures, software is divided into abstract components that can be specified to suit the developer's needs and that can be flexibly integrated with other components using a standard framework (Krueger, 1992). At a minimum, the way a system is divided into components has an impact on reuse, because each component can be enhanced or replaced without having to modify the other components. As ACLS systems are distributed applications with multiple users, one common implementation of these systems follows a client-server architecture, with an interface client provided for each student and a central server containing multiple components responsible for managing the collaborative sessions (e.g., Baghaei et al., 2007; Tedesco, 2003; Vizcaíno et al., 2000). Collaboration between interface clients is often facilitated using a "what you see is what I see" policy, where objects are coupled in shared workspaces so that an action taken on a coupled object in one user's client is broadcasted to the parallel objects in collaborators' interfaces (Suthers, 2001). Similarly, text-based interaction tends to follow a traditional instant messaging format, where a user's partner sees every utterance he or she submits (e.g., Vieira et al., 2004). The tutoring functionality of these systems is then generally located on the server. Many systems subdivide the tutoring module into different components, and although the components are named differently across systems, the underlying purpose is parallel across systems. ACLS systems generally include an *expert model*, which compares student actions to an ideal model of collaboration, and a *feedback model*, which contains the logic for how feedback should be delivered to students (e.g., Kumar et al., 2007; Israel & Aiken, 2007; Tedesco, 2003). The two components handle all types of support the system offers. For instance, in the case of *COMET*, they support both information pooling and reciprocal interaction (Suebnukarn & Haddawy, 2006). One or more *translator* components are sometimes also included to convert the low-level user actions into high-level representations of their collaboration that can

be input to the expert model (e.g., Kumar et al., 2007, Israel & Aiken, 2007, Vieira et al., 2004). A variation of this approach to developing a tutoring module is to include both individual expert models and a group expert model on the server, with the group model being either a parameterization of the individual models (Hoppe, 1995) or containing its own specifications for good collaboration (Baghaei et al., 2007).

Based on this description of components, the reuse facilitated primarily involves the ability to modify one aspect of tutor functionality without altering other aspects of tutoring functionality. For example, Kumar and colleagues (2007) discuss how their expert model, translator, and feedback model are separate from each other, such that each component then can be iteratively improved without altering any others. However, another way to facilitate reuse is by adding new components directly to existing configurations: In *COLLECT-UML* (Baghaei et al., 2007), group modeling components are added to augment the individual modeling components already present. Once an integration framework has been developed for the components, they can be more easily substituted for one another or combined in novel ways. For instance, Mühlenbrock and colleagues have created an integration framework where individual user interfaces register with the *DALIS* server, which then invokes a pre-specified set of support agents (Mühlenbrock, Tewissen, & Hoppe, 1998). Essentially, the *DALIS* server acts as the facilitator in a federated system (Genesereth, 1997). Similarly, *LeCS* treats tutors as clients, with a central facilitator managing the interaction between tutor clients and interface clients, although with no explicit integration framework (Rosatelli & Self, 2004). Although the described designs for reuse can make it easier to increase the sophistication of a single type of adaptive support, they do not necessarily facilitate the integration of multiple types of adaptive support and the efficient implementation of comparison conditions. Few ACLS systems specifically include multiple tutor components which each provide a different level of tutoring. One exception is *COLER*, which includes three expert model tutoring components: a "Participation Monitor", a "Difference Recognizer", and a "Diagram Analyzer" (Constantino-González et al., 2003). This division of tutoring components by functionality can make it easier to incrementally add tutoring complexity by integrating multiple tutoring components, particularly if there is a framework in place so that new tutoring models can be integrated with existing tutoring models.

As a main goal of this dissertation is to integrate existing domain models with custom-built collaborative models, a large technical challenge is to develop a framework that enables this integration (investigating *Q2-T1*; "How can existing and custom components be integrated?"). Here again, we look to individual intelligent tutor architectures, which are structured so that custom-built interface and tutor components can be integrated with existing components. This type of reusability can be found in Ritter and Koedinger's (1996) component-based framework for facilitating the development of intelligent tutoring systems. Framework components are divided into tools and tutors, and a standard protocol for interchanging messages is defined to make it easier to swap different components in and out. So that off-the-shelf components can be used, the framework also includes a translator component to convert messages sent from the off-the-shelf components into the standard format, and convert messages sent to the component into a format that it understands. Although Ritter and Koedinger (1996) demonstrated how the

Table 6. Evaluations of ACLS support. Evaluations range from technical validations of the models behind the systems, usability studies of interactions within the system, and controlled experiments to evaluate student learning.

System	Evaluation Purpose	Evaluation Specifics
COLER (Constantino-González et al., 2003)	Feedback validation	Expert ratings of system support, comparison of expert & system support
COLLECT-UML (Baghaei et al., 2007)	Controlled experiment	2 conditions (adaptive collaboration support vs. no collaboration support), classroom study, effects on learning and interactions
COMET (Suebnukarn & Haddawy, 2004)	Model validation	Predict individual & group solution paths
CycleTalk (Kumar et al., 2007)	Controlled experiment	2 (collaborative, individual) x 3(adaptive, static, no support) design, classroom study, effects on learning & interactions
GroupLeader (Israel & Aiken, 2007, McManus & Aiken, 1996)	Model validation, usability study	Assess student dialogue acts, single-condition evaluation of the effects of the system on learning
HabiPro (Vizcaíno et al., 2000)	Model validation	Assess need for assistance, off-topic behaviors, & passivity
LeCS (Rosatelli & Self, 2004)	Design-related study	Students use a non-adaptive system to inform design
MArCO (Tedesco, 2003)	Usability study	Students use adaptive and non-adaptive versions of the system to explore its effects
OXEnTCHÊ (Vieira et al., 2004)	Usability study	Usability, student ratings of system assistance

framework could be used with two separate tutoring applications, their emphasis was on the use of off-the-shelf applications for individual tutoring, rather than on the addition of metacognitive or collaborative components. However, further iterations of the cognitive tutor (e.g., the *Help Tutor*) have experimented with using a similar framework to add metacognitive tutoring; the *Help Tutor* module was added to the traditional cognitive tutor, and feedback from the two tutor modules were integrated as necessary (Aleven et al., 2004). Allowing multiple tutors, and providing an integration framework for the tutors, might allow us to provide more complex tutoring to collaborating students.

2.3.3 Learning Sciences

Much of the evaluation of ACLS systems has been conducted on the technological aspects rather than on the effects of the assistance on student interactions and learning outcomes. See Table 6 for a summary of the evaluations that have been conducted on ACLS systems. In some cases, a technological evaluation

meant evaluating the effectiveness of the collaborative assessment. For example, Mühlenbrock (2004) in evaluating *CARDDALIS* described how well the model represented the student interactions. In other cases, it meant evaluating the predictive power of the models used. *COMET* used kappa to demonstrate the relationship between expert-constructed group solutions and system-predicted group solutions, with positive results (Suebnukarn & Haddawy, 2004). Finally, sometimes feedback itself was evaluated. For an evaluation of *COLER*, 73% of the advice the system provided to collaborative students was rated as "worth saying" by an expert.

Research that has not focused directly on validating the system technology has tended to fall under the category of design-related and usability studies rather than controlled experiments. To inform the development of the adaptive component of *LeCS*, data from dyads interacting using the *LeCS* interface were collected and analyzed (Rosatelli & Self, 2004), and after *OXenTCHÊ* had been implemented, the usability and the benefits of the assistance were rated by student users (Vieira et al., 2004). The few full studies that have been conducted using adaptive systems have been promising. As described in Table 6, to evaluate *COLLECT-UML*, Baghaei and colleagues (2007) compared an adaptive collaboration support condition to a no support condition and found that while there were no differences in domain learning gains, the experimental condition gained more collaborative knowledge. Even more encouraging was the study conducted by Kumar and colleagues (2007), which manipulated two variables: adaptive versus fixed support, and collaborative versus individual learning. They found that the adaptivity and collaboration interacted to produce a significantly higher learning result compared to the other conditions. As technical merits of the reviewed systems have been established, a logical step will be to investigate their potential interaction and learning benefits (as investigated in *Q1-L1* and *Q1-L2*, described in Table 1; "What are the effects of ACLS on student collaborative interactions and learning?").

This dissertation also investigates whether building collaboration components on top of an existing tutoring system might accelerate the evaluation process (*Q2-L1*; "How can intelligent tutoring-style data logs augment the analysis of collaborative study data?"). There are several obstacles to conducting controlled experiments with ACLS systems. Large amounts of data are often required to develop the assessment components of the systems, but the data can be difficult to collect. After expending the effort it takes to build an adaptive collaborative system, it can be too time-consuming to build appropriate control conditions for evaluation. Finally, once appropriately calibrated conditions exist, it can be difficult to find enough participants for the study, and even more difficult to conduct the study in an ecologically valid setting. As intelligent tutoring systems are older than ACLS, there exists more infrastructure surrounding these systems that can facilitate evaluation studies. The Cognitive Tutor Algebra (*CTA*), for example, can be found in thousands of schools across the US, and therefore vast amounts of data are logged every day (www.carnegielearning.com). Tutor data is often mined in service of investigating learning science hypotheses and ultimately informing the improvement of intelligent tutoring systems (Beck, Mostow, & Bey, 2004). Similarly, it has become common practice to perform embedded experiments, making small modifications to already deployed tutoring systems (Mostow & Aist, 2001;

Koedinger & Aleven, 2007; Koedinger et al., 2009). Finally, because the tutors are so widespread, there are well-established relationships with schools that can be leveraged to gain access to classrooms and ecologically valid participants. Taking advantage of these relationships is one main goal of the Pittsburgh Science of Learning Center, which connects researchers and classrooms, and then instruments those classrooms so that it is easier to collect data and evaluate learning interventions (www.learnlab.org). Developing ACLS systems on top of existing intelligent tutoring systems holds great promise both in making such systems more available and in using them as a platform for research on users' interactions, collaborative learning, and methods for adaptive support.

In this subsection, I summarized the current research on ACLS, and the potential for advancement in their design, implementation, and evaluation using intelligent tutoring approaches and components. In the following section, I talk more specifically about the design context, which involves extending the CTA for reciprocal peer tutoring.

2.4 Context: Reciprocal Peer Tutoring

As discussed above in *2.2*, collaborative activities have been demonstrated to have several benefits in classroom environments, including improved individual and group learning outcomes (Lou et al., 2001), and deep elaboration of the learning content (Teasley & Fischer, 2008). One key interaction in collaboration is the *help* given from one student to another student in a collaborative group, encompassing the exchange of hints, feedback, and information (e.g., as described in Johnson and Johnson's promotive interactions, 1990). This dissertation focuses on how the act of help-giving between novices can be supported using a reciprocal peer tutoring activity. While I focus on the reciprocal peer tutoring activity, the dissertation work could potentially generalize to other collaborative activities where help is exchanged between novices.

2.4.1 Learning from Help-Giving in Peer Tutoring

The act of giving help can improve learning of both the help giver and the receiver (see Ploetzner, Dillenbourg, Preier, & Traum, 1999), as it stimulates students to engage in several cognitive processes that lead them to acquire deep knowledge: attentional processes, reflective processes, elaborative processes, and co-constructive processes (see Table 7). On the help-giver's side, there is some evidence that the accountability that students feel when they are told they will be taking on a peer tutor role leads them to attend more to the domain material, and thus learn more. For these reasons, having students prepare to tutor can in itself increase learning (Ploetzner et al., 1999). While tutoring, Roscoe and Chi (2007) concluded that peer tutors further benefit from knowledge-building activities, where they reflect on the current state of their knowledge and use it as a basis for constructing new knowledge. During tutoring, peer tutors must monitor their own and their partner's knowledge. If they become aware of gaps in their own knowledge, they may move to repair those gaps, improving their mastery of the domain (Ploetzner, Dillenbourg, Preier,

Table 7. Processes involved in learning from help-giving and help-receiving. While help does not have to always be correct for tutees to benefit, tutees should receive enough correct help to lead them to correct problem-solving steps.

Processes	Tutor Behaviors	Tutee Behaviors
Attentional	Accountability to partner	Accountability to partner
Reflective	Reflect on tutee steps and misconceptions	Prompted to self-explain, receive (correct) help targeted at misconceptions
Generative elaborative	Construct conceptual elaborated help, prompts	Ask specific questions, receive (correct) conceptual elaborated help, use help constructively
Co-constructive	Discuss solution alternatives	Discuss solution alternatives

& Traum, 1999; Van den Bossche, et al., 2006; *reflective processes* in Table 7). Additionally, peer tutors may develop structured networks of knowledge by asking and answering questions and giving and receiving explanations, leading them to make inferences about the subject material and better integrate their knowledge (Ploetzner, Dillenbourg, Preier, & Traum, 1999; Roscoe & Chi, 2007; *generative elaborative processes* in Table 7). On the other hand, there is evidence that benefits for the help receiver are more contingent on the quality of help given. One set of findings illuminates the importance of appropriate help at impasses: When the tutee reaches an impasse, they should be prompted to find and explain the correct step, and only be given help if they fail to do so (VanLehn, Siler, Murray, Yamauchi, & Baggett, 2003). One might assume that this variety of help activates *tutee* attentional processes, by leading tutees to attend more to the relevant aspects of the problem. Another set of findings investigates the properties of help that relates to tutee learning: Help should be correct, be conceptual and elaborated, and address tutee misconceptions (Webb, 1989; Webb & Mastergeorge, 2003). This variety of help may engage tutees to engage in *reflective processes*, reflecting on their misconceptions and repairing them. It has also been found that tutees learn when they ask their partners specific questions, and then use the help they receive constructively, suggesting that *generative elaborative processes* also come into play (Webb, Troper, & Fall, 1995). Finally, when both the help-giver and help-receiver are novices, *co-constructive processes* can also come into play. If peer tutors and tutees have conflicting ideas about the correct way to proceed in the problem, they can mutually benefit from resolving the conflict through discussion.

It is important to note here that while many of the activities that the help-giver engages in leads them to benefit from the activity, regardless of their tutoring ability, it appears it is more difficult for the help-receiver to benefit (Robinson, Schofield, & Steers-Wentzell, 2003). Simply by being placed in the role of the peer tutor and by observing tutee problem steps, peer tutors are likely to benefit from attentional and reflective processes. On the other hand, peer tutors are less likely to engage in elaborative and co-constructive processes unless the help they give includes conceptual elaborated content (Webb & Mastergeorge, 2003). However, whether tutees engage in these processes are highly contingent on the

abilities of the peer tutor. Unfortunately, most students do not exhibit positive helping behaviors spontaneously (Roscoe & Chi, 2007), and thus during collaboration students may fail to help each other well or even at all. Expanding on Clark's analysis (2007), there are two preconditions to peer tutors giving effective help: tutoring competence and tutoring motivation. Peer tutors must know how to execute tutoring behaviors that contribute to their knowledge building, like giving elaborated explanations (Fuchs et al., 1997). They also must monitor the tutee's progress and know when to execute particular behaviors; a didactic explanation is probably not as beneficial if the tutee has not been given a chance to correct his or her own error (VanLehn et al., 2003). Finally, they must have domain competence, or sufficient knowledge about the correct solution to help their partner (Dillenbourg, Baker, Blaye, & O'Malley, 1996).

However, even if students have mastered these competencies, there are motivational factors that may affect their tutoring behaviors: namely, their perceptions of their role as a tutor, their beliefs that they can fill that role, and their engagement in the tutoring process. There is evidence that tutors have the tendency to adopt a knowledge-telling strategy rather than a knowledge-building strategy (Roscoe & Chi, 2007), and that this approach may be due to their perceptions of the tutoring scenario; they may see their role as transmitting their knowledge through lengthy explanations rather than as seeking to improve their tutee's and their own understanding. This knowledge-telling approach is less beneficial to conceptual learning than knowledge-building for both students (Chan, 2001). Another motivational factor that may be relevant is peer tutors' feelings of efficacy, which was related to the effort they expended during tutoring (Medway & Barron, 1977). Perhaps the most convincing work linking motivation to tutoring involves studies that adopt the Slavin (1996) philosophy of individual accountability for group performance. When peer tutors are rewarded for tutee outcomes (Fantuzzo, King, & Heller, 1992) or even simply observe their tutee take a test after tutoring (Biswas et al., 2005), they learn more. Effective peer tutoring has a competence component and a motivational component that needs to be supported.

2.4.2 Supporting Help-Giving Behaviors Using a Reciprocal Peer Tutoring Script

One particular set of methods that have been applied to supporting help-giving involve a reciprocal peer tutoring script, where first one student is given artificial expertise in a particular domain and is told to regulate the problem-solving of a second student, and then the roles are reversed and the second student becomes the expert. As part of their role, the expert must monitor their partner's problem-solving and offer appropriate help when it is needed. The reciprocal schema is one of the basic schemas proposed by Dillenbourg and Jermann (2007) in their SWISH design model for collaborative learning. Examples of this class of collaborative activities are dyadic activities such as reciprocal teaching by Palincsar and Brown (1984), mutual peer tutoring by King, Staffieri, and Adelgais (1998), and reciprocal peer tutoring by Fantuzzo, Riggio, Connelly, and Dimeff (1989). Dillenbourg and Jermann (2007) argue that the nature of the reciprocal task leads students to interact and construct shared understanding, that is, learn collaboratively. Even though only the tutee solves the complete problem, with the peer tutor acting as the regulator, peer tutoring among students of similar abilities has much in common with other collaborative

learning scenarios. The ultimate goal of peer tutoring is for both students to develop a deep understanding of domain concepts, just as in other forms of collaborative learning. To that end, tutees and tutors construct domain knowledge in the process of either solving or explaining problem steps. Additionally, both students take initiative, creating a transactive interaction: The peer tutor determines when to give help by monitoring tutee problem-solving, but the tutee must monitor their own understanding in order to know when to request help or question peer tutor explanations. In reciprocal peer tutoring, where students take turns being tutees and tutors, all students have the opportunity to engage in the same cognitive activities.

Several of these reciprocal activities have been successful at increasing student learning in classroom environments compared to individual and unstructured controls (Fantuzzo et al., 1989; King et al., 1998; Fuchs et al., 1997), and have been effective for both low and high ability students. However, it is critical to provide support to students in order to assist them in helping each other effectively. As described in the introduction, collaborating students are often supported using fixed scripts that outline roles and activities that relate to the desired interactive behaviors. Scripting has also been used successfully in the context of peer tutoring. For example, King, Staffieri, and Adelgais (1998) found that having tutors ask their tutees a series of questions at different levels of depth had a significantly positive effect on tutor learning. Even relatively limiting scripts that leave peer tutors with little freedom in their interactions have had beneficial effects on tutor learning in the classroom (Fantuzzo, King, & Heller, 1992). Another way of increasing the benefits of peer tutoring is to provide students with pre-collaboration training on good tutoring behaviors. Fuchs et al. (1997) trained students to deliver conceptual mathematical explanations and give elaborated help, and showed that their mathematical learning was significantly better than training on elaborated help alone or an individual learning control. In these scenarios, domain assistance generally takes the form of preparation on the problems and scaffolding during tutoring (e.g., by giving the tutors the answers to the problems; Fantuzzo et al., 1989). While these fixed methods of support have been effective, they do have the same drawbacks as fixed support to collaboration more generally: When support is not adapted to student needs, there is the danger that any particular pair will be over- or under-supported.

2.4.3 Promise of Adaptive Collaborative Learning Support in Reciprocal Peer Tutoring

In peer tutoring, where it is necessary to support the peer tutor's ability to give both good help and correct help, using adaptive support methodologies might play a highly beneficial role by tailoring traditional fixed script support to student needs. Potentially more interestingly, using intelligent tutoring *components* has promise both for developing support and for data analysis. Much of the success of the peer tutor hinges around aspects of the problem-solving domain: peer tutors must not only give correct help, but they must recognize tutee errors and then support tutees in recognizing them. While fixed support techniques cannot recognize these opportunities for learning, adaptive support techniques could potentially use domain models to do so. Student learning from peer tutoring interactions is often analyzed by collecting tutor and tutee dialogues, using video or audio recording. Dialogues are transcribed, and researchers code the interaction for particular help-giving and help-seeking behaviors. For instance, Webb, Troper, & Fall

(1995) have developed an extensive coding scheme for specific and general help requests made by tutees, and levels of elaborated help given by tutors. These types of analyses allow researchers to infer from student dialogue that particular cognitive processes are occurring, link those processes to learning, and link a given intervention to those processes. However, collecting interaction process data in the context of the classroom and in a complex learning setting (e.g., with alternating individual and collaborative phases) is extremely difficult. In addition, these types of dialogue analyses are very time costly and can often only be performed for a small fraction of the process data. Therefore, on top of the potential pedagogical benefits of providing adaptive assistance to peer tutoring, combining intelligent tutoring and peer tutoring might give the researcher access to the rich problem-solving log data common in intelligent tutoring systems, which is not generally recorded in peer tutoring interventions. Using this computer-mediated data would both place the student interaction in context and potentially make it easier to automate parts of the data analysis.

Despite this promise, the majority of the previous work done in integrating intelligent tutoring and peer tutoring has been surrounding the idea of placing a student in the tutor role in an intelligent tutoring system and providing adaptive support. Chan and Chou (1997) outlined the space of possibilities for interactions between real learners, real tutors, virtual learners, and virtual tutors, and described two relevant scenarios: One where an agent tutors a human tutoring a human, and one where an agent tutors a human tutoring an agent. They then implemented a distributed reciprocal tutoring system involving two students alternating between learner and tutor roles. Peer tutors were provided with a scaffold, based on a domain model, which helped them to diagnose errors made by the tutee and select a relevant hint. An evaluation of this scenario with five learners showed promising posttest scores. Another "helping the helper" system has been implemented by Vassileva, McCalla, and Greer (2003), where computer agents use peer learner and helper profiles to negotiate tutoring partnerships between students. A further addition to this system provides the helper with more information about the request context, a plan for providing help, and even information about the learner's preferred delivery method (Kumar, McCalla, & Greer, 1999). Finally, people have investigated a human teaching an agent (e.g., Uresti, 2000), or even reciprocal learning scenarios between a human and an agent (e.g., Scott & Reif, 1999), but many of these systems have not implemented adaptive assistance for the peer tutor. One exception is the Betty's Brain system (Leelawong & Biswas, 2008), where a human student tutors "Betty", a computer agent, with the help of another agent "Mr. Davis". This scenario has been found to be effective at promoting learning compared to a traditional intelligent tutoring scenario. Based on these results, there is promise in developing adaptive domain assistance for human-human tutoring for the purpose of improving student interaction and learning.

2.5 Outlook & Discussion

In this section, I reviewed the potential benefits for adaptively supporting collaboration, surveyed design, implementation, and evaluation aspects of ACLS, and then discussed where ACLS might apply to reciprocal peer tutoring. In the process, I motivated the nine research questions that are addressed in this dissertation (see Table 1). In the remainder of this paper, I discuss several iterations of design and

implementation of assistance for collaboration. I also explore the effects of adaptive assistance on collaborating students from a learning sciences perspective, looking at how the context of interaction and assistance students receive informs their interactions and learning (see Figure 2). In each phase, the peer tutoring context and the assistance students are given is discussed. Then, when examining the effects of assistance, we look at problem-solving actions, tutorial actions, dialogue, and student use of the assistance itself. These interactions are linked to various learning, motivational, and social outcomes.

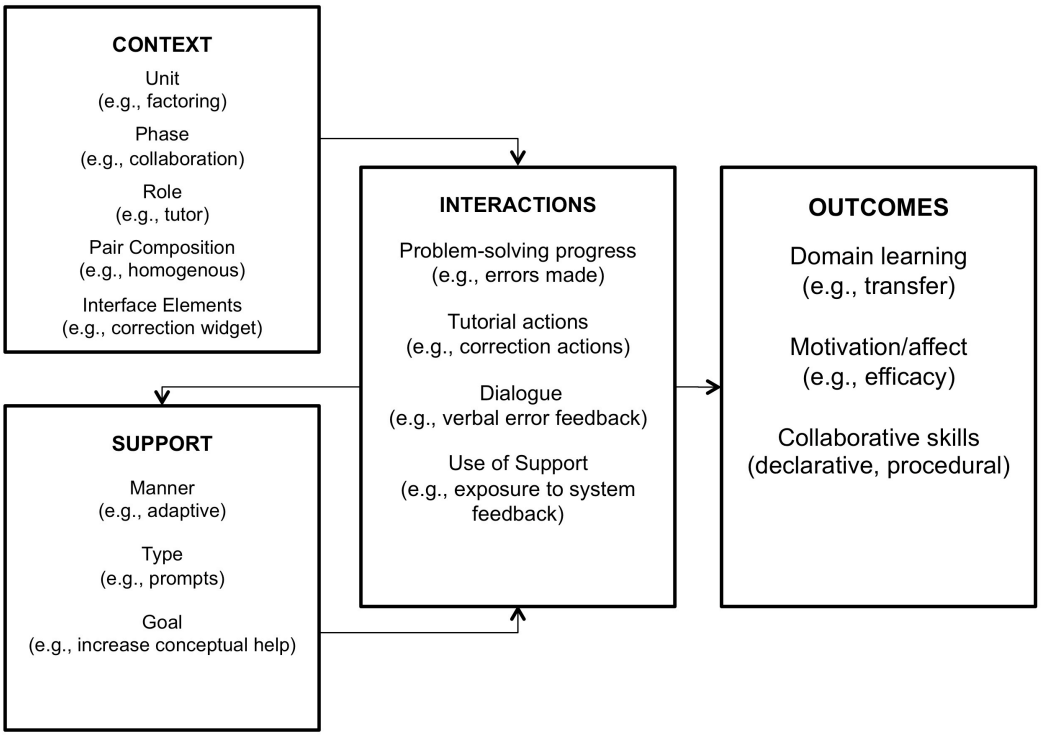

Figure 2. Context, support, interactions, and outcomes in *APTA*, the learning environment described in this dissertation. Context and support influence interactions, which influence outcomes. In adaptive systems, support varies based on student interactions.

3 Phase 1: Peer Tutoring Learning Environment

3.1 Introduction

The first step in my program of research was to design a basic peer tutoring script, drawing elements from other successful peer tutoring scripts, and determine how peer tutors used the script without intelligent support (*3.2*). I implemented the script as a collaborative extension to the Cognitive Tutor Algebra (*CTA*; *3.3*). I compared two designs in a classroom experiment: A basic version, and one with elements encouraging students to reflect on their collaboration. By examining how peer tutors used the script without extensive support, I was able to discover where adaptive support might be effective (*3.4*). The conclusions of this phase are summarized in *3.5*. The work in this phase was discussed elsewhere in Walker (2005) and Walker, Rummel, and Koedinger (2007a).

3.2 Design: Basic Peer Tutoring Script + Reflection Elements

3.2.1 Interactions: Basic Script Design

I designed the basic version of the peer tutoring script as an extension to the literal equation solving unit of the CTA, which is consistently identified by classroom teachers as one that is particularly difficult for students to master. In this unit, students are given a prompt like "Solve for *x*," and then given an equation like "$ax + y = bx + c$." To solve these problems, students must be able to manipulate an equation to move constant (e.g., *y*) and variable terms (e.g., *bx*) from one side of an equation to another, factor *x*, and divide by a coefficient (e.g., *a* - *b*). In addition to learning these procedural steps, they must conceptually be able to recognize the difference between constant and variable terms, the difference between positive and negative terms, the distinction between the four major operators on the equation (multiplication, addition, division, subtraction), and the conceptual basis for factoring. Students use menus in an equation solver tool to manipulate the equation, selecting operations like "add *x*" or "combine like terms" (see Figure 3). 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., the denominator of a fraction).

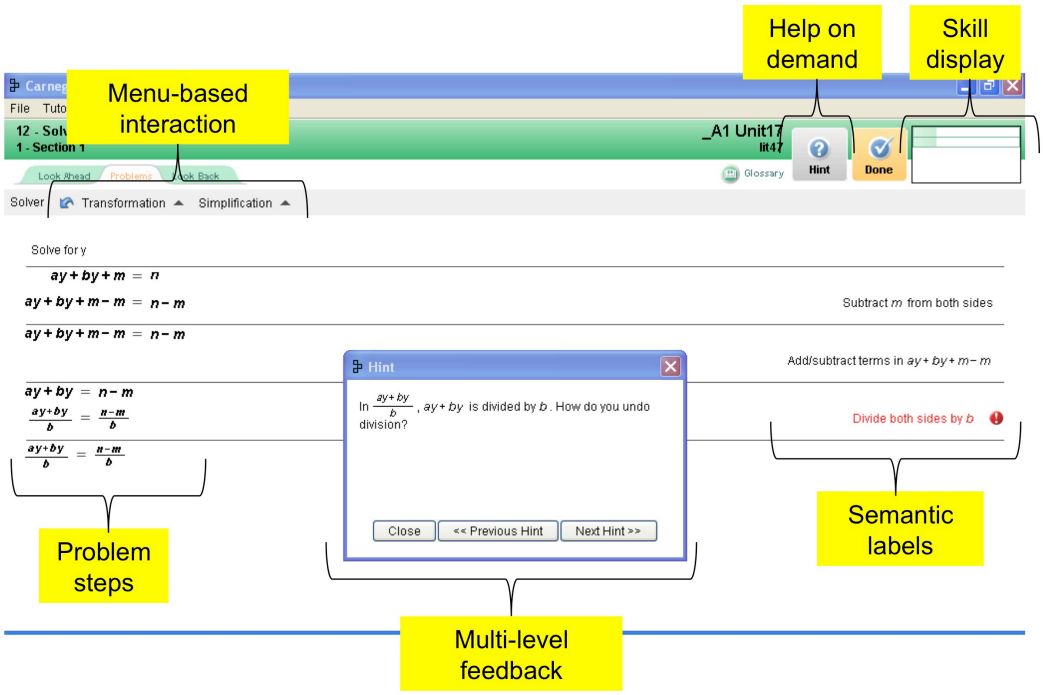

Figure 3. Individual use of the Cognitive Tutor Algebra. Students solve problems using the menus on the top left, and their steps are displayed on the left side of the screen. Students can request hints and receive feedback on their problem-solving actions.

The basic peer tutoring script attempts to create interaction conditions conducive to the display of positive tutoring behaviors. As part of the script, the classroom teacher grouped students into same-gender pairs of similar abilities. The script included two phases: a preparation phase and a collaboration phase. For the first half of a class period, students engaged in a preparation phase, peer tutors were given a chance to practice with the material ahead of time by solving problems using the individual version of the CTA. Pair members were each given different sets of problems to solve in the preparation phase. During the second half of a class period, students took part in a collaboration phase, pair members collaborate with each other at different computers, taking turns being peer tutors and peer tutees. For example, if Phil and Sara were partners, and Sara was the tutor for the first problem, Phil would be the tutor for the second problem. As a tutee, Phil would solve a problem that Sara had solved in the preparation phase, using the same equation solver interface. Sara, in the role of the tutor, can see Phil's problem-solving steps and the results of her type-in entries, but cannot solve the problem herself (see Figure 4). Instead, she can mark Phil's actions right or wrong, and monitor his knowledge by raising or lowering the values of her skillometer bars. Phil sees every action Sara takes to correct him or give him feedback on his knowledge. Phil and Sara can interact with each other in natural language using a chat tool, where, for example, tutees can ask questions and tutors can give hints and feedback. When Phil decides that the problem is done, he would click "done" on her interface, and then Sara would be given the choice to agree or disagree. If both students agree that

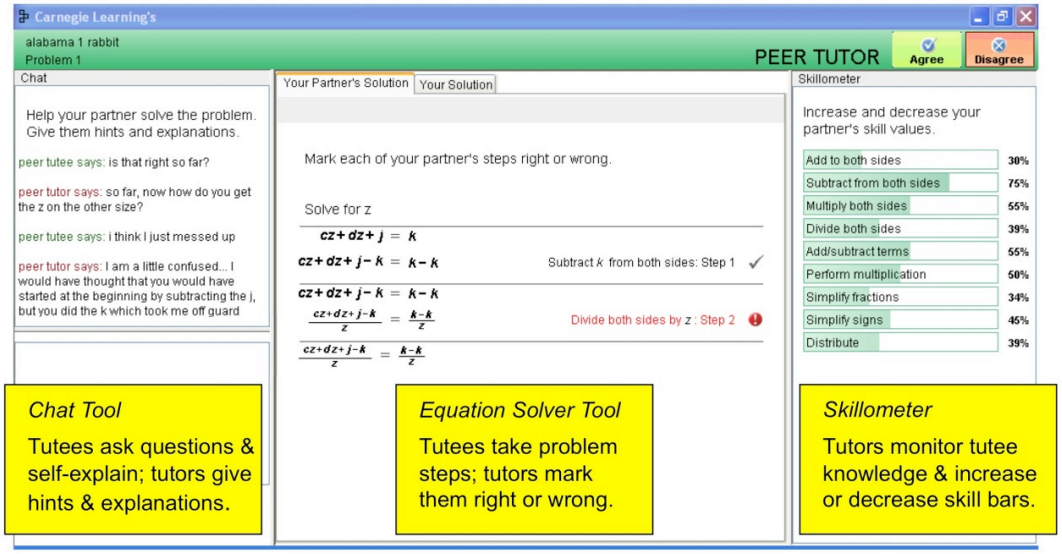

Figure 4. Peer tutor's interface. Tutors can chat in the chat tool, correct peer tutee actions in the solver, and increase or decrease tutee skills in the skillometer.

the problem is done, they move to the next problem, and their roles switch. Peer tutors do not have to mark every step or adjust all relevant skills before they can move to the next problem. Overall, students walked through the same fixed sequence of problems they followed during the preparation phase. To support peer tutors in giving correct help, we gave them a packet with their own answers to the problems they were tutoring printed out. When students received a problem to tutor on the computer, they could find the solved example of the problem in their packet and use it to give hints.

3.2.2 Model: Expected Learning

Approach & Assumptions. My first attempt at modeling peer tutoring was a black-box model, based on previous research on learning from tutoring and being tutored, that links tutoring behaviors to cognitive processes. Thus, it has one main assumption that carries through to the other models in the dissertation: by encouraging students to engage in particular positive behaviors, we increase the likelihood that they will experience beneficial cognitive processes that lead to learning. Unlike future models that I developed in peer tutoring, this model is not a computational model; it does not suggest under what conditions or in what order peer tutors should engage in those behaviors, and thus is not necessarily suitable as a basis for assessing student actions or providing adaptive support to students. However, the model can be used as a lens for interpreting student interaction with the system: Are students engaging in the behaviors prescribed by the model? Do they appear to be benefiting in the ways that we expect from those behaviors?

Effective Behaviors. There are several potential benefits of the script for students who take on the role of peer tutors, as long as they engage in effective tutoring behaviors (see Figure 5). First, we hypothesized that the script would trigger accountability between peer tutors and their partners, leading them to attend more to the activity (*attentional processes*). In the preparation phase, students solve problems anticipating that they will have to tutor another student. Ideally, they then attend more to the domain material as they study because they feel more responsible for acquiring knowledge. Then, when students actually tutor their partner, their responsibility for their partner's learning makes them try harder to be a better tutor. Additionally, the script is designed to encourage peer tutors to be accountable for giving correct help to their tutees, triggering *reflective processes*. In the collaboration phase, because peer tutors must mark the tutee's actions correct or incorrect, adjust the skill values of the tutee, and give the tutee next-step help and error feedback, they will be observing problem-solving actions and reflecting on their partner's knowledge and the steps that it takes to solve the problem. These reflective processes might lead them to notice misconceptions in their own knowledge and repair them. Finally, the script is designed so that students are also accountable to giving well-explained help to their partners, triggering *generative elaborative processes*. In asking each other questions and giving each other explanations in the chat tool, students have to articulate their reasoning, leading them to elaborate on existing knowledge and generate new knowledge. This process may lead the two students to discuss differences in opinion on how to solve the problem, giving them the opportunity to engage in *co-constructive processes.* The benefits of the script for students taking on the tutee role are less clear. The better the tutor at giving help, and the better the script is at encouraging those help-giving behaviors, the more the student taking on the tutee role is likely to benefit.

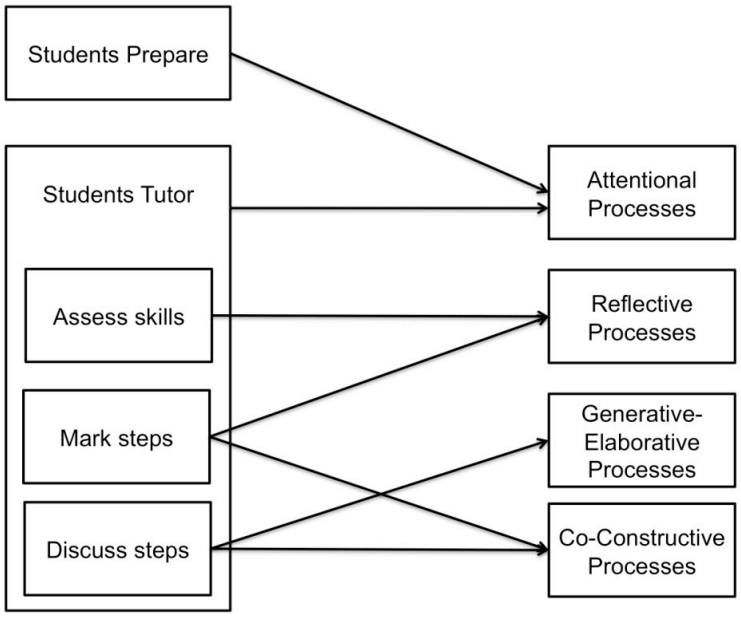

Figure 5. Abstracted model of learning from tutoring, for the peer tutor. Through preparation to tutor, assessing tutee skills, marking problem-solving steps, and discussing the steps, peer tutors engage in attentional, reflective, generative elaborative, and co-constructive processes.

Ineffective Behaviors. There are particular ways to deviate from our intentions with the script design that might lead peer tutors to fail to benefit. If peer tutors fail to engage with any of the script behaviors (i.e., they fail to mark problem steps or give verbal help to their partners), they are less likely to engage in the reflective and elaborative processes that would lead one to learn from the script. Further, even if they engage with the script behaviors, there is no guarantee that they will perform the behaviors in a way that will benefit them. If peer tutors are unable to give correct help on the next step, or unable to construct an explanation, it is less likely that they will benefit. Even if peer tutors do execute the behaviors well, they still may not lead to beneficial processes (just because a tutor gives an explanation does not mean they are elaborating on their own knowledge, even if the explanation is in itself elaborated). Additionally, the script is not set up so that students receive as many benefits from taking on the tutee role as taking on the peer tutor role. While, as described in the introduction, tutees tend to benefit greatly from good one on one tutoring, there is no guarantee in the script as designed that students receive tutoring that is tailored to their misconceptions, elaborated, or even correct. To explore these issues, we piloted the script in a classroom, and indeed found that students were not interacting as we had hoped. In particular, the peer tutor typically focused on correcting the tutee, barely using the chat tool at all. This situation was problematic because the natural language interaction between students is expected to be one of the primary benefits of the peer tutoring situation. To encourage peer tutors to give better help, we added some fixed support to the script, described in the following section.

3.2.3 Support: Problem Solutions & Reflective Elements

To support peer tutors in giving conceptual elaborated help, I added three additional activities that encouraged students to reflect on good collaboration. First, to ensure that students were familiar and comfortable with the instant messaging tool provided for their chat interaction, I engaged dyads in the discussion of three questions over chat, before they collaborated (*chat practice*). For example, I asked them to collaboratively rank how useful five specific questions about an algebra problem might be in helping them learn. Second, during the preparation phase, I gave students questions to prepare them for the collaborative challenges of tutoring as well as the cognitive ones (*preparation reflection*). For example, after they had solved a problem, we told them: “You should ask your tutor questions about the problem. A good question is specific. It asks why something is done, or what would happen if the problem was solved a certain way. What is a good question to ask about the step you chose in Question 2?” Finally, I gave students three additional reflection questions after they had just finished tutoring a problem (*collaboration reflection*). For example, the reflection asked them: “What was the best question asked by the tutee? If the tutee didn't ask any questions, what was a good question he/she could have asked?” This additional scripting should give students the expectation that they be good tutors, and then help them to develop a better model of effective peer tutoring, equipping them better to co-construct knowledge and engage in self-reflection.

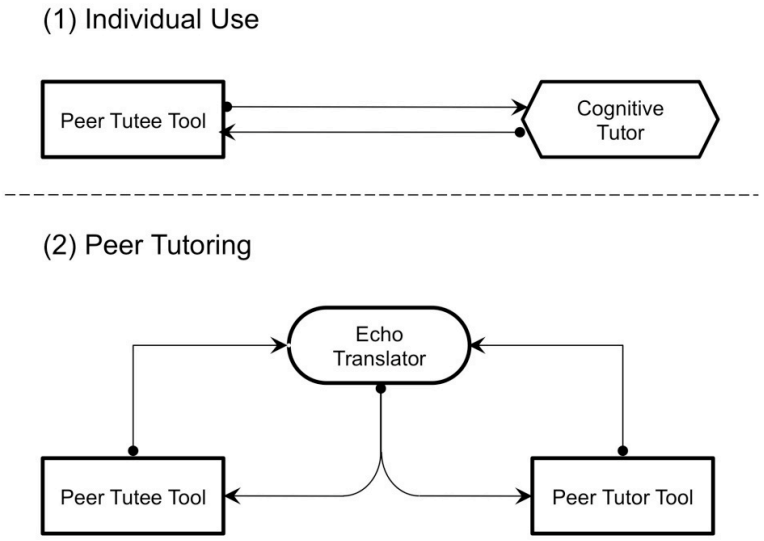

Figure 6. Components used in Phase 1. The individual use scenario was transformed into a peer tutoring scenario, by removing the cognitive tutor, and adding an echo translator to share students' actions with their partners.

3.3 Implementation: Refactoring the Cognitive Tutor Algebra for Collaboration

To implement the peer tutoring script, I made code extensions to the Cognitive Tutor Algebra (*CTA*) so that its components could function in a collaborative setting. First, I refactored the *CTA* so that its *tool component* (the interface to the user and logic for changing the problem-solving state) functioned independently from its *tutor component* (the model of problem-solving behavior and relevant support). #1 in Figure 6 depicts the individual use of the cognitive tutor scenario. While the *CTA* was designed to be refactored in this way, following the architecture outlined in Ritter and Koedinger (1996), development constraints led its current state to evolve from this ideal. I then modified the components so that they could be launched and quit remotely, and send remote messages to one another. In the *CTA*, components had already been designed to send networked messages using TCP/IP sockets, so this is the protocol we used within the mediator to send the low-level remote messages. High-level responsibility for managing sessions was not fully factored (e.g., beginning a session, moving to the next problem), so we used Java RMI to make the remote message calls for accomplishing these functions. We added a central control module to act as a switchboard for passing messages between components, both for the high-level and low-level messages. I also used RMI to implement a client-server setup for running multiple tutoring sessions at once, so that multiple students in the class could use the system. After refactoring the CTA so that components were separate and could be passed remotely, I was able to add a second tool component, and then use a *translator* component to echo messages from one component to another. The translator receives messages reflecting the actions that users took in one tool, and broadcasts them to the other tool. Finally, I

modified the duplicated tool components to create a tool for the peer tutor and a tool for the peer tutee. I added a chat tool by modifying Jeti, an open-source Jabber client (http://jeti.sourceforge.net). I also modified the peer tutor's tools, disabling widgets for inputting answers, and adding widgets for approving and flagging to the title bar. The end result of this process was a system greatly resembling the original CTA that students could use to tutor each other (see #2 in Figure 6).

3.4 Evaluation: Study 1

3.4.1 Experimental Design

I conducted a small-scale study to evaluate the effectiveness of our baseline learning environment: a peer tutoring addition to the CTA, without problem-solving or help-giving support. The study allowed me to begin forming a corpus of data to support the ultimate goal of supporting the peer tutor in giving better help. I compared two conditions, one in which students simply tutored each other using the CTA interface (*collaboration condition*), and one in which students tutored each other using the CTA interface and were given additional reflection exercises (*collaboration + reflection condition*). I hypothesized that peer tutoring would increase student learning of the relevant algebra skills in both conditions, but giving students additional instruction would improve the effects of the peer tutoring on learning.

3.4.2 Method

Participants. Participants were 30 high-school students from two first-year algebra classes at a vocational high school. One class had 18 students, one class had 12 students, and both classes were taught by the same teacher. Due to the potential disruptiveness of students in the same class using different interventions, the manipulation was between-class. The class with the most participants was assigned to the *collaboration+reflection* condition. Unfortunately, there were significant between-class differences. The classroom teacher informed us prior to the study that students in the two classes were at different levels in the course, and that the students in the smaller classroom were generally more motivated and engaged. This information was confirmed by quantitative data on student progress. Students in the *collaboration+reflection* condition were, on average, working on a significantly lower unit in the Cognitive Tutor Algebra (*Ms* = Unit 8.3 and 11.6, *SDs* = 1.25 and 2.76, $F(1,12) = 8.22$, $p = 0.01$). Over half the students in the collaboration condition had reached Unit 12, the literal equation unit used in the study, while none of the people in the *collaboration+reflection* condition had arrived at that unit. Additionally, the study was run in the last week of the semester, and as a result there was a lot of attrition. Only 14 participants participated in all phases of the study (pretest, preparation for tutoring, peer tutoring, and posttest): seven in the tutoring condition, and seven in the *collaboration+reflection* condition.

Procedure. The study took place over the course of a week, spanning one 35 minute classroom period and two 70 minute classroom periods. See Table 8 for an overview of the procedure. During the first period of

the week, students took a 10 minute pretest. During the second period, students were given a 15 minute overview of the study, and then 15 minutes of practice using the chat interface (the chat practice described in the previous section). Dyads in the *collaboration* group were asked to answer three questions about what they would do if they were stranded on a desert island, while dyads in the *collaboration+reflection* group we asked to answer three questions about good collaboration. Students then spent 40 minutes solving problems they would be tutoring in the collaborative phase. They worked individually using the *CTA*. After each preparation problem students in the *collaboration+reflection* group answered reflection questions on paper intended to prepare them for tutoring (the preparation reflection described in the previous section). During the third classroom period, students in both conditions spent 50 minutes taking turns tutoring each other. After each problem, *collaboration+reflection* groups discussed reflection questions related to what had just occurred over chat (the collaboration reflection described in the previous section). Students were given a 10 minute posttest.

Table 8. Study procedure in Phase 1. The study took place over three days in a single week.

Week	Day	Time (minutes)	Collaboration	Collaboration + Reflection
1	1	10	Domain Pretest	Domain Pretest
1	2	15	Instruction	Instruction
1	2	15	Chat Practice (desert island)	Chat Practice (collaboration)
1	2	40	Preparation	Preparation + Reflection
1	3	50	Collaboration	Collaboration + Reflection
1	3	10	Domain Posttest	Domain Posttest

Measures. To assess students' individual learning there was a counterbalanced pretest and posttest, each containing 8 questions at 4 levels of difficulty and drawn from the same unit as the intervention questions. As it was unlikely that most students to be able to solve all the questions in the time allotted, students were given instructions to attempt as many questions as they could. The tests were administered on paper. To analyze the collaborative process of the students, the cognitive tutor log data was used. All tutor and tutee actions were logged, including tutee problem-solving actions, tutor correction actions, tutor skill-adjusting actions, student chat actions, and student done and quit actions. I computed the number of problems solved by students in each phase on each study day, and whether each problem was successfully completed or unsuccessfully completed.

3.4.3 Results

Learning Outcomes. I conducted a two-way (condition x test-time) repeated-measure ANOVA, with test-time as the repeated measure. Posttest scores were significantly higher than pretest scores in both

the collaboration and the *collaboration+reflection* condition ($F[1,12] = 15.25$, $p < 0.002$, $\eta^2 = 0.56$), but there were no significant differences between conditions, and no interaction (see Table 9). Although students' cognitive tutor unit prior to the study was weakly correlated with student posttest scores ($r(12) = 0.47$, $p = 0.09$), it was not predictive of the gain in scores from pretest to posttest ($r(12) = 0.11$, $p = 0.70$).

Table 9. Domain pretest and posttest scores in Phase 1. Scores are percent correct.

	Pretest		Posttest	
Condition	**M%**	**SD%**	**M%**	**SD%**
Collaboration	31.1	25.4	45.8	31.8
Collaboration+Reflection	22.9	15.3	42.8	22.0

Helping Behaviors. To assess the quality of student helping behaviors I conducted a qualitative analysis using log data and notes from classroom observation. During peer tutoring, students appeared engaged. They exhibited many of the positive collaborative behaviors that we were attempting to encourage with our script and that have been shown to correlate with knowledge construction and self-reflection. Table 10 shows an excerpt from an interaction between two of the students that had many desired elements. The students were solving the problem, "$cz + dz + j = k$" for z. Such explanation and reflection behavior by the students should lead to better learning of the course material, and indeed, the peer tutor went from 60% on the pretest to 73% on the posttest, and the tutee went from 7.5% on the pretest to 23% on the posttest. While most of the student interactions did not achieve this quality of dialogue, most students appeared to engage with the idea of giving and receiving typed help.

Problem-solving progress. The major obstacle peer tutors faced appeared to be their struggle with the domain knowledge, where they frequently had difficulty knowing how to proceed with the problem their tutee was solving. In fact, the dyad discussed above took 30 steps in the equation solver to solve that particular problem, where 5 steps should have been sufficient. In general, peer tutors did not appear to connect the preparation that they had done with their tutoring during the collaboration phase. For instance, they often did not consult their answer printouts when they did not know the next problem step and thus had to rely on teacher assistance to successfully complete a problem. As a result, tutees became frustrated and skipped problems without completing them correctly. In particular, students in the *collaboration+reflection* condition found it more difficult to tutor, and asked the teacher for more support. The classroom teacher observed that while students generally stayed on task, the problems appeared to be frustrating to them, and that they had a lot of questions.

Table 10. Positive interaction in Phase 1. Students were solving for *z* in the problem $cz + dz + j = k$.

Problem Action	***Interaction***	***Positive Behavior***
The tutee subtracts k and combines like terms	Tutee: Is that right so far?	Tutor asks a specific question
	Tutor: So far, now how do you get the *z* on the other side??	
	Tutee: I am getting there	
Tutee factors z, divides by c+d, and multiplies terms	Tutee: I think I just messed up	Tutee evaluates her actions
	Tutor: I am a little confused… I would have thought that you would have started at the beginning by subtracting the j, but u did the k which took me off guard	
	Tutee: I know im difficult	Tutor attempts to resolve a misconception
	Tutor: lol. I understand that. Ummmmm I think you need to erase your last step.	
The peer tutor marks the step wrong, and the peer tutee undoes them.	Tutor: Now you need to get the z on the other side… so you prob. Need to divide by z.	Tutor provides an elaborated explanation
The tutee divides by z.	Tutee: I know I realized that after I looked at it	Tutee constructively processes the explanation

Looking further at potential consequences of this frustration, there were differing effects on students' problem-solving behavior for the *collaboration* condition and the *collaboration+reflection* condition (see Table 11). Students in the *collaboration* condition attempted more problems than students in the *colllaboration+reflection* condition ($F[1,8] = 4.33$, $p = 0.071$), and appeared to complete more problems as well ($F[1,8] = 2.95$, *ns*). The average number of problems completed by dyads in the *collaboration+reflection* condition was quite low; in fact, students in this group took an average of 11 minutes to complete a single problem, compared to a 6 minute average in the *collaboration* condition. Students in the *collaboration* condition were apparently more willing to skip past the problems they could not solve than students in the *collaboration+reflection* condition, thus they completed less than 60% of the problems they attempted. Immediately before skipping a problem, students would generally either state their inability to solve the problem, "I don't know how to do this one," or express their lack of motivation, "Just do something and I'll agree or something." Only in a few cases did students show an apparent lack of awareness that their answer was wrong. Skipping problems is undesirable behavior in this script because

students then do not learn how to solve the difficult problems. Completing a low number of problems is undesirable because students then are not given as much of an opportunity to master different skills found in different problems. Students in the *collaboration+reflection* condition also showed their frustration with the collaboration by interacting less than students in the *collaboration* condition; in general, dyads in the *collaboration+reflection* condition took fewer actions in the interface than dyads in the *collaboration* condition ($F[1,8] = 4.12$, $p = 0.077$). This lack of interaction was likely related to the fact that the *collaboration+reflection* dyads had more long pauses (i.e. periods of inaction greater than a minute) than the collaboration dyads ($F[1,8] = 3.55$, $p = 0.096$). In some cases, a long pause meant that students were pausing to think. However, in other cases students made a clear statement of lack of knowledge, and were likely pausing to ask a question to the teacher (e.g., the peer tutee says "Help me", the peer tutor says "Hold on," and there is a pause). More worrisome was when a negative statement preceded the long pause, such as "They let us out of math three weeks early last year. I refuse to participate in this," or when students would have several long pauses in a row. Because long pauses disrupt the interaction, and were in many cases indicators of frustration, they are also undesirable.

Table 11. Problem-solving interactions in *Phase 1*: *The Peer Tutoring Learning Enviornment*.

	Problems Attempted		Problems Completed		Number of Interactions		Number of Long Pauses	
Condition	**M**	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
Collaboration	14.2	8.47	8.4	5.13	622	307	3.0	2.6
Collaboration + Reflection	5.8	3.11	4.4	0.89	319	133	6.4	3.1

Use of Support. As there was no effect of condition on pre-post gain, I examined the activities of the students in the *collaboration+reflection* condition, in particular their participation in the reflection activities of the script during the preparation and the collaboration phases. Students appeared to engage with the *preparation reflection* questions. Of the fourteen students who participated in this phase, only one student left the questions blank or gave irrelevant answers. The other 13 students gave answers of one or two sentences that indicated that they had put some thought into the question (see Table 12 for examples). For example, in response to the question, "What was the hardest step in solving this problem", one student answered "The second step -- dividing both sides by 0.75." However, students did not appear to put a lot of thought into their answers in the collaboration reflection phase. One group did not answer any of the questions at all; a member of the group said, "I think we're skipping the questions we're supposed to answer," at which point her partner said "Who cares." When students did answer the questions, their answers were often no more than a couple words, and only one student tended to compose the answer, even though they were supposed to discuss the answers with their partners. For example, in this phase in response to the question, "What was the hardest problem step for the tutee", one group answered, "THE

WHOLE THING". During this phase, students seemed anxious to skip to the next problem. Additionally, the *chat practice* phase appeared to be unnecessary, as students were already heavy IM users.

Table 12. Use of reflection exercises in collaboration + reflection condition in Phase 1.

Preparation Reflection Question	**Sample Preparation Reflection Answer**	**Collaboration Reflection Question**	**Sample Collaboration Reflection Answer**
What was the hardest step in solving this problem?	The second step - dividing both sides by .75	What was the hardest problem step for the tutee?	THE WHOLE THING
What is a good question to ask about the hardest step?	Do you agree with my answer?	What was the best question asked by the tutee?	none
What is a good way to explain to your students how to do the step?	Take the one number or letter that you're solving and factor that by itself	How did the tutor answer the question?	Poorly

3.5 Outlook and Discussion

3.5.1 Introduction

In this chapter, I described a peer tutoring addition to the Cognitive Tutor Algebra (*CTA*) that incorporated elements of previous successful peer tutoring scripts. I implemented two versions of a baseline condition that used the CTA interface to structure student collaboration, but did not incorporate problem-solving or interaction tutoring. A small-scale study was conducted to evaluate the effectiveness of this baseline condition for improving student domain knowledge. Here, I discuss the design (*3.5.2*), technological (*3.5.3*), and learning sciences (*3.5.4*) implications of this chapter. I then discuss the potential for iteration on the system (*3.5.5*).

3.5.2 Design

As I did not computationally model collaboration or provide adaptive support in this chapter, none of the design-related research questions were addressed. But, by designing and piloting student interactions with each other and the system, this chapter is a necessary step for then designing adaptive support for those interactions.

3.5.3 Technology

While this chapter does not describe clear technological contributions, it does lay the foundation for future technological work. By refactoring the CTA and extending it for collaborative activities, it becomes

possible to combine those collaborative activities with domain tutoring found in the individual version of the CTA. In the next chapter, I discuss a component-based architecture for implementing flexible ACLS and relevant comparison conditions (addressing *Q2-T1*; "How can existing and custom components be integrated?"), and throughout the dissertation, I discuss how this architecture is used to provide different varieties of adaptive support. Without the work in this chapter, these later contributions would not have been possible.

3.5.4 Learning Sciences

Although students learned as a result of the peer tutoring, the condition with additional reflection exercises did not learn more than the condition without the reflection. Both groups did exhibit positive collaborative behaviors while tutoring, and the *collaboration+reflection* group could indeed provide reasonable questions and explanations during the preparation reflection. Perhaps the additional reflection exercises were simply not necessary in this particular context, as students might have already had a well-developed schema for how to tutor from their own prior experience with the *CTA* (see debate on internal vs. external scripts in Carmien, Kollar, Fischer & Fischer, 2006). On the other hand, maybe the *collaboration+reflection* group did not significantly benefit from the additional instruction because they did not fully engage with the exercises. Their general lack of participation in the collaborative reflection would support this theory. A third, but perhaps less likely explanation, would be that the apparent lack of effect of the additional instruction might simply be due to between-class differences and the small sample size. Perhaps the preparation exercises did help the *collaboration+reflection* group, but the *collaboration* group did not need the extra help to arrive at the same results as they were at a higher unit at the beginning of the study. In sum, it would appear that the reflection support had no effect.

Nevertheless, students had difficulty using the peer tutoring script effectively. In some ways, students did not comply with the script. Students in the *collaboration+reflection* group did not fully engage with the collaboration reflection activity. Students in the collaboration group tended to skip past problems they could not solve, denying themselves the opportunity to master the skills required for those problems. In other ways, students attempted to comply with the script, but had difficulty tutoring each other. In particular, compared to the *collaboration* condition, the *collaboration+reflection* condition completed fewer problems than the tutoring group, interacted less, and paused more (either in frustration or to ask the teacher for help). Given these results, it may seem surprising that there were no significant differences between the pre-post gains of the two groups. However, it should be noted that students in the collaboration group were able to solve more problems on the pretest than the *collaboration+reflection* group. Therefore, during the collaboration phase, students in the *collaboration* group were facing more problem types that they already knew how to solve than those in the *collaboration+reflection* group. They were able to complete those problems and move on to problems they had not yet mastered, completing more problems overall, but completing a similar number of problems that they were not able to complete on the pretest. As a result, even though there were differences in the absolute scores of the groups, the pre-post gain of both

groups was similar. Increasing the number of problems that students are able to complete while collaborating and decreasing the amount of help they require from the teacher should further improve student learning, because students will be given more of an opportunity to master the skills required by different problems and spend less time waiting for the teacher to respond.

In summary, while all students learned using the system, and students did exhibit positive collaborative behaviors, the findings of the classroom study suggested that the reciprocal peer tutoring script we designed exhibited the problems of other fixed script approaches surveyed in the background (*2.2*). The reflection support added to the system may have been support that students did not need, and in many cases, students did not comply with the activities. However, the fixed problem solutions provided by the current system did not provide sufficient support for the lack of students' domain knowledge. Consequently, students struggled to complete problems, skipped past difficult problems, relied too much on teacher assistance, and ultimately became discouraged. These findings suggest that the collaborative activities added to an intelligent tutoring system do need to be adequately supported, and support needs to be tailored to student needs. In particular, if students were to receive domain support from an adaptive system, they may benefit more from the activity. In *Phase 2*, I explore the effects of adaptive domain support compared to fixed domain support, addressing in part the research questions relating to the effects of adaptive support (*Q1-L1*, *Q1-L2*; "What are the effects of ACLS on student interaction and learning?"; see Table 1).

3.5.5 Implications for Iteration

There are several improvements to the peer tutoring script suggested by the study results. The first set relate to iterating on the fixed support given to peer tutors:

1. The collaboration phase should follow immediately after the preparation phase, instead of being on a different day. In this way the problems that peer tutors have just solved will be fresher in their minds, and they will be more able to link the preparation to the tutoring.
2. Peer tutors should be given answers to the problems that they are tutoring within the solver interface, rather than on paper. This change will make it easier for them to connect the answers with current problem.
3. The reflective elements of the script should be removed or improved.

There were also evident places were adaptive support might be useful for peer tutors:

4. Students should not be allowed to skip ahead without finishing the current problem. The intelligent tutor should check whether a tutee's done action is correct, and give the peer tutor feedback if the students incorrectly decide to move to the next problem.
5. The peer tutor should not be allowed to incorrectly mark the peer tutee's steps. If the peer tutee takes a correct action, and the peer tutor says it's incorrect, the cognitive tutor should provide feedback. If the peer tutee takes an incorrect action, and the peer tutor

says it's correct, the cognitive tutor should provide feedback. This mechanism would give the peer tutor a chance to get immediate feedback on their tutoring and the peer tutee would not be given incorrect guidance.

6. The peer tutor should be able to ask for a hint from the cognitive tutor. The cognitive tutor would then give the peer tutor the hint it would ordinarily give to the tutee, plus an instruction to talk to his or her partner about the hint. This mechanism gives the peer tutor a resource to consult other than the teacher if the tutor does not know how to proceed.

Intelligent tutoring support would be a next logical step in facilitating the peer tutoring, both in helping students to correctly master a greater number problems independently of teacher assistance, and in maintaining their engagement in the collaborative activity. In the next chapter, we describe the framework we developed to allow us to provide intelligent domain and collaborative tutoring to the collaboration. We will later describe the details of our incorporation of cognitive tutoring into the peer tutoring script, and the effects of this modification.

4 Development 1: The Collaborative Tutoring Research Lab

4.1 Introduction

The next step in our program of research was to add adaptive support to collaboration, and it became clear that to do so, we needed a better structure to support our ACLS implementation. In this section, we introduce the Collaborative Tutoring Research Lab (*CTRL*), a research-oriented framework for adaptive collaborative learning support that facilitates the collection of multiple streams of process data, the development and integration of assistance based on the data, and the implementation of relevant comparison conditions for experimental control. *CTRL* extends the individual tutoring scenario (one student, one tutor) to a collaborative multi-tutor setting (multiple students and multiple tutors, with different roles or for different purposes). One of the strengths of the framework is that it focuses on reusability: it facilitates the addition, removal, and integration of components. In *CTRL*, adaptive collaborative conditions can be developed more rapidly by using existing computational models. For example, a meta-tutor for sharing information with a teammate would be able to use results provided by a domain tutor about whether the facts shared were correct. *CTRL* facilitates the addition and removal of components in order to create appropriate comparison conditions for adaptive support.

In this chapter, I outline the basic components involved, the way they interact with each other, and the way they can be integrated. Throughout the rest of the dissertation, I describe how *CTRL* is used to implement our iterations of adaptive peer tutoring support. The focus of *CTRL* is on facilitating interactions between different components, and I define and discuss each of those components in more detail in *4.2*. In *4.3*, we describe how the various components communicate with each other. *4.4* outlines how the control module interacts with the research management store to allow the flexible integration of components and construction of multiple collaborative conditions. The work in this phase was discussed elsewhere in Walker, Koedinger, McLaren, and Rummel (2006) and Walker, Rummel, and Koedinger (2009a).

4.2 Component Functionality

CTRL consists of six different types of components, based in part on Ritter and Koedinger's (1996) description of plug-in tool and tutor components (se Figure 7):

1. *Tools*: Used by the student to take problem-solving actions
2. *Tutors*: Provide students with assistance
3. *Translators:* Facilitate inter-component communication and implementation of scripts
4. *Learner Management:* Stores curriculum information and student model data
5. *Research Management:* Stores protocol logs and information about how the components involved can be integrated with each other (session types)

6. *Control Module:* Constructs and manages collaborative sessions, both on a problem-to-problem level (session manager) and on an action-to-action level (mediator)

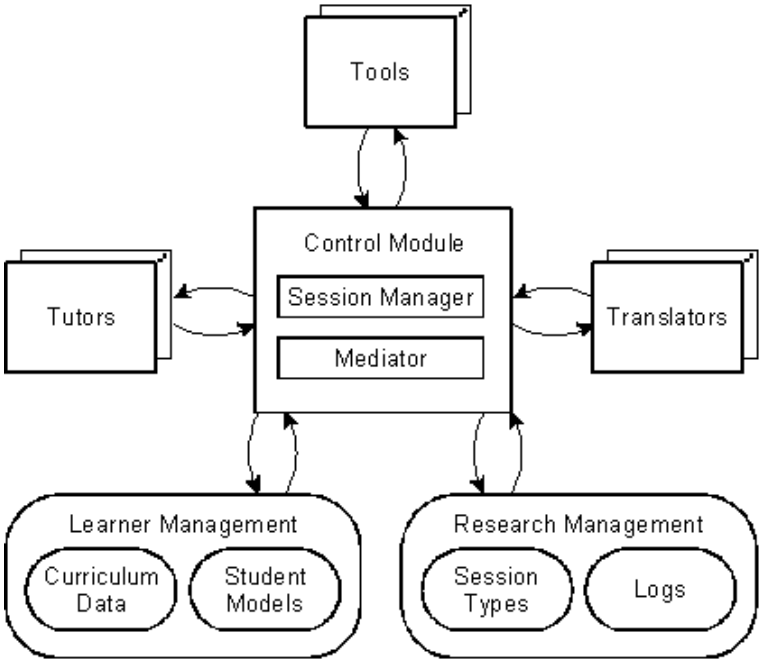

Figure 7. High level overview of CTRL. CTRL consists of tool, tutor, and translator agents, learner and research management data stores, and a central control module.

A *tool* is a piece of software that a student interacts with in order to solve problems in a particular domain. A tool could be as simple as a text-editor that allows students to write essays or as complex as a simulation environment for chemistry experiments. *CTRL* allows any number of tools to be involved in the learning scenario. Multiple users can collaborate remotely while each one uses different tool components. There is not necessarily a one-to-one mapping between students and tools; a single student could have access to multiple tools (e.g., an instant messaging tool in addition to the text-editor), and two students could conceivably be using the same tool at the same computer. However, we assume for the purposes of this discussion that in a condition with multiple users, a tool represents a single user's interaction with the system as a whole. Tool components contain the user interface, a domain model, and meta-knowledge of tutoring. The interface is the point of interaction between the user and the system. The domain model is present so that the tool can update its state without input from an additional component. A user can then interact with a tool without input from any tutoring component, and therefore a tool is not bound to a given tutor. For example, in a chemistry simulation environment, the interface might allow students to mix different chemicals, and the domain model might calculate and display the result of mixing the chemicals.

Although tools should be able to share domain models, this behavior is currently not explicitly supported by *CTRL*, in part because of our focus on using pre-existing components that already have a domain model. Tools also contain meta-knowledge so that they can convert feedback from a tutor agent into a format appropriate for display. Thus, tutors can be used with any tool because they do not need to send tool-specific messages. When the chemistry simulation tool mentioned above receives a hint message from the tutor, it might display it in a pop-up dialogue in the interface, while if a collaborative discussion tool receives the same message, it might display it as part of the chat interaction. The functionality that we have described is ideal, but it is likely that many pre-developed tools we may want to use will not incorporate all functionality, and may be difficult to modify. In these cases, we use *translator* components to compensate for the missing functionality.

Translator components are all-purpose facilitators that bridge communication between other components. They have two general functions. First, they make it possible to integrate components that do not conform exactly to the framework specification by providing missing functionality (e.g., an implementation of tutoring meta-knowledge) or by converting individual component messages into the standard message format. For example, if a particular collaborative discussion tool does not know how to handle a hint message, a translator would need to be built to convert the abstract message (e.g., *giveHint*[hint]) into a format the tool understands (e.g., *displayInChat*[hint]). This aspect of translator functionality is very much in line with the translators discussed in Ritter and Koedinger (1996) and Kumar et al. (2007). Second, translators can impose a structure on the collaborative interaction by communicating certain actions across tool components (such that a user action on one component is displayed on all other relevant components) and by triggering changes related to collaboration scripts to the tool components. For example, a translator could be used to allow some actions made by one student to appear on the other student's screen, but not others. This approach, where translators facilitate collaboration, is different from the more traditional object coupling approach in CSCL systems (Suthers, 2001), where students can automatically see all actions made in a shared workspace. There may be cases during a student interaction where actions that would generally be collaborative should not be shared (e.g., when one collaborating student makes an error, it may not always be desirable to broadcast the error to group members). We chose this implementation so that a designer of a learning environment has more control over structuring the interaction between students. Like tools and tutors, there can be any number of translator components incorporated in a learning scenario. The implementation of a given translator depends on its function.

Tutor components are any components that provide adaptive support to students, generally by comparing their actions to a model, providing assistance based on the model, and assessing skills based on the model. Tutors might range from a domain tutor for writing grammatical sentences based on a constraint-based model to a metacognitive tutor for proofreading a paper based on a cognitive model. Any number of tutors can be involved in a learning scenario, and any type of tutor can be used in our framework. Tutor components should contain an expert model, a feedback model, and a student model. Like in regular intelligent tutoring system functionality (as described in VanLehn, 2006), the expert model

does model tracing by evaluating the student action, the feedback model determines the sort of feedback that is given, and the student model does knowledge tracing by assessing the student performance (or in some cases, the group performance). As with tools, any pre-existing tutor components used that do not have the desired functionality can be augmented with a translator component.

4.3 Message Protocol

All components communicate with each other using a standardized set of messages, providing guidelines for the development of new components that can be incorporated into the framework (see Table 13). As components may be running on different machines, messages are sent remotely. In these messages, details specific to the implementation of individual components are hidden as much as possible and only abstract semantic content is communicated. In this paragraph, we will enumerate the high-level representations that form the parameters and return values of the messages sent, and in the following paragraph we will discuss the types of messages themselves. First, a *Student Interaction,* or a step that can be taken by a user in the interface, is represented using four parameters:

1. *Student* – the student taking the action
2. *Selection* – the widget being acted upon
3. *Action* – the action performed upon the widget
4. *Input* – any additional information necessary for the action

For example, a student with an id of jmiller entering 25 in a table might be represented as (student = jmiller, selection = cell A1, action = enterValue, input = 25). The concept of a selection-action-input triple can be traced back to Anderson and Pelletier (1991). A *Tutor Response* to a student interaction is represented by four parameters:

1. *Tutor* – the tutor sending the message
2. *Action Evaluation* – the type of message (e.g., correct, incorrect, highlight)
3. *Feedback Message* – any message the tutor wants to send
4. *Skill Assessment* – the change in student skill values

For example, a domain tutor might approve the student action in cell A1 (indicating it was correct), send a feedback message for encouragement (e.g., "Keep it up! What goes in cell A2?"), and increase the value of the relevant skill (e.g., set the skill "entering values in a table" to 60%). As described in Table 13, information that is not a Student Interaction or Tutor Response (such as current problem details) is communicated as a set of Properties**,** which is a conventional data structure containing any number of attribute-value pairs.

Table 13. Messages passed between components.

Message Name	Input	Output	Sending Components	Receiving Components
launchComponent	Component properties	Success or failure	Session Manager	Tool, Tutor, Translator
quitComponent	None	Success or failure	Session Manager	Tool, Tutor, Translator
getNextProblem	Problem-selection properties	Problem properties	Session Manager	Tool, Tutor, Translator
changeProblem	Problem properties	Success or failure	Session Manager	Tool, Tutor, Translator
processInteraction	Interaction	None	Tool, Translator	Tutor
scriptInteraction	Interaction	None	Translator, Tutor	Tool
processFeedback	Interaction, Response	None	Tutor, Translator	Tool
setProperty	Component property	None	Translator, Tutor	Tool, Translator, Tutor
getValue	Attribute	Value	Translator, Tutor	Tool, Translator, Tutor
putData	Data properties	None	Mediator	Learner Management, Research Management
getData	None	Data properties	Session Manager	Learner Management, Research Management

These data structures are then used as parameters and return values for the message types exchanged between components (see Table 13). For example, when a session is started a *getData* message would be used to retrieve relevant curriculum and student information, and *launchComponent* messages would be used to start and configure all the relevant components. While elements of this message protocol are taken from Ritter and Koedinger (1996), the protocol is more abstract than the protocol that they defined, in order to facilitate a variety of potential learning environment interactions. Because the problem-

solving interactions are the core messages of *CTRL*, here we present an in-depth example of how those messages might be used by the different components (see Figure 8). The example includes two tools (representing two collaborating students, Bo and Jan), two tutors (representing a domain and collaborative tutor), one translator to implement the shared collaborative workspace, one research management component, and the mediator subsection of the control model. In the example, the tool receives input from the user and sends information about the user action to the control model, using a *processInteraction* message. Once the control module receives the message, it logs it, and then redirects it to all components that should receive it (in this case, the translator and the two tutors). The translator takes the message and transforms it into a *scriptInteraction* message in order to reproduce a student action on another interface, which is sent back to the control module. Meanwhile, the domain (math) tutor evaluates the user action, and sends its feedback to the control module, which passes it along to the collaboration (chat) tutor using a *processFeedback* message. The collaboration tutor, using the user action and the

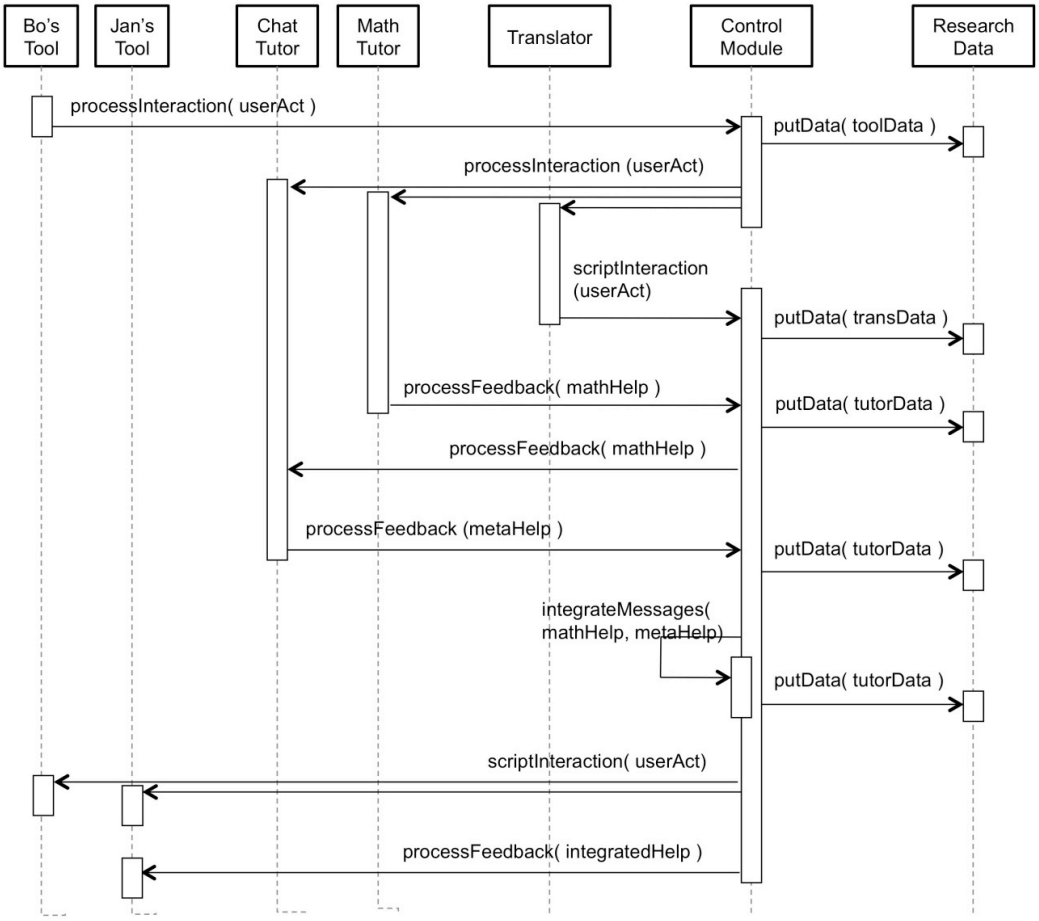

Figure 8. Message-passing between two tools, two tutors, and a translator. Each tool represents a collaborating student. One tutor supports student interaction and one supports problem-solving.

feedback as input, evaluates the action and sends its feedback back to the control module using a *processFeedback* message. The control module has now received messages from the translator, the collaboration tutor and the domain tutor. The control module integrates the messages, passes the *scriptInteraction* message along to both tools, and then sends the feedback message to Jan's tool, as specified by the integration logic in the control module. Although not all collaborative scenarios will operate in exactly this way, these messages form the building blocks for handling interactions between tool, tutor, and translator components.

We have explicitly chosen to leave some elements necessary for implementing a computer-supported collaborative learning system unspecified, because they are outside of our main focus. As the system is distributed, some components of the system (e.g. the control module) will run on a central server, and some components (e.g. the tool components) will run on various clients. However, the way components are distributed may depend on the deployment environment, so we leave it purposefully ambiguous. Also, because components are distributed, all messages need to be sent remotely, and we leave the implementation of the specific protocols up to the developer. Finally, to be deployed in a classroom, multiple sessions handling multiple student pairs need to be run at once, meaning that a server needs to handle client logins and launching the collaborative sessions.

4.4 Component Integration

In addition to illustrating how messages are passed between components, there are several notable elements of the above example that highlight the centrality of the control module during a session. All messages sent go through the control module, which logs the messages prior to sending them to the relevant components. In this manner, the logging of different streams of interaction is combined within a single framework. Further, the control module is in control of which components are involved, where messages get sent, and how messages are integrated. Using the control module, a translator component can be built to echo messages from one tool to another, facilitating collaboration. Additionally, the output of one tutor module can be used as input to a second tutor module, facilitating the integration of different tutor components. While *CTRL* is not the only architectural framework to use a federated system (see Rosateli & Self, 2004; Mühlenbrock, Tewissen, & Hoppe, 1998), its contribution is that it focuses specifically on integrating different tutor components and on the efficient implementation of comparison conditions.

The control module facilitates the integration of different components, helping to meet our goals of providing complex adaptive functionality and making it easier to create control conditions in experiments. In standard use of the intelligent tutoring system, each individual component has knowledge of where it is sending and receiving messages, and this configuration works because the system is so simple (the tutor sends messages to the tool, the tool sends messages to the tutor). With multiple components, a central body is needed to manage all the communication. The control module uses a representation of the session characteristics in order to determine how to route the messages. Each condition facilitated by *CTRL* is represented as a *session type* stored in research management. Each session type contains three arrays

corresponding to three different types of components (tool, translator, tutor). Session types also contain a set of logical rules for how messages are passed between components. These rules can be as simple as:

IF *m* is a message
THEN send *m* to every tutor

However, some rules will need to be more complex, as they should also represent how to integrate feedback messages from different tutors. For example, if there is a participation tutor and a domain tutor involved in a session, a rule represented in the session type might be:

IF step *s* is incorrect
AND *m* is a domain feedback message
AND student *a* has not participated sufficiently
AND *n* is a participation feedback message
THEN aggregate *m* and *n* and send *m* + *n* to *a*.

Rules can involve any information available to the mediator, including the components involved in the message, the parameters of a particular message, curriculum or student parameters, and a pre-set priority of the message.

Once a session type has been created, the session manager and mediator can use it as a guideline for how different components should be interacting. When a collaborative session is started, the session is associated with a given session type. How this association is made is left open: it can be based on user login, or a particular curriculum, or even be selected by the user. The details of the particular session type discussed in the above paragraph are then retrieved from research management and stored locally in the control module. The session manager iterates through the components involved to send a high-level message (e.g., launching each component). The mediator's function is to control the low-level message passing between components by intercepting all messages sent by a component and directing them to the appropriate targets, following the rules outlined in the session type. Therefore, based on the session type activated, the same components can be used in different ways. Adding or removing a component can be as simple as creating a new session type, without the need to modify the other components involved in the interaction.

The central control module also facilitates the creation of an integrated log of collaborative interactions. In *CTRL*, each semantically meaningful action occurring within a component is sent to the control module, which transforms the action into xml, and sends it to a data store in the research management component. In this manner, logs from each component are automatically integrated and can be reviewed together after a study without any further processing. The logging protocol of the architecture is based on the Pittsburgh Science of Learning Center protocol (PSLC, 2009), which records semantic-level messages sent from tool and tutor components. These tool and tutor logs follow the concept of a transaction described by VanLehn and colleagues (2007), where a user action and the tutor response to the action are linked. In our framework, a *processInteraction* message is logged as "tool message" to the learner management module, with the student interaction parameters, a unique id, and a timestamp being

represented in the log (see Figure 9). A responding *processFeedback* message is logged as a "tutor message" to the learner management module, with the student interaction parameters, tutor response parameters, and a timestamp being captured. The relationship between the tool and tutor messages is also represented, as the tutor message contains the ID of the tool message that triggered it. Logs include *context messages,* which are initiated by the control module, and record information about the problem being solved, the settings of the learning environment, or the experimental design. Once a relevant context message has been logged, both tool and tutor messages are linked to it, containing the context message id.

Because *CTRL* is designed for adaptive collaborative learning systems rather than individual intelligent tutoring systems, the logging supported needs to be broader than the protocol discussed by VanLehn and colleagues (2007). Thus, an additional type of message is supported: a *scripting* message, logged whenever a module changes the problem state of a tool. In this case, the student interaction parameters, the timestamp, the relevant context message id, and the relevant tool message id are logged. Second, because CTRL supports multiple users on multiple tools, it is important not only to record the user of the message (part of the student interaction parameter), but the collaborative session of the user, and the role of the user within that session. I incorporate this information into the context message, which logs the learning environment settings. Third, because CTRL supports multiple tool responses, the relevant metaphor for analyzing the data is not a single tool-tutor transaction but a chorus of responses to a tool action. Not only does each tutor response need to be logged, but also the final message constructed by the mediator to be sent to each tool.

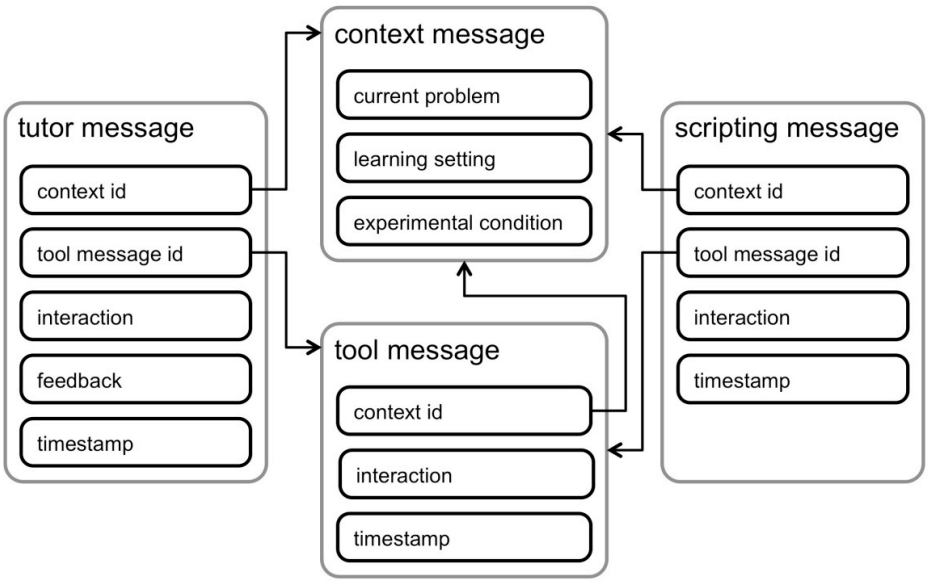

Figure 9. Logging format for student-tutor interaction. Logs consist of context messages, tutor messages, tool messages, and scripting messages.

4.5 Summary of Technical Contribution

The construction of *CTRL* addresses research question *Q2-T1* as outlined in Table 1: How can existing and custom components be integrated to enable ACLS? *CTRL* captures rich process data, integrates feedback from multiple tutor components, and ideally makes it easier to implement comparison conditions. All semantic messages from components are sent to the control module, which creates a log of all student interactions including verbal interaction, collaborative problem-solving actions, and the intelligent tutor responses. Multiple preexisting and custom-built intelligent tutors can be incorporated into the system by changing the definition of a session type in the mediator. Domain-general intelligent tutors can use the output of domain-specific tutors as input into their models. Finally, because components are designed to be independent, it becomes possible to remove components from collaborative sessions in order to create multiple comparison conditions.

CTRL focuses directly on the interaction between collaborating students and intelligent support, and thus there are a number of ACLS applications that are outside its scope. It is appropriate for use in scenarios where a small number of students have been placed in a group and are collaborating on a particular task. *CTRL* is not designed to adaptively assign students to particular groups or tasks; that is, it is not a tool for manipulating the preconditions of the interaction. However, *CTRL* can be applied in conjunction with a wide variety of different sets of preconditions, once they have been specified, and there is nothing inherent in *CTRL* that restricts the domains for which it is used. *CTRL* is also not specifically designed for macro-scripting the interaction (e.g., by specifying a sequence of phases for students to follow, such as alternation between individual and collaboration phases). Although it is possible to implement a macro-script using a translator, the challenges of managing a macro-script are not addressed by the design of *CTRL*, and there may be simpler ways for doing so within a given adaptive system. Despite this limitation, *CTRL* can be applied to manage interactions *within* the phases of macro-scripts. Finally, although *CTRL* could be applied to asynchronous interaction, it was designed with synchronous interaction in mind, and it is likely there are other frameworks more appropriate for managing asynchronous communication. Within these parameters, adaptive or fixed micro-scripting of synchronous interaction between a small number of students given a particular task, *CTRL* actively facilitates the implementation of different types of interactions.

In determining what is necessary for other researchers to use *CTRL*, it is important to make the distinction between the conceptual framework itself and our specific implementation using existing *CTA* components. The mediator component of *CTRL* is simple to implement, and one could imagine other researchers adopting this concept in order to develop their systems. However, the difficult part of applying *CTRL* is refactoring the components of existing systems to separate the tool, translator, and tutor functionality. For large and complicated systems whose code has been developed iteratively and by multiple people, this process can be a challenge. Ideally, once the code has been refactored, it would not be necessary to make modifications to the existing components. However, practically, this is not the case; it still can be difficult to interpret and modify the code relating to the existing tutor components. In cases

where existing components are used, we need to do more work towards reducing the need for them to be modified. As more systems are developed with a component-based approach, *CTRL* will become more and more effective.

In this chapter, we have outlined a conceptual framework called *CTRL* that supports educational technologists in developing adaptive support for collaboration and educational psychologists in investigating its effects. The framework enables researchers to integrate different types of adaptive support and, in particular, allows domain-specific models to be used as input to domain-general components in order to create more complex tutoring functionality. Additionally, the framework helps researchers to implement comparison conditions by making it easier to vary single aspects of the adaptive intervention through removing tool or tutor components from a system. We see one of the main contributions of this work as the development of a framework that supports the integration of pre-existing and custom-built components, with a particular focus on tutoring components. Throughout the remainder of the work discussed in this dissertation, *CTRL* made adaptive support for collaboration possible to implement, and enabled the development of relevant comparison conditions.

5 Phase 2: Adaptive Correction Support

5.1 Introduction

In this chapter, I describe the first iteration of *APTA,* an Adaptive Peer Tutoring Assistant that delivers adaptive support to peer tutors. As a result of the work in *Development 1: Collaborative Tutoring Research Lab* (Chapter 4), I now had a flexible framework for combining existing and custom-built components to create ACLS. The results of *Phase 1: Peer Tutoring Learning Environment* (Chapter 3), suggested that I should first focus my attention on providing adaptive support for peer tutors that helps them to know how to solve the problem they are tutoring and communicate that knowledge to their partner. Giving tutors more adaptive access to correct problem-solving information might have cognitive benefits, leading them to reflect more on problem-solving steps, and motivational benefits, in that they will feel like better tutors. Further, it is likely that tutees will then receive more correct help and benefit more from the activity, as well as feel less frustrated as they are solving the problems. The implementation of *CTRL* made it possible to develop and evaluate this adaptive support.

I explored whether adaptive domain assistance would indeed have these positive effects using both techniques from individual intelligent tutoring and the already developed domain models found in the CTA. First, I designed a simple computational model of good peer tutoring, focusing on the domain challenges of the task. I designed hints and feedback for the peer tutor that combined existing cognitive hints found in the CTA with collaborative prompts (*5.2*). As part of this design, I explored the role existing domain information plays in the design of the collaborative models and support (investigating *Q2-D1* and *Q2-D2*; see Table 1). By leveraging the existing intelligent tutoring domain model present in the *CTA* and using *CTRL* to integrate it with a custom-built collaborative model, the new components that had to be developed to provide the support were minimal (described in *5.3*). This implementation demonstrates an instantiation of CTRL, furthering work on *Q2-T1* ("How can existing and custom components be integrated?").

Then, I conducted a study investigating the effects of collaboration and adaptive support on student interaction and learning by comparing three conditions: 1. Students used the CTA individually (*individual condition*), 2. Students tutored each other (*fixed collaboration condition*), and 3. Students tutored each other with adaptive domain support (*adaptive collaboration condition*). I expected the adaptive collaboration condition to learn more than the fixed collaboration condition because of the adaptivity of support, and to learn more than the individual condition because of the benefits of collaboration. The study addresses research question *Q1-L1* ("What are the effects of ACLS on student collaborative interactions?"), and is described in *5.4*. In the analysis of the study data, I rely heavily on information derived from CTA components, exploring question *Q2-L1* ("How can intelligent tutoring-style data logs augment the analysis of collaborative study data?"). The work in this phase was discussed elsewhere in Walker, McLaren, Rummel, and Koedinger (2007b), Walker, Rummel, and Koedinger (2008), and Walker, Rummel, and Koedinger (2009b).

5.2 Design: Adaptive Domain Hints & Feedback

5.2.1 Interactions: Script Cohesiveness

After *Phase 1: Basic Peer Tutoring Script* (Chapter 3), I kept the basic activities present in the peer tutoring script constant, with only three major modifications: phase restructuring, removal of reflection activities, and digital problem solutions. These modifications were designed to help students make better connections between script phases and activities.

Phase Restructuring. I modified the alternation between the preparation and reflection phases, such that students did the preparation phase during the first half of a class period and then did the collaboration phase during the second half of a class period. As peer tutors struggled with giving domain help, I wanted the preparation problems to be fresh in peer tutors' minds as they began to help their tutees. I further made a small change by having one student tutor during one class period, and their partner tutor in the second period. In theory, this design makes it possible to assess learning from tutoring and learning from being a tutee separately.

Removal of Reflection Activities. Second, based on the results of the previous study, I removed the majority of the reflection exercises I had incorporated. I kept the preparation reflection, as students had appeared to engage well with that, but removed the chat practice, as students did not appear to need it, and the collaboration reflection, as students did not use it properly. Given my research questions, I decided to focus my attention on providing adaptive support instead of providing additional training and scaffolding to encourage students to use the collaboration reflection more appropriately.

Digital Problem Solutions. Finally, I addressed a serious problem with the previous version of the script by incorporating student problem solutions in the interface rather than giving them to students on printouts. It was a lot of effort for students to find the appropriate problem solutions on the printout, and it appeared that students found them difficult to read. By adaptively presenting the relevant solution in the interface along with the current problem, peer tutors could more easily compare the tutee's solution to the problem solutions. For similar reasons, in this scenario, peer tutors were presented with ideal problem solutions rather than their own problem solutions – for some students, their own problem solutions took upwards of 15 steps, and thus were very difficult to decipher when used as a resource for tutoring.

5.2.2 Model: Correct Help-Giving

Approach & Assumptions. Using rational task analysis, I designed a model of peer tutoring that represents the basic mechanics of peer tutoring, in order to support peer tutors in giving more correct help. In addition to the benefits of introducing more correct help into the discussion, supporting the simple mechanics of the tutoring task may lead peer tutors to be freer to engage in the more complex cognitive aspects of the task, such as generating elaborated explanations targeted toward tutee misconceptions.

For simplicity, the model is constrained in three ways. First, it assumes that peer tutor and tutee actions are synchronous, in that every action by the tutee is followed immediately by a tutor response. In practice, this is unlikely and may be undesirable; a fast tutee should not be held up by a slow tutor. Second, our model assumes that certain actions will always be taken even if those actions have been made redundant by other actions. For example, the peer tutor marks an answer incorrect before giving the peer tutee feedback. In a real situation, the peer tutor might simply tell the tutee that their answer is incorrect while giving them feedback, and therefore would not need to explicitly mark it. The model treats this single action as two separate actions. Finally, for the time being, the model represents tutor-tutee discussion at a very high level.

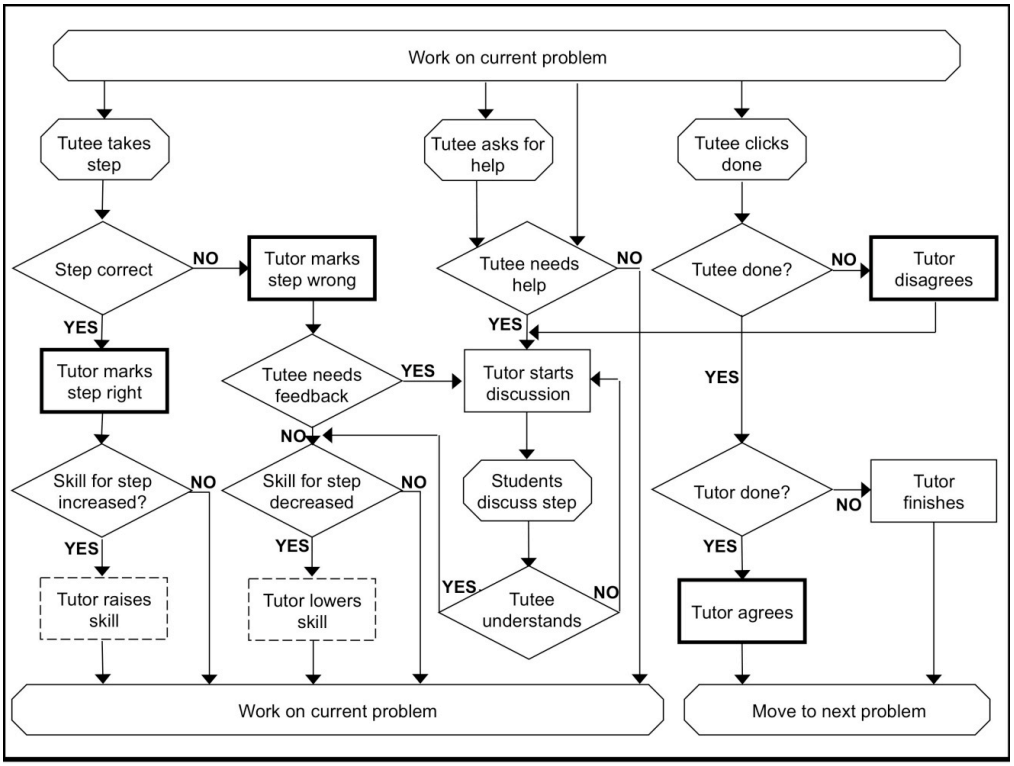

Figure 10. "Ideal" model of basic peer tutoring. Bold boxes are correction actions, dashed boxes are skill assessment actions, and regular boxes are discussion actions.

Ideal Behavior. The model starts when the students are in a state of "working on the problem" (see Figure 10). The peer tutee can take one of three actions: take a problem step, ask for help from the peer tutor, or indicate they are done. The peer tutor can also begin a model sequence by determining that the peer tutee needs help. The model can then be divided into three general types of peer tutor responses: correction (bold boxes), skill assessment (dashed boxes), and discussion (regular boxes). The model assumes that correction is the immediate response to the tutee taking a step or selecting done. If the tutee action is correct, the tutor should mark it right. If the step or action is incorrect, the tutor should mark it wrong. The peer tutor starts a

discussion whenever the tutee needs help or feedback, which may be the case after an incorrect answer by the tutee, when the tutee requests help, or when the tutee appears to be struggling. We have divided the discussion into three substeps: Initiation ("Tutor starts discussion"), the bulk of the discussion ("Students discuss step"), and termination ("Tutee understands?"). The peer tutor engages in skill assessment after correction and discussion actions, adjusting the tutee's skill bars as appropriate. Skill assessment is the lowest priority; the tutee needs feedback in order to continue but can move to the next step as the tutor adjusts skills.

Buggy Behavior. Using data from *Phase 1: Peer Tutoring Learning Environment* (Chapter 3), I was able to identify three departures from the model that students tend to take that might interfere with their learning:

1. *Being unable to provide domain help.* In the *tutoring + reflection* condition from *Phase 1*, students completed on average less than five problems during the collaborative session. Peer tutees would ask tutors for hints, and tutors would be unable to provide guidance, saying "I don't know" in response to questions or waiting for teacher help. If peer tutors lack basic knowledge on how to proceed with the problem, it is unlikely that tutoring will be effective for either party.
2. *Incorrectly moving to the next problem.* In *Phase 1*, students in one class only completed roughly 60% of the problems they attempted. For 40% of the problems, they skipped to the next problem even if the previous problem was not yet done. Agreeing to move to the next problem when the problem is not yet done, and agreeing that a step is correct when in fact it is incorrect, may prevent students from identifying or repairing their misconceptions.
3. *Being overenthusiastic about moving skill bars up and down.* During *Phase 1*, we noticed that peer tutors would tend to raise their partner's skill bars all the way to the maximum during the first or second problem. While it may be beneficial for students to use skill bars to support their partner's efforts in problem solving, students probably do not receive the full benefits of the skill bars if they are not used to reflect on the tutee's actual skills.

5.2.3 Support: Peer-Mediated Hints & Feedback

In the design of the adaptive support, peer tutors were given support by the intelligent tutoring system in response to the buggy behaviors identified in the previous section. First, in cases where the peer tutor does not know how to solve the problem, *APTA* (the Adaptive Peer Tutoring Assistant) provided hints on demand to support the peer tutor in giving help to the tutee. The peer tutor could request a hint from the computer tutor at any time. Hints were multi-level; each level consisted of a randomly selected prompt to collaborate, and then the domain help tutees using the individual *CTA* would have originally received at that level. Domain help included both instrumental help on how to solve the problem (typically at the last level), but also conceptual hints and explanations related to the next step. Thus, it contained a lot of material peer tutors could use to construct their own explanations.

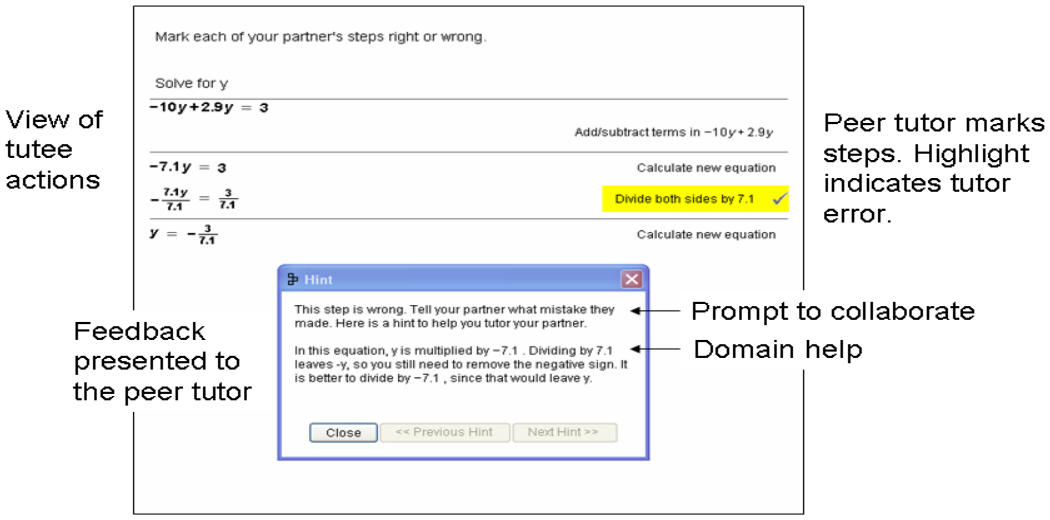

Figure 11. Adaptive correction support presented to the peer tutor after the peer tutor has marked an incorrect step correct. The error is highlighted, and the peer tutor receives a feedback containing a prompt to collaborate and domain help.

The next two cases of support related to feedback on peer tutor correction actions in the interface. After the tutee clicked the done button, if the peer tutor incorrectly agreed that the problem was done, he or she would be notified and told to ask for a hint about how to complete the problem. Similarly, if the peer tutor incorrectly disagreed that the problem was not done, he or she would be told that the problem was in fact done, and the students would be moved to the next problem. Next, if the peer tutor marked something incorrectly in the interface (e.g., they marked a wrong step by the tutee correct), the computer tutor would highlight the answer in the interface, and present the peer tutor with an error message (see Figure 11). Like the hint message, error messages were composed of a prompt to collaborate and the domain help the tutees would have received had they been solving the problem individually. In the last support case, peer tutors were sent a feedback message if they tried to raise a skill bar more than 15% per problem that read, "Slow down! Before increasing more, wait until your partner has shown this skill on another problem." Peer tutors received a similar message if they tried to lower a skill bar more than 15%.

The intention was that the domain hints and feedback would stimulate the peer tutor's reflective processes, while incorporating more correct and conceptual content into the interaction. The feedback provides an incentive for peer tutors to mark tutee steps, as they get information on whether the steps are right or wrong that they can then use to tutor their partner. Not only is the feedback designed to trigger reflective processes on the part of the peer tutor by encouraging them to mark steps more frequently, the feedback serves to draw the peer tutor's attention to misconceptions by letting them know when they have made an error marking a problem step, further encouraging him or her 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 make peer tutors feel more efficacious, empowered tutors, but also to further introduce more correct and conceptual content into the interaction. This additional input into the interaction triggers co-constructive processes, potentially leading to more learning. The skill bar feedback is also intended to induce students to reflect more on the skills required to solve the problem.

There are two aspects to the design of the adaptive domain support that depart from typical ACLS feedback. First, the feedback delivered was peer-mediated, in that domain feedback ultimately intended for the tutee was presented to the peer tutor, and it was the peer tutor's responsibility to explain it to the tutee. By using peer-mediated feedback, the tutee could focus on solving the problem, and the peer tutor could process the feedback and tailor it directly to the tutee. Additionally, having the peer tutor explain the feedback in his or her own words was intended to lead tutors to elaborate on his or her own knowledge. Second, feedback was based on the peer *tutor's* actions, and not solely on the peer *tutee's* actions. In this manner, students received less support than they would have if they were using the tutoring system alone, because peer tutees did not receive feedback after every mistake. Peer tutors were only intended to receive feedback when they could not help the tutee on their own. It is important to note, the goal in providing students with feedback was not simply to force the peer tutor to reproduce all help the *CTA* would have provided. The more the peer tutor elaborates on the feedback, the more both students will benefit.

5.3 Implementation: Using CTA Models to Add Adaptive Correction Support

The adaptive support was implemented in Java as an instantiation of the *CTRL* framework described in *Development 1: Collaborative Tutoring Research Lab* (Chapter 4), with a mixture of custom-implemented components and components that were originally part of the *CTA*. The instantiation included two tool components (the peer tutor's interface and the peer tutee's interface), a translator component (to echo actions from one tool to the other tool), and two tutor components (a cognitive tutor component to evaluate the peer tutee's problem-solving actions, and a *correction tutor* component to evaluate the peer tutor's collaborative actions). Also included were a learner management component, a research management component, and a control module to integrate all the components. As described in *Phase 1: Peer Tutoring Learning Environment* (Chapter 3), the tool components were implemented based on the equation solver tool already found in the *CTA*, and further modified to create the peer tutor's and peer tutee's interfaces. I also used the custom-made translator component described in Chapter 3 to facilitate collaboration between the two users. The translator was constructed based on the *CTA* tools, and is therefore not a general component for facilitating collaboration.

5.3.1 New Tutor Component: Correction Tutor

Assessment. The correction tutor had four types of input (see Table 14). I used the cognitive tutor component evaluation of tutee problem-solving steps and information about the next step hints. I also used the correction and skill assessment actions logged as part of the peer tutor component.

Table 15. Assessment in the correction tutor in *Phase 2.* The assessment of student interaction is based on the cognitive tutor evaluation of tutee actions and on the peer tutor actions.

Input	Component	Description
Evaluation of tutee steps	cognitive tutor	Whether a problem step is correct or incorrect.
Next step hint	cognitive tutor	The hint the CTA would have given on the next step.
Correction actions	peer tutor	Whether peer tutors mark a step right or wrong.
Assessment actions	peer tutor	Whether peer tutors change the assessment of a skill.

Table 14. Model tracing in the correction tutor in *Phase 2*. Rules are written from the peer tutor perspective. The "++" represents effective behaviors, and the "—" represents ineffective behaviors. The support column indicates whether support is given or not.

#	Skill	Type	Rule	Agent	Support
1	correction	++	IF tutee takes step x AND x is correct THEN mark x correct	interface CTA interface	no
2	correction	++	IF tutee takes step x AND x is incorrect THEN mark x incorrect	interface CTA interface	no
3	correction	--	IF tutee takes step x AND x is correct THEN mark x incorrect	interface CTA interface	yes
4	correction	--	IF tutee takes step x AND x is incorrect THEN mark x correct	interface CTA interface	yes
5	skill assess	++	IF tutee displays skill y on problem p THEN increase assessment of y by <= 15%	judgment interface	no
6	skill adjustment	++	IF tutee displays lack of skill y on problem p THEN increase assessment of y by <= 15%	judgment interface	no
7	skill adjustment	--	IF tutee displays skill y on problem p THEN increase assessment of y by >15%	judgment interface	yes
8	skill adjustment	--	IF tutee displays lack of skill y on problem p THEN decrease assessment of y by >15%	judgment interface	yes

Model Tracing. The correction tutor component consisted of eight simple rules related to peer tutor effective and ineffective behaviors, outlined in Table 15. The rules span two skills, the peer tutor ability to correct tutee steps (*correction*), and the peer tutor assessment of tutee skills (*skill adjustment*). Each skill has two rules relating to effective behaviors (represented by the "++" in the type column, and two rules relating to ineffective behaviors (represented by the "--"). Rules related to correction involved a simple comparison between the cognitive tutor and peer tutor response to a tutee problem step. In the case where the responses match (rules 1 and 2 in Table 15), the peer tutor has exhibited effective behavior. In the case where the responses do not match (rules 3 and 4 in Table 15), the peer tutor has made an error in marking tutee steps, exhibiting ineffective behavior. The skill adjustment rules involve a specification for how much the peer tutor should increase or decrease a particular skill per problem. At this point, peer tutors behavior is considered to be ineffective if he or she increases or decreases the skill by more than 15% (rules 7 and 8). However, the peer tutor makes the judgment about whether knowledge or the lack of knowledge of a skill has been displayed, without correction tutor input. In theory, the cognitive tutor assessments of which skills have been displayed could be linked to the peer tutor assessments, to provide more sensitive modeling in this area, and this would be a good area for future work.

There is considerable overlap between the idealized model of peer tutoring described in the design subsection (*5.2.2*) and the implementation of the model, although some of the model assumptions have been relaxed even further. The *correction*-related production rules map to the paths in the model where the tutee takes a problem step, and the peer tutor marks the step right or wrong based on whether the step is correct or not. However, unlike the model assumptions, which specify that each tutee action should be followed by a peer tutor response, the implementation of the rules does not require that the peer tutor respond to any given step with a correction action or skill adjustment action. This modification adds two additional dimensions to the implementation of the model. First, it gives more flexibility to the peer tutor, who can now decide which steps are important to respond to. If the peer tutor does respond to a step, it may be because they themselves want feedback on their correction response, in addition to representing their desire to communicate the correctness of a step to tutees. Second, the implication of this implementation is that all rules representing peer tutor actions are optional. While the model represents *sequences* of tutee-tutor actions and checks to see whether a sequence is effective, it does not limit peer tutors to the sequences described by the model.

Knowledge Tracing. The rules in the correction tutor relate to two overall skills: Peer tutor *correction* abilities, and peer tutor *skill-adjustment* abilities. However, in this phase I did not assess peer tutor mastery of these skills.

Support Construction. When one of the correction bug rules fire, the tutoring model considers the type of problem step (e.g., a solver action) and peer tutor response (e.g., the peer tutor marked it incorrectly *wrong*)

in choosing from a fixed set of collaboration-oriented meta-feedback (e.g., "Your partner is actually right. Why don't you talk to them about why they took this step"). Then, if the domain tutor has appropriate feedback, the meta-tutor appends the cognitive tutor message to the meta-feedback message. The tutoring model sends a message to highlight the problem step on the peer tutor's screen and to present the feedback to the peer tutor. Hint requests from the peer tutor work in a similar manner, combining the cognitive tutor hint on the step with a prompt to collaborate. If a skill-adjustment bug rule fires, the correction tutor sends a message telling peer tutors that they have over-raised or over-lowered student skills, and then prevents them from continuing to do so on that problem (e.g., "Slow down! Before increasing more, wait until your partner has shown this skill on another problem."). The correction tutor is domain-independent, and thus could be effective in combination with any intelligent tutor, as long as a translator exists to translate the intelligent tutor messages into an appropriate message format.

5.3.2 Integration of Components

In general, components communicate using the *CTRL* message protocol, and the way components interact in a given session is defined in the control module. All peer tutee solver actions, peer tutor correction actions, peer tutor skill ranking actions, and student chat actions are logged as tool messages by the control module. All cognitive and meta tutor feedback and hints are logged as tutor messages. See the left side of Figure 12 for a diagrammatic representation of the message-passing logic in the adaptive support condition (all interactions occur via the mediator). In this configuration, when the peer tutee takes an action, the echoing translator sends the action to the peer tutor's screen. In addition, the cognitive tutor evaluates the action, and sends the evaluation to the meta-tutor. All these interactions occur via the mediator. When the peer tutor takes an action, it is sent to the echo translator, which echoes the action onto the peer tutee's screen, and to the correction tutor, which compares the peer tutor evaluation to the cognitive tutor evaluation. If a bug rule fires, the correction tutor sends feedback to the peer tutor. The peer tutor can also request a hint from the correction tutor, which has stored the cognitive tutor hint for that step.

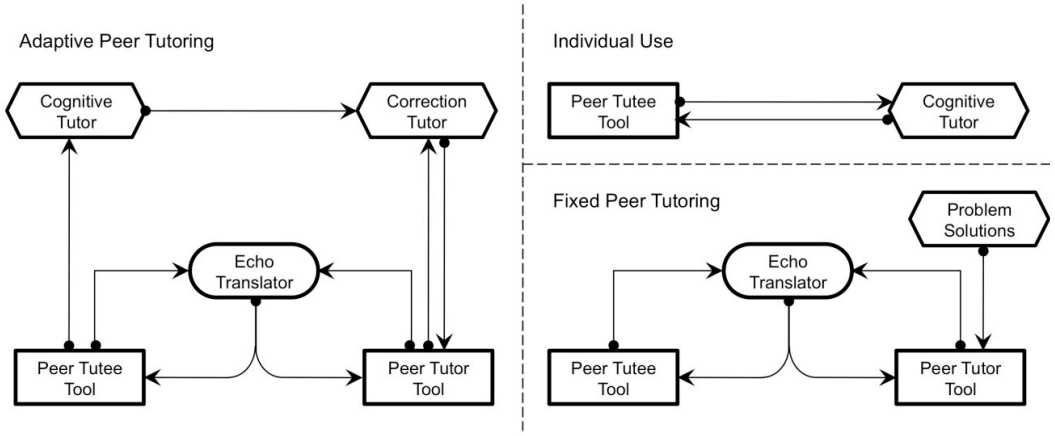

Figure 12. Message passing logic in the mediator for the three scenarios involved in *Phase 2*: *Adaptive Correction Support*.

As the logic of which components are involved in the session and how they communicate exists in the control module, it is simple to use the module to implement the relevant conditions. The components involved were defined in the same manner as in the *CTRL* framework, where all the components involved in a session and their component types were enumerated. However, instead of the message passing logic being defined in a rule-based manner, it was initially defined in the form of several *message groups*, which specify that messages of a particular origin should be sent to a particular target, with a given priority (represented pictorially in Figure 12). Message groups can be considered a template for automatically authoring simple rules. Upon receiving a message from a component, the mediator would match the component to all message groups that have that component as an origin, and then send the message to the targets in each relevant message group. In the case of messages sent to non-tool components, the control module then waits for a response from all the components that have received messages, before sending the messages out in the order of the specified priority.

5.3.3 Comparison Conditions

Within *CTRL*, we created two comparison conditions for our system to evaluate both the effects of the adaptive support and the effects of the collaborative activity. First, we were able to remove the correction tutor from the collaborative condition to create a peer tutoring scenario without adaptive support (see the right side of Figure 12). Second, we were able to remove the second tool and the correction tutor to create an individual use scenario with the cognitive tutor that mirrored typical use of the cognitive tutor. Implementing these conditions was as simple as creating new session types in the mediator that changed the message passing logic between components, and did not necessitate any major code changes. In the following subsection, we discuss a study in which we compared these conditions to the adaptive condition.

5.4 Evaluation: Study 2

5.4.1 Experimental Design

The goal of this study was to investigate the effects of collaboration and adaptive support on student interaction and learning by comparing three conditions: 1. Students used the CTA individually (*individual condition*), 2. Students tutored each other using the peer tutoring script (*fixed collaboration condition*) described in *5.2.1*, and 3. Students tutored each other with adaptive domain support described in *5.2.3* in addition to the peer tutoring script (*adaptive collaboration condition;* see Table 16). Both the adaptive and fixed collaboration conditions included peer tutoring; both the adaptive collaboration condition and individual learning condition included adaptive domain support. We expected the adaptive collaboration condition to learn more than the fixed collaboration condition because the adaptive domain support would be presented when students need it, and therefore as they try to apply it to their situation they are more likely to engage in reflective processes. We expected the collaborative conditions to learn more than the individual condition because of the deep interaction involved in the activity, and the elaborative processes that natural language discussion triggers.

Table 16. Conditions for *Phase 2* study. I varied whether students collaborated and whether they received adaptive domain support. I hypothesized that the adaptive peer tutoring condition would be best for learning (represented by the *).

		Adaptive	
		yes	no
Collaborative	yes	adaptive peer tutoring*	fixed peer tutoring
	no	individual learning	n/a

5.4.2 Method

Participants. Participants were 62 high-school students (34 male, 28 female) from five second-year algebra classes at a vocational high school in the United States, taught by the same teacher. There were 10 students in 10th grade, 41 students in 11th grade, and 11 students in 12th grade. Students spent half the day at this high school taking math and various technical subjects (e.g., nursing, electronics). The other half of the day was spent at their "home school" learning conventional subjects. The high school used the individual version of the CTA as part of regular classroom practice. The literal equation solving unit was a review unit for the students, and one that they had already covered in their first algebra class. Based on the assessment of the classroom teacher, the concepts in the unit were difficult for the students to understand, and review was necessary. Students in the collaborative conditions were put in pairs by the classroom teacher, who was told to pair students of similar abilities who would work well together. Because students benefit from being tutors in addition to tutees, and even low-ability students benefit from being placed in the tutor role (see Robinson, Schofield & Steers-Wentzell, 2003, for review), it was important to pair students who felt like they could tutor their partner. Pairing students of similar abilities ensured that students could plausibly function as both tutors and tutees.

Students from each class were randomly assigned to one of the three conditions. 11 students were excluded from the analysis because either they were absent during a collaborative part of the intervention, or their partner was absent and they could not be re-paired with another student. Another 12 participants did not take the delayed posttest, but were included for all other analyses. The total number of participants included in the analysis was thus 51 for the pretest and posttest (17 students in the adaptive peer tutoring condition, 14 students in the fixed peer tutoring condition, and 20 students in the individual use condition), and 39 students for the delayed posttest (11 in the adaptive peer tutoring condition, 10 in the fixed peer tutoring condition, and 18 in the individual use condition). There were an odd number of students in the adaptive condition because students were retained in the analysis who had an absent partner during an intervention day but were placed with a new partner in the same condition.

Procedure. The study took place over the course of five weeks (see Table 17). Students were given a 15 minute pretest on Monday or Tuesday of the first week, depending on their class schedules. The

intervention then took place on two days where students would typically be using the CTA, over two 70 minute class periods. The first intervention day was on Thursday or Friday of the first week, the second was on Thursday or Friday of the following week. On both intervention days, students in the peer tutoring conditions spent half the period in the preparation phase and spent the remaining classroom time tutoring each other in the collaboration phase. In any given pair, one student tutored on one intervention day and the second student tutored on the second intervention day. Students in the adaptive collaborative condition received adaptive cognitive support while tutoring. Students in the individual use condition used the CTA throughout the preparation and collaboration phases. The week after the intervention, students were given a 15 minute posttest. Two weeks later, students were given a 15 minute delayed posttest to assess their long-term retention. On non-intervention days, students continued with their typical algebra curriculum, which involved different units than the literal equation solving unit.

Table 17. Study procedure in *Phase 2*. The study took place on 5 days over the course of 5 weeks.

Week	Day	Time (minutes)	Individual Learning	Fixed Support	Adaptive Support
1	1	15	Domain Pretest	Domain Pretest	Domain Pretest
1	2	5	Instruction	Instruction	Instruction
1	2	30	Individual use	Preparation	Preparation
1	2	35	Individual use	Collaboration + Fixed Support	Collaboration + Adaptive Support
2	3	35	Individual use	Preparation	Preparation
2	3	35	Individual use	Collaboration + Fixed Support	Collaboration + Adaptive Support
3	4	15	Domain Posttest	Domain Posttest	Domain Posttest
5	5	15	Delayed Domain Posttest	Delayed Domain Posttest	Delayed Domain Posttest

Measures. To assess students' individual learning I used counterbalanced pre-, post-, and delayed posttests, each containing 8 questions. The tests were developed by the experimenter and approved by the classroom teacher. The first two questions were scaffolding questions, in that students were either given a problem solution and asked to label each step or given a sequence of step labels and asked to provide the problem solution. The next three questions were parallel to the questions solved during instruction. The final three questions were transfer questions, and asked students to apply their skills in a different context. The questions across the different test versions were parallel but used different numbers and symbols. The tests were administered on paper. I scored answers on the three tests by marking whether the solutions were correct or incorrect. If students got a completely correct solution or reached a nearly correct solution but made a copying error, they received a 1. If students performed one or more conceptual steps incorrectly

they received a 0. Points on all the questions were summed. I computed normalized gain scores (Hake, 1998) between the pre- and post-tests and pre- and delayed tests by using the formula *gain* = (*post* - *pre*) / (1 - *pre*). If posttest scores were lower than pretest scores, we used the formula (*post* - *pre*) / *pre*.

In order to analyze student collaborative process, we logged all tutee actions, peer tutor actions, and intelligent tutor responses. I computed the number of problems solved by students in each phase on each study day, the amount of time it took to solve each problem, and whether each problem was successfully completed, unsuccessfully completed (only possible in the fixed condition), or interrupted by the end of the classroom period. Then, for each problem, I computed the number of correct and incorrect problem-solving steps students took. For each step, I computed the number of hint requests to the cognitive tutor that students made, and the amount of feedback from the cognitive tutor that students received. I also calculated the number of tutoring-related interface actions: such as the number of times peer tutors marked a step right or wrong (and whether they were correct in their assessment), and the number of times they consulted the problem answers. All these metrics are data typically available in intelligent tutoring systems.

Next, I adapted an approach widely used in collaborative learning research and classified all tutee and tutor chat actions. In general, I segmented the dialog by chat messages, creating a new segment every time students hit enter. However, consecutive lines of chat where the student was uninterrupted by another interface action were classified as the same segment (e.g., a student typed "do you need" and then immediately typed "help", with no other action being logged between the two chat actions). The experimenter and a second trained rater then independently coded the chat dialogs on two dimensions: help-seeking behavior (Cohen's kappa = 0.80) and help-giving behavior (Cohen's kappa = 0.86). The coders trained on 20% of the data and agreement was computed on the remaining 80%. Disagreements on all data were resolved through discussion. The different dimensions are described below.

Our first step was to categorize tutee help-seeking behavior. While in the individual learning condition students could click a hint button to request help, in the collaborative condition students had to make verbal requests to the peer tutor. For the coding, we adapted the coding scheme by Webb, Troper, and Fall (1995), who coded help requests as any statement that was a request for help or indicated confusion. The data did include direct *requests*, where it was clear that the tutee was expecting an immediate response, often because a question was posed or help was demanded (see Table 18 for examples of all codes). However, tutees also made several *problem-related* statements, where the tutee was not demanding a response from the tutor, but where an on-topic response would be appropriate, such as self-explanations or statements of confusion. All other tutee statements were divided into *activity-related* and *off-topic* categories, depending on whether or not they related to the collaborative activity. Next, I defined *help given*, expanding on Webb's definition of elaborated and unelaborated help (Webb, Troper, & Fall, 1995). Webb divided help received into several degrees of elaboration, ranging from a fully labeled verbal explanation to simply delivering the answer. While these levels mapped to our data, I chose to simply label these forms of help as unelaborated or elaborated, because from a preliminary inspection students either tended to give straightforward instructions or more complex tutoring advice. I also coded hints, where peer

tutors provided an explanation for the problem step but did not directly instruct the tutees on what to do, as elaborated help. The categorization of tutor utterances had five codes: elaborated help, unelaborated help, feedback, activity-related, and off-topic (see Table 18).

5.4.3 Results & Discussion

I began by evaluating our primary hypothesis that the adaptive support condition is better for student domain learning than the fixed support condition and individual learning condition. I then looked at the process data on each level discussed in the above section, moving toward finer and finer granularity. I analyzed the data by individual, so that a given student's actions can be linked to his or her own learning gains and his or her partner's learning gains. For example, the number of errors committed by a student while in the tutee role can be correlated with learning, but so can the number of errors viewed by a student while in the tutor role.

Learning Outcomes. I conducted a two-way (condition x test-time) repeated-measures ANOVA, with test-time (pretest, posttest, or delayed test) as the repeated measure. There was a significant effect for test-time ($F[2,72] = 41.303$, $p < 0.001$), but there were no significant differences between conditions ($F[2,36] = 0.881$, $p = 0.423$), and no interaction ($F[2,36] = 0.859$, $p = 0.432$). A priori contrasts revealed that the effect was due to the difference between the pretest and the other two tests ($t[36] = 69.541$, $p < 0.001$) and not due to the difference between the posttest and the delayed test ($t[36] = 2.544$, $p = 0.119$). Thus, the different

Table 18. Coding scheme for tutor and tutee dialogue. We used two codes that related to both students, two additional tutee-specific codes, and three additional tutor-specific codes.

Role	Category	Description	Examples
Tutee	Request	Statement relating to the problem that requires a response from the tutor	"how do I get b by itself", "help"
Tutee	Problem-related statement	Tutee statements containing problem-related content	"so I get w on one side", "I'm lost"
Tutor	Elaborated help	Explanation of a step, hint on how to complete a step, describing an error	"now get m by itself"
Tutor	Unelaborated help	Direct instruction on how to complete all or part of the next step	"factor out t", "then divide"
Tutor	Feedback	Indication of whether a step was right or wrong	"good", "no"
Both	Activity-related statement	Coordination and activity-related statements	"what are you doing?"
Both	Off-topic	Statements not related to the problem or activity	"He's dating her"

conditions did not have different effects on delayed or immediate learning, and overall students did not show differences between the delayed and immediate measures. For the correlational analyses in this paper described in the following sections, we use the student gain scores between the pretest and posttest and pretest and delayed test, computed as described in *5.4.2*. Table 19 contains the absolute scores of the students who took all three tests. It is interesting to note that pretest scores were near floor, despite students' prior familiarity with the unit.

Table 19. Absolute scores on pretest, posttest, and delayed test. Each test had a maximum score of 8.

	Pretest		**Posttest**		**Delayed Posttest**	
Condition	M	SD	M	SD	M	SD
Individual	1.28	1.60	3.00	1.75	3.67	1.78
Fixed	0.90	0.88	3.50	2.17	3.60	2.17
Adaptive	0.82	1.08	2.36	1.57	2.82	1.78

Problem-solving progress. Our next level of analysis involved the number of problems completed per hour by each condition during the intervention. Because students learned equal amounts across the three conditions, one might expect that the problem-solving rate of each condition would be similar. However, students working collaboratively tend to solve problems slower than students working individually. I further expected the fixed condition to solve fewer problems successfully than the adaptive condition, since it had less relevant domain support. Problems solved may have an impact on the immediate posttest, but is less likely to relate to long-term retention, which is a sign of deeper learning.

I conducted a one-way (condition: individual, fixed, adaptive) ANOVA on the number of problems successfully completed per hour in the collaboration phase of the study (which, for individual learners, was simply the second half of the period). For this particular analysis, we grouped the students in the collaborative conditions by dyad, as the number of problems that one pair member completes is dependent on the number of problems the other pair member completes. Condition was indeed significantly related to problems solved ($F[2,34] = 8.76$, $p = 0.001$), where the adaptive collaboration condition ($M = 17.7$, $SD = 6.69$) and fixed collaboration condition ($M = 13.3$, $SD = 7.71$) solved fewer problems per hour than the individual conditions ($M = 47.0$, $SD = 30.2$). However, there were no differences between the fixed and adaptive conditions. In order to determine if problems completed were related to learning, I correlated total problems *successfully* completed per hour by each student as a tutee with their posttest and delayed test gain scores. Indeed, across all conditions, problems successfully completed per hour were marginally correlated with student learning on the posttest ($r[49] = 0.233$, $p = 0.100$), but not on the delayed test ($r[37] = 0.020$, $p = 0.906$).

Looking more closely at the collaborative conditions, differences in the design of the two conditions also led to differences in the number of problems *unsuccessfully* completed. In the fixed

condition, students were able to move to the next problem when they thought they were done, regardless of whether they were actually done. Tutees claimed that they were done, and tutors agreed, a mean of 2.50 times (SD = 1.61). The mean percentage of times that this exchange occurred out of the number of total problems seen (M = 8.00%, SD = 2.63%) was negatively correlated with the immediate learning gains of the tutee (r[12] = -0.597, p = -0.024) and the delayed learning gains of the tutee (r[8] = -0.714, p = 0.020). It was also negatively correlated with the delayed learning gains of the tutor (r[7] = -0.686, p = 0.040), but not the immediate learning gains of the tutor (r[10] = -0.214, p = 0.504). In the adaptive collaboration condition, the counterpart of incorrectly moving to the next problem would be the tutee attempting to move to the next problem, the tutor agreeing, and then both being blocked from doing so by the system. Students acting as tutees faced this situation a mean of 2.18 times (SD = 2.56). The percentage of times tutees witnessed this exchange out of total problems seen (M = 5.00%, SD = 2.09%) was negatively correlated with learning gains on the delayed posttest (r[9] = -0.667, p = 0.025), but not on the immediate posttest (r[15] = -0.007, p = 0.980). Surprisingly, being the tutor during this exchange was positively correlated with learning gains on the delayed posttest (r[9] = 0.652, p = .030), but not on the immediate posttest (r[15] = 0.280, p = 0.275). It would seem that being faced with these impasses in the adaptive condition led peer tutors to reflect more on how to overcome them and move to the next problem, an opportunity that they did not have in the fixed condition.

In summary, I found that progress as tutee was correlated with learning on the posttest but not on the delayed posttest. Further, moving on without solving the previous problem was negatively related to learning on the delayed test. On the other hand, witnessing one's tutee getting blocked from moving on, which was only possible in the adaptive condition, was correlated with peer tutor's learning gains. While struggling with the problem may have been detrimental to tutees, viewing this process may have been beneficial for tutors. It may be critical for tutor learning that tutees reach these problem-solving impasses. In the following section, I explore the relationships between student progress, impasses, and learning.

One explanation for the difference in problems solved was that students struggled with the problems more in the collaborative conditions than in the individual condition, because they did not have the same level of support from the intelligent system. Further, it seems that students in the fixed support condition might commit more errors than students in the adaptive support condition, again due to a lack of sufficient domain assistance. To investigate this hypothesis, we looked at the average number of errors (or incorrect attempts) students made per problem during the collaboration phase. We conducted a one-way (condition: individual, fixed, adaptive) ANOVA, with pretest as a covariate. Pretest was significantly predictive of errors per problem (F[1,47] = 5.41, p = 0.025), but there were no significant effects of condition (F[2,47] = 1.738, p = 0.187). The number of errors made by students in the fixed and adaptive peer tutoring conditions were not significantly different from errors made by students working alone (see Table 20, Row 1). We then looked at how the errors made related to learning. As each error was a learning opportunity, we focused on the total error counts, rather than the per problem average. Total errors made were not related to gains on the immediate posttest or delayed test. Viewing errors as a tutor was also not

correlated with learning gains on the immediate posttest. However, viewing errors was positively correlated with delayed learning gains (although non-significantly). It appeared that viewing errors was related to learning from tutoring, just as observing your tutee unable to proceed to the next problem was related to learning from tutoring. These two correlations put together suggest that peer tutors are indeed benefiting from the reflective aspects of tutoring: viewing impasses and considering what might be necessary to overcome them.

Table 20. Frequencies of student progress variables and correlations with learning.

#	Type	Frequencies / problem			Learning gains from being tutored		Learning gains from tutoring	
		Individual	Fixed	Adaptive	Posttest	Delayed Test	Posttest	Delayed Test
1	Errors	M = 1.46 SD = 1.26	M = 1.81 SD = 1.04	M = 2.46 SD = 1.87	$r(49) = -0.113$ $p = 0.432$	$r(37) = -0.063$ $p = 0.702$	$r(27) = -0.058$ $p = 0.763$	$r(18) = 0.354$ $p = 0.126$
2	Help Requests	M = 0.65 SD = 0.81	M = 0.66 SD = 0.52	M = 1.18 SD = 0.52	$r(49) = -0.208$ $p = 0.144$	$r(37) = -0.133$ $p = 0.418$	$r(27) = -0.175$ $p = 0.364$	$r(18) = 0.429$ $p = 0.059$
3	Yes-No Feedback	N/A	M = 0.38 SD = 0.27	M = 0.25 SD = 0.40	$r(27) = 0.153$ $p = 0.427$	$r(18) = 0.051$ $p = 0.832$	$r(29) = 0.353$ $p = 0.052$	$r(19) = 0.454$ $p = 0.038$

Given this relationship between viewing errors and reflective processes, one question is whether peer tutors in the adaptive condition were made more aware of errors because of the adaptive domain support. It is difficult to determine whether tutor awareness of errors did increase in the adaptive condition compared to the fixed condition, as tutors did not often verbally indicate that they knew a particular step was an error, and did not approve or flag every step made by the tutee. However, I can indirectly infer that tutors were more aware of tutee errors in the adaptive condition and that the errors did in fact constitute learning opportunities, because tutor gains from the pretest to the delayed test were correlated with tutee errors per problem in the adaptive condition ($r = 0.523$, $p < 0.10$), but not in the fixed condition ($r = 0.272$, $p = 0.479$). Table 22 gives an example of the peer tutor being made aware of a tutee error. The dyad was asked to solve the equation "$3q - xq = x$" for q. In this example, the tutor first marked the step correct, but then received feedback from the intelligent system that it was in fact incorrect. The peer tutor, after being alerted to the error, determined how to repair the error and take the next correct step. Although the outcome of his reasoning was communicated to the tutee, the process itself was not made transparent, potentially explaining why the delayed gain of the tutor was 0.375, while the tutee showed a delayed gain of 0.125. In general, tutors appeared to benefit even from simply viewing more errors, while tutees did not benefit from committing them.

Table 21. Learning opportunity created by tutee feedback. Students are solving the problem *3q-xq=x* for *q*.

Step Description	Analysis
Tutee selects "factor q", but types "3q = x".	The tutee knows what to do, but is not sure how to complete the step.
Peer tutor approves the calculation, and receives error feedback from the cognitive tutor.	The peer tutor initially thinks the step is correct, but is made aware from the system that it is an error, creating a learning opportunity.
The peer tutor tells the tutee "undo that step", but the tutee proceeds by dividing by 3. The tutee clicks the done button, but the peer tutor disagrees.	The peer tutor understands that the tutee has not solved the problem.
The students have the following exchange: Peer tutor: undo it Tutee: why? U marked it right? Peer tutor: the step is right but it said you made a typing error when you factored The dialog continues until the tutee confirms which step to undo.	The peer tutor identifies the error for the tutee in an unelaborated way.
The tutee undoes the step, and the tutor explicitly tells the tutee what to do, after asking for a hint: Now factor out q. It should be q(3 – x) + x. q(3 – x) = x, sorry	The peer tutor then tells the tutee how to complete the step, correcting his own error.

Helping Behaviors. The next level of analysis involved the interaction between the tutee, the peer tutor, and the tutoring system. First, we looked at tutee help-seeking behaviors. Active help-seekers may have been better learners because they were more likely to receive help when it was most appropriate. Additionally, as errors made were related to learning from tutoring, it is possible that tutee help-seeking actions were also related to learning from tutoring. We only used hint requests from the individual condition which occurred during the time period of the collaboration phase (see Table 20, Row 2). I conducted a one-way (condition: individual, fixed, adaptive) ANOVA on hints requested per problem, and found that the number of hints requested in the individual condition was not significantly different from the number of hints requested in chat in each collaborative condition ($F[2,50] = 1.68$, $p = 0.198$). I then determined if we could link making and receiving hint requests to learning gains. Making hint requests was not correlated with immediate or delayed learning gains. Receiving hint requests as a tutor, while not correlated with immediate posttest gains, was marginally correlated with delayed posttest gains ($r[18] = 0.429$, $p = 0.059$). Perhaps the help requests prompted the same reflective processes in the peer tutor as viewing impasses.

In the individual condition, the kind of help given to tutees did not vary, but in the collaborative conditions, the peer tutor chose what kind of help to give, and when to give it. First, we examined the role that verbal yes-no feedback played in the student interaction (Table 21, Row 3). We conducted a one-way (condition: individual, fixed, adaptive) ANOVA on yes-no feedback per problem, and found no significant differences between conditions ($F[1,29] = 0.925$, $p = 0.334$). Feedback given by the tutor was marginally correlated with learning gains as a tutor on the immediate posttest, and significantly correlated with learning gains as a tutor on the delayed test. Feedback received by the tutee was not related to tutee learning gains. Here, because the peer tutor was simply providing yes or no responses, it is not likely that it was the content of the responses that related to learning benefits, but rather the reflective processes that led them to produce the responses. In general, responses with better content were not directly related to learning gains. For example, giving elaborated help was not predicitive of gains on the delayed test for tutors ($r[19] = 0.191$, $p = 0.407$), nor was receiving elaborated help for tutees ($r[18] = 0.108$, $p = 0.649$).

Although giving and receiving elaborated help was not found to be important in isolation, it may be that the quality of the help interacted with the tutee's need for help in order to produce learning gains. In the individual condition, students always received help or feedback after an error or help request, but in the collaborative conditions, that may not be the case. Here, we examine the two extreme examples of the quality and timing of help. First, giving elaborated help after a help request is likely an extremely productive behavior: The tutee needs the help, and using the elaborated help given, the tutee should be able to overcome his or her impasse and complete the next problem step. Table 22 displays the percent of requests that tutees made that were answered by the peer tutor with elaborated help, out of total requests made. While this value was not significantly different between conditions ($F[1,29] = 0.136$, $p = 0.715$), answering requests with elaborated help was significantly correlated with learning as a tutor, unlike overall instances of elaborated help. However, as the tutee, having requests answered was not correlated with learning gains. On the other hand, tutees are unlikely to need help immediately after making a correct step, and in particular, they do not need an unelaborated instruction on how to complete the next step. This help is unlikely to be beneficial, and may in fact hinder tutees by preventing them from reflecting on the next step. While percent unelaborated help when not needed was not significantly different between conditions ($F[1,27] = 0.011$, $p = 0.918$), it was significantly negatively correlated with tutee delayed learning. To summarize: Giving good help when needed was positively related to tutor learning, while receiving poor help when not needed was negatively related to tutee learning. Interestingly, it is unclear which features of help had a positive effect on tutee learning, potentially because of the rareness of good help.

Table 22. Percent good help given when needed and bad help given when not needed.

#	Type	Frequencies / problem			Learning gains from being tutored		Learning gains from tutoring	
		Individual	Fixed	Adaptive	Posttest	Delayed Test	Posttest	Delayed Test
1	Good help when needed	N/A	M = 31.2 SD = 21.2	M = 23.6 SD = 26.7	$r(27) = 0.301$ $p = 0.113$	$r(18) = 0.267$ $p = 0.255$	$r(27) = 0.398$ $p = 0.033$	$r(18) = 0.476$ $p = 0.034$
2	Poor help when not needed	N/A	M = 12.6 SD = 15.7	M = 13.1 SD = 11.6	$r(27) = -0.331$ p = 0.079	$r(18) = -0.521$ p = 0.019	$r(27) = -0.309$ p = 0.103	$r(18) = -0.055$ p = 0.817

In addition to giving elaborated help, giving *conceptual* elaborated help may be important for peer tutor and tutee learning. Fuchs and colleagues (1997) suggested that tutor provision of conceptual help is critical for seeing benefits from peer tutoring. In APTA, tutors that give conceptual help would likely benefit because they are engaging in knowledge-building processes, and their tutees would likely benefit because they are receiving good help. Table 23 is an example of a conceptual exchange observed between students. The tutor in Table 23 had a gain score of 0.625 on the delayed test. Although this exchange is the type of interaction we were hoping to see, this kind of conceptual help was rare among students, with a mean of 1.67 (*SD* = 1.61) total instances per tutor in the fixed condition, and 0.80 (*SD* = 1.01) in the adaptive condition. Students introduced concepts into their elaborated help 35% of the time (*SD* = 40%). This percentage was marginally correlated with tutee learning, $r(19) = 0.426$, $p = 0.06$, suggested that I should indeed be encouraging students to give more conceptual help in addition to more elaborated help.

Table 23. Conceptual interaction about problem $ay + by + m = n$. Students are solving for y.

Step Description	Analysis
The tutee factors *y*. The tutor checks the problem answers (which say to subtract *m* from both sides). The tutor marks the problem step wrong, and the tutee undoes the step.	Tutor (incorrectly) flags the tutee because her solution doesn't match the problem-solving action
The students have the following dialogue: Tutor: ok um what variable is by itsself Tutor: that is the one you need to get on the other side Tutee: right now just "n" but i have to get "y" by itself Tutor: look at the equation ay+by+m ...wat 1 is bby itself Tutee: m	Tutor conceptually explains the first step as she sees it.
The tutee adds *m*. The tutor gives a hint: Tutor: look at the sign b4 n	Tutee makes a conceptual error, and the Tutor immediately moves to correct it.
The tutee combines like terms. The tutor checks the problem answers and flags the step. The tutee undoes both steps.	Tutor uses fixed resource to verify her thinking, then marks step wrong.
Tutor: look at the sign b4 the m is it a plus or a minus Tutee: it a plus so i would wnt to minus it from the rest of the problem Tutor: yup	Tutor continues giving the conceptual hint. The tutee self-explains her reasoning.

Use of Support. I conducted a more exploratory comparison of the effects of domain support on learning, using only the adaptive condition. The adaptive condition had both fixed and adaptive feedback available, and thus we could conduct a finer-grained examination of the uses of both types of support. Out of 17 tutors in the adaptive condition, 12 received hints and error feedback from the computer. The other 5 did not ask for hints or mark the problem steps of their tutees, focusing instead on chat communication. Out of an average of 3.50 instances of CTA help (*SD* = 3.15), tutors communicated the help to the tutee a mean of 63% of the time (*SD* = 42%). In general, percent feedback communicated was positively correlated with

tutee learning gains on the delayed posttest ($r[10] = 0.852$, $p = 0.015$) for the 12 students that used the adaptive feedback. Below, Table 24 contains an example drawn from a different pair than Table 24, where the peer tutor receives information that they are not actually done, and then successfully communicates hint feedback to the tutee. In this pair, the tutee had a gain score of 0.375 on the delayed posttest. Here, the tutee benefitted from committing an error and engaging in a dialog with the tutor.

Table 24. Example of peer-mediated feedback. Students are solving for *t* in the equation: $t = (-bh+mn)/(-v)=r$. They need to simplify the equation by getting rid of the negative sign in front of the *v* in the denominator.

Step Description	Analysis
Tutee selects the done button. Peer tutor incorrectly agrees, and receives feedback from the system.	Both students are surprised to hear that they are not done.
The tutee says, "do u kno wat i should do". The tutor looks at the problem solution.	The tutee asks for a hint, and the tutor consults the worked example to help her.
Students have the following dialog: Tutor: look at the neg sign on the denominator Tutee: but wat do i do to get rid of the negative? Tutor: the neg has to disappear u ll find it in trans Tutee: will u please just tell me already? Tutor: i don't remember what it's called The dialog continues until the tutor realizes that he does not actually know the specific next step.	The tutor begins to give elaborated help, but lacks the knowledge to fully identify and explain the step. The tutor is unsuccessful at helping the tutee.
The peer tutor asks for a hint from the cognitive tutor. She communicates the help, saying "use common factor". The tutee simplifies fractions and then promptly undoes it. The tutor says, "-1" , and the tutee factors -1. Finally, the tutor says, "now simplify." The tutee simplifies and completes the problem.	The peer tutor uses a hint to provide a series of procedural instructions to the tutee. The tutee successfully completes the problem.

When feedback from the intelligent system was not communicated to tutees, it appeared to lead to damaging confusion on the part of the tutee. The following displays a situation where the tutee did not receive the feedback given by the tutor, to unfortunate results. After marking a step right, the peer tutor received a feedback message telling him that the step was actually wrong and giving him a hint on the step. At this point, the peer tutee said: "that doesn't look right, im sorry I suck at math lol", and then "k, nevermind." The peer tutor did not respond. Then the peer tutee clicked done, the peer tutor agreed, and the peer tutor was given another feedback message saying the problem is not done. This message was not communicated to the tutee either. Given such lack of communication, not only were tutees not getting the assistance needed, but they were getting misleading feedback. To the tutee, it appeared as if the steps were correct, even if they were not. In this pair, the tutee had a gain score of 0.000.

I also investigated the effects of fixed feedback use (looking only at students who had access to both adaptive and fixed feedback). 13 students used the fixed feedback provided. These students viewed the problem solution a mean of 8.38 times ($SD = 7.96$), over twice the amount of time students received adaptive assistance. 45.8% of the fixed assistance accessed by the tutor was communicated to the tutee (*SD*

= 32.2%), but the percent fixed assistance communicated from the tutor was not correlated with learning from being the tutee. However, communicating fixed assistance was correlated with *tutor* learning gains on the delayed posttest ($r[11] = 0.683$, $p = 0.062$), suggesting that when students actively processed the problem answers they benefitted (or that the good students were more likely to actively use them).

Integrating results across data sources. As a final step, we conducted two regression analyses to better compare the abilities of the variables discussed to predict student delayed learning. We focus here on delayed learning because it indicates long-term retention, and thus is likely a better indicator of deep learning than the immediate posttest. The inferential statistics on the regression results should be taken as suggestive, not conclusive, both because of the small sample (Tabachnick and Fidell, 1996) and because of the correlational nature of the data. Nevertheless, the results can be used to generate causal hypotheses that can then be tested with further experimentation.

First, we constructed a model to predict domain learning using the three variables common to *tutees* in all conditions: problems completed per hour, errors made, and help requested. We further included whether the learning was individual or collaborative as a dummy coded variable (condition; individual = 0, collaborative = 1), and added the interaction terms between condition and all the other variables, as the individual condition differed from the collaborative conditions in several ways. As a whole, the model explained roughly 30% of the variance in delayed gain ($R^2 = .299$, $F[7,38] = 1.891$, $p = 0.105$). Four variables significantly predicted delayed gain: errors made ($\beta = 0.620$, $t[38] = 2.281$, $p = 0.030$), hints requested ($\beta = -0.465$, $t(38) = -2.104$, $p = 0.044$), condition by errors made ($\beta = -1.134$, $t[38] = -3.191$, $p = 0.003$), and condition by hints requested ($\beta = 0.730$, $t[38] = 2.466$, $p = 0.019$). While these results are correlational, it appears that errors made were positively related to delayed gain, but hints requested were negatively related to delayed gain. Interestingly, from the direction of the interaction coefficients it appeared that it is better to try steps in the individual condition than in the collaborative conditions, but better to ask for a hint in the collaborative conditions than in the individual conditions.

Next, we conducted a second regression analysis to predict delayed learning in the two collaborative conditions. We included all the variables that were somewhat correlated with delayed learning and were found in both conditions: errors viewed, help requests received, feedback given, elaborated help given when needed, and unelaborated help received when not needed. We also included errors made and help requests made, as those were significantly predictive of learning in the first regression. The model accounted for a significant proportion of the variance in the delayed gain ($R^2 = 0.783$, $F[7,19] = 6.190$, $p = 0.003$), although due to the small sample size it is likely that this value is inflated (Tabachnick & Fidell, 1996). Table 25 contains the beta values, t statistics, and p-values for each variable. Elaborated help given when needed was the only variable that was not significantly predictive of delayed gains as a tutor. The variable that was most significantly predictive of delayed gains as a tutor was the yes-no feedback given. Again, these results are correlational, but two interesting elements stand out from this analysis. First, given the positive relationship between learning gains and errors viewed, requests

received, and feedback given, students appeared to benefit more from the reflective aspects of tutoring than the articulation of their thoughts. Second, based on the negative relationships between errors made, unelaborated help when not needed, and learning gains, in general tutees may have not received the support they needed to overcome the problem-solving impasses they encountered. In the following section, we will discuss the implications of these results with respect to which aspects of peer tutoring might most benefit from the introduction of adaptive support.

Table 25. Regression results used to predict student delayed learning in collaborative conditions.

Variable	*b*	*t*(19)	*p*
Errors made	-0.354	-2.225	0.046
Errors viewed	0.365	2.336	0.038
Requests made	0.353	2.245	0.044
Requests received	0.332	2.091	0.059
Feedback given	0.647	4.403	0.001
Elaborated help given when needed	0.262	1.676	0.120
Unelaborated help received after correct step	-0.375	-2.203	0.048

Discussion. Although learning gains between the individual and collaborative conditions were parallel, students in the two different types of conditions took different paths to learning. It took students in the collaborative conditions far fewer problems to achieve the same learning gains than students in the individual condition (although an equivalent amount of time). This result is in line with other collaborative results that suggest that learning in collaborative conditions is more efficient than learning individually (e.g., Diziol et al., 2008). In domains where problem-authoring is difficult, collaborative conditions may require fewer problems to be designed in order to facilitate student learning. On the other hand, it is possible that had we controlled for the number of problems solved and not for time, students in the individual conditions would have learned as much as students in the collaborative conditions in a shorter amount of time. Further, other than the stark differences between the problems completed in each condition, the individual and collaborative conditions were remarkably similar on the surface. Students made parallel numbers of errors and asked for help at the same rate.

Against the initial prediction, the adaptive and fixed collaboration conditions led to similar domain learning gains. However, each collaborative condition had particular design elements that had unique effects on student interaction. For example, preventing students from moving to the next problem until the previous problem was complete in the adaptive condition may have had an indirectly beneficial effect on

tutor learning by leading them to reflect on their misconceptions at these critical moments. Allowing students to move to the next problem without finishing the previous problem appeared to be a design flaw in the fixed condition, as it did not give students the opportunity to reach these beneficial impasses. Even though peer tutors appeared to benefit from adaptive feedback given by the cognitive tutor, not forcing them to communicate it to their tutees may also have been a design flaw, as this event was negatively correlated with tutee learning. Surprisingly, communicating fixed feedback was not related to benefits for the peer tutee, but was related to benefits for the peer tutor. It may be important to give tutors access to materials that they can use to construct conceptual elaborated explanations, and future designs should encourage this behavior.

The problem-solving and collaborative dialog data collected in each condition gave us insight into how students benefitted from being tutors and tutees across both collaborative conditions. Viewing errors, fielding help requests, and giving feedback were all correlated with tutor delayed learning, suggesting the tutors benefitted from the reflective processes triggered by tutee problem-solving actions. The evidence supporting the theory that tutors benefitted from constructing help was more mixed. Although learning was related to communicating fixed support and giving good help when needed, tutors did not benefit from giving good help in general or communicating adaptive assistance received from the CTA. It is possible that increased domain learning led to these good tutoring behaviors, rather than the other way around. Roscoe and Chi (2007) hypothesized that while tutors benefit from knowledge-building, they do not benefit from communicating knowledge that they already know, and it is possible that when looking at student elaborated help, we cannot distinguish knowledge-building from knowledge-telling. Additionally, the benefits of being a tutor may have been offset by the disadvantages of being tutored by a peer. It is potentially problematic that the same elements that led tutors in the collaborative conditions to learn (viewing errors and fielding help requests) are elements that signify a lack of tutee knowledge. Further, we did not find many relationships between collaborative process and tutee learning, although we did find some evidence that receiving help when needed related to tutee learning. It is possible that tutors did not exhibit a sufficient number of positive tutoring behaviors to have a noticeable beneficial effect on tutee learning. It is striking that students in the peer tutor role benefitted from the same interactions that related to less learning for students in the tutee role. It may have been the design of the peer tutoring script itself that lead to the lack of differences between the individual and collaborative conditions in the current experiment. These results do not correspond to other collaborative learning experiments that have demonstrated benefits for collaboration (Lou et al., 2001).

5.5 Outlook and Discussion

5.5.1 Introduction

In this chapter, I described the design (*5.2*) and implementation (*5.3*) of adaptive *correction* that supports a peer tutor in reflecting on the peer tutee's problem-solving steps and ultimately in giving more correct help. I compared this support to a fixed control where the peer tutor simply had problem solutions and an

ecological control where students simply used the *CTA* (*5.4*). While I had hypothesized that the adaptive support condition would be better than the fixed support condition and individual use condition at increasing domain learning, I found that students learned equally across all conditions. However, there were differences in the effects each condition had on the path students took to learning, informing future adaptive support. In this subsection, I talk about the design (*5.5.2*), technology (*5.5.3*), and learning sciences (*5.5.4*) contributions of this chapter, and then discuss future directions for adaptive support (*5.5.5*).

5.5.2 Design

This phase makes contributions to all three of the design research questions: *Q1-D1, Q1-D2,* and *Q2-D1* (see Table 1). I constructed a cognitive tutor-style model representing the basic actions inherent in peer tutoring: correcting problem steps, assessing skills, and giving help (*Q1-D1;* "How do ITS approaches to modeling apply to ACLS?"). The design of the model had some fairly inflexible assumptions: peer tutors had to respond in specified ways to tutee actions and respond to each tutee action. However, those assumptions were relaxed in the implementation of the model. Instead, only a small subset of peer tutor actions were responded to, and peer tutors did not have to engage in the actions that would lead to feedback. Under this model, peer tutors had more freedom to interact without influence from the intelligent tutor than they would have in an individual cognitive tutoring scenario. I also departed slightly from intelligent tutoring paradigms in designing support based on the model (*Q1-D2*; "How do ITS approaches to support apply to ACLS?"). *APTA* relied heavily on peer-mediated feedback, where domain feedback intended for the tutee was presented to the peer tutor. By communicating the feedback to the tutee, peer tutors might reflect and elaborate on it, benefiting from the mediation and providing the tutee with tailored help. This concept was cautiously successful here, and will be followed up on in later chapters (*Development 2*, *Phase 3*, *Phase 4*). Finally, this chapter demonstrated in the design of the model of peer tutoring interaction that relevant collaboration support can be given to interacting students that relies primarily on a domain model (*Q2-D1*; "What role does domain information play in collaboration models and feedback?"). Domain information serves as input to the cognitive tutor models, to let peer tutors know which steps are correct and which ones are incorrect. It is also incorporated in feedback, to give peer tutors specific information that they can pass on to their tutees. The success of this approach is discussed in *5.5.4*.

5.5.3 Technology

The main contribution of this chapter is not a technical one, but the chapter does further research on *Q2-T1* ("How can existing and custom components be integrated?"). Using *CTRL*, introduced in *Chapter 4*, I integrated the *CTA* domain model with a custom-built correction model. The correction model was a simple, production-based implementation that covered a subset of the designed model presented in *5.2*, and triggered feedback to the peer tutor based on four bug rules. Despite the simplicity of the model, the support was capable of helping peer tutors in giving correct help to tutees at important impasses. Essentially, by leveraging the existing *CTA* components and *CTRL* infrastructure, it became possible to

give students support for their interaction without a lot of additional implementation. Additionally, *CTRL* allowed the implementation of the two control conditions – individual use of the cognitive tutor and fixed support for peer tutoring – with a minimum of code changes. This phase was the first use of the *CTRL* architecture, with promising results.

5.5.4 Learning Sciences

While I had hypothesized that the adaptive support condition would be better than the fixed support condition and individual use conditions at increasing domain learning (*Q1-L2*; "What are the effects of ACLS on student learning?"), I found that students learned equally across all conditions. However, using both collaborative dialog and problem-solving data I was able to see differences in the effects each condition had on the path students took to learning (*Q1-L1*; "What are the effects of ACLS on student interaction?"). Further, using intelligent tutor logging components, I could take a deeper look at which aspects of student interaction related to learning outcomes (*Q2-L1*; "How can intelligent tutoring-style data logs augment the analysis of collaborative study data?"). While the results do not provide evidence that adaptive support is more effective than fixed support, they do contribute to the peer tutoring literature by suggesting that peer tutors do indeed benefit from reflective processes. Actions as simple as observing tutee errors and giving yes-no feedback were correlated with peer tutor learning. The results were also interesting in that they suggested that in our script, the quality of student dialog was not high enough for either student to benefit from elaborative processes. Additionally, the rich data set collected by integrating problem-solving logs (including intelligent tutor responses) and chat logs allowed the analysis of the peer tutoring data in a way that has not been done before, by linking tutor and tutee problem-solving actions to their discussion. The success of this approach is evidence for the utility of incorporating problem-solving information into collaborative *data analysis* (*Q2-L1*).

5.5.5 Implications for Iteration

The results suggest several ways to improve learning from the script. Future designs for adaptive support suggest that it may improve peer tutor benefits from the script to improve the likelihood that they reflect on tutee errors. By encouraging tutees to make more errors, or encouraging peer tutors to notice more errors, it may be possible to amplify this effect. Perhaps more importantly, it appears that peer tutors need more support engaging in elaborative processes and giving good help to their tutees than they are receiving. First, peer tutors should be supported in constructing conceptual, elaborated help, by automatically detecting when their help is unelaborated (e.g., "subtract *x"*), and then by providing assistance at this critical moment. It may be necessary to both remind tutors that they should be giving better help and provide them with sufficient scaffolding to ensure they are capable of doing so. Second, peer tutors should be supported in providing relevant help at moments when tutees have reached impasses. It may be necessary to automatically detect these points where tutees need help, determine whether tutors have provided relevant help, and if not, scaffold them in constructing help that targets tutee

misconceptions. In this manner, peer tutors may be able to surpass the helping abilities of intelligent tutors; they will be giving tutees help when they need it, but the help might be more tailored to the tutee's level of understanding. However, without sufficient support, peer tutors might continue to fall short in providing support to tutees.

Additionally, the results suggest that perhaps the way support is presented should be examined. In this phase, I presented peer tutors with *peer-mediated support*: Domain support that was ultimately intended for tutees, with the hypothesis that peer tutors would benefit from communicating the support, and even that tutees would benefit from receiving support from the peer tutor. This support design is unlike typical designs in ACLS, which involve simply presenting direct feedback to an inefficient collaborator. In the following section, I explore that and other support designs that may have a more beneficial effect on student collaboration than simple direct and explicit support. Then, in Chapter 7, I discuss my implementation of adaptive support for student help-giving.

6 Development 2: Student Needs & Design Space for Adaptive Support

6.1 Introduction

The results of *Phase 2: Adaptive Domain Support* (Chapter 5) suggested that a logical iteration of *APTA* would be to adaptively support peer tutors in giving good help. However, the review in *2.3.1* indicated that previous researchers have not explored in depth how collaborating students perceive and respond to different types of adaptive support. Without a good understanding of how adaptive support can fit into the social context of two collaborating students, it is possible that support developed will not have the desired effect. Thus, before implementing interaction support for the peer tutor, I conducted a preliminary design exploration to determine what students in a peer tutoring context need in terms of support, and how they perceive different forms of support. I generated several different ideas for adaptive support, and then used a design method called Speed Dating (Davidoff, Lee, Dey, & Zimmerman, 2007) to gather preliminary impressions of how students reacted to the ideas. These reactions gave information about the quality of a particular idea, but, more importantly, exposed three principles that guide the design of ACLS more generally. This chapter addresses the research question *Q1-D2* at length (see Table 1), as it explores adaptive support paradigms beyond intelligent tutoring feedback that may be more appropriate for collaborating students. A subset of this work was discussed in Walker, Rummel, and Koedinger (2009c).

6.2 Ideation

Drawing inspiration from a variety of existing forms of support for individual and collaborative learning, I generated several ideas for adaptively supporting reciprocal peer tutoring that went beyond the traditional individual learning model of presenting explicit feedback to the collaborator who is in need of support. In this section, I describe four of these ideas: Reflective prompts, peer-mediated feedback, adaptive opportunities, and adaptive resources. I discuss the origin of each idea, and why they may have a positive effect on student help-giving. These ideas then served as a basis for soliciting student reactions to assistance.

These four ideas can be mapped onto a two-dimensional design space for support: Whether the action that students should take is explicitly described in the feedback or implicitly arises as a result of the support (*explicit* or *implicit* instruction), and whether it is presented directly to the person it targets or presented indirectly to another party or through a change in the learning environment (*direct* or *indirect* presentation; see Figure 13). As described in *2.3.1* most existing ACLS systems provide direct explicit support, and are located in the lower right quadrant of Figure 13. By ensuring that these ideas cover this whole design space, multiple different designs for support are considered.

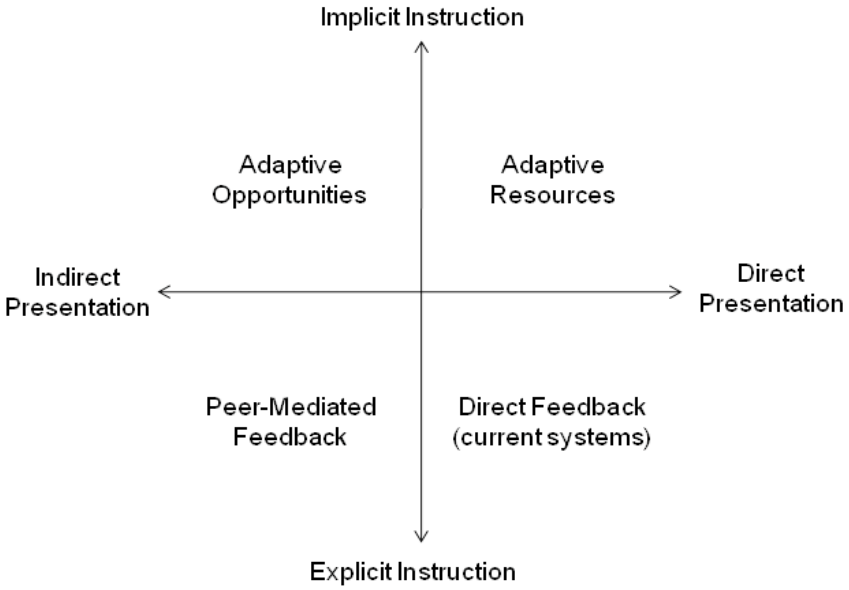

Figure 13. Design space for adaptive collaborative learning support, varying the explictness of instruction and directness of presentation. Most current systems are in the direct feedback quadrant.

6.2.1 Reflective Prompts

One idea for delivering adaptive support to collaborators is to mimic the kind of support that human facilitators provide in face-to-face groups. In accountable talk as described by Resnick, O'Connor, and Michaels (2007), a teacher directs a classroom using several different reflective "moves", ranging from asking a given student to apply their reasoning to someone else's utterance ("Do you agree or disagree and why?") to asking a student to expand on his or her own utterance ("Why do you think that?"). Instead of presenting a single student with very explicit feedback, it may be beneficial to present all students involved in the interaction with questions that prompt further reflection and reasoning. Such open-ended questions might be much less of a threat to students' feeling of self-efficacy, control, and psychological safety than more explicit criticisms. Further, presenting the prompts to all collaborators means that both may benefit from the reflection triggered by the prompts, and that the person addressed by the prompt may feel more accountable to incorporate it. While other adaptive systems have presented feedback publicly to both users (e.g., Constantino-Gonzalez, Suthers, & de los Santos, 2003), it is rare for ACLS to pose these kinds of open-ended reflective prompts.

This idea could be applied to adaptive help-giving assistance by prompting students to reflect on the help exchanged in a collaborative interaction. For example, if peer tutors give impoverished instrumental help like "subtract *x*", the system could prompt the peer tutor for more explication (e.g., "Why do you say that?"), or prompt the tutee to reflect on his or her partner's help (e.g., "Do you understand why your partner said that?"). If both students see the help, then they both may be prompted to reflect on the

concepts underlying the help. This technique could also be used to deliver positive feedback to students. This assistance lies in the middle of the direct/indirect design dimension, as it is presented to both students. It is also in the middle of the explicit/implicit dimension; while it does not explicitly tell students how to improve the collaboration, it does prompt students to take specific action.

6.2.2 Peer-Mediated Feedback

Some effective fixed collaborative learning scripts attempt to get individual students to elicit certain responses from their partners; for example, by having students ask their partners a series of questions at increasing levels of depth (King, Staffieri, & Adelgais, 1998). In our second idea, peer-mediated feedback, the system follows up on this concept by providing interaction guidance to a student other than the student whose behavior we would like to change. For example, if one student is not self-explaining their problem-solving steps, we can prompt their partner: "Did you understand what your partner did? If not, ask them why", rather than telling the first student, "Tell your partner why you took that step." For the students whose behavior we would like to change, receiving a prompt from their partner might feel more natural and be more comprehensible than receiving computer feedback. For students who receive the prompt, the approach encourages them to self-regulate their own learning by prompting them to request the help they need from their partner. This design idea is located along the indirect/explicit quadrant of our design space: The feedback is presented to another collaborating student instead of to the target of the support, and the next desired action is explicitly addressed. To our knowledge, the only instance of testing this concept has been in an adaptive collaboration scenario is in the version of *APTA* presented in *Phase 2*, where peer tutors were prompted to give tutees domain feedback. While this intervention was fairly effective at inducing peer tutors to paraphrase and communicate domain help, it is yet to be seen whether it will be effective for interaction assistance.

One area where this idea might be applied to adaptive interaction assistance is for instances where tutees receive help from the tutor, but it is likely that they do not understand the concepts involved with the help. Here, the system could deliver indirect explicit feedback to the tutee such as: "Wait -- do you understand why you should subtract x? If not, ask your partner why." This approach is in contrast to a direct and explicit feedback approach, where the prompt would generally be given to the peer tutor: "Why don't you tell your partner why they should subtract x." In this proposed mediated feedback, it is not so clear that blocking other tutee actions (e.g., problem-solving actions) as they receive this feedback is the best direction, as it takes away some tutee control over their environment. How to balance student control with partner confusion is still an open question. Nevertheless, it is possible that this mediated feedback will promote better self-regulation of learning and a deeper interaction.

6.2.3 Adaptive Opportunities

The third idea, adaptive opportunities, assesses whether the current state of the learning environment facilitates high-quality interactions and, if not, adapts the environment to create learning opportunities. For

example, many fixed collaboration scripts implement sentence starters or classifiers, where students engaging in a dialogue are scaffolded to indicate what their intentions are when composing utterances (e.g., Kollar, Fischer, & Slotta, 2005). In an adaptive system, the content of the starters could be changed to reflect the abilities and needs of the students, without providing any explicit feedback to students at all. Adjusting the conditions of the interaction rather than delivering instruction to change interaction behaviors may be perceived as less intrusive by students and may avoid threatening their control over the situation. While this design idea is a concept that has been explored in intelligent tutoring systems (e.g., in adaptive problem selection; VanLehn, 2006), it has not been a main feature of ACLS. However, there are a few adaptive systems that use this approach; For example, for groups with low participation and low motivation, *HabiPro* introduces two script elements: forced turn taking and rewards for the correct solution (Vizcaino et al., 2003). While these systems have not been evaluated for their effects on interaction, the approach may indeed be promising. This idea is situated in the implicit/indirect quadrant, because the intervention involves adaptively modifying the learning environment (in this case, changing the materials available to students prior to collaboration) to affect all students involved.

One area where this design idea could be applied to *APTA* is by creating the conditions where errors may occur by removing those two obstacles to committing errors. As in the individual version of the *CTA*, we might assess the skills that tutees have mastered, and adaptively select problems where tutees are likely to make errors that might lead both parties to benefit. Simultaneously, we could assess the peer tutor tendency to provide unsolicited help before a step has been attempted, and, if it is high, select problems for the tutee that the peer tutor has not yet mastered. Hopefully, if the peer tutor is struggling with the concepts in the problem, he or she will be less able to simply walk the tutee through the problem, and more joint knowledge construction will occur. This intervention is potentially advantageous because of its subtlety; students are unlikely to notice the manipulation, but it has the potential to increase the opportunity for tutees to make errors and therefore the potential for learning. Adaptively selecting problems to improve learning conditions is an example of an indirect presentation, as the support is not directly delivered to the student, and implicit instruction, as it does not make the next interaction steps clear to students.

6.2.4 Adaptive Resources

In adaptive resources, instead of explicitly telling students how to improve their behavior, they are provided with resources to help them to make the necessary changes. This approach is drawn from adaptive hypermedia, where the information that is available to students changes in accordance with their knowledge (Brusilovsky, 2001). In a fixed support approach developed by Fuchs et al. (1997), students are trained in delivering conceptual mathematical explanations, using an alternating program of video clips and classroom discussion. In an adaptive system, the video related to each concept could be presented when a student may be thinking of applying the concept (for example, while preparing explanations for a given problem), and additional materials surrounding the video could incorporate specific information about the current problem or collaborating students. This idea differs from adaptive resources in that it involves

presenting a collaborator with relevant resources (direct support) rather than restructuring the interface (indirect support). An advantage to this approach is that students have more freedom in how they use the information presented. A potential disadvantage is that they do not process as much relevant information. While this specific approach has rarely been used in ACLS systems, visualization systems have been developed that simply mirror back to students aspects of their collaborative performance (Soller, Martinez, Jermann, & Mühlenbrock, 2005). These systems are a first step towards developing adaptive resources; augmenting these systems to incorporate more information presented to students about reaching ideal performance might be a fruitful area of research. This design idea is located in the direct/implicit quadrant of the design space, the resources are made directly available to the collaborator that needs them, but the collaborator has to determine how to use them to improve their collaboration.

In the current version of our system, the only resources available to the peer tutor are worked-out problem solutions. By redesigning the resources available to the peer tutor and by making them adaptive, it may be possible to encourage deeper interaction amongst the students. One could explore two types of adaptivity in delivering resources to students: Changing the content of the resources based on the current problem state, and changing the content of the resources based on an assessment of student knowledge. There are several different types of resources that can be provided to peer tutors other than a worked out problem example, such as:

R1. Conceptual description of how to solve the problem rather than problem steps

R2. Example of a similar problem, but using numbers in place of constant terms

R3. An annotated worked-example with conceptual explanations for each step

Additionally, the content of the resources themselves could be adapted based on information about the current problem-state, skill mastery, or student interaction. For example, *R2* could also display the errors made by the tutee on the problem using numbers in place of letters, or *R3* could derive the conceptual explanations using language that students have used previously. Hopefully, providing the peer tutor with resources that incorporate conceptual information about correct and incorrect problem steps will encourage peer tutors to incorporate those elements into their interaction with tutees. Making those resources adaptive means that it would be possible to tailor the presentation of each resource to the particular problem situation and abilities of the tutee.

6.3 Speed Dating Process

Our next step was to use these assistance concepts as a basis for exploring user perceptions relating to adaptive support. We applied a design method called Speed Dating (Davidoff, Lee, Dey, & Zimmerman, 2007), which takes a sketch-based approach to give the designer insight into user needs. The aspect of Speed Dating we leveraged involves the use of focus groups to discuss several potential design scenarios in rapid succession. Because peer tutoring involves social interaction, a design method that asked users to give their reactions in a social context was a logical way to proceed.

Table 26. Ideas presented to students as part of Speed Dating. Students were given three scenarios, and then shown twelve support sketches associated with one of the scenarios. Some support sketches were related to multiple ideas for adaptive support.

Scenario	Description	Type
Help when not needed	Direct feedback to peer tutor	Traditional ACLS
Help when not needed	Feedback given to peer tutee	Peer-mediated feedback
Help when not needed	Feedback given in chat to both students	Reflective prompts
Help when not needed	Problems selected adaptively to adjust difficulty for collaborators	Adaptive opportunities
Unsure how to proceed	Adaptive hint	Traditional ACLS
Unsure how to proceed	Worked problem solutions	Adaptive resources
Unsure how to proceed	Conceptual description of solution	Adaptive resources
Unsure how to proceed	Simplified example relating to solution	Adaptive resources
Instrumental help	System explains tutee's error and asks tutor to transfer	Peer-mediated feedback, adaptive resources
Instrumental help	System explains problem to peer tutor, then gives reflective prompt in chat	Traditional ACLS, Reflective prompts
Instrumental help	System gives reflective prompt to peer tutor, and then gives hint instructions	Reflective prompt, peer-mediated feedback
Instrumental help	System scaffolds hint construction for peer tutor	Reflective prompt, peer-mediated feedback, adaptive opportunities

I sketched 12 scenarios for adaptively supporting a reciprocal peer tutoring activity, based both on the ideas described above and on traditional ACLS. The support sketches varied the collaborative situation that triggered the support, with four sketches designed to support peer tutors unsure how to give help, four sketches designed to prevent peer tutors from giving help when it was not needed, and four sketches designed to prevent peer tutors from giving simple instructions (see Table 26). Each scenario leveraged particular aspects of the ideas described in the previous section. Figure 14 shows a sample scenario that we presented to students representing peer-mediated feedback. In response to the peer tutor giving unasked-for help, the tutee is told to ask his partner to let him try the step before helping.

I then assembled four groups of volunteer high-school students with four high-school students in each group. One group was a set of high-achieving students and was interviewed in the lab, and the other three groups were from a lower-achieving class and were interviewed at a school. To gather information on

students' knowledge about collaboration, I first presented a multiple-choice questionnaire to each student describing the three collaboration situations that served as the foundation for the design sketches, and the four potential courses of action in response to each situation. I asked students to first individually select the action they would take in response to each situation, and then discuss their answers with the group. Next, I presented the 12 support sketches to each group of students, and asked for their reactions to each idea. As a result of this activity, I identified the level of students' knowledge of collaboration, their needs regarding adaptive support to peer tutoring, and their expectations of adaptive support.

6.4 Results

6.4.1. Knowledge of Good Help-Giving Behaviors

One aim of this design exercise was to qualitatively assess what students thought good help was, and how that matched up to existing research on good help. Many students from both the lab and the classroom groups indeed had a general understanding of help-related skills. Students knew that help should come at student impasses and errors ("You want to stop them before they do the entire problem wrong" – Lab Group), and that they should not be giving help if students are problem-solving effectively ("You don't

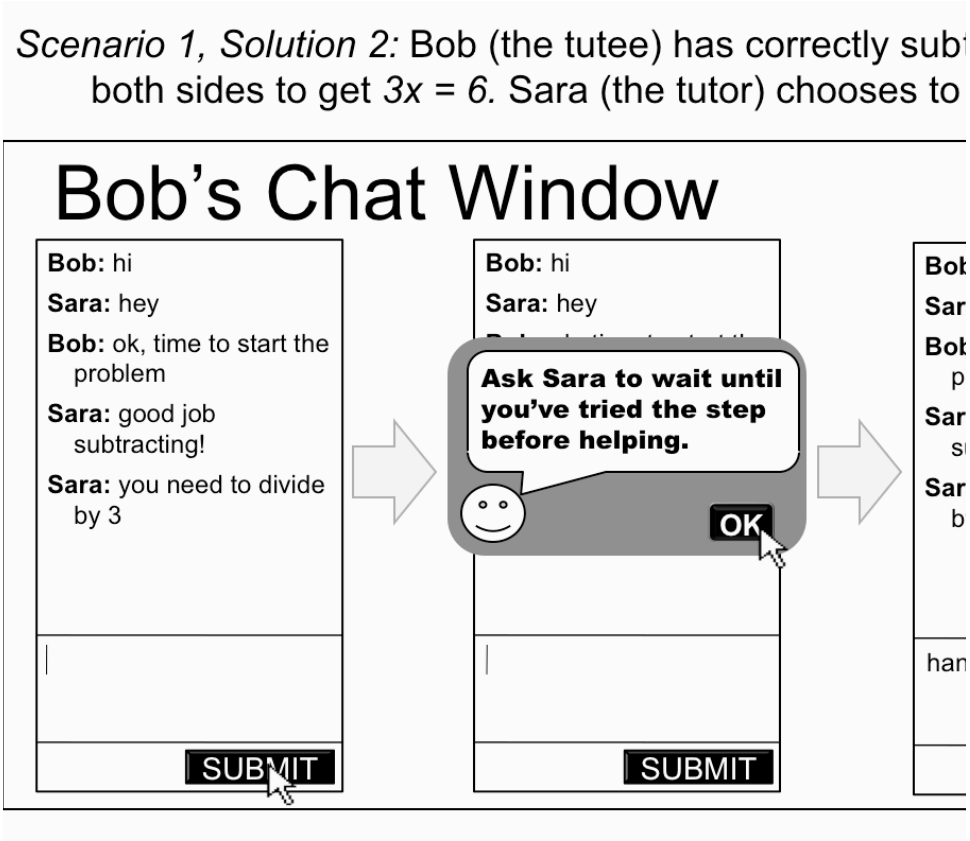

Figure 14. Speed Dating scenario. In this scenario, the tutee is encouraged to self-regulate their own learning by asking the peer tutor to refrain from helping until the tutee has tried the step. Students were presented with 12 scenarios in rapid succession and asked discussion questions.

want to do it for them, you want them to do the thinking" – Lab Group). Students also had a good sense of how the level of their tutees' understanding might relate to the help that they would give ("[You] want to see what part they're confused on" – Group 1). Finally students were aware of the characteristics of good help, describing an example of good help as "it explains it a lot better… it's not just giving you the answer" – Group 3, and referring to an example of bad help as "It's like she's telling him what to do and he's not learning how to do it himself "– Group 3. However, two problematic aspects did stand out from student discussions of good collaboration. First, while many of the students could identify aspects of good help, there were still several students who did not know (or could not express) what good help was. In Group 2, student descriptions of good help were somewhat misguided ("Then you can show him what you have to divide – that way he understands"), and one student, when talking about giving help, emphasized personal preference over everything else ("Whatever you're into, I guess"). Second, the differences between the descriptions of the lab group and the classroom groups suggest that some students who could recognize good help were unable to generate good help.

This analysis leads one to the conclusion that scaffolding students on how to generate good help might be just as necessary (and perhaps even more important) as giving them explicit descriptions of good help. If students are more likely to be able to recognize good help than to construct good help, giving them assistance on constructing good help for their partners is more important than describing good help for them. For example, showing students alternatives to just giving their partner the answer, such as providing examples of hints, may help novice peer tutors figure out how to be better tutors. Nevertheless, it was clear that knowledge of how to collaborate well was not the only aspect of collaboration that required adaptive support. I further explored motivational factors that may contribute to ineffective peer tutoring.

6.4.2 Accountability and Control Design Principles

Two motivational influences reappeared in student discussions: feelings of accountability for tutee learning, and a desire for tutoring efficacy. Students appeared to take their potential role as peer tutors very seriously, saying when considering a tutoring error: "Maybe he's going to be messed up – I wouldn't want that to happen" (Group 1). They wanted to feel like good tutors and be perceived as good tutors, responding very positively to a scenario where the computer offered public praise in the chat window: "I really like the one where the computer joins in on the IM… You gave that person good advice, both students see it" (Group 1). Interestingly, when taking the perspective of tutees, students had high expectations of their prospective tutors, describing: "If Sara's the tutor, shouldn't she know what she's doing, so she can help" (Group 2). Based on this analysis, students who do not feel like capable tutors may disengage with the activity or

Table 27. Three principles for designing adaptive support in a peer tutoring context.

#	Principle	Description
1	Accountability	Make peer tutors feel accountable to incorporate support.
2	Efficacy	Make peer tutors feel like good tutors, in control of the situation.
3	Relevance	Make peer tutors feel like support is relevant

simply give their partners the answer in order to increase their feelings of efficacy. These results confirm existing literature on peer tutoring, which suggests that the primary motivational influences on the activity are feelings of accountability for one's partner (Fantuzzo, Riggio, Connelly, & Dimeff, 1989) and feelings of being a capable tutor (Robinson, Schofield, & Steers-Wentzell, 2005).

There are two main implications of these motivational factors with respect to designing assistance provided to peer tutors (see Table 27). First, assistance could be designed to leverage the feelings of accountability already present in tutoring interactions in order to encourage peer tutors to give help in effective ways (*Accountability Design Principle*). For example, presenting interaction feedback and praise publicly in the chat window where both students can see it might encourage peer tutors to apply the advice. Second, it is necessary for assistance in general, and in particular for assistance designed to increase accountability, to avoid threatening peer tutors' beliefs that they are capable tutors, but instead to increase their sense of control over the situation (*Efficacy Design Principle*). Any assistance given by the computer should avoid undermining the peer tutor's control over the interaction, and for this reason, students overwhelmingly rejected the idea of peer-mediated feedback being given to the tutee, saying that this was "like your teacher talking over your shoulder" (Group 2). Students even pushed back against pop-up dialogs telling peer tutors what to do, saying "I wouldn't listen to that thing that said help at the wrong time – if it popped up, I would click no" (Group 2). Instead, students preferred assistance that put computers and peer tutors on more equal footing, such as reflective prompts delivered by computers in the chat window ("the computer's asking – I kind of like that… I think the computer should just go ahead and do it in the chat window" – Group 3). By positioning computers and peer tutors as collaborators (see Chan and Chou, 1997, for examples of this strategy in individual learning), we may be able to preserve tutoring efficacy, and increase peer tutor motivation to give good help.

6.4.3 Relevance Design Principle

When exploring student perceptions of different support designs, I also found that students particularly focused on how relevant the help appeared to be to their task, and how little it disrupted their interaction. On a broad level, it was clear that students wanted to get system feedback that they could use ("If it [the computer] says something we needed to know then it would be ok" – Group 2). By and large, students cited domain help on how to solve the problems as useful feedback, but surprisingly, what they wanted to receive was not simply a hint targeted at the next problem step. Students said that the adaptive hints were not always very informative ("the hint – doesn't really tell you much" – Group 2), and admitted that therefore they would be likely to take advantage of the hints ("You could just be clicking the hint button, to like, get the answers" – Group 3). Instead, students stated that they preferred hints that gave both the high-level concepts relevant to each problem step, and specific illustrations of the concepts. One student even suggested support that "give[s] you an example problem, but explains the steps to you and explains how they get the answer" (Group 3). Despite all this discussion about the usefulness of cognitive feedback, there was nearly no talk about the usefulness of interaction feedback, suggesting that students perceived

interaction feedback as less relevant than cognitive feedback. This finding is unsurprising: people generally have the illusion that they are capable of collaborating, as collaboration does not seem like something one would need support on.

This analysis leads one to recognize the importance of designing adaptive support so that it appears relevant to students (*Relevance Design Principle*; see line 3 of Table 27) in order to motivate them to incorporate the assistance into their own interactions. As students believe that cognitive support is relevant but do not recognize the relevance of interaction support, it might be that any interaction feedback given to students should be linked to cognitive feedback, to make the interaction feedback more concrete and immediately applicable. Telling students: "You should explain why to do a step in addition to what to do. For example, on the next step your tutee should be trying to isolate the y" might make the help seem more relevant than simply just telling them, "You should explain why to do a step in addition to what to do." Another technique for making the collaboration support more relevant to student interaction is by clearly linking the support to what peer tutors themselves want to do. By linking interaction support to the interaction-related intentions of the students, they might perceive the support as more relevant. For example, help on how to give an explanation would be perceived as maximally relevant when students are actively trying to give their partner an explanation.

6.5 Summary of Design Implications

By generating ideas for support that covered a two-dimensional design space (indirect/direct and implicit/explicit), and then using a needs-validation method called Speed Dating, I generated three design principles for supporting students in collaborating with each other: accountability, efficacy, and relevance. These principles are in line with research that suggests that accountability to one's partner and feelings of tutoring efficacy contribute to tutor success (Robinson, Schofield, & Steers-Wentzell, 2005), as well as more general research on adaptive assistance that suggests that assistance must be perceived as useful in order to be adopted (Dey, 2009). Further, the design exercise revealed particular implications of these principles for accepting or rejecting certain varieties of assistance. For example, students' opinion that feedback directed at the peer tutor should never be delivered solely to the tutee is an important insight into how manipulating the target of the support might affect feelings of efficacy, and argues for rejecting peer-mediated feedback delivered to the tutee. Similarly, I eliminated the varieties of adaptive opportunities from consideration where peer tutors could tell that something had changed, but could not tell why; if students could not see the relevance, they reacted negatively to the support. The insights gleaned from the Speed Dating activity formed the basis for our design of adaptive support for peer tutor help giving, which is described in the next section. As a result of this chapter progress was made on *Q1D2* ("How do ITS approaches to support apply to ACLS?"), where individual intelligent tutoring support paradigms may indeed be inadequate for collaboration.

7 Phase 3: Adaptive Help-Giving Support

7.1 Introduction

As a result of the work in *Development 2: Student Needs & Design Space for Adaptive Support* (Chapter 6), I now had design guidelines for constructing ACLS that students would be likely to incorporate into their interaction. The results of *Phase 2: Adaptive Correction Support* (Chapter 5) suggested that peer tutors needed support on how to give high-quality help so that they would be more likely to engage in elaborative processes that lead students to benefit from tutoring. To this end, I designed a simple model of good peer tutoring, focusing on the skills required to give high quality help, and using domain context to inform the model (exploring *Q2-D1:* "What role does domain information play in collaboration models and feedback?"; see Table 1). I then designed several different supports for the peer tutor, based on the principles identified in Chapter 6. I implemented the collaboration model by using multiple sources of information: the domain model found in the CTA, student interface actions, and machine learning classifications of student chat. The implementation assesses students on four collaborative skills and provides support when appropriate (adding to *Q1-T1*: "Can collaborative skills be knowledge traced?"). I then evaluated the adaptive support in a classroom setting, comparing it to a fixed support condition representing typical assistance in a peer tutoring activity (investigating *Q1-L1*: "What are the effects of ACLS on student collaborative interactions?"). I hypothesized that if the adaptive support appeared to students when they needed it and followed design principles that made students more likely to incorporate it, students would use the support to improve their interaction. If student interaction quality increases, their learning outcomes may as well. The work in this phase has been discussed elsewhere in Walker, Rummel, and Koedinger (2009d).

7.2 Design: Help-Giving Support

7.2.1 Interactions: Discussion Scaffolding and Practicality

After *Phase 2: Adaptive Correction Support* (Chapter 5), I kept the majority of the basic activities present in the peer tutoring script constant, but made seven modifications. The two most important modifications were modifications to the micro-script, to help scaffold the quality of student discussion: I added sentence classifiers to the interface, and implemented an adaptive problem order. Three of the changes to the script were macro-script changes that increased the practicality of the script for a classroom setting: I expanded the units covered by the system, modified the sequence of the phases, and also modified the way students were paired with each other. The other two changes were minor; I made a usability improvement to the way peer tutors interacted with problem steps and skill bars, and had the cognitive tutor (in addition to the peer tutor) adjust the skill assessments displayed in the skill bars. These changes are described in detail below. Figure 15 is a screenshot of the tutee's interface, and Figure 16 is a screenshot of the peer tutor's interface.

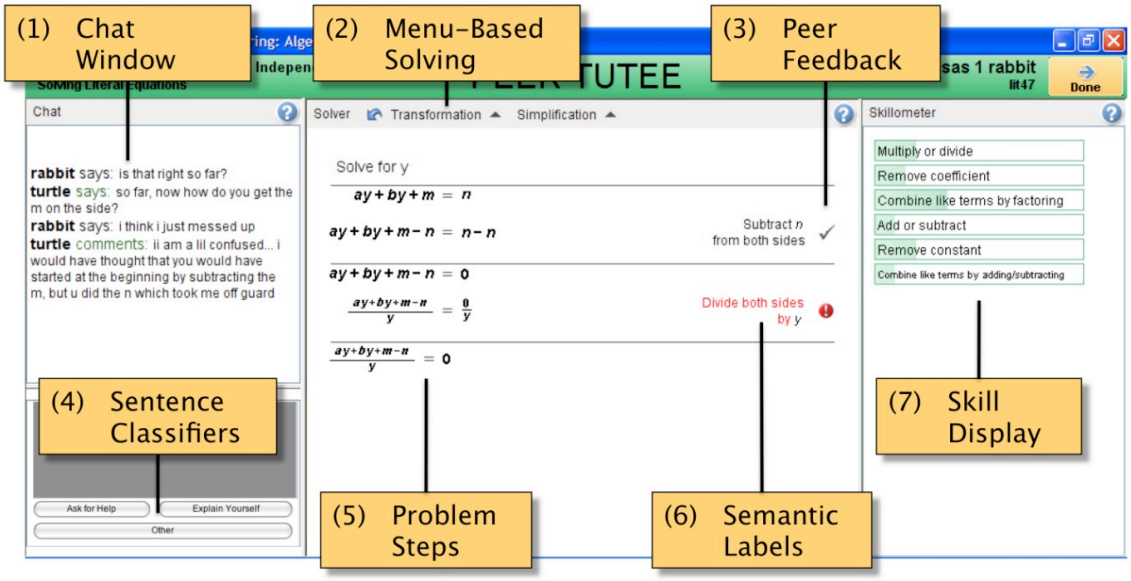

Figure 15. Tutee's problem-solving interface. The tutee solves problems using the menu, chats with their partner in the chat window, and receives feedback in the solver and skillometer.

Sentence Classifiers. To facilitate the discussion in the chat window, I 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, Ertl, Fischer, & Mandl, 2005). I asked peer tutors to label their utterances using one of four classifiers: "ask for explanation, explain mistake, give hint, and explain next step" (see #8 in Figure 16). Students had to select a classifier before they typed in an utterance, but they could also choose to click a neutral classifier ("comments"). 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 x". Tutees were also asked to self-classify each utterance as one of three categories: a "help request", "explanation", or "comment". Making those behaviors explicit in the interface encouraged students to put more consideration into what they said and why, facilitating them in engaging in generative elaborative processes.

Adaptive Problem Selection. Some problems were too easy for tutees in the previous study, and thus they experienced few opportunities for discussion during their tutoring session. To combat this problem, I used the *CTA* model to provide adaptive problem selection for tutees, where the next problem selected was based on the *CTA* knowledge tracing model of their problem-solving performance. While this modification meant that tutees did not necessarily follow the same sequence of problems as their tutors, it did mean that problems were more likely to be at the right difficulty.

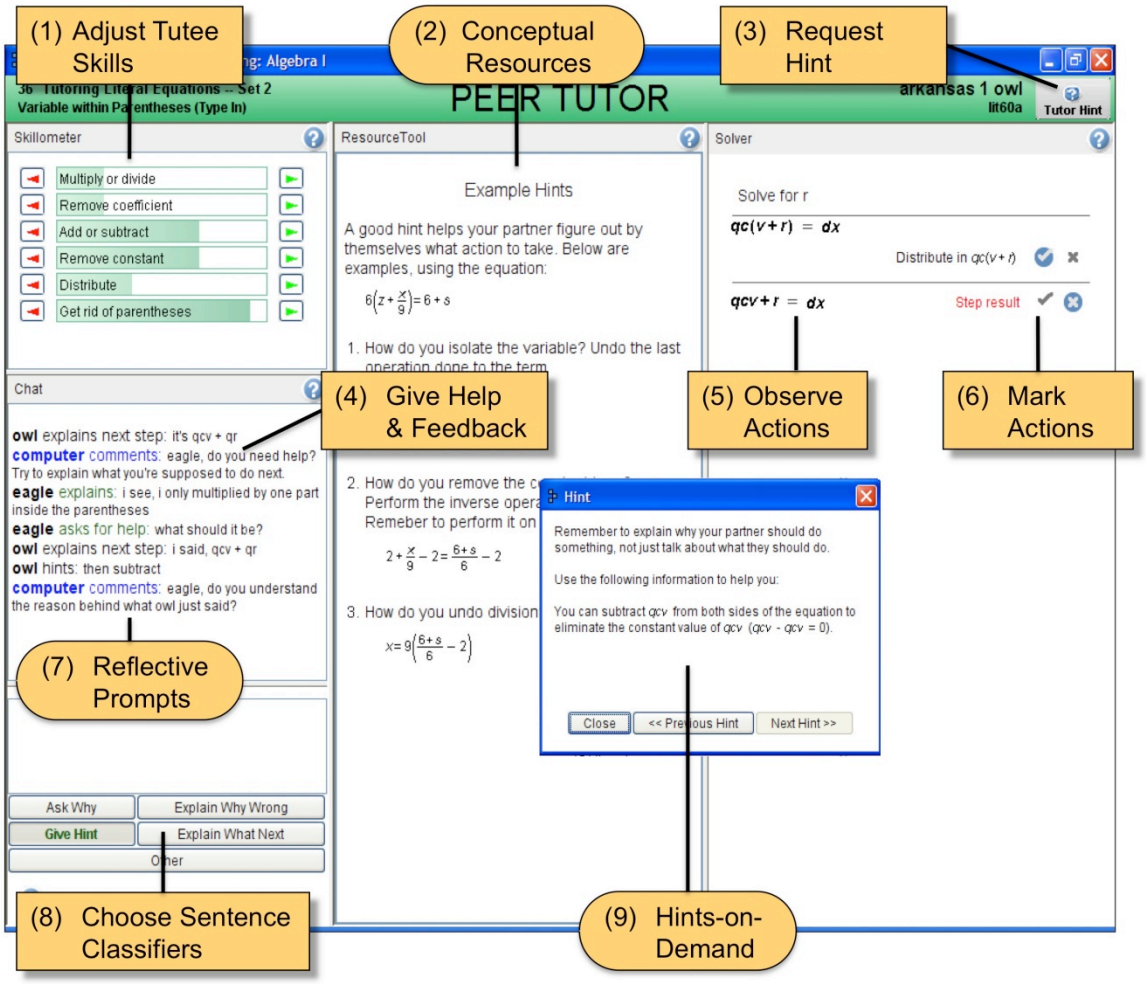

Figure 16. Peer tutor's interface in *Phase 3.* Square labels represent possible peer tutor actions in the interface. Round labels represent the support peer tutors received from the adaptive system. Students are solving for *r*.

Units Covered. In previous iterations of the system we had students solve literal equation solving problems that required them to master the procedures and concepts surrounding factoring. In this iteration, I added an additional unit that required students to master the knowledge associated with distributing (e.g., $a(x+y) = ax + ay$). During the preparation phase, one member of each pair solved factoring problems, and the other member of each pair solved distributing problems. Thus, tutors had different expertise then their tutees.

Phase Restructuring. In this iteration, students did the preparation phase in one class period, and the collaboration phases in subsequent class periods. I made this change for coordination reasons; in the study, periods were very short, and switching between phases would have been time-consuming.

Forming Pairs. Instead of having classroom teachers assign pairs that lasted throughout the course of the study, I paired students randomly and had them take on different partners for each new phase. In previous studies, certain pairs would get stuck in negative interaction patterns that lasted them the whole study. By giving students different pairs, it avoided having them get stuck with partners that they did not work well with. Additionally, data analysis is made easier if students switch pairs; it is clearer how to deal with cases of attrition if all partners do not stay together throughout the course of the study. To avoid pairing students with people that they do not get along with, teachers indicated who should *not* be paired with each other, and those pairs were not made.

Correction Interface. Our previous interface for marking steps and adjusting skills was modal, and this was very problematic for students. Students would have to click an "Approve" or "Flag" button, and then click on the relevant widget, but this process was difficult to understand without instruction. The interface was changed so there were approve and flag widgets for each problem step and skill bar, making it clearer how students were to use them (#6 in Figure 16).

Skill Bar Adjustment. Students in the adaptive condition previously had little guidance in how to increase or decrease student skill bars. The support was modified so that the peer tutors and computers could manipulate the skill bars collaboratively. The cognitive tutor would increase or decrease the relevant skill bar with each tutee problem step, and the peer tutor could then modify the value to suit the particular situation. This change was intended to lead peer tutors to make more informed decisions about how to manipulate the skill bars.

7.2.2 Model: Effective Help-Giving

Approach & Assumptions. The next step was to design a model to support peer tutors in help-giving, intended to lead to learning for both the giver and receiver. Here, the approach was to use theory on good peer tutoring, and combine it with data from previous studies. The majority of this current model replaces the high-level "discuss problem" box in the model presented in *Phase 3: Adaptive Cognitive Support* (Chapter 5). Like in the model presented in *Phase 3*, this model addresses each problem-solving step in isolation, without making connections between the help given across problem-solving steps. There are two design decisions implicit in the model: a focus on behaviors compared to processes, and a built-in flexibility to avoid overconstraining tutor interaction. First, in the model, the type and timing of student help are the main focus, and the model does not explicitly represent cognitive elaborative or reflective processes. As these processes are only visible through student behaviors, the model represents when and how students should display effective behaviors, with the hypothesis that engagement in beneficial processes will follow naturally. Second, despite the use of an intelligent tutoring approach, interaction is not limited to the paths represented in the model, as this overstructuring could have negative consequences (Dillenbourg, 2002). Student collaboration is more open-ended than traditional intelligent tutoring domains,

and thus a departure from ideal model paths is not necessarily suboptimal. If a peer tutor makes a decision inconsistent with the intelligent tutor estimate of the situation, it may not be appropriate for the intelligent tutor to intervene, as it is possible that students have a better understanding of the context. However, if deviations from the model accumulate, the intelligent tutoring system can pinpoint the ineffective student behaviors and act.

Effective Behavior. The model, depicted in Figure 17, begins when the tutee starts a new step in a given problem. For tutee behavior (the dark-shaded area of the diagram), I have adapted a model for good help-seeking developed by Aleven, McLaren, Roll, and Koedinger (2004). The model encourages tutees to solve problems on their own, but ensures that tutors provide scaffolding when appropriate. In our adaptation, tutees can perform two behaviors: trying a step or asking for a hint. Tutees should ask for a hint when they begin an unfamiliar step, after they make an error they do not know how to fix, or after they have received a hint they do not know how to use. They should try a step if it is familiar, if they understand the help given to them, or if they understand the error they just made on the step. I added two further elements to the model, based on tutee dialog moves found in the literature and in *Phase 2: Adaptive Correction Support*

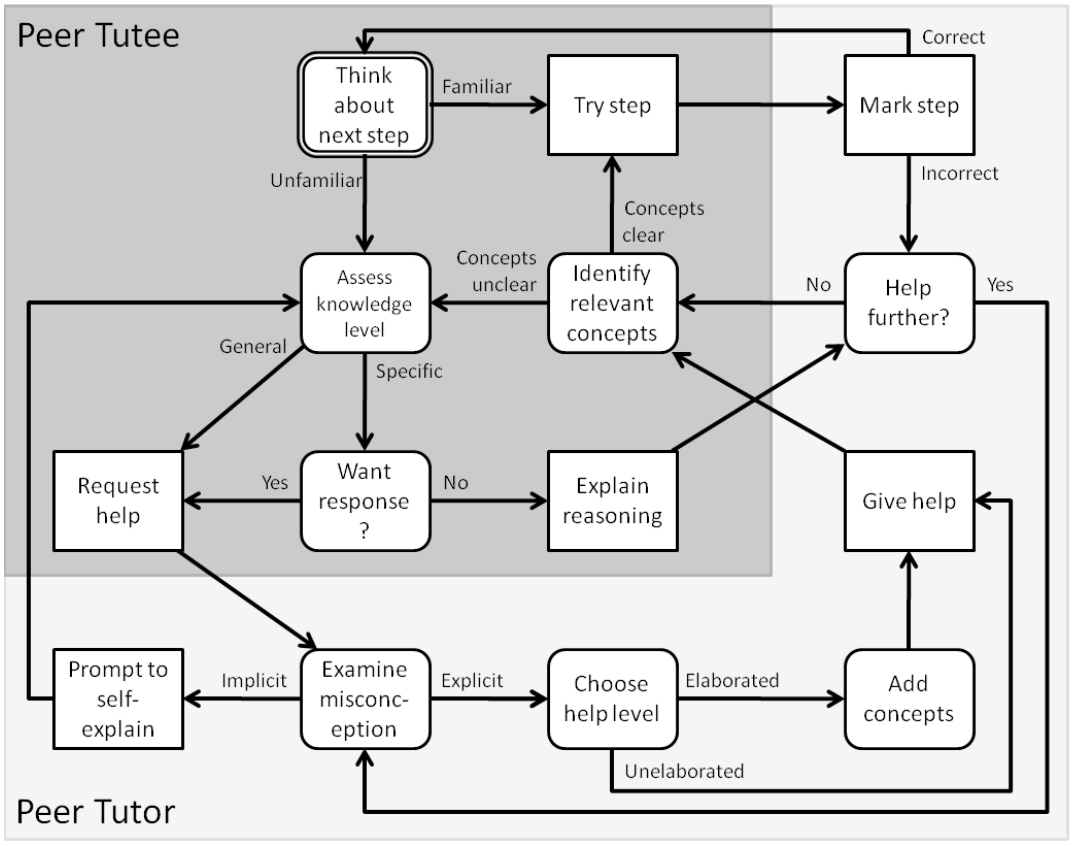

Figure 17. Model of tutor and tutee helping behavior, designed to contribute to learning of both parties. The dark area represents tutee behaviors, and the light area represents peer tutor behaviors.

(Chapter 5). First, tutees can choose to self-explain instead of requesting help. Self-explanations have been shown to be very beneficial for student learning (Chi, DeLeeuw, Chiu, & LaVancher, 1994), and may also allow tutors to reflect on their content and target explanations toward tutee misconceptions. Further, if students choose to request help instead of self-explain, requests that include specific references to the problem have been shown to be more useful than general requests (Webb & Mastergeorge, 2003). Therefore, if tutees have specific knowledge about the appropriate next step, our model suggests that they add specific content to their request.

The peer tutor side of the model (the light-shaded area) is based on a combination of findings (see Webb & Mastergeorge, 2003; VanLehn et al., 2003). Here, the peer tutor's behaviors include giving yes-no feedback, giving help, and prompting the tutee to self-explain. Yes-no feedback is beneficial for tutee learning in that it provides them with feedback on their problem-solving, and potentially beneficial for tutor learning in that tutors reflect on the nature of correct and incorrect problem-solving steps (triggering *reflective processes*). Peer tutors can then deliver help after an incorrect step, after a help request, or after a self-explanation, and take several cognitive steps in constructing the help. VanLehn and colleagues (2003) show that help tailored toward a tutee misconception is beneficial for the tutee. Therefore, when tutees ask for help, if they have recently committed an error, peer tutors should identify the tutee misconception. If they cannot, they should prompt the tutee to self-explain until the misconception becomes clear. The self-explanation benefits the tutee as well (Chi et al., 1994). After peer tutors have identified the misconception, they can begin constructing the help, deciding whether the help should be elaborated or unelaborated. Elaborated help, where the tutee elaborates on the content of the help, has been shown to be beneficial for both tutor and tutee learning, as the process of constructing the help leads tutors to reflect on their own knowledge and move to repair gaps (triggering *generative elaborative processes*). Tutees can then use the elaborated help to build on their knowledge. In fact, in *Phase 2: Adaptive Correction Support* (Section 5), I found that the right help delivered at the right time was indeed beneficial for student learning. Augmenting elaborated help with conceptual content further facilitates these processes (Fuchs et al., 1997). However, there may be some cases where unelaborated help is the most appropriate kind of help. If the tutee lacks the relevant knowledge to continue with the problem it may be better for the peer tutor to give the answer, treating the problem as a worked example.

Ineffective Behavior. I then used student data to identify three categories of suboptimal behaviors: departing from the model, not engaging in theoretically positive behaviors, and over-engaging in certain model-related behaviors. With respect to departures from the model, I had found that when peer tutors gave help after a correct step (a behavior not found in the model) it was negatively correlated with tutee learning. Thus, I was able to identify this as a sign of inefficient collaboration, and use it as a target for feedback. Another way students could display ineffective collaboration was by not engaging in model behaviors. For example, peer tutors in the previous study rarely prompted students to self-explain or gave error feedback. One goal of intelligent support should be to increase the frequency of these positive behaviors. Finally, peer

tutors in the previous study sometimes over-engaged in model behaviors; for example, giving far too much unelaborated help *compared to* elaborated help. Implicit in our model is a ratio between particular types of behaviors that should be approached. Thus, the buggy behaviors specified did encompass traditional buggy behaviors (e.g., paths not found in the model), but also spanned the amount of model behaviors engaged in and the ratios of particular model behaviors.

7.2.3 Support: Hints on Demand, Conceptual Resources, and Reflective Prompts

I augmented this preexisting correction assistance with three types of help-giving assistance, designed based on the principles identified in *Development 2: Student Needs and Perceptions of Support* (Chapter 6). The first type of assistance, *hints on demand*, is used for instances when the peer tutor (let's call her Sara) does not know how to help the tutee (let's call him Phil). There may be moments where Phil has asked for help, and Sara does not know what the next step to the problem is or how best to explain it. In this case, Sara would click on a hint button, found in the top right corner of the interface (see Figure 16*,* #3), and receive a multi-level hint on both how to solve the problem and how to help his or her partner (see Figure 16, #9). The hint opens with a collaborative component (e.g., "Remember to explain why your partner should do something, not just what they should do"), and then contains the domain component that the tutee would have originally received had they been using the *CTA* individually (e.g., "You can subtract *qcv* from both sides of the equation to eliminate the constant value of *qcv* [*qcv* – *qcv* = 0]."). If Sara still does not understand what to do and clicks next hint, both the collaborative and the domain component become more specific, until the domain component ultimately reveals the answer to Sara. The collaborative component uses several strategies to encourage students to give more conceptual help, and is adaptively chosen based on the current problem-solving context (e.g., it varies depending on whether the tutee has most recently taken a correct step or an incorrect step). Sara is intended to integrate the cognitive assistance for how her tutee, Phil, should proceed in the problem with the collaborative assistance on what kind of help she should give. In this case, Sara might use the information she received to tell Phil, "Eliminate the constant value of *qcv*", information that does not reveal the answer to the tutee, but includes relevant and correct domain content. This integration is intended to trigger *reflective* processes on the part of the peer tutor, while ensuring that the tutee receives correct help.

There may be cases where even after examining the adaptive hints, Sara is still unsure how to use the hints to give the tutee feedback (e.g., how to give help that refers to information Phil already knows). We designed the *adaptive resources* to further assist the peer tutor in constructing good help. When Sara clicks the "give hint" sentence classifier to prepare to compose a hint to her partner (#8 in Figure 16), she is presented with a resource (#2 in Figure 16), with content tailored to the current problem type, which provides examples of what a good hint would be within the context of this problem type. We had four separate sets of resources mapping to each type of sentence classifier (one for "ask why", one for "explain why not", one for "give hint", and one for "explain next step"). As the resource presents several sample hints for the whole problem, Sara has to actively process the resource in order to determine which kind of

hint might apply to the information she has to convey. The goal was for Sara to use the adaptive hints and resources together to construct more conceptual help, potentially triggering *generative elaborative* processes on the part of the peer tutor.

Once Sara has given help to her partner, she might receive a *reflective prompt* in a chat window that appears simultaneously to both students and targets peer tutor help-giving skills that need improvement. For example, if Sara is a novice tutor she may give a novice 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 its assessment of Sara's help-giving skill to say in the chat window (visible to both Sara and Phil), "Phil, do you understand the reason behind what Sara just said?" (Figure 16, #7). This utterance is designed to get both Phil and Sara reflecting on the domain concepts behind the next step, and to remind Sara 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. Prompts could be addressed to the peer tutor ("Sara, can you explain your partner's mistake?") or the tutee ("Phil, do you know what mistake you made?"), and were adaptively selected based on the computer assessment of help-giving skills. For example, *APTA* gave prompts based on student use of sentence classifiers, which were an integral component of the assessment of peer tutor utterances, and had potential benefit for the students. When students failed to use the sentence classifiers, they received prompts suggesting that they do so ("The buttons underneath the chat (e.g., "Give Hint") can help you let your partner know what you're doing"). 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"). The prompts contained both praise and hedges, such that the computer's voice never publicly threatened the peer tutor's voice. Only one reflective prompt was given at a time, and parameters were tuned so that students received an average of one prompt for every three peer tutor actions. There were several different prompts for any given situation, so students rarely received the same prompt twice. Again, these prompts were intended to lead peer tutors to engage in the model behaviors, leading them to experience more *reflective* and *generative elaborative* processes while providing better help to their tutees.

7.3 Implementation: Adding Adaptive Help-Giving Support

As in *Phase 2: Adaptive Correction Support* (Chapter 5), the adaptive support was implemented in Java as an instantiation of the *CTRL* framework described in *Development 1: Collaborative Tutoring Research Lab* (Chapter 4), with a mixture of custom-implemented components and components that were originally part of the *CTA*. A new element of the implementation was a tutor component to assess the peer tutor's help-giving quality and provide assistance. The implementation also included all the tool, tutor, and translator components present in Chapter 5: two tool components (the peer tutor's interface and the peer tutee's interface), a translator component (to echo actions from one tool to the other tool), and two tutor components (a domain tutor component to evaluate the peer tutee's problem-solving actions, and a

correction tutor component to evaluate the peer tutor's correction actions). In this section, I discuss the construction of the new tutor component and the integration of all the components.

7.3.1 New Tutor Component: Help-Giving Tutor

Assessment. To assess peer tutor help, the help-giving tutor used a combination of several inputs (see Table 28). As in *Phase 2: Adaptive Correction Support* (Chapter 5), the tutor component used the CTA assessment of tutee problem-solving steps and calculation of the next step hint. The component used two additional metrics to aggregate information about student chat. First, it used student self-classifications of chat actions, based on the sentence classifier selected (e.g., "give hint"). Second, a machine classifier of student help, constructed using *Taghelper Tools* (Rosé et al., 2008), that could determine whether students gave help or not and whether it was conceptual or not. This classifier is discussed in more detail in *Development 3: Assessment of Help Quality* (Chapter 8).

Table 28. Assessment in the help-giving tutor in *Phase 3*. The assessment of student interaction is based on the cognitive tutor evaluation of tutee actions, student self-classifications of their chat, and machine classifications of their chat.

Input	Component	Description
Evaluation of tutee steps	cognitive tutor	Whether last problem-solving step was correct or incorrect.
Next step hint	cognitive tutor	The hint the CTA would have given on the next step.
Self-labeling of chat	tutee	The label tutees used prior to sending a chat message (representing request, self-explanation, or other).
Self-labeling of chat	peer tutor	The label peer tutors used prior to sending a chat message (representing prompt, feedback, hint, explanation, or other).
Machine labeling of help	text classifier	A machine classifier that labeled each peer tutor utterance as help or not.
Machine labeling of elaborated help	text classifier	A machine classifier that labeled each peer tutor utterance as containing elaborated content or not.

Table 29. Productions in the help-giving model, spanning four skills: timely help, targeted help, elaborated help, and appropriate use of classifiers. The "++" indicates an effective behavior, the "+" indicates a somewhat effective behavior, the "-" indicates a somewhat ineffective behavior, and the "—" indicates an ineffective behavior.

#	skill	type	rule	agent	support
1	timely	++	IF tutee makes a help request THEN peer tutor gives help	self self	yes
2	timely	+	IF tutee makes an error THEN peer tutor gives help	CTA self	no
3	timely	+	IF tutee self-explains THEN peer tutor gives help	self self	no
4	timely	--	IF tutee makes 2 help requests in a row THEN tutee makes a third help request	self self	yes
5	timely	-	IF tutee makes a help request THEN tutee makes an error	self CTA	yes
6	timely	--	IF tutee makes 2 errors in a row THEN tutee makes a third error	CTA CTA	yes
7	targeted	+	IF tutee makes a correct step THEN peer tutor asks for explanation	CTA self	no
8	targeted	+	IF tutee makes a correct step AND tutee requests help THEN peer tutor asks for explanation	CTA self self	no
9	targeted	-	IF tutee makes a correct step THEN peer tutor gives next-step help	CTA self	yes
10	targeted	++	IF tutee makes an error THEN peer tutor gives previous step help	CTA self	yes
11	targeted	+	IF tutee makes an error AND tutee requests help THEN peer tutor gives previous step help	CTA self self	yes
12	targeted	-	IF tutee makes an error THEN peer tutor gives next step help	CTA self	yes
13	elaborated	+	IF peer tutor gives next step help THEN help is elaborated	self machine	yes
14	elaborated	-	IF peer tutor gives next step help THEN help is not elaborated	self machine	yes
15	classifiers	+	IF peer tutor labels help THEN peer tutor gives help	self machine	no
16	classifiers	-	IF peer tutor does not label help THEN peer tutor gives help	self machine	yes

Model Tracing. The above inputs were fed into a production rule model with 16 rules (see Table 29), modified from the idealized model described in *7.2.2*. Rules were divided into four categories: effective behaviors, somewhat effective behaviors, somewhat ineffective behaviors, and ineffective behaviors.

Effective behaviors, represented by the "++" in the table, were paths in the model that were considered to be beneficial for collaboration quality the majority of the time. For example, explaining a tutee error after a tutee makes it (rule 10 in Table 29) was considered to be an ideal behavior. The tutee commission of the error was detected by the *CTA*, and then explanation of the error was detected using the self-classification of the peer tutor. 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 29, 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 seven 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 11 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 seven somewhat ineffective behaviors. Finally, ineffective behaviors, represented by the "--" in the table, were considered to be detrimental to collaboration quality in most cases. The model consisted of two ineffective behaviors total: one where the peer tutor allows the tutee to commit three errors in a row (rule 4), and another where the peer tutor allows the tutee to commit three unanswered help requests (rule 6). 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. They are indicators that the tutee is in trouble, and the peer tutor, for whatever reason, is struggling to help. Rules were represented in a fully configurable xml file, and then parsed as part of the java code.

The implementation of this model is a subset of the designed model described in *7.2.2*. It focuses on peer tutor and not tutee behaviors, because the system is designed to support the peer tutor in supporting the tutee. It also does not check whether the peer tutor is marking the step correct, as the correction tutor handles that aspect of peer tutoring. The implemented model uses the state represented by tutee behaviors to identify the correct peer tutor responses. For example, peer tutors are encouraged to give next-step help after a tutee incorrect step or help request (rules 1 and 2), but not after a correct step (rule 9). The subset of the model that is implemented is relatively faithful to the designed model, with one notable exception: the idea of error feedback or previous-step help appears repeatedly in the implemented model, but is not explicitly represented in the designed model. I added error feedback to parallel the benefits of providing help targeted at tutee misconceptions, which is represented in the designed model. By operationalizing that concept as error feedback, the system is better able to target that behavior as one that should be increased. Another element of the designed model that is not evident from the implementation is the peer tutor's judgments, which gave the designed model flexibility. In the production rules, giving elaborated help is "somewhat effective", and giving unelaborated help is "somewhat ineffective", while in the designed model, the peer tutor makes a judgment about whether to give elaborated or unelaborated help. In the

implementation, this flexibility is not actually lost, but is represented in the way feedback is triggered through the assessment of collaborative knowledge. The implementation of the model is a conceptual hybrid between a production system and a constraint system. While the model is represented as a series of if-then rules, the chain of student behavior is often only two steps long (i.e., if the tutee makes a help request, answer it), and thus the model is very state-based. Additionally, parts of the model are highly similar to constraints, such as the condition of how peer tutor next-step help should be elaborated. This analysis will be resumed in the discussion section of the paper.

Knowledge Tracing. In addition to providing a localized assessment of student actions, each rule contributed to an overall knowledge-tracing assessment of the degree to which students had mastered one of four skills: timely help, targeted help, elaborated help, and classifier use. *Timely help*, covered by model rules 1-6, represented whether the peer tutor gave help when tutees needed it. *Targeted help*, covered by rules 7-12, represented whether peer tutors gave the *type* of help that tutees needed. *Elaborated help,* covered by rules 13-14, represented whether peer tutors gave help that included an explanation. Finally, *use of classifiers,* represented by rules 15-16, covered whether peer tutors used sentence classifiers appropriately. These skills were derived from our model of behavior described in the *Design* section (*7.2*), and from the theory on good peer tutoring described in the *Background* section (*2.4*).

We used Bayesian knowledge tracing to update a running assessment of peer tutor mastery of these four skills (Corbett & Anderson, 1995). Knowledge tracing computes the likelihood that students have mastered a skill for any particular opportunity to do so based on four parameters:

- $p(L_0)$ *Initial Learning.* The probability that students had mastered the skill before the opportunity.
- $p(T)$ *Acquisition.* The probability that students will learn the skill at the next opportunity.
- $p(G)$ *Guess.* The probability that students will give a correct response if they do not know the skill.
- $p(S)$ *Slip.* The probability that students will give an incorrect response if they know the skill.

For each student step, the algorithm first calculates the probability that students had mastered the skill prior to taking the step, using one of the following two formulas:

$$p(L_{n-1}) = (p(L_{n-1}) * (1 - p(S))) / ((p(L_{n-1}) * (1 - p(S))) + p(G) * (1 - p(L_{n-1})))$$

$$p(L_{n-1}) = (p(L_{n-1}) * (p(S))) / ((p(L_{n-1}) + (1 - p(G)) * (1 - p(L_{n-1})))$$

Then, using those values, the algorithm calculates the probability that students currently know the skill.

$$p(L_n) = p(L_{n-1}) + p(T) * (1 - p(L_{n-1}))$$

While this type of knowledge tracing has been used in individual settings, it has not to my knowledge been used in collaborative settings. I made a few modifications to the basic knowledge tracing

algorithm to make it more appropriate for collaborative settings. First, given the ill-defined nature of the collaborative task, I decided to give peer tutors the benefit of the doubt with respect to their mastery of help-giving skills. At the beginning of the tutorial session, we set $p(L_0)$ 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), I 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. I 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

Table 30. 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	help	unelaborated	incorrect step

(3) Model and Knowledge Tracing

	timely	**targeted**	**elaborated**	**classifiers**
$p(L_0)$	0.903	0.933	0.903	0.794
rule fired	2	12	14	15
valence	+	-	-	+
$p(L_n)$	0.956	0.864	0.81	0.9

(4) Feedback Selection

	timely	**targeted**	**elaborated**	**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
target	n/a	chat	chat	n/a
chosen	no	yes	no	no

(5) Message Choice

possible prompts	"Tutee, why did you take that last step", "Tutee, do you know what mistake you made?", "Tutor, before you help your student on the next step, you may want to talk to them about the previous step.", "Tutor, can you explain your partner's mistake?", "Tutor, is there anything your partner doesn't understand right now?"
prompt chosen	"Tutor, is there anything your partner doesn't understand right now?"

the system characterizes student responses as ineffective even if they have mastered the skill. Finally, I had $p(T)$ toggle based on whether students received feedback or not – it is unlikely that students who did not know the collaborative skill would learn it if they had not received support.

In Table 30, there is an example model and knowledge trace taken from the study described in *7.4*. The students were solving the problem "$-wn + 3n = w$" for w, and the tutee had just subtracted 3n from both sides, which was incorrect (#1 in the Table 30). The tutor then said "factor out n" and labeled it as a "hint". The computer classified the chat as unelaborated help (#2 in the table), and recognized that it came immediately after an incorrect step. This action fires model rules 2, 12, 14, and 15, meaning that the knowledge tracing assessments for all four skills are updated (#3 in the Table 30).

Support Construction. I used a combination of the model tracing and knowledge tracing to decide when to give students the reflective prompts in the chat window. 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 knew a skill. Each rule was linked to a set of feedback thresholds that specified that: (1) if a rule gets fired, and (2) the skill adjustment associated with the rule falls within one of the feedback thresholds linked to the rule, then (3) add the rule and threshold to a list specifying the possible 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 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 *targeted* and *elaborated* help fell within the feedback threshold, and were added to the list (#4 in the table). 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 the table), the prompt "Tutor, is there anything your partner doesn't understand right now?" was sent to the students.

Hint messages were implemented in a less adaptive way. The collaborative prompts given to students when they requested a hint (integrated with the CTA domain hint) were based on the hint level: First level hints related to the *timely* skill, second level hints were related to the *targeted* skill, and third level hints were related to the *elaborated* skill. Hints were tailored to the problem-solving situation (e.g., there was a different set of hints based on whether tutees had just made and error or correct step), but hints were otherwise randomly chosen.

Conceptual resources were set to appear based on the classifier students selected. Different conceptual resources were presented based on the problem type students were working on, but conceptual

resources of a particular category did not vary between problems. The peer-mediated correction feedback was taken directly from the version of *APTA* implemented in *Phase 2: Adaptive Correction Support* (Chapter 5).

7.3.2 Integration of Components

As in *Phase 2*, components communicated using the *CTRL* message protocol, and the way components interacted was defined in the control module. See the left hand side of Figure 18 for a diagrammatic representation of the message passing logic in the adaptive support condition. The only new component here was the help-giving tutor, which received messages from the two tools, the cognitive tutor, and the text classifier, and as described above, used those inputs to decide whether and how to send feedback. To account for the new complexity in communication between components, I modified the "message group" definition to allow for communication between components limited to particular actions or tools.

7.3.3 Comparison Conditions

I created a fixed comparison condition that swapped out the adaptive support tutors in favor of parallel fixed resources (see the right side of Figure 18). To create a fixed parallel to the *adaptive cognitive support*, where peer tutors were given domain hints and feedback, they were instead given annotated solutions to the current problem (#2 in Figure 18), a technique that had been used as part of other successful peer tutoring scripts (e.g., Fantuzzo, Riggio, Connelly, & Dimeff, 1989). With this fixed assistance, peer tutors could consult the problem solutions at any time, but would not receive feedback on

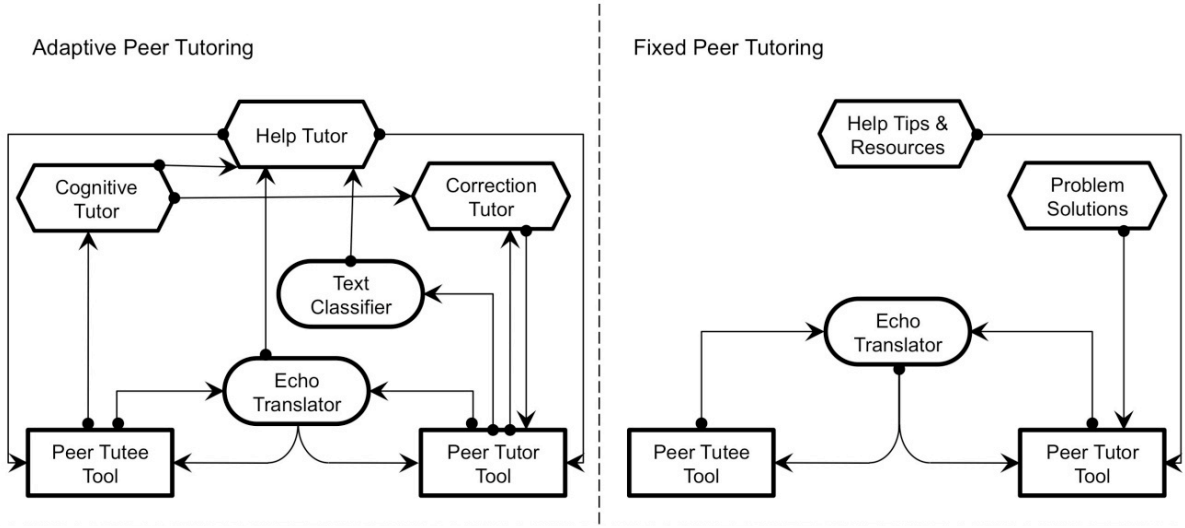

Figure 18. Message passing in *Phase 3.* Adaptive help-giving support components are on the left. Fixed support components are on the right.

whether the current problem was completed or whether their help was correct. To parallel the *hints on demand,* students were given access to a "random tip" button that yielded multi-level randomly selected tips (#4 in Figure 19). While the overall content of tips was the same as the hints on demand, the tips were randomly selected rather than chosen adaptively. The random tips did not contain any adaptive cognitive content. For *adaptive resources,* I gave students access to the same resources as they had in the adaptive condition, but the resources did not change based on the sentence classifiers students selected – instead, students had to select which resource they wanted to view without additional guidance (#1 in Figure 19). Finally, instead of receiving *reflective prompts* in the chat window, I gave students reflective collaborative tips between each problem, with parallel content to the reflective prompts present in the adaptive condition (#3 in Figure 19). Both students were presented with five randomly chosen reflective statements after each problem was complete such as "Good work! Remember, hinting or explaining the reason behind a step can help your partner learn how to do the step correctly." I chose that form of support also because it is common for students using a collaborative script to receive reflective prompts at fixed intervals. This approach was a reasonable way to provide students with similar content to the adaptive condition. I ran the cognitive tutor, correction tutor, and help-giving tutor to log evaluations of student actions, but did not use

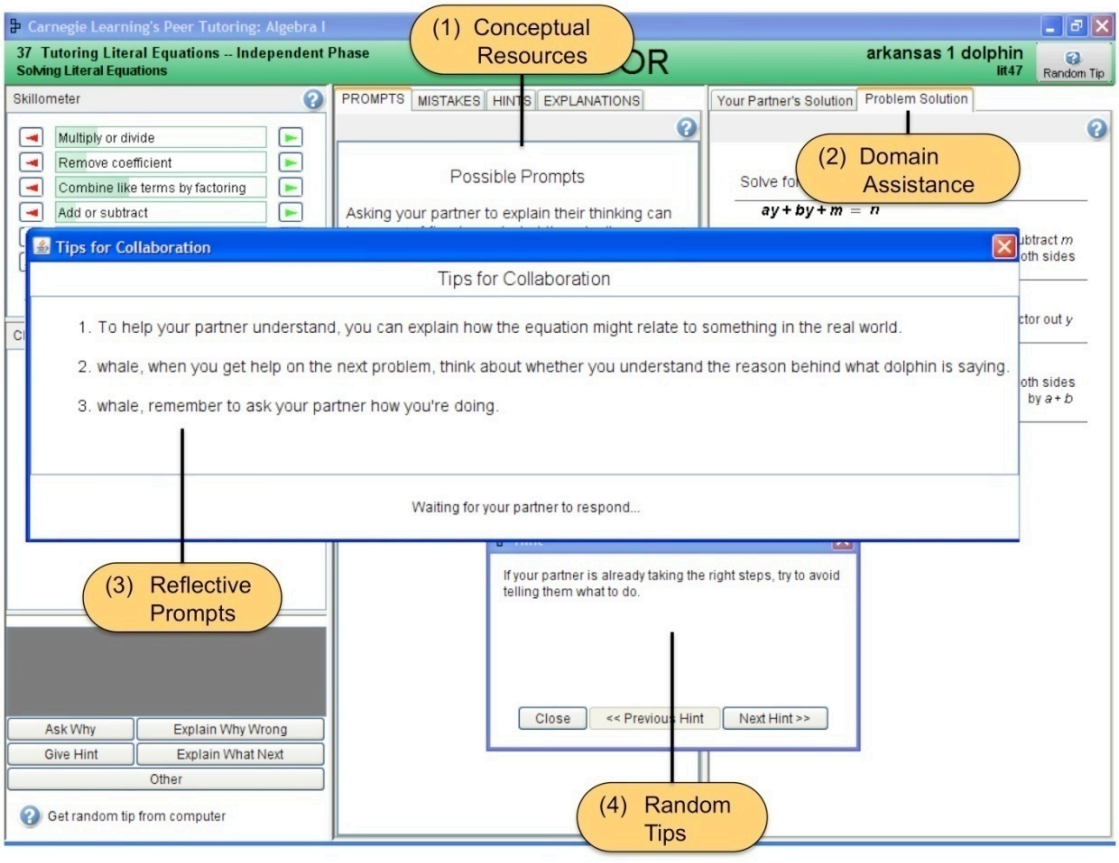

Figure 19. Peer tutor's interface in fixed support condition. Conceptual resources are not connected to sentence classifiers, domain assistance is in the form of fixed problem solutions, reflective prompts are randomly delivered between problems, and the students can request randomly selected collaboration tips.

those components to provide support to students in the fixed condition.

7.4 Evaluation: Study 3

7.4.1 Experimental Design

The goal of this study was to investigate the effects of combined adaptive help-giving and domain support on student interaction and learning. The new system was deployed in a classroom experiment to examine the influence of the adaptive support on peer tutor help-giving behaviors, and on how our design related to peer tutor accountability, efficacy, and the perceived relevance of the support. In order to determine whether it was indeed the way the support was designed that produced a change in student behavior, I compared it to fixed support that provided the same *collaborative knowledge*, but did not include adaptive elements. I hypothesized that the adaptive support will lead to greater improvements in interaction and learning compared to the fixed support because it will give students guidance on how to improve their help at moments where they can apply the guidance. In this section, I first describe a quantitative analysis of the effects of the adaptive support as compared to fixed support on interaction and learning. I then discuss a qualitative analysis exploring to what extent our designs for accountability, efficacy, and relevance had the desired impact.

7.4.2 Method

Participants. Participants were 104 high-school students (54 male, 50 female) from two high schools, currently enrolled in Algebra 1, Algebra 2, or Pre-Calculus. Both high schools used the individual version of the CTA as part of regular classroom practice. The literal equation solving unit that we used was a review unit for many of the students, one that they had already covered in Algebra 1. Nevertheless, the concepts in the unit were difficult for the students to understand, and teachers were in favor of reviewing the unit. Students from each class were randomly assigned to one of the two conditions, and to either the initial role of tutor or tutee. This analysis focuses on those students who interacted with the system as a tutor, and thus excluded 27 students who only took on the role of tutees; that is, they were absent on one or both supported tutoring days and were tutees on the days they were present. We further excluded 1 student who was partnered with a teacher when tutoring, and 2 students who played the role of tutor in both collaboration periods. A total of 74 students were included in the analysis of interactions. Of those 74 students, another 23 were excluded for the learning analysis for having not turned in a pretest or a posttest.

Procedure. The study took place over the course of a month, spread across six 45-minute classroom periods (see Table 31). During the first period, students took a 15-minute pretest measuring domain learning. Then, in the second period, students spent 45 minutes in a preparation phase, solving problems individually using the CTA. Students worked on one of two problem sets; focusing on either factoring in literal equation solving or distributing in literal equation solving. Periods 3 and 4 were collaboration periods, where students were given partners, and tutored them on the problems they had solved in Period 2, with either

adaptive or fixed support. Students were given different partners for each of the two collaboration periods. They were paired with students who were in the same condition, but who had solved a different problem set during the preparation phase. Within these constraints, we assigned pairs randomly, with the exception of not pairing students teachers explicitly told us would not get along. Within a pair, students were randomly assigned to the tutor or tutee role during the first collaboration period, and then they took on the opposite role during the second collaboration period. In Period 5, students collaborated with new partners without any adaptive support (test phase), and in Period 6, students took a delayed domain posttest.

Table 31. Study procedure in *Phase 3*.

Week	Day	Time (minutes)	Fixed Support	Adaptive Support
1	1	15	Pretest	Pretest
1	2	45	Preparation	Preparation
1	3	5	Instruction	Instruction
1	3	40	Collaboration + Fixed Support	Collaboration + Adaptive Support
2	4	45	Collaboration + Fixed Support	Collaboration + Adaptive Support
2	5	45	Unsupported Collaboration	Unsupported Collaboration
4	6	20	Posttest	Posttest

Measures. Students' individual learning was assessed using counterbalanced pretests and posttests, each containing 10 conceptual items, 5 procedural items, and 2 items that required a verbal explanation. Some of the conceptual items had multiple parts. The tests were developed by the experimenter, but adapted in part from Booth's measures of conceptual knowledge in Algebra (Booth & Koedinger, 2008). Tests were approved by the classroom teacher, and were administered on paper. Answers on these tests were scored by marking whether students were correct on each item part, computing the scores for each item out of 1, and then summing the item scores to get a total score.

In order to analyze student collaborative process, all semantic actions students took within the system 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, computer tutor responses were logged, which included both the system's evaluation of the action and the assistance students received. Using this data, I computed the number of problems viewed by each student, and the number of problems correctly solved (in the fixed condition, students could move to the next problem without having correctly solved the previous one). I calculated the number of errors viewed by students when they took on the peer tutoring role, and the number of times peer tutors used each type of sentence classifier. Finally, I computed

peer tutor exposure to the assistance in APTA, including the number of times tutors received reflective prompts and the number of times they requested a cognitive hint.

The dialog was segmented by chat messages (creating a new segment every time students hit enter), and two raters coded the chat data on several dimensions. I computed interrater reliability 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. First, each help segment was coded for whether it constituted previous-step help, that is, help relating to an action tutees had already taken (e.g., "no need to factor because there is only one g"; kappa = 0.83), or whether it was next-step help, that is, help relating to a future action in the problem (e.g., "how would you get rid of 2h?"; kappa = 0.83). Finally, each help segment was coded for whether it contained a concept (e.g., "add ax" is purely instrumental help, while "add ax to cancel out the –ax" is conceptual). I decided to code for conceptual instead of elaborated help because there were few instances of elaborated help that was not conceptual, and conceptual elaborated help was considered to be better than elaborated help alone (Fuchs et al., 1997). Kappa for conceptual help was 0.72.

7.4.3 Quantitative Results

I used quantitative interaction and learning data to determine if peer tutors' help quality increased because of the assistance they received, and if an increase in help quality translated into a learning improvement.

Learning Outcomes. I first looked at whether learning outcomes varied between the two conditions. The adaptive condition had a mean pretest score of 33.53% (*SD* = 25.11%) and posttest score of 40.55% (*SD* = 21.50%). The fixed condition had a mean pretest score of 39.13% (*SD* = 23.92%) and posttest score of 47.10% (*SD* = 26.28%). I conducted a two-way repeated-measures ANOVA with condition as a between-subjects variable and test-time as a within-subjects variable. Only students who had participated in the pretest, posttest, and an intervention phase as a peer tutor were used. Students in both conditions learned ($F[1,49] = 11.97$, $p = 0.001$), but there were no significant differences between conditions ($F[1,49] = 0.048$, $p = 0.828$).

Problem-Solving Behavior. To get a sense of the context of student interaction, I examined whether there were systematic high-level differences between the two conditions in the way students solved problems and gave help. I used a MANOVA with condition as the independent variable to evaluate the differences between conditions for the following variables: problems viewed, problems completed correctly, tutee errors viewed by tutors, and help given by tutors. The analysis revealed significant differences between conditions (Pillai's Trace = 0.30, $F[1,72] = 7.68$, $p = 0.001$). Table 32 displays the results of one-way ANOVAs for each dependent variable. The students in the fixed condition saw significantly more problems than students in the adaptive condition (row 1). Students in the fixed condition could skip past problems that gave them trouble (and occasionally did not realize they had made a mistake), while students in the

Table 32. Differences in peer tutoring context across conditions.

	Adaptive		Fixed		ANOVA results	
Context variables	*M*	*SD*	*M*	*SD*	*F*(1,74)	*p*
Problems seen	7.90	4.51	10.43	5.76	4.483	0.038
Problems completed	7.26	4.42	7.60	4.37	0.113	0.738
Errors viewed	15.71	8.56	12.63	8.41	2.44	0.122
Help given	11.92	6.23	12.17	8.87	0.019	0.890

adaptive condition had to overcome every impasse they reached. However, both conditions completed similar numbers of problems correctly (row 2), and the total number of errors viewed by peer tutors was not significantly different across conditions (row 3). Finally, the amount of help given by peer tutors was not significantly different across conditions (row 4). The ratio between errors viewed by the peer tutor and help given was roughly 4:3 in the adaptive condition and 1:1 in the fixed condition. In the following, we present count data of particular aspects of student interaction, and use Mann-Whitney non-parametric tests to evaluate the relationship between variables. Unless otherwise noted, I perform statistical tests on the raw data counts, but to better illustrate what occurred, I may also present ratios between the count data and context variables like errors viewed or total amounts of help.

Helping Behaviors. The main goal in the design of the adaptive assistance was to improve the quality of help given in the adaptive condition. This goal can be operationalized as improving the amount of conceptual help given, since conceptual help is an indicator of elaborative processes in peer tutoring, and a predictor of learning gains for both students. The effects of condition on conceptual help were significant (see Table 33, row 1). In total, roughly 20% of the help was conceptual in the adaptive condition, nearly double the percentage of help that was conceptual in the fixed condition (10%). I also hypothesized that students would give more previous-step help targeted towards tutee errors in the adaptive than in the fixed condition. In fact, previous step help was not significantly different between conditions and was overall rather low; peer tutors explained roughly 1 out of every 8 errors in both conditions (Table 33, row 2).

Table 33. Help quality across conditions.

	Adaptive		Fixed		Mann-Whitney results	
Interaction variables	*M*	*SD*	*M*	*SD*	*U*	*p*
Conceptual help (n=74)	2.67	2.83	1.34	2.14	468.5	0.015
Previous step help	2.25	1.78	1.78	1.86	608.5	0.17
Classifiers used (n=74)	7.95	6.77	4.28	5.78	371.5	0.001
% help given with classifiers (n=71)	56.84%	33.7%	31.0%	34.4%	348.00	0.001
% non-help with classifiers	14.8%	25.9%	10.1%	23.4%	657.5	0.334

In addition to improving the quality of student interaction, I intended that the adaptive help would improve student use of interface features, and in particular, encourage students to use the sentence classifiers while chatting. As described in the *Background* (*2.2*), sentence classifier use is theoretically related to help quality, and thus should be related to the amount of conceptual help that students give. Further, the more appropriately students use classifiers, the better intelligent systems are at determining the content of student chat. Thus, one hypothesis was that students would use help-related classifiers (i.e., not the neutral "comments" classifier) more *frequently* in the adaptive than in the fixed condition, regardless of the content of their utterances. This hypothesis was supported by the data (see Table 33, row 3). Students used roughly 2 classifiers for every 3 errors in the adaptive condition, compared to 1 classifier for every 3 errors in the fixed condition. However, while this measure reflected how *often* students used classifiers, it did not reflect the student's purpose in using the classifiers. Another prediction was that when peer tutors gave help to tutees, they would be more likely to label their utterance with one of the help-related classifiers than the "comments" classifier. The percentage of help given using help-related classifiers was significantly greater in the adaptive condition than in the fixed condition (see Table 33, row 4), suggesting that students used classifiers *appropriately* more often in the adaptive condition. The percentage of non-help chats given using help-related classifiers were not significantly different between conditions, suggesting that it was not increased classifier use overall that was driving the effect (Table 33, row 5).

We further explored the relationship between condition, sentence classifiers used, and conceptual help given. The number of classifiers used and conceptual help given were correlated ($r[72] = 0.442$, $p < 0.01$), but it was not clear whether condition had separate effects on classifiers used and conceptual help given, or whether the number of classifiers used influenced the amount of conceptual help given (as suggested by prior research on sentence classifiers). To explore these separate possibilities, we conducted a regression analysis to predict the amount of conceptual help given controlling for the number of classifiers used. We used student condition, the number of sentence classifiers used, and the amount of help given overall as predictor variables. The model was indeed statistically significant (*likelihood chi ratio* = 33.287, $df = 3$, $p < 0.001$). Condition was a significant predictor of conceptual help given (*Beta* = 0.687, *SE* = 0.284, $p = 0.016$), as was the amount of help given (*Beta* = 0.087, *SE* = 0.0182, $p < 0.001$). Classifiers used were marginally predictive (*Beta* = 0.042, *SE* = 0.217, $p = 0.052$). This analysis suggests that classifiers used partially mediated the relationship between condition and conceptual help, but condition still had an independent effect on conceptual help given..

Use of Assistance. In an attempt to determine which forms of assistance may have contributed to the effects of the adaptive condition as a whole, we examined how often students were exposed to each type of assistance, and qualitatively looked at how they reacted. In the case of the reflective prompts, students in the adaptive condition received the prompts a mean of 6.95 times per session (*SD* = 4.51). Thus, 62% of the total help peer tutors gave was followed by a computer prompt. Although that seems high, only 30% of total peer tutor chat actions received feedback (including both help and coordination actions), which

matched our design intentions. Interestingly, the peer tutors in the fixed condition had far more access to the reflective content, as they received 5 prompts each time they completed a problem, and completed a mean of 11 problems.

Because of the ways the resources were linked to sentence classifier use, it was difficult to determine when students used the adaptive resources that were presented. However, we can examine how often students had the opportunity to incorporate resources into their interaction. On average, students clicked on a sentence classifier to compose a message, without necessarily sending the message, 9.90 (SD = 6.99) times in the adaptive condition, compared to 5.32 (SD = 7.33) times in the fixed condition, U = 366.5, $p < 0.001$. An indication of whether students used the resources was how often they selected a classifier without sending a message, suggesting that they selected the classifier solely to look at the resources. The mean number of classifiers that were not used to compose a message was 2.10 (SD = 0.44) in the adaptive condition compared to 1.24 (SD = 0.26) in the fixed condition (U = 407.5, $p < 0.001$). Interestingly, there was also evidence that students in the fixed condition found the resources to be useful, viewing resources tabs an average of 2.59 times per problem (SD = 3.18). That is, students actively sought out resources in the fixed condition with the same frequency as students selected classifiers without sending a message in the adaptive condition.

The third type of interaction support provided to students was the context-based interaction hint that accompanied the cognitive hint students received. In the adaptive condition, students requested a hint a mean of 3.53 times (SD = 5.24), compared to the mean of 0.14 times that students requested a random hint in the fixed condition (SD = 0.48). This difference was significant (U = 232.5, $p < 0.001$). However, it appeared that students were primarily requesting hints in the adaptive condition to access cognitive help. In fact, a mean of 79.4% (SD = 38.3%) of the time, after students received a cognitive hint, they communicated the hint to their partner on the next help turn, often without modifying the hint in any way. It is likely that when students incorporated the cognitive component of the hint, they were not attending to the interaction component.

Despite this mixed evidence on the effectiveness of the help-giving support, there was strong evidence that the domain support was effective. In cases where peer tutors *incorrectly marked* a tutee action, they received adaptive domain feedback in the adaptive condition but not in the fixed condition. In the adaptive condition, peer tutors changed their incorrect response a mean of 66.21% of the time (SD = 26.83%), compared to only 6.16% of the time in the fixed condition (SD = 2.29%). This difference was significant (U =36.5, $p < 0.001$). Here, the adaptive domain support was perceived as highly relevant by peer tutors, and led them to give more correct feedback to their tutees.

Discussion. The adaptive support improved both the quality of help given and the use of sentence classifiers by peer tutors compared to the fixed support condition. While these interaction improvements did not transfer into learning improvements, it is possible that students would have had to participate in the intervention for a longer time for any effects on learning to be seen. Nevertheless, the effects on student

interaction are encouraging. Not only is it promising that we were able to increase conceptual content in student help, but our increase of the accuracy of student classifier use may be beneficial in itself. Indeed, sentence classifiers are typically included in fixed support to collaboration because they are hypothesized to improve the quality of student interaction, by making expectations on student dialog clear to the students. From that perspective, an adaptive intervention that increases student frequency and accuracy of classifier use would be a positive augmentation to many existing fixed scripts. In the following section, I examine cases to shed some light on the potential mechanisms for how adaptive support influenced student interaction.

7.4.4 Qualitative Results

While the quantitative analysis could tell us that the adaptive support had a positive influence on peer tutor help-giving, it was not clear *why* this effect occurred. I next investigated, on the basis of case analyses, to what extent the positive influence of the adaptive support was related to the hypothesized desired effects on student motivational factors, following the design principles identified in Chapter 6. I present one case representative of the positive effects of accountability on student interaction, and one case representative of the negative effects of a lack of perceived relevance. I use both cases to discuss the influence of efficacy on student interaction.

A Case of Accountability & Elaborative Processing. With this case study of Dyad 1, I illustrate how feelings of accountability to be good tutors engendered by the adaptive support encouraged dyads to engage in elaborative processing. In this dyad, the peer tutor scored 55% on the posttest, and the tutee scored 20%. The interaction occurred on the second tutoring day, and concerned the problem $kj - mk = fr$, solve for k. It was the second problem the dyad had seen that day, but the first with this form. Over the course of the interaction, the different assistance types increased the peer tutor's accountability to knowledge and to reasoning – that is, her effort to give the correct answer and to give a conceptual explanation for her answer. The interaction began with the tutee asking for help (see Table 34, row 1). When the tutee asked for help, and the peer tutor clicked on the sentence classifier "explain next step" to compose her response, the peer tutor received a resource on how to construct good explanations. On first glance, the resource appeared to have little effect, as only 10 seconds pass between the time the resource was presented in the interface and the time the peer tutor's response was submitted, and the peer tutor gave instrumental help ("add *mk* to both sides"; row 4). However, the simple presentation of this resource began to establish the expectation that peer tutors are expected to put thought into the help that they give. A second type of assistance was presented immediately after the peer tutor had delivered her instrumental help: the computer said in the chat window, where both collaborators could see it ("Tutee, did you understand the reason behind what the tutor just said?"; row 5). Not only was the computer prompting the tutee to reflect, but also publically reminding the peer tutor that help should include an explanation in addition to an instruction, further increasing the peer tutor's accountability for giving elaborated help. In fact, the tutee responds to

Table 34. Positive effects of adaptive support on student interaction. Support may trigger feelings of accountability.

	Solve for *k*: *kj – mk = fr*	
1	Tutee: [*both*]	What should I do first?
2	Tutor: [*self*] *classifier*	*chooses "explain next step"*
3	Agent: [*tutor*]	*gives resource on explanations*
4	Tutor: [*both*]	Add mk to both sides.
5	Agent: [*both*]	Tutee, do you understand the reason behind what the tutor just said?
6	*Tutee: [both] problem*	*adds mk to both sides of the*
	Solve for *k*: *kj = fr + mk*	
7	Tutee: [*both*]	Does it matter that there's a k on the right side?
8	Tutor: [*both*]	*marks the "add mk" step correct*
9	Agent: [*tutor*]	*highlights step*
10	[*tutor*]	This step is wrong. Give your partner some advice on what to do next
11	Tutor: [*tutor*]	*chooses "comments" classifier*
12	[*both*]	wait!!! I completely messed up… the computer wants you to subtract *kj* from both sides, because of the other *k* in the problem. sorry =(
13	Tutee: [*both*]	haha, it's alright, these problems are so simple but confusing.

this prompt with evidence of deep processing, saying in row 7 of Table 34: "Does it matter that there's a *k* on the right side?" The tutee was reflecting on features of the problem that were relevant for attaining the problem solution. After the tutee had in fact added *mk*, and the peer tutor had marked the step wrong, the computer further enforced the peer tutor's accountability to give the correct answer by saying privately to the peer tutor: "This step is wrong. Give your partner some advice on what to do next." At this point, the peer tutor's response represented a breakthrough in the peer tutor's helping behaviors. The peer tutor responded with a conceptual statement, saying "the computer wants you to subtract *kj* from both sides, because of the other *k* in the problem" (row 12). This statement explained what the tutee should do,

explained why, and alluded to the concept that all *k*s in this problem have to be on the same side, suggesting that the peer tutor was reflecting on the next step and elaborating on her knowledge. It was the first conceptual statement made by this particular peer tutor. This insight on the part of the tutor, and articulation of the insight to the tutee, had benefits for both parties. The error that Dyad 1 made during this problem related to the concept that to solve for a given variable all instances of the variable need to be moved to the same side of the equation. Both the tutor and the tutee in the dyad got a similar problem correct on an individual posttest, suggesting that as a result of this interaction, they had mastered the discussed concepts.

A Case of Support Relevance & Shallow Processing. While peer tutors appeared to find adaptive help on how to solve the problem extremely relevant, they did not have a similar response to adaptive assistance on how to give good help, potentially leading them to process the problem shallowly. The case of Dyad 2 in Table 35, who engaged in suboptimal interaction, is from the first tutoring day (Period 3); the dyad was solving the problem $6t - qt = wr + qv$. This problem was their ninth problem of the day, but the first problem they had encountered where they had to move two variable instances to the same side. The peer tutor had scored 23% on the pretest, and the tutee had scored 38%. The following dialogue began when the tutee had reached the equation $6t - wr = qt + qv$, but then incorrectly divided both sides by vt instead of $v + t$. The tutee triggered the exchange using a question that shows the tutee is reflecting on the situation ("It won't let me get rid of the *v* and *t.* Help me"; row 1 in Table 35). The peer tutor asks for a hint, but then only transferred the *instrumental* component of the hint to the tutee ("Multiply both sides by *vt*"; row 6 in Table 36), suggesting that while the peer tutor felt that the domain help is relevant, he did not perceive the conceptual scaffolding as relevant. As in the previous scenario, the computer prompted the tutee for further explanation ("eagle, can you talk about why you took that last step?"; row 7 in Table 35), but this only served to confuse the students further ("what last step?"; row 8 in Table 35), suggesting that the vague wording of the prompts may be a liability in this case. After getting more content-related feedback and another hint, the peer tutor relayed the hint again to the tutee, indicating his lack of understanding of the situation (if the tutee were to undo the steps, there would be no need to multiply the sides by *vt*; row 15 in Table 36). After this exchange, the tutee realized his error, and proceeded to solve the problem, without interacting further the peer tutor. This lack of communication has effects on the posttest results: For Dyad 2 to solve this problem correctly, they needed to master the concept that to isolate the *x* in an expression like $x(a+b)$ you need to divide by $(a+b)$. Neither member of the dyad got the related conceptual question right on the posttest they received at the end of the study. Not surprisingly, the peer tutor in this interaction came out of the session unsure of how to use the computer help, saying the following class period when he was the tutee: "yeahh the tutor is confusing cuz it gives youu all this stuff to write about but I had no clue what to write when i was the tutor." In summary: When this peer tutor gave help, he ignored the collaborative components of the hint he received and focused on the cognitive component, exactly the opposite of what was intended by the design.

Table 35. Problem with perceived relevance of adaptive support. While the peer tutor perceived adaptive correction support as relevant, he did not perceive the help-giving support as relevant.

Solve for *q:* $(6t - wr)/vt = (qt + qv)/vt$			
1	Tutee:	[*both*]	It won't let me get rid of the v and t. Help me.
2	Tutor:	[*agent*]	*requests hint*
3	Agent:	[*tutor*]	A good hint explains why in addition to what. What can you do to both sides to get the q by itself? In $(qv + qt)/vt$, $qv+qt$ is divided by vt. How do you undo division? Multiply both sides by vt.
4	Tutor:	[*self*]	*chooses "give hint" classifier*
5	Agent:	[*tutor*]	*gives resource on hints*
6	Tutor:	[*both*]	Multiply both sides by vt
7	Agent:	[*both*]	Tutee, can you talk about why you took that last step?
8	Tutee:	[*both*]	What last step? The simplifying fractions?
9	Tutor:	[*both*]	*marks the simplifying fractions step right*
10	Agent:	[*tutor*]	This step is wrong. Get your partner to think about what to do next.
11	Tutor:	[*agent*]	*requests hint*
12	Agent:	[*tutor*]	Remember to explain what your partner did wrong, in addition to what to do next. What can you do to both sides to get the q by itself? In $(qv + qt)/vt$, $qv+qt$ is divided by vt. How do you undo division? Multiply both sides by vt.
13	Tutor:	[*self*]	*chooses "give hint" classifier*
14	Agent:	[*tutor*]	*gives resource on hints*
15	Tutor:	[*both*]	Delete the last 3 steps and multiply both sides by vt

Transfer of Control. We had also attempted to design feedback to maintain the peer tutors' sense of control of the situation and belief in their tutoring ability. To a certain extent, the design appeared successful, and in some cases, the support we gave helped peer tutors to take control of the situation. For example, after one tutee added ax to both sides in the problem $ax - y = 8$, and the peer tutor marked it right, the peer tutor received the feedback: "This step is not right. Tell your partner what mistake they made. Here is a hint to help you tutor your partner. Since $a*x$ is positive, you should subtract to remove it from the left side. Erase your last step and subtract $a*x$ from both sides." In consequence, the peer tutor *changed* their response, marking the step wrong, and then smoothly gave the conceptual hint "It's a positive ax you wouldn't add u would subtract." This peer tutor was adept at using the cognitive tutor hints to give their partner guidance, and while the peer tutor didn't acknowledge his error, he did give error feedback to the tutee. However, sometimes students would attribute hints to the computer in order to indicate their uncertainty and to convey to their peer tutee a sense that they (peer tutor and tutee) are in the same boat. A good example of this phenomenon is in the first case study, where the peer tutor both attributed the hint to the computer, and apologized for the confusion ("wait!!! I completely messed up… the computer wants you to subtract *kj*

from both sides, because of the other *k* in the problem. sorry =(“). Interestingly, the peer tutor gave a much more elaborated hint than the one she had received from the computer, but still attributed the hint to the computer, probably to indicate her own lack of confidence in the solution. The same students from Dyad 1 verbally expressed similar sentiments at several points, bonding over their own inexpertise: The peer tutor said, “wow… this is so confusing!”, and the peer tutee replied, “I’m glad I’m not the only one who’s confused! hahaha”. Those two students went on to be successful at solving the problem. Against our designs, it appeared that the peer tutor indicating uncertainty and attributing help to the computer was beneficial for the tutor-tutee relationship.

7.5 Outlook and Discussion

7.5.1 Introduction

In this chapter, I described the design (*7.2*) and implementation (*7.3*) of adaptive *help-giving* support, where we provided peer tutors with multiple types of assistance in giving conceptual help targeted at tutee misconceptions. I compared this support to a fixed control where peer tutors received traditional forms of fixed support with parallel content, but not adapted to student needs (*7.4*). I found that the adaptive support improved the quality of student interaction but not their learning, compared to the fixed control. In this subsection, I discuss the design (*7.5.2*), technical (*7.5.3*), and learning sciences (*7.5.4*) contributions to the work, and future directions (*7.5.5*).

7.5.2 Design

This chapter adds to *Q1-D1, Q1-D2,* and *Q2-D1* (see Table 1). With respect to *Q1-D1* (“How do ITS approaches to modeling apply to ALCS?”), I modeled student collaboration as a production system, focusing on peer tutor help-giving behaviors. Like in *Phase 2: Adaptive Correction Support* (Chapter 5), it became apparent how important it was to build flexibility into the model, so that support came at moments where peer tutors needed it, but did not prevent peer tutors from behaving in the way they saw fit. With respect to *Q1-D2* (“How do ITS approaches to support apply to ACLS?”), I took the support ideas from *Development 2: Student Needs and Design Space for Adaptive Support* (Chapter 6), and implemented the most promising ones into *APTA*. It appeared that public help-giving support and private correction support was most effective, and this issue is revisited in the overall design discussion (*10.2*). For *Q2-D1* (“What role does domain information play in collaboration models and feedback?”), I investigated the use of domain information in collaborative models by making the domain context a main component of the help-giving model. This choice allowed the modeling of behaviors important to peer tutoring, such as help at impasses, and allowed the presentation of domain support integrated with collaboration advice.

7.5.3 Technology

This chapter makes technological contributions to the understanding of whether collaborative skills can be knowledge traced (exploring *Q1-T1*; "Can collaborative skills be knowledge traced?"), and is a second instance of the use of the *CTRL* architecture to implement collaborative support (*Q1-T2*; "How can existing and custom components be integrated?"). In this chapter, Bayesian knowledge tracing, a method used to assess domain knowledge in individual tutoring, was applied to collaborative skills with some success, looking into *Q1-T1*. It was used primarily as a technique for giving students support based on skill assessments, rather than as a ground truth assessment of those skills. As such, it was relatively successful, and allowed us to build inherent flexibility for peer tutor actions into *APTA*. The Bayesian parameters had to be adjusted to make this approach effective, and there is future research to be done on whether these parameter adjustments are, in fact, good representations of collaborative skills. This subject is taken up again in *Phase 4: Cognitive and Motivational Benefits of Adaptive Support* (Chapter 9), and then again in the discussion of overall technical contributions in *10.3*. The second technical contribution in this chapter relates again to *Q2-T2*, and involves the use of *CTRL* to implement the adaptive help-giving support. Here, *CTRL* was applied in a more complex scenario, with an adaptive help-giving tutor added to the adaptive collaborative support session. The help-giving tutor received input from the two tool clients, the cognitive tutor, and the text classifier in order to model peer tutor actions. Using *CTRL,* it was relatively easy to integrate the new components in with the overall system, and to create a control condition for the study by removing messages sent from the tutor components to the tool components.

7.5.4 Learning Sciences

These results add to the small but growing body of evidence that adaptive support can improve the quality of student collaboration, addressing *Q1-L1* ("What are the effects of *ACLS* on student collaborative interactions?"). Previous research in the effects of adaptive support compared to a fixed control has found that adaptive support can increase student learning (Kumar et al., 2007), but little is known about how adaptive support affects collaborative process compared to fixed support. This research provides evidence of the direct effects of adaptive support on student interaction; the adaptive support aided students to increase the conceptual help content of their utterances compared to fixed support. This successful intervention could potentially be applied to other collaborative scenarios that seek to improve the conceptual quality of student discussion (e.g., integrated with the peer tutoring script of Fuchs and colleagues, 1997). It is true that we did not find effects of adaptive support on student learning, compared to fixed support (addressing *Q1-L2*; "What impact does adaptive support have on student learning?"). While this is a concern, our study was a short-term study, and attrition between the intervention and the posttest was rather large, making effects on learning difficult to find. In theory, well-designed adaptive support will, in the end, have a positive effect on student learning.

APTA also improved the way students used sentence classifiers, in that they chose to use help-related sentence classifiers more often and more accurately. This result suggests that adaptive support can

be used to make existing fixed scaffolds (here, the sentence classifiers) more effective. Sentence classifiers have benefits of their own, but students often fail to use them appropriately (Lazonder, Wilhelm, & Ootes, 2003). Thus, students may not benefit from the classifiers, and any adaptive system that uses sentence classifiers as input to its models (e.g., *GroupLeader*; Israel & Aiken, 2007) may not get accurate information on student interaction. As our system uses simple adaptive prompts to improve student use of sentence classifiers, applying a similar method to other systems may make other interventions more effective.

The qualitative results contributed to the research into how students are motivated by peer tutoring. While most studies have looked broadly at how reward structures increase student accountability (e.g., Fuchs et al., 1997), they have not examined how this mechanism might be working *during* the interaction. The qualitative analysis in this phase supported the conclusions of these experimental manipulations by suggesting that students indeed feel accountable to be good peer tutors to their partners, and that this accountability increases when relevant and public support is given to tutors (i.e., when peer tutor responsibility is primed). With the increase in accountability, students put more effort into constructing help and applying the assistance they received to their help, potentially engaging in more cognitive elaborative processes associated with good help-giving. This analysis suggests that it may not be the *adaptivity* of the support that is creating these results, but the *perceived adaptivity*. In other words, a tighter fixed control that takes the form of random prompts in the chat window may have the same effect as the adaptive support, as long as students perceive the prompts as adaptive and are consequently motivated to feel more accountable for the help they construct.

7.5.5 Implications for Iteration

The next logical step in this program of research would be to tease apart to what extent adaptivity has cognitive benefits (i.e., students can apply the support at the correct time and thus benefit more), and to what extent it has motivational benefits (i.e., students feel accountable for incorporating support they receive because they believe it is adaptive). In order to take this step, it is necessary to improve the adaptivity present in the system to ensure that it is responsive enough to student behavior to be distinguished from random support. *Development 3: Assessment of Help-Giving* (Chapter 8) focuses on taking that step, and *Phase 4: Cognitive and Motivational Benefits of Adaptive Support* (Chapter 9) describes a study where the effects of actual and perceived adaptivity are teased apart.

8 Development 3: Assessment of Help-Giving

8.1 Introduction

After the study described in *Phase 3: Adaptive Help-Giving Support* (Chapter 7), one component of our system that needed to be improved was the accuracy of the assessment of peer tutor actions. While *APTA* was relatively accurate at classifying whether students were giving help or not, with kappas of around 0.7, it was not accurate at classifying conceptual help content, with kappas of around 0.3. Further, the old version of *APTA* depended on the student selection of sentence classifiers to determine whether students gave next-step help or error feedback, and the accuracy of this judgment might be improved by moving to automatic classification. In the iteration of *APTA* described in Chapter 7, I trained a machine learning model on previous study data using *Taghelper* (Rosé et al., 2008). This chapter explores how incorporating domain context and self-classification features might improve the automated classification of peer tutor dialogue. I describe details of the corpus, and then four classification approaches: baseline classification of student dialogue based solely on text features (*B*), baseline classification with additional domain features (*B+D*), baseline classification with additional self-classification features (*B+SC*), and baseline with problem-solving and self-classification features (*B+D+SC*). I discuss the results of comparing the classifiers and their implications. The work in this phase was discussed elsewhere in Walker, Walker, Rummel, and Koedinger (2010).

8.2 Context

Improving the adaptivity of *APTA* involved classifying two aspects of peer tutor dialogue: help type and conceptual help.

> *Help type.* Are peer tutors giving next-step help, error feedback, both, or no help at all? 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 tutors are giving the appropriate kind of help.
>
> *Conceptual content.* Are peer tutors giving help that explains concepts rather than simply stating what to do next? Being able to identify this aspect lets *APTA* know whether tutors are providing enough conceptual help.

I used the corpus drawn from the classroom study described in *Phase 3: Adaptive Help-Giving Support* (Chapter 7), where I compared adaptive and fixed support for peer tutoring. 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 are a total of 84 tutoring sessions from both conditions, consisting of an average of 21.77 tutor lines of dialogue per session ($SD = 10.25$). As described in Chapter 7, two raters coded tutor utterances for help type and conceptual content. 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. First, each help segment was coded for help type by determining whether it consisted of *previous-step help* relating to an action tutees had already taken (e.g., "no need to factor because there is only one g"; kappa = 0.83), and whether it consisted of *next-step help* relating to a future action in the problem (e.g., "how would you get rid of 2*h*?"; kappa = 0.83). 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". Second, each help segment was coded for whether it contained a concept (e.g., "add *ax*" was purely instrumental help, while "add *ax* to cancel out the *–ax*" was conceptual). Kappa for conceptual help was 0.72. 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 coded as conceptual help.

I explored two different approaches for improving the accuracy of dialogue classification: incorporating information about the domain context, and incorporating student self-classifications. First, the domain context of the interaction was used as additional features for a machine learning classifier. This context includes 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 has referred to another student's incorrect steps 10 times over the course of the whole interaction). There is a precedent for this approach: The few adaptive collaborative learning systems that have used domain information have shown gains both in the variety of support that those systems provide and in the effects of support (e.g., Baghaei, Mitrovic, & Irwin, 2007), but they have not applied these models to the classification of collaborative dialogue. However, this approach has been applied successfully in asynchronous collaborative contexts (Wang et al., 2007), and domain features have been successfully used to enhance the ability of automatic classifiers in other fields (Dybowski et al., 2003). In addition to domain context, student self-classifications of their own chat dialogue were used as a potential source of features for improving the accuracy of the machine classifier. It is common in adaptive collaborative learning systems to ask students to classify their own utterances. While these classifications are not always accurate, they may still be relevant for assisting the machine.

8.3 Method

8.3.1 Baseline Classification

I 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. In this particular dataset, *Taghelper* generated

641 features. I used a chi-squared feature selection algorithm to rank the most predictive features, and selected 150 features for help type and 125 features for conceptual content. Then, I used 10-fold cross validation to train a support vector machine classifier for each dimension.

8.3.2 Incorporating Domain Features

I augmented the dialogue features generated by Taghelper with domain context features. After assembling the problem-solving context, text substitution, and history features described below, I again used a chi-squared feature selection algorithm to rank the most predictive features. I used 10-fold cross validation to train a support vector machine classifier for help type and conceptual content.

Problem-Solving Context. In general, features describing the tutee's problem-solving progress may provide information about the type and quality of the help peer tutors tend to give (e.g., peer tutors may be more likely to give error feedback after the tutee has made an error). Thus, I added a total of 10 features for the classifier, created using information from the CTA models of student problem-solving. This information included whether the last step taken on the problem was correct or incorrect, the student's progress in the problem (i.e. the number of correct, incorrect, and total steps taken), and the student's problem-solving momentum (e.g. the number of incorrect steps the student had made in a row).

Text Substitutions. I then added features representing whether peer tutors *referred* to problem-solving elements in their utterances (e.g., "subtract x" refers to a specific problem-solving action). By treating different references to the problem as members of a higher-level category, it was possible to compensate for a lack of training data and enable the classifier to transfer between different units of the problem. More specifically, I extracted a list of problem-related actions from the CTA menu options that tutees were able to select in the unit (e.g., {factor, distribute, add, subtract}), and a list of problem-related variables from the problem-statement (e.g., $x = a + b$ would return $\{x,a,b\}$). I then substituted specific occurrences of a problem-related action or term in the text with general terms (see the "Substituted Text" column in Table 37), and used the new text as input into Taghelper. I also added features that indicated that a substitution had been made ("Action Present" and "Term Present" in Table 36).

Further, by tracking which specific aspects of the problem to which tutee utterances referred, it might be possible to be able to better identify the target of the help given by the peer tutor. Thus, a feature was added representing whether the substituted terms referred to the tutee's last correct step or last incorrect step. For example, in the second row of Table 36, "add x to the left side" sets the Term Present feature to "last-correct", indicating that there is a term in the problem that references the last correct step. Substitutions were also made based on whether peer tutors referenced terms that appeared in the problem-solving hints generated by the cognitive tutor. I created a list of verbs and nouns found in the hints, and then substituted a generic "concept" word for these words (as in the third row of Table 36). An additional feature was added representing whether a concept term had been substituted. Finally, substitution features

were created to indicate whether multiple substitutions of the different types had occurred. The presence of multiple substitutions in an utterance makes it more likely that a reference to the problem actually occurred. This approach emphasized those utterances where multiple substitutions were done, while deemphasizing utterances where only a single substitution took place. Overall, seven text substitution features were added.

Substitution History. Finally, 6 history features were added in an attempt to provide holistic information about the overall quality of the interaction. The history features were based on the numbers of each type of substitution made; features were created for what percent of the peer tutor's total number of utterances referred to a concept, what percent referred to a correct or incorrect action, and what percent referred to a correct or incorrect term. A simple yes/no feature was also included to indicate whether or not a substitution of a specific type was made at any point, under the rationale that somebody who has given a certain kind of help in the past would be more likely to give that kind of help in the future. All history features were updated with each tutor utterance; that is, history features were only computed based on all utterances that had occurred *prior* to the current utterance, so that the algorithm could be applied to a learning situation as it unfolds. History features were based on the substitutions rather than on the machine classifications to avoid being stuck in a state where, for example, because the machine has not yet classified an utterance as conceptual help, it is likely to never classify an utterance as conceptual help.

Table 36. Selected features created from particular tutor utterances. If the tutee's last action was "factor x", and this action was correct, the following are the substitutions that would be made.

Chat Text	Substituted Text	Action Present	Term Present	Term-Concept Present	Action-Term Present
now factor	now action	last-correct	no	no	no
add *x* to the left side	action term to the left side	yes	last-correct	no	yes
isolate the *p*	concept the term	no	yes	yes	no

8.3.3 Adding Self-Classification

In addition to creating domain features, we also added two features that involved students' self-classification of their actions. As introduced in *7.2.1*, before composing an utterance peer tutors were asked to label their utterance as a prompt, error feedback, a hint, an explanation, or a comment. The label selected by the peer tutor, as well as his or her overall use of sentence classifiers, may be predictive of the type of help the peer tutor gave in a particular utterance. The self-classification specified by the tutor and the number of help-related sentence classifiers used by the tutor in total were added as features in the machine classification.

8.4 Results

I hypothesized that both the domain features (*B+D*) and the self-classification (*B+SC*) would lead to an improvement over the baseline classification (*B*), with a classifier containing all three sets of features being the most effective (*B+D+SC*). I compared Cohen's kappa for all classifiers in the Weka Experimenter using 10 repetitions of 10-fold cross-validation (see Table 37; Hall et al., 2009). Kappa was used instead of percent accuracy due to the imbalanced frequency distribution between categories (for example, there was over 10 times more non-conceptual help utterances than conceptual help utterances). Weka uses paired t-tests corrected for dependence between samples to compare classifiers. For help type, *B+D+SC* was significantly better than the *B* classifiers ($p < 0.05$). For conceptual help, only *B+D* was significantly better than baseline ($p < 0.05$). It is encouraging that the help type kappa for *BS+D+SC* approached the kappa we

Table 37. Kappas for the baseline (*B*), baseline plus self-classification (*B+SC*), baseline plus domain (*B+D*), and baseline plus self-classification plus domain feature sets (*B+D+SC*). Kappas are reported for both the help type and conceptual help classifications

	Help Type Kappa		**Conceptual Help Kappa**	
Classifier	M	SD	M	SD
B	.78	.04	.59	.10
B + SC	.78	.04	.60	.10
B + D	.80	.04	.66*	.10
B + D + SC	.81*	.04	.65	.11

achieved for human interrater reliability, and that the conceptual help kappa improved substantially between the *B* and *B+D*.

Examining which features were ranked highly by the chi-squared feature selection algorithm for the *B+D+SC* feature set, we can see that our domain context features consisted of 7 of the top 10 features for the help type classification, and 7 of the top 10 features for the conceptual help classification (see Table 38). In addition, one highly ranked feature for help type was the sentence classifier used, part of the *SC* feature set. Overall, for the help type classifier, only three of the domain context features created were not selected to be part of the machine classifier, and two of these features related to the number of correct steps that had recently been taken by the tutee. It is interesting that while incorrect problem-solving actions were somewhat predictive of the type of help given, correct problem-solving actions were not. This result makes sense, as it is more likely that tutors would refer to a previous incorrect step than to a previous correct step. For the conceptual help classifier, 14 of the 25 conceptual help features were not selected, suggesting that conceptual help classification is less dependent on domain context.

Table 38. The top ten ranked features in chi-squared feature selection for help type and conceptual help for the baseline plus problem-solving plus self-classification feature set.

Rank	Help Type Kappa	Conceptual Help Kappa
1	action present	concept present
2	"action"	"concept"
3	term present	concept & term present
4	"term"	"concept_term"
5	"BOL_action"	line length
6	action & term present	"you_concept"
7	classifier used	percent concepts used
8	"term_EOL"	"how_do"
9	"BOL_undo"	"you"
10	"undo"	"term_by"

8.5 Summary of Technological Contribution

The focus of this section was to increase the accuracy of automated classification of peer tutor utterances in order to improve the ability of an intelligent tutoring system for peer tutoring to provide appropriate support. To do so, I explored the use of domain context features, extracted from individual domain models found in the *CTA*, as input to dialogue classifiers. We also examined whether student self-classifications of their own utterances might improve the machine classification. We found that domain context features in combination with self-classifications significantly improved the accuracy of an automated classifier with respect to help type, but only domain context improved the accuracy of conceptual content classification. This result provides support for *Q2-T2* (see Table 1), suggesting that domain components can improve the assessment of collaborative dialog.

We incorporated three different types of domain context features into the machine classifier: problem-solving context, text substitutions, and substitution history. Of those features, relevant text substitution and substitution history features were highly related to the machine classification for each dimension; for example, substitutions of references to tutee actions were highly predictive of help type, while substitutions of references to concepts were highly predictive of conceptual help. In contrast, self-classifications were less effective; they appeared to augment the results of the help-type classifications, but inhibit the results of the conceptual help classification. This result is not unexpected, as the self-classifications that students made were far more relevant to the help type dimension than to the conceptual help dimension. Perhaps student self-classifications that were more related to whether the utterance included conceptual help would have a positive effect on a conceptual help classifier.

These results make the argument for an emphasis on designing adaptive support for collaboration that is rooted in problem-solving context, suggesting that *Q2-T1* is an important research question ("How can existing and custom components be integrated?"). If domain context information can improve the accuracy of automated collaborative dialogue classification, it would make sense for intelligent tutoring systems for collaborative learning to incorporate domain models. While domain models are difficult to build from scratch, integrating adaptive collaborative learning systems with existing individual intelligent tutoring systems may be a way to leverage sophisticated domain information in order to improve the effectiveness of *ACLS*.

9 Phase 4: Cognitive and Motivational Benefits of Adaptive Support

9.1 Introduction

The results of *Phase 3: Adaptive Help-Giving Support* (Chapter 7*)* suggested that the *perception* that the system is adaptive (*motivational effects*), rather than the actual adaptivity of the system (*cognitive effects*), may have lead peer tutors to feel more accountable for the help they gave and improve its quality. If this hypothesis is true, it has important implications for the development of ACLS, suggesting that it may not be necessary to develop sophisticated adaptive systems. Instead, effort could be spent developing systems that could convincingly pretend to be adaptive.

As a result of the work in *Development 3: Assessment of Help-Giving* (Chapter 8) to improve the machine classification of student help, the ability of APTA to be adaptive was increased, making the comparison between cognitive and motivational effects possible. To explore whether adaptive help-giving assistance has cognitive effects, motivational effects, or both, I conducted a lab study comparing three conditions (see Table 39). In one condition, students received adaptive support and were told it was adaptive (*real adaptive condition*). In another, students were given nonadaptive support, but were still told it was adaptive (*told adaptive condition*). This manipulation was intended to affect student perceptions of support, where students in the *told adaptive condition* would believe support was adaptive, even if it was not. In the third, students were given nonadaptive support and told it was nonadaptive (*real random condition*). Because of a limited number of subjects, I eliminated the fourth condition where support was adaptive but students were told it was nonadaptive, as this condition is unlikely to be deployed in a real situation. In this phase, the adaptive support manipulation involved only the reflective prompts presented to the peer tutor and collaborative hints; these prompts and hints were the most adaptive aspect of the help-giving support, and thus seemed like good candidates for this manipulation.

I hypothesized that the two conditions where students were told support was adaptive would learn more than the condition where students were told support was random, because of an increase in the accountability peer tutors felt to provide good help to tutees. Here, to separate peer tutor and tutee learning, students were randomly assigned to role, and did not change roles. This phase involved a lab rather than a classroom study to achieve more experimental control. However, I tried to maintain ecological validity by running the study at a high school as an after-school program and running multiple pairs at once.

Table 39. Study conditions in *Phase 4.* We manipulated whether the reflective prompts peer tutors received were actually adaptive, and whether we told students they were adaptive. The hypothesized effect is signified by the *.

		Actually adaptive	
		yes	no
Told adaptive	yes	real adaptive*	told adaptive*
	no	n/a	real random

9.2 Design: Moving to a Lab Setting

9.2.1 Interactions: Lab Setting

Few changes were made to the design of student interactions since *Phase 3: Adaptive Help-Giving Support* (Chapter 7), but several changes were made to the overall script design, to reflect that this version of the system was deployed as an after-school lab study. I changed the way roles were distributed to students, the topics covered, and the way the preparation phase was structured and supported.

Role Distribution. Unlike in previous phases, peer tutors and tutees did not switch roles over the course of the script. As the planned evaluation for this phase was in a lab setting., there was the opportunity to compare the effects of being a peer tutor and a tutee without risking the classroom ramifications of negatively impacting students who did not take on the peer tutor role.

Units Covered. Because students were not switching roles, and thus did not need to cover separate topics, only the factoring unit was used in the study.

Preparation Problems. To be able to fairly compare tutor and tutee learning in the study, their exposure to the domain content needed to be kept constant. Thus, peer tutors did not prepare on the problems that they tutored during the collaboration phase, meaning that they may not have received full benefit from the preparation phase. Instead, peer tutors and tutees prepared on the same set of problems, representing the easier problems in the unit (e.g., "$ax + bx = cy - dz$, solve for x"). Peer tutors then helped their partners on a harder set of problems from the unit (e.g., "$ax – dz = bx + cy$, solve for x").

Preparation Reflection. For time considerations, the preparation reflection was removed from the study. The preparation reflection, introduced in *Phase 1*, prompted students to reflect on how a particular problem would be tutored. As there was not a long preparation phase in this study, and students did not prepare on the problems they would be tutoring, the study time was better spent on having the students solve more problems.

9.2.2 Model

The basic model was not changed from *Phase 3*. The implementation of the model and assessment was changed, and I will talk about that in *9.3*.

Table 40. Assessment in the help-giving tutor in *Phase 4*. The assessment of student interaction is based on the cognitive tutor evaluation of tutee actions, student self-classifications of their chat, and machine classifications of their chat.

Input	Component	Description	Update from *Phase 3*
Evaluation of tutee steps	cognitive tutor	Whether last problem-solving step was correct or incorrect.	none
Next step hint	cognitive tutor	The hint the CTA would have given on the next step.	none
Self-labeling of chat	tutee	The label tutees used prior to sending a chat message (representing request, self-explanation, or other).	none
Self-labeling of chat	peer tutor	The label peer tutors used prior to sending a chat message (representing prompt, feedback, hint, explanation, or other).	none
Machine labeling of help type	text classifier	A machine classifier that labeled each peer tutor utterance as no help, next-step help, previous-step help, or both.	4 help types, not "help" or "no help"
Machine labeling of conceptual help	text classifier	A machine classifier that labeled each peer tutor utterance as containing conceptual content or not.	conceptual, not elaborated help

9.2.3 Support: Targeted Reflective Prompts

In order to increase experimental control and focus on the effects of publically presented support on the peer tutor, I modified all prompts directed at the tutee so that they were directed at the peer tutor, and always gave reflective prompts in the chat window rather than as pop-ups. Further, to explore peer tutor perceptions of support, each reflective prompt presented by the computer in the chat window was accompanied by a thumbs up and thumbs down widget, visible only to the peer tutor. Peer tutors were able to click "thumbs up" if they liked the support, click "thumbs down" if they did not like the support, or to ignore both options. Appendix B contains xml code for all the feedback messages peer tutors received.

9.3 Implementation: Improving Adaptivity

9.3.1 New Tutor Component: Help-Giving Tutor

Assessment. To assess peer tutor performance, the updated help-giving tutor used mainly the same inputs as in *Phase 3: Adaptive Help-Giving Support* (Chapter 7), but with modifications to the machine classification based on the work conducted in *Development 3: Assessment of Help-Giving* (Chapter 8). Table 40 has the full list of inputs.

Model Tracing. I updated the implemented model from *Phase 3* to reflect the increased capacity for adaptiveness in *APTA*, creating a 20 rule production system (see Table 41). Because of the improved machine classification, we updated several rules from the old model by using machine classifications instead of self-classifications (rules 1, 2, 3, 7, and 14 of the new model). I was also able to make particular rules more specific; instead of just tracking previous-step help, in rules 8-14 of the new model I separated

Table 41. Production rules in *Phase 4.*

#	skill	type	rule	agent
1	timely	++	IF tutee makes a help request	self
			THEN peer tutor gives help	machine
2	timely	+	IF tutee makes an error	CTA
			THEN peer tutor gives help	machine
3	timely	+	IF tutee self-explains	self
			THEN peer tutor gives help	machine
4	timely	--	IF tutee makes 2 help requests in a row	self
			THEN tutee makes a 3rd help request	self
5	timely	-	IF tutee makes a help request	self
			THEN tutee makes an error	CTA
6	timely	--	IF tutee makes 2 errors in a row	CTA
			THEN tutee makes a third error	CTA
7	timely	-	IF tutee makes a correct step	CTA
			THEN peer tutor gives next step help	machine
8	prompt	++	IF tutee makes an error	CTA
			THEN prompt for explanation	self
9	feedback	++	IF tutee makes an error	CTA
			THEN explain the mistake	self
10	prompt	-	IF tutee makes an error	CTA
			THEN explain the mistake	self
11	prompt	+	IF tutee makes an error	CTA
			AND tutee makes a help request	self
			THEN prompt for explanation	self
12	feedback	+	IF tutee makes an error	CTA
			AND tutee makes a help request	self
			THEN explain the mistake	self
13	prompt	-	IF tutee makes an error	CTA
			AND tutee makes a help request	self
			THEN prompt for explanation	self
14	prompt,	-	IF tutee makes an error	CTA
	feedback		THEN give next step help	machine
15	concepts	+	IF the peer tutor gives help	self
			THEN help is conceptual	machine
16	concepts	-	IF the peer tutor gives next step help	self
			THEN help is not conceptual	machine
17	classifiers	+	IF peer tutor labels help	self
			THEN give help	machine
18	classifiers	+	If peer tutor labels no help	self
			THEN don't give help	machine
19	classifiers	-	If peer tutor labels no help	self
			THEN give help	machine
20	classifiers	-	IF peer tutor labels help	self
			THEN don't give help	machine

the model tracing based on whether students gave prompts or error feedback (compared to rules 7-12 in the *Phase 3* model). Also because of the increased detection abilities of the system, I was able to add rules 19 and 20 of the new model as indicators of the way students were using sentence classifiers. As per the design decisions made after *Phase 3*, the classification of elaborated help was changed to conceptual help, and that is reflected in rules 15 and 16 of the new model (13 and 14 of the old model).

Knowledge Tracing. The knowledge tracing here was very similar to the knowledge tracing in *Phase 3*, with two modifications. Because of the increased adaptivity of the system, I was able to decompose the "targeted" skill into two subskills "prompts" and "error feedback," as the system could now better discern when prompts were being used and when error feedback was being used. Also, because I was now detecting conceptual and not elaborated help, the "elaborated" skill was changed to "conceptual". After tweaking the parameters, we also decided to modify $p(T)$ to stay at a fixed value, to give the system better performance.

Support Construction. Support was triggered in the same manner as in Phase 3.

9.3.2 Integration of Components

There was no change to component integration between *Phase 3* and *4*.

9.3.3 Comparison Conditions

To implement the nonadaptive comparison condition, we gave students pseudo-random prompts that ensured that the timing and content of the prompts that was *noncontingent,* not adaptive. Every time students *would have* received a reflective prompt were they in the adaptive condition, they never receive a prompt in the fixed condition. However, we ensured that they receive a prompt within the next three turns, essentially yoking the fixed 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. Similarly, hints were selected randomly rather than adaptively. All other support across conditions was parallel (all students received adaptive correction support and other forms of adaptive help-giving support).

9.4 Evaluation: Study 4

9.4.1 Experimental Design

The primary goal of this study was to differentiate between the cognitive and motivational effects of adaptive interaction support on student learning by comparing three conditions: 1) Students received adaptive support and were told it was adaptive (*real adaptive condition*), 2) students received fixed support and were told it was adaptive (*told adaptive condition*), and 3) students received fixed support and were

told it was fixed (*real fixed condition*). Based on the results of the previous study, we hypothesized that in the conditions where students perceived support as adaptive, they would be more motivated to construct good help, and thus would learn more than the students who perceived the support as random. In this study students played the role of peer tutor or tutee, and did not switch roles, so that we could determine the effects of each role on student learning. In this section, we describe the quantitative effects of the different support conditions and roles on motivation and learning, and preliminary results of the effects of different conditions on student interactions.

9.4.2 Method

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 completed, the teacher again identified it as a challenging unit for the students, and, in fact, many students did not remember seeing the material before. Unlike in previous studies, this school did not use the *CTA* as part of regular classroom practice. The study was run at the high school, either immediately after school or on Saturdays. 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. Students came with partners that they had chosen, except in the case of 4 students assigned partners by the researchers. 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

Table 42. Study procedure in *Phase 4*.

Week	Day	Time (minutes)	Real Adaptive	Told Adaptive	Real Fixed
1	1	5	Instruction	Instruction	Instruction
1	1	20	Pretest	Pretest	Pretest
1	1	20	Preparation	Preparation	Preparation
1	1	30	Collaboration (*told adaptive, with adaptive support*)	Collaboration (*told adaptive, with fixed support*)	Collaboration (*told fixed, with fixed support*)
1	1	10	Motivation Survey	Motivation Survey	Motivation Survey
1	1	30	Collaboration	Collaboration	Collaboration
1	1	20	Posttest	Posttest	Posttest
1	1	25	Unscripted Collaboration	Unscripted Collaboration	Unscripted Collaboration
1	1	5	Debriefing	Debriefing	Debriefing

students who worked alone over the course of the session. Thus, a total of 122 students were included in the majority of the analyses (40 in the *real fixed* condition, 44 in the *told adaptive* condition, and 38 in the *real adaptive* condition). For the motivation analysis, students who did not respond to one or more questions on the relevant surveys were excluded from those particular analyses.

Procedure. This study took place over the course of 3 hours (see Table 42). Students received a brief 5 minute introduction to the study, and then took a 20 minute pretest that consisted of a 10 minute conceptual component and a 10 minute procedural component. Next, students spent 20 minutes in a preparation phase, solving problems individually using the CTA. All students worked on Sections 1 and 2 of the factoring problem set in the literal equation solving unit, which consisted of problems where the variable terms were on the same side of the equation. Students then spent 30 minutes in the tutoring phase, with one student tutoring another student on Sections 3 and 4 of the factoring problem set, which consisted of problems where the variable terms were on both sides of the equation. Students took up to 10 minutes to answer several survey questions on their motivational state, and then spent another 30 minutes in the tutoring phase. At this point, students took a 15 minute break, and then took a 20 minute domain posttest, again consisting of a 10 minute conceptual component and 10 minute procedural component. Students concluded the study by tutoring without support for 25 minutes, and answering some demographic questions.

In the tutoring phase, we varied whether students received adaptive support or not and whether they thought it was adaptive or not. The fixed support was implemented as described in *9.3.3.* Prior to the tutoring phase, we gave students instructions that told them that the support was either adaptive or fixed. 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 fixed 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 the tutoring system, they were given further instruction, including directions to indicate how they felt about the reflective prompts using thumbs up and thumbs down widgets (as described in *9.2.3*). 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 fixed* 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 posttests, 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. Appendix A contains both forms of the domain learning tests.

We further assessed student motivational state. We gave students five items asking them about how adaptive they thought the system was (e.g., for the 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"). We also assessed how positively students perceived themselves in the interaction (e.g., for the tutor: "I think I was a good tutor"), and how positively they perceived their partner (e.g., for the tutee: "I think my partner learned a lot from being a tutor"). Finally, we adapted individual learning orientation questionnaires (Elliot & McGregor, 2001; Finney, Pieper, & Barron, 2004) to assess peer tutor mastery and performance goals for being a good tutor (e.g., "While tutoring, I was worried that I might not learn enough about tutoring", "While tutoring, my goal was to show my partner I was a good tutor"), and tutee mastery and performance goals for helping his or her partner be a good tutor (e.g., "While being tutored, I wanted my partner to understand how to tutor", "While being tutored, it was important for me that my partner look like a good tutor").

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. To analyze this interaction data, several additional steps needed to be performed. While these steps also were necessary for the studies in *Phase 2: Adaptive Correction Support* (Chapter 5) and *Phase 3: Adaptive Interaction Support* (Chapter 7), I describe them here because they have not yet been completed for this phase, and thus all presented interaction results should be interpreted as preliminary. First, the data needs to be converted from raw data logs into a format suitable for analysis. In the raw data logs, following DataShop logging protocols (Koedinger et al., in press), each tool and tutor component logs their actions separately. For example, a peer tutor chat action is logged independently from the interaction tutor's response to the chat action. Custom code needs to be written to convert these messages into an analyzable format (in this case, an excel worksheet that aggregates all information about each user action into one row), and this step has been taken on this data set.

Next, the formatted data needs to be carefully checked for consistency with the raw data logs, and cleaned to remove any logging errors. In previous studies, logging errors have included both technical problems, such as missing or duplicated logs resulting from network issues, and practical problems, such as incidents where a student leaves his or her computer and is replaced by a teacher. When problems are found, new code needs to be written to further process the log data to make it an accurate representation of the student interaction. This step has not yet been taken on the data, and thus the interaction results presented may not be an accurate representation of what actually occurred. With that caveat, it is currently clear from the data logs that there were fewer technical and practical problems in this study than in previous studies, perhaps as a result of the increased experimental control in this study. Finally, student dialogue needs to be human coded along several dimensions, both to verify the accuracy of the machine

coding, and for the purposes of data analysis. The initial plan for this data would be to code student dialogue along the same dimensions as the machine classifier, and I intend to take this step in the future. In summary, while the raw data logs have been converted into an analyzable format, they have not yet been checked for consistency with the raw logs, cleaned, or been human coded. Thus, all interaction results presented in the following section should be interpreted as preliminary, and may not reflect what actually occurred.

9.4.3 Results

Learning Outcomes. For reliabilty purposes, the learning outcomes are analyzed by overall posttest score (*Cronbach's alpha* = 0.652) instead of separately by the conceptual test (*Cronbach's alpha* = 0.573) and procedural test (Cronbach's alpha = 0.554). 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[1,115]=120.43, $p < 0.001$; see Table 43 for means). There was a significant effect of condition on posttest (F[2,115] = 4.20, $p = 0.017$, eta^2=0.068), indicating that the adaptiveness of support had a positive effect on student posttest performance. A planned comparison of the effects of receiving actual adaptive support (collapsed across role) revealed that it indeed had a significant effect (F[1,115] = 7.47, p=0.007), while a planned comparison of the effects of receiving support that students were told was adaptive (collapsed across role) revealed that this manipulation did not have a significant effect (F[1,115]=0.393, $p = 0.532$).

Interestingly, while the effect of role on posttest was not significant (F[1,115] = 0.751, $p = 0.338$), there was a marginally significant interaction effect between condition and role (F[1,115] = 3.334, p = 0.039, $eta^2 = 0.055$). Applying the planned comparisons to the interaction effect revealed that while the effects of real adaptivity did not differ across the two roles (F[1,115] = 2.660, $p = 0.106$), the effects of told adaptivity had differential effects on peer tutors and tutees (F[1,115] = 6.561, $p = 0.012$). Inspecting the data of student learning across role and condition (see Table 43) reveals that peer tutors benefit more from the perceived adaptive condition than the real fixed condition, but tutees benefit far more from the real fixed condition than the perceived adaptive condition. It is possible that the perception of adaptivity does indeed have an effect on *peer tutor* motivation, but that the deception impedes the tutoring abilities of the peer tutor, which leads to less tutee learning in the real fixed condition.

Table 43. Pretest and posttest scores in *Phase 4*.

	Tutee Scores				**Tutor Scores**			
	Pretest		**Posttest**		**Pretest**		**Posttest**	
Condition	M	SD	M	SD	M	SD	M	SD
Real Fixed	0.23	0.16	0.33	0.21	0.29	0.15	0.28	0.18
Told Adaptive	0.28	0.18	0.29	0.14	0.24	0.12	0.27	0.16
Real Adaptive	0.28	0.15	0.36	0.21	0.27	0.16	0.39	0.17

Motivation Outcomes. As a manipulation check, I evaluated how adaptive and effective students thought the system was, using the average of the five items relating to student perceptions of the system (see Table 44, row 1). In a two-way (condition x role) ANOVA, there were no significant effects of condition on perceived adaptivity ($F[2, 96] = 1.046$, $p = 0.355$), no significant effects of role ($F[1,96] = 0.00$, $p = 0.992$), and no interaction ($F[2,96] = 1.741$, $p = 0.181$). I will address this discrepancy in the discussion.

Table 44. Motivational effects in *Phase 4*.

	Real Fixed		**Told Adaptive**		**Real Adaptive**	
	Tutor	Tutee	Tutor	Tutee	Tutor	Tutee
perceived adaptivity	4.75 (1.60)	4.79 (1.56)	5.13 (1.01)	4.54 (1.30)	4.92 (1.32)	5.48 (0.79)
positive feelings	4.53 (1.28)	5.35 (1.37)	4.68 (1.86)	4.90 (1.39)	54.3 (1.03)	6.14 (0.67)
mastery orientation	4.93 (1.10)	5.15 (1.24)	4.94 (1.05)	5.10 (1.17)	5.28 (1.10)	5.46 (1.52)

The manipulations did appear to have other positive motivational effects. We assessed how positively students felt about their and their partner's performance during the tutoring, using a two-way (condition x role) ANOVA (see Table 44, row 2). We found that condition significantly affected student positive feelings ($F[2,102] = 5.58$, $p = 0.005$), as did role ($F[1,102] = 5.10$, $p = 0.026$). There was no significant interaction ($F[2,102] = 0.542$, $p = 0.583$). Similarly, condition affected student desire to be good peer tutors (see Table 44, row 3, for marginal means); controlling for student overall motivation, condition had a significant effect on mastery motivation ($F[1,94] = 4.73$, $p = 0.01$), and role had a marginal effect ($F[2,94] = 0.629$, $p = 0.06$). There was no interaction ($F[2,94] = 0.046$, $p = 0.955$).

Problem-Solving Progress. I next looked at the preliminary problem-solving progress variables generated from the log data to further explore the link between condition and learning. As in *Phase 3*, I first examined whether there were systematic high-level differences between the three conditions in student problem-solving and dialogue. While differences in the problem-solving context might indicate the relative effectiveness of the experimental condition, they may also reveal other confounds affecting the experimental manipulation, and thus it is important to check for their existence. I used a MANOVA with condition as the independent variable to evaluate the differences between conditions for the following variables: the number of problems seen by the students, the number of errors tutees made, and the total amount of chat engaged in by the students. This analysis was done by dyad, rather than by individual, as the dependent measures were the same for each dyad member. The analysis revealed no significant differences between conditions (Pillai's Trace = 0.070, $F[2,58] = .691$, $p = 0.738$). Table 45 displays the means and standard deviations for each dependent variable. Problems seen were not significantly different across conditions ($F[2,58]=0.306$, $p=0.706$), and neither were errors made by tutees ($F[2,58]=0.350$, $p=0.603$) or lines of dialogue ($F[2,58]=1.54$, $p=0.223$).

Table 45. Differences in peer tutoring context across conditions.

	Real Fixed		**Told Adaptive**		**Real Adaptive**	
Context variables	M	SD	M	SD	M	SD
problems seen	6.20	1.99	6.32	2.23	6.74	2.44
errors made	16.60	10.19	19.00	9.65	17.32	8.35
lines of dialogue	98.05	40.01	93.55	41.54	77.95	26.80

In *Phase 2*, the number of errors viewed by the peer tutor was correlated with peer tutor learning gains. To test if this relationship held true in this study, I conducted a one-way ANOVA with condition as an independent variable and errors as a covariate, but included the condition*errors interaction term. I used peer tutor gain score as the dependent variable. I found that while condition did not significantly affect gain (F[2,55] = 0.998, p = 0.375), and errors was not significantly predictive of gain (F[1,55] = 0.032, p = 0.859), the condition*errors interaction was significantly predictive of gain (F[2,55] = 4.70, p = 0.013), where more errors viewed was related to more learning in the real adaptive condition but less learning in the real fixed condition. This result suggests that viewing errors was positively related to peer tutor learning in the real adaptive condition (r[1,20]=0.365, p = 0.125), negatively related to peer tutor learning in the real fixed condition (r[1,20] = -0.481, p = 0.032), and had no effect in the told adaptive condition (r[1,22] = 0.087, p = 0.701).

Helping Behavior. I then looked at the machine classification of their dialogue as a preliminary assessment of student helping behavior. One of the main hypotheses of this work is that the better the peer tutor help, the more both peer tutors and tutees will benefit. As per the model of good peer tutoring implemented in *APTA*, I was particularly interested in two aspects of the peer tutor dialogue automatically labeled by the machine classifier: Whether students gave conceptual help and whether they gave help that targeted previous tutee steps. First, we examined whether these dimensions were predictive of learning using a MANOVA with tutor and tutee gain scores as the dependent variables, condition as the independent variable, and machine classification of conceptual help and error feedback as covariates. All three predictor variables were significant or marginally significant in the whole model (see Table 46). Looking separately at the effects of tutor and tutee learning, we see that conceptual help was significantly predictive of both tutor and tutee learning, but error feedback was more predictive of tutee learning. In contrast, machine classification of help in general was not predictive of tutor or tutee learning. These results suggest that our machine classifiers can predict the likelihood that both tutees and tutors are going to learn.

Table 46. Relationship between student interaction and learning.

	Overall Model		**Tutor Learning**		**Tutee Learning**	
Predictor variables	F	p	F	p	F	p
condition	4.555	0.002	4.212	0.020	4.149	0.021
conceptual content	7.178	0.002	4.984	0.030	5.572	0.022
error feedback	3.065	0.055	1.224	0.273	3.400	0.070

We then examined whether condition had an effect on any of the covariates (see Table 47). Using a one way ANOVA to look at the effects of condition on machine-classified conceptual help, we found that condition did not significantly affect conceptual help (F[2,58]=1.782, p = 0.177). Similar results were found for the effect of condition on error feedback (F[2,58]=0.910, p=0.408). Thus, while relevant interaction variables were predictive of learning (as we expected), condition did not appear to have an effect on those variables.

Table 47. Differences in interaction variables between condition.

	Real Fixed		**Told Adaptive**		**Real Adaptive**	
Dependent variables	M	SD	M	SD	M	SD
conceptual content	4.55	3.72	2.64	2.87	4.47	4.54
error feedback	2.55	2.24	2.54	2.24	3.36	2.11

Use of Assistance. Another important element of student interaction is the amount of reflective prompts they received from the computer, and how they reacted to the prompts. If students received different numbers of prompts in different conditions, this would represent an experimental confound that could affect the results of our study. A one-way ANOVA revealed there was no significant difference in the number of prompts received by students between conditions ($F[2,58] = 1.070$, $p = 0.350$; see Table 48). When presented with a reflective prompt, students could choose to engage with it by indicating in the interface that they liked it or disliked it. While the percentage of prompts liked out of total prompts received was not significantly different between conditions ($F[2,55] = 1.234$, $p = 0.299$), the percentage of prompts disliked out of total prompts received was significantly different between conditions ($F[2,55] = 3.603$, $p = 0.034$), where students in the real adaptive condition disliked more prompts than students in the other two conditions. The percent of prompts liked was positively correlated with learning ($r(53) = 0.331$, $p = 0.011$) as was the percent of prompts disliked ($r(53) = 0.237$, $p = 0.071$), suggesting that students who engaged with the prompts, either positively or negatively, benefitted more from the peer tutoring activity. However, the percent of problems disliked was negatively correlated with our outcome measure of

perceived adaptivity ($r(51) = -0.353$, $p = 0.009$), suggesting that students who disliked more prompts thought the system was less adaptive (but not enough to make the adaptive system seem non-adaptive).

Table 48. Student responses to reflective prompts across condition.

	Real Fixed		**Told Adaptive**		**Real Adaptive**	
Context variables	M	SD	M	SD	M	SD
prompts received	13.55	8.41	17.95	9.36	16.47	11.57
prompts liked	3.05	2.40	3.50	5.27	4.33	3.52
prompts disliked	1.75	2.53	1.86	2.64	4.22	6.16

9.5 Outlook & Discussion

In this chapter, I described a second iteration on the adaptive *help-giving* support outlined in Chapter 7, including its design (*9.2*) and implementation (*9.3*). I then discussed a study where I compared reflective prompts related to the adaptive help-giving support to random prompts that students were told was adaptive, and random prompts that students were told was random (*9.4*). The results of the study indicated that the real adaptive prompts had positive effects on student learning, compared to the two fixed prompt conditions. There were no substantial design contributions made in this chapter, and the technological contribution was related to the increased adaptivity of *APTA*, which drew heavily from the contributions made in *Development 3: Assessment of Help-Giving* (Chapter 8). Thus, in this subsection, I focus the discussion on the learning sciences contributions of the work.

The study conducted in this phase was the first study in this dissertation work to demonstrate positive effects of adaptive support on learning compared to fixed support (addressing *Q1-L2*; "What are the effects of ACLS on student learning?"). The results suggested that real adaptivity had a *cognitive* benefit, where peer tutors who received assistance at moments they needed it gained more between the pretest and posttest, and these gains were also transferred to their partners. Real adaptivity also had motivational benefits, in that students who received adaptive support had more positive feelings about the collaboration and were more mastery oriented in their peer tutoring – perhaps because they felt their collaboration improving as a result of the support. These results encourage the continuation of a program of research relating to adaptive support for collaboration.

I had hypothesized that by simply telling students that support was adaptive, even when it was not, students would feel more accountable for their collaboration and learn more from the collaboration. While the data suggested that peer tutors benefited from this manipulation (although not as much as when the support was actually adaptive), tutees fared the worst in this condition, potentially suggesting that the deception had a negative effect on the help they received. An examination into the interaction data should shed more light into the effects of the motivation manipulation on students. It should also be noticed that during the manipulation check, when students were asked how adaptive they found the system, there were

no significant differences between responses in the three conditions. This confusing result would suggest that either our manipulation was not strong enough, or the manipulation check did not get at student actual perceptions of adaptivity. It is interesting that students in the actual adaptive condition did not appear to perceive the system as more adaptive, but did experience beneficial effects of adaptivity.

It is not yet clear from the preliminary interaction analysis which aspects of student interaction mediated the relationship between condition and learning. It is encouraging that important components of our model of peer tutor help-giving such as conceptual help and error feedback were indeed predictive of student learning across conditions. Further, it was the machine classification of these variables that was predictive of student learning, suggesting that there is potential to use machine classification of text to assess collaboration and learning outcomes more generally. However, there were no significant differences between conditions on these predictor variables, potentially suggesting that here, unlike in *Phase 3: Adaptive Help-Giving Support* (Chapter 7), condition did not have a direct effect on the quality of student interaction. It is also possible that a human coding of these variables will be more accurate than the machine coding, and yield between-condition differences. Finally, it appeared that students in the real adaptive condition engaged more with the prompts they received in the chat windows, "disliking" them more frequently than in the other two conditions. The percent feedback disliked was positively correlated with learning outcomes, suggesting that this engagement was leading students to reflect more on the prompts. This potential motivation effect will be explored more in future analysis.

Unlike *Phase 3*, *Phase 4* consisted of a tight manipulation, simply varying the actual adaptivity and told adaptivity of a single type of assistance incorporated in the system: the reflective prompts. Additionally, while *Phase 4* took place at a school, it took place outside of school hours, and students were paid for their participation. It is possible that the reason learning differences were found in this study was the tighter control and the fact that students were more motivated to engage with the system. Thus, the study lacks the ecological validity of the previous studies surveyed in the dissertation, but does imply that there are benefits to adaptive support worth further exploration. The next step in this program of research will be to do a more thorough analysis of the interaction logs from the study to determine what mechanisms are driving the effects of adaptivity on learning, and how the effects of condition differed depending on role.

10 Outlook & General Discussion

10.1 Introduction

This dissertation explored the design, implementation, and effects on interaction and learning of adaptive collaborative learning support (ACLS; see Table 49). It addressed two broad research questions: (1) Where and how can intelligent tutoring approaches be applied to the development of ACLS, and (2) Are there benefits to using existing domain models developed as part of individual intelligent tutoring systems in ACLS? In each of Chapters 3,5,7, and 9, I discussed a full design-implementation-evaluation phase related to ACLS development, with each phase building on the results of the previous phase. In each of Chapters 4,6, and 8, I discussed a single research contribution that made the next phase possible.

I began in *Phase 1* by implementing a learning environment for peer tutoring as an addition to a successful intelligent tutoring system, the Cognitive Tutor Algebra (*CTA*). I conducted a study exploring where adaptive support to peer tutoring might be most useful in this context, and discovered that peer tutors needed the most support in providing correct help to their tutees. In *Development 1*, I discussed *CTRL,* an architecture developed to facilitate the extension of individual intelligent tutoring systems for collaborative learning and the integration of existing intelligent tutoring components with custom-build components. I then added adaptive support for peer tutors in giving correct help. I discovered that while peer tutors benefitted from reflecting on their partner's errors, they needed additional support in giving tutees conceptual help (*Phase 2*). I used human-computer interaction design methods to explore potential designs for adaptive support for collaboration that departed from individual learning paradigms, and found that good support would likely prime students' feelings of accountability to their partner, increase their feelings of being good tutors, and be relevant to their tutoring choices (*Development 2*). Using the results from *Development 2*, I designed and implemented adaptive help-giving support for peer tutors, and found effects of this support on interaction in a classroom study compared to an ecologically valid fixed support control (*Phase 3*). Qualitative analysis suggested that this result might have been due to students' increased feelings of accountability from the adaptive support. I improved the adaptivity of the support in the system by improving the automated assessment of peer tutor actions (*Development 3*). Then, in a lab study, I compared the effects of actually adaptive support to two fixed support controls: One in which students were told the support was adaptive, and one in which students were told the support was fixed (*Phase 4*). Results suggested that the actually adaptive support had positive effects on student learning and motivation.

I have found that intelligent tutoring techniques can be applied to the development of ACLS, but modifications need to be made to account for the ill-defined nature of the collaborative setting and the social context of the student interaction. Intelligent tutoring components are an important part of this process, and can improve the assessment of peer tutor actions, the modeling of their behaviors, the feedback that is given, and the analysis of student log data. Overall, it appears that ACLS is a promising research direction for improving student collaboration quality and domain learning. In this discussion, I examine the design implications (*10.2*), technological contributions (*10.3*), and empirical findings *(10.4*) of

the work in more detail. I then explore the potential impact of my methodology for the development of *ACLS* and future opportunities in the field (*10.5*).

Table 49. Two research questions related to adaptive collaborative learning support, and the findings of this dissertation relating to design, technology, and learning sciences implications of the support.

#	Research Question	Design	Technology	Learning Sciences
Q1	Where and how can ITS approaches be applied to the development of ACLS?	Production-based modeling relevant, modified for flexibility (*Q1-D1*). Traditional ITS support valid, but other approaches may increase accountability, efficacy, and perceived relevance (*Q1-D2*).	With some parameter adjustments, collaborative skills can be knowledge traced, and this approach appears useful (*Q1-T1*).	Adaptive support can improve student interactions over traditional fixed support (*Q1-L1*). Adaptive support can improve student learning over fixed prompts (*Q1-L2*).
Q2	Are there benefits to using existing domain models in ACLS?	Domain information improves collaborative models and specificity of feedback (*Q2-D1*).	Existing and custom components be integrated using a centralized architecture (*Q2-T1*). Domain features improve assessment of help type and conceptual content in chat (*Q2-T2*).	Intelligent tutoring data logs augment analysis of collaborative data by allowing dialogue to be linked to problem-solving steps and evaluation information (*Q2-L1*).

10.2 Design Implications

10.2.1 ITS Approaches to Modeling and ACLS (Q1-D1)

I applied a cognitive tutoring approach to the initial design of the correction and help-giving models. Each model represented a complete peer tutor-tutee interaction triggered by a tutee problem-solving step, with the assumption that peer tutors would address each tutee problem-solving step separately. In the correction tutor model, described in *Phase 2: Adaptive Correction Support* (Chapter 5), basic tutorial actions were represented, involving marking tutee problem-solving steps and adjusting tutee skills. In the help-giving tutor model, described in *Phase 3: Adaptive Help-Giving Support* (Chapter 7), the focus was the specifics of tutor-tutee discussion, in particular when and how peer tutors should give help. In both models, there

were idealized procedures for peer tutors to follow that were intended to maximize their likelihood that peer tutors and tutees would engage in beneficial cognitive processes.

When the model was implemented, there were areas where typical cognitive tutor approaches were relaxed in favor of more flexible approaches. One theme employed was the idea of abstraction, where the production rules used in the model represented a subset of possible peer tutor actions, rather than covering the whole space. This technique created a less deterministic model than typical cognitive tutor models, and in fact was similar to constraint-based modeling (Mitrovic & Weerasinghe, 2009): if peer tutors were operating outside the production rules implemented, their actions were assumed to be correct, not incorrect. This flexibility also carried through to the incorporation of peer tutor judgments in the model. In the correction and help-giving tutor models, I explicitly allowed peer tutors to make judgments about the kinds of help to give tutees, without specifying in the model what those judgments should be. Further, this approach employed abstraction, in that I did not model the whole domain, but instead supported peer tutors in the choices they made. This judgment-based approach was reified in the implementation of collaborative knowledge tracing, also discussed in *10.2.3*, where peer tutors were given the benefit of the doubt with respect to any given collaborative action unless they showed a pattern of ineffective collaboration.

One goal of this work was to model aspects of collaboration and provide support without overly constraining the collaboration, and in many respects, the efforts in this area were a success. Much of the emphasis in the computer-supported collaborative learning literature on supporting collaboration is on providing students with scaffolds and tools to achieve collaborative goals, and intelligent tutoring approaches are criticized for forcing students down particular pre-defined paths, which would be inappropriate in a collaborative setting. This dissertation work has avoided these pitfalls by providing support that is adaptively tailored to the current situation, but at moments when peer tutors need and are looking for support, rather than at moments where support would limit what peer tutors are trying to do. For example, correction support comes at moments that peer tutors can predict (when peer tutors have marked a problem-step). Peer tutors do not have to mark problem steps, and thus when they do, the support they receive is perceived as relevant and highly useful to the current situation. While there is further work to be done in this area, overall the collaborative models developed as part of this dissertation were successful.

10.2.2 ITS Approaches to Support and ACLS (Q1-D2)

As most of the work in *ACLS* has used direct and explicit paradigms of feedback drawn from individual intelligent tutoring systems, one of the contributions of this dissertation has been to systematically explore other paradigms of support that might be more appropriate for collaborative scenarios. Not only did *APTA* vary whether support targeted domain skills or interaction skills, but it also varied whether support was presented to the peer tutor or to both students, and whether support explicitly told students what to do or was more implicit. This work, beginning in *Phase 2* with peer-mediated feedback, expanding in

Development 2, and then carrying through in *Phase 3* and *Phase 4*, provided more information about the kinds of adaptive support that work in collaborative scenarios.

There were two clear successes in assistance design in this dissertation. Early in the process, it became apparent that making domain support peer-mediated was effective. This technique likely activated peer tutor feelings of efficacy by giving them information that tutees did not have. It also appeared to increase peer tutor perceptions of relevance, as peer tutors would often communicate the assistance they received to tutees. This pattern of results occurred in *Phase 2* and *Phase 3*. On the other hand, it appeared to be useful to present interaction feedback in the form of reflective prompts in the chat window. This public chat support may have engaged peer tutor's accountability, leading to the beneficial results of the adaptive help-giving support in *Phase 3* and *Phase 4.* This combination of peer-mediated domain support and public reflective prompts formed an effective assistance combination in *APTA*.

There were other assistance designs attempted that were not so successful. The conceptual resources provided to students in *Phase 3* and *Phase 4* did not appear to be used, despite being linked to student choices. This was probably due to two interacting factors: The support provided in the resources was implicit, in that it did not give peer tutors explicit direction, and contained a lot of text, which was probably difficult for students to parse. In this case, it may have been better for the assistance to be more similar to individual intelligent tutoring support, to make it easier for students to use. Another conclusion to draw from the support results was the importance of managing student cognitive load. The peer tutoring task is complex, and by giving students multiple kinds of assistance, they needed to pay attention to elements of the task not directly related to the problem-solving, potentially interfering with their learning. Future iterations should do more work in integrating assistance organically with the collaboration, keeping the activity simple enough to facilitate student learning.

10.2.3 The Role of Domain Context in Interaction Models and Support (Q2-D1)

The final contribution this dissertation makes to the design of *ACLS* is an examination of the benefits of including domain context in models of collaboration and in support given to the students. By using domain context in the correction and help-giving models, I was able to represent peer tutoring behaviors that have been identified in educational psychology literature as beneficial. For example, giving correct help is a necessary skill of a good peer tutor (Webb, 1989). In *APTA*'s modeling of peer tutor correction actions, it gets information about whether the peer tutor help is correct using the *CTA* domain model. It is difficult to imagine a way to get that information without the use of a domain model. Domain information also plays a crucial role in the help-giving model, and is used in 9 of the 16 rules. By identifying tutee errors, it is possible to identify impasses they may be facing, which are critical in helping tutees benefit from being tutored (VanLehn et al., 2003). It is true that *APTA* only models a subset of tutee-tutor interactions, and there are other subsets that are potentially beneficial for learning from tutoring that would not require domain context. However, given that domain context plays a critical role in learning in the peer tutoring literature, using the *CTA* components as input to the collaborative models is likely to be beneficial.

It is less clear that the explicit use of domain information in the *support* delivered by *APTA* was beneficial. On the one hand, in *Phase 2* the domain support that prevented students from advancing to the next problem when they were not done with the current one was correlated with peer tutoring learning. Additionally, it was evident in *Phase 3* that peer tutors used the correction support to give more correct help to tutees, as per the design of the system. On the other hand, the system was designed for peer tutors to conceptually elaborate on the domain support when communicating it to their partner, and when peer tutors asked for a hint, we integrated domain support with collaborative prompts, one of the design ideas generated in *Development 2*. The results of *Phase 3* indicated that while peer tutors were enthusiastic about communicating the domain portion of the hints to their partner, they communicated it in a very instrumental, unelaborated way. It is possible that domain information has to be used in a different way in support in order to get the desired result. For example, perhaps it should only be presented to peer tutors in full when students lack the knowledge to move forward with the problem. More exploration is needed in this area.

10.3 Technological Contributions

10.3.1 Adapting Intelligent Tutoring Methodology to Collaborative Activities (Q1-T1)

Applying Bayesian knowledge tracing to collaborative skills is a technical contribution that addresses how intelligent tutoring methodology applies to supporting collaboration. To my knowledge, no other adaptive collaborative learning system has knowledge traced collaborative skills (see Soller et al., 2005, for review), so the deployment of this method in *APTA* is a proof of concept of the viability of this approach. In the collaborative interaction, *APTA* begins by assuming students know the skill rather than assuming that they do not. *APTA* also inflates two parameters to reflect uncertainty in the computer assessment of the interaction state: the probabilities of students taking an effective path when they have not mastered a skill ($p[G]$) or taking an ineffective path if they have mastered a skill ($p[S]$). These modifications are appropriate for a collaborative setting because they give students the benefit of the doubt in the collaborative situation, and thus give them more flexibility in their behaviors than traditional intelligent tutoring approaches. However, this approach has not yet been validated, and could probably benefit from estimating parameter values contextually, rather than using theory (see Baker, Corbett, and Aleven, 2008). This contextual estimation and subsequent validation are important next steps.

Another facet to this technical contribution is the use of the skill assessments as triggers for the support presented to students. By linking the support given to the more persistent assessment of student skills, rather than to the problem state (in terms of either immediate feedback or solution-based feedback), it became possible to deliver support appropriate to student understanding levels. This concept seems particularly relevant to collaboration, where students need a lot of support as poor collaborators, but a lot of freedom as skilled collaborators. While I did not contextually estimate skill parameters, as discussed in the previous section, I did iterate on parameter estimates during piloting, until the timing and content of reflective prompts (an extension of the skill estimates) were tuned to what was occurring in the

collaborative interaction. While the initial *Phase 4* learning results suggested that this approach was successful, a more thorough interaction analysis will illuminate whether the skill assessments led to appropriate support, and whether they were good indicators of actual student skill levels.

10.3.2 Integration of Existing and Custom Components (Q2-T1)

In *Development 1: The Collaborative Tutoring Research Lab* (Chapter 4), I discussed *CTRL,* a framework that supports the integration of pre-existing and custom-built components, with a particular focus on tutoring components. Using *CTRL*, I combined the pre-existing *CTA* domain model with models of correction and help-giving behavior in order to support peer tutors in giving more correct (*Phase 2*) and higher quality help (*Phase 3* and *Phase 4*). Without the creation of *CTRL,* it would not have been possible to investigate the design, technical, and learning sciences research questions surrounding *Q2* ("Are there benefits to using existing domain models developed as part of individual intelligent tutoring systems in ACLS?").

There are difficulties to relying heavily on existing tutoring systems for components, because it may be necessary to refactor the components or deal with legacy code that is difficult to appropriate for new purposes. However, in multiple iterations of adaptive support, we leveraged *CTA* logging protocols, interface components, and cognitive models, which would have been time-consuming to reconstruct from scratch. These components made it possible to develop a classroom-functional adaptive collaborative learning system, which is currently a rarity. Another concern with relying too much on existing components was that it might overly constrain the design of adaptive support interventions. It is true that in *Phase 2*, considering the full design space of adaptive collaborative learning support, our system did not depart very much from the current functionality of the CTA. It substituted peer tutoring for cognitive tutoring and collaborative domain support for individual domain support, but did not explore collaborative scenarios that did not involve tutoring or forms of interaction support other than collaborative domain support. However, in *Phase 3* and *Phase 4*, we extended *APTA* to provide adaptive help-giving support. This extension was a much clearer departure from the original functionality of the *CTA*, but used *CTA* components as a means of incorporating the domain context into the help-giving models and support. Further, using *CTRL*, it will eventually be possible to apply our domain-general collaborative components to provide collaborative tutoring for other tasks with pre-existing domain models. As a final caveat, in discussing *CTRL* and the contribution to this research question, it is important to differentiate between *CTRL* as a framework and the implementation of the framework presented in this dissertation. The implementation was very dependent both on *CTA* components and on the *RMI* networking protocols for facilitating the collaboration, and this is certainly a limitation of the work. However, the conceptual framework outlined by *CTRL* relates to the distinction between tool and tutor components, describes how messages are passed between them, and describes a components can be easily added or removed. It is that conceptual work that is generalizable to a variety of different scenarios, and could potentially contribute to others' research on ACLS.

CTRL, the collaborative tutoring research lab, is an initial step toward supporting research into complex forms of adaptive assistance for collaborative learning. There have generally between two types of work in this area: Research that attempts to understand from an educational psychology perspective whether and how adaptive assistance can be effective to promote collaborative learning, and research that attempts to understand from a technological perspective how to construct models of collaboration and provide automated adaptive assistance. In the first case, educational psychologists often lack the technological tools required to implement adaptive systems, and thus conduct wizard-of-oz studies or work with programmers to implement technologically less than optimal interventions. On the other hand, technologists focus their energies on determining how to create complex systems, but the output is often a research prototype that is not generally evaluated to determine its effect on student collaboration and learning. What this dissertation offers is a way to bridge the gap between the two approaches, making it easier to move from implementing adaptive systems to evaluating them, and iterate upon existing adaptive systems to improve the quality of the support that they can provide. Such a bridge is necessary in order to create adaptive systems that can have a real impact on classrooms; it does not matter if impressive adaptive systems are being developed if they do not have a positive effect on collaboration and learning, and psychology experiments may develop a restricted theory of adaptive assistance if they only experiment with suboptimal, low-tech solutions. It is the hope that the structure of CTRL, and in particular its integration framework, facilitates more complex forms of support by leveraging domain-specific models, a more controlled evaluation by allowing the construction of comparison conditions using pre-existing components, and iteration on the development of adaptive support.

10.3.3 Using Domain Components to Improve Assessment (Q2-T2)

This dissertation also makes a technical contribution by demonstrating that the use of domain context can improve the assessment of collaborative learning dialogue. The results of *Development 3: Assessment of Help-Giving* (Chapter 8) demonstrated that domain features indeed had a significant effect on the classification of help type and conceptual help. These results alone suggest that pursuing a course of research where *ACLS* systems are integrated with individual intelligent tutoring models would be beneficial. However, there is some doubt about the practical implications of these results, in that the absolute difference in kappa between models that include problem features and models that do not was small. This practical limitation may have been due to the small size of the data set, and thus the experiment should be replicated on a larger data set. In addition, the models were trained and tested on the same data set, and it will be important in the future to look at the accuracy of the models in classifying the data set from *Phase 4: Cognitive and Motivational Benefits of Adaptive Support*. Nevertheless, the contribution in this area was encouraging.

10.4 Empirical Results

10.4.1 Benefits of Adaptive Support for Collaboration (Q1-L1, Q1-L2)

The primary empirical contribution of this work relates to the benefits of adaptively supporting collaboration using a tutoring system, addressing the research questions: "What are the effects of ACLS on student collaborative interactions (*Q1-L1*) and learning (*Q1-L2*), compared to fixed forms of support?" The results of the four studies presented in *Phase 1*, *Phase 2*, *Phase 3*, and *Phase 4* present a compelling case for the hypothesis that adaptive support is better than fixed support at improving collaborative learning, and suggest that further research should be pursued in the area. *Phase 1: Peer Tutoring Learning Environment* (Chapter 3) established that the peer tutors do indeed need support in the context of our system, and that an initial attempt at designing fixed support for this context provided too little domain support for certain students. *Phase 2: Adaptive Correction Support* (Chapter 5) suggested that there may be some interaction benefits for providing adaptive over fixed correction support, but no clear learning benefits. While peer tutors appeared to benefit from the reflective aspects of tutoring, the help exchanged between tutors and tutees was poor, indicating that support that improves the quality of help given may benefit both students. In fact, *Phase 3: Adaptive Help-Giving Support* (Chapter 7) demonstrated benefits of adaptive support over a far fixed control on help quality, and *Phase 4: Cognitive and Motivational Benefits of Support* (Chapter 9) demonstrated benefits for adaptive support over two close fixed controls on learning.

The approaches to the studies in *Phase 3* and *Phase 4* were complementary. In *Phase 3*, there was a far control; while the content was relatively parallel between conditions, I varied both adaptive correction and help-giving assistance, and presented students with multiple types of assistance that differed between conditions. The motivation behind this choice was to make the fixed assistance similar to fixed assistance that might typically be presented in a peer tutoring scenario. I intended to conduct a fine-grained process analysis to differentiate between the effects of different types of assistance, rather than using close controls experimentally. While this comparison may have been ecologically valid, it was more difficult than I had expected to determine which aspect of the assistance might have triggered any effects – there were few direct links between assistance use and behavior. Thus, in *Phase 4*, I instead used a very close control, varying only the adaptiveness of *reflective* prompts on peer tutor help-giving, and varying the information we gave students about the prompts. With this manipulation, there were effects on learning of the actual adaptiveness of the prompts. This manipulation is not necessarily as ecologically valid as the one in *Phase 3* (presenting students with random prompts is not a traditional fixed method of support), but provides needed insight into *why* adaptive reflective prompts might have a beneficial effect. While it would have been ideal to link the support to interaction and learning benefits in the same study, that link has not yet been established. This dissertation joins a small number of other programs of research (Baghaei et al., 2007, Kumar et al., 2007, Gweon et al., 2006) that provide a strong justification for implementing adaptive support in a collaborative context.

After the study in *Phase 4*, there is further evidence of the mechanisms behind adaptive support for collaboration, including *who* benefits and *why* they benefit. The results suggested that it is the cognitive

aspects of the adaptive support that have the majority of the effect, where the support that was actually adaptive was the support that had the majority of the benefit on tutor and tutee learning. This finding is in line with results exploring the benefits of adaptivity in individual intelligent tutoring systems (Koedinger et al., 1997; VanLehn, 2006). In fact, the problem of delivering adaptive assistance to collaboration can be considered an instantiation of a more general assistance dilemma (Koedinger & Aleven, 2007), where in order to discover how best to deliver assistance to optimize student learning, one must manipulate the amount, type, and timing of help provided to students. In the case of collaborative learning, there are several levels on which assistance can be delivered, ranging from assistance on domain skills to assistance on elaborated verbal interactions. In cases where assistance on multiple levels might be appropriate at a single time, how best to integrate the different levels is an open question. The results of *Phase 4* suggest that further exploration along these lines would be appropriate.

Phase 4 also suggested that the motivational aspects of the support may have also played a role in improving the learning of the peer tutor. As the *Phase 3* results also suggest that the perception of adaptivity may motivate students to feel more accountable for the help they give, and put more thought into it, the motivational effects of adaptivity should be explored more in the future. Interestingly, it appeared that deceiving peer tutors by telling them the system was adaptive when it was not had a negative effect on the tutee, suggesting that tutees may have been receiving more confusing help as a result of the support. An analysis of the interaction data from *Phase 4* should further tease out these effects. Additionally, while the results from *Phase 2* implied that peer tutors might be deriving more benefit from the activity, the results from *Phase 3* and *Phase 4* suggested that this was not the case. It may have been that the increased support led both students to benefit from the activity, or it may have been that, in fact, the *Phase 2* results are misleading, and students in this scenario benefit equally from peer tutoring. As there were no learning differences between the collaborative and individual conditions in *Phase 2*, including an individual control in the *Phase 4* studies would have shed some clarity on what peer tutees and peer tutors learned compared to individual use of the Cognitive Tutor Algebra. Future work should establish an individual learning baseline against which the peer tutor and tutee learning results could have been compared.

Despite these positive results, the effect sizes of the adaptive support were not large, and thus while this appears to be a theoretically sound direction of research, is it a practical one? Given the vast effort that needs to be put in to developing adaptive support, it may be that focusing energies on improving fixed techniques may ultimately be more beneficial for improving the quality of collaboration in the classroom. To develop the adaptive support described in *Phase 4* required the five years of design, technical, and empirical work surveyed in Chapters 3-8 of this dissertation. I would argue that the level of adaptive support achieved in *Phase 4* would be the *minimum* level required to have a cognitive impact. On the other hand, given the results presented in this dissertation, it is plausible to suggest that the benefits of adaptivity will increase as the quality of adaptive support increases. Finding empirical results on interaction and learning given the youth of this technology is encouraging. As this technology develops, it will be

important to base iterative improvements on deployments of the technology with real students, and I will return to this idea in *10.5*.

10.4.2 Intelligent Tutoring Components and Data Analysis (Q2-L1)

This dissertation addressed another research question with respect to using intelligent tutoring *components* in the evaluation of adaptive support: How can intelligent tutoring data logs augment the analysis of collaborative study data (*Q2-L1*)? In order to get a full picture of the effects of collaboration and adaptive support to collaboration on student learning, it helps to be able to link different experimental interventions to student interactions and learning outcomes (as described in Strijbos, Martens, & Jochems, 2004). In the scenarios in this dissertation, where students go through multiple phases of learning (individual and collaborative) and take on multiple roles (tutor and tutee), the data is particularly complex. Ideally, by integrating intelligent tutoring support with a computer-supported collaborative learning activity, it is possible to view the study data at multiple levels of analysis: student activities, problems solved, attempts at problem-solving steps, and student interactions with each other and with system feedback. While such an in-depth approach was not historically possible in a classroom environment, the combined logging capabilities enabled by intelligent tutoring systems and computer-supported collaborated learning offer us a unique opportunity. Each level of data analysis consists of student interactions with each other and with the system, which is common in computer-mediated collaboration. However, it also consists of the system assessment of the student problem-solving actions, which has been refined particularly in intelligent tutoring system approaches. The student and system inputs at each level provide insight into what is occurring in the collaboration, and how it might relate to domain learning. These multiple data sources enable analyses that would not typically be possible: They encourage quantitative analyses of problem-solving data, qualitative analyses of student dialogue, and links to be made between these interaction variables and outcome variables.

While this is an idealized picture of the data analysis procedure, the only study where I successfully implemented this procedure was in *Phase 2*. Multiple data sources improved our understanding of the benefits of peer tutoring and adaptive assistance. I were able to specifically link tutor gains to problem-solving behaviors that would logically trigger reflection, such as errors, help-requests, and tutor feedback. Further, the most interesting results required data sources to be combined in a single analysis. Help needed (which links tutee problem-solving and tutor help) and assistance communicated (which links cognitive tutor feedback and tutor help) are the two clear examples of this. These empirical results are not common in other work, potentially because this data is rarely available in an integrated form. However, I was unable to duplicate this analysis in *Phase 3* and *Phase 4*. While *Phase 3* contained an interaction analysis combining quantitative and qualitative measures, there was difficulty with the learning measures, preventing a proper correlational analysis between positive student interactions and learning. In *Phase 4*, I improved the learning measures and found learning differences between conditions, but have not

yet had the time to look at the interaction data. Overall, this integrated data approach appears highly useful for the analysis of the benefits of collaborative learning, and should be applied to *Phase 4*.

10.5 Final Thoughts

The overall research approach used by this dissertation has been one of *iterated in vivo experimentation,* characterized by iterative design, the reuse of existing components to create new components, and a balance between experimental control and ecological validity. This approach evolved in part from the *in vivo experiments* described by Koedinger and colleagues (2009). *In vivo* experiments lie at the intersection of psychological experimentation and design-based research, as defined by Collins (1999). Like psychological experimentation, an *in vivo* experiment involves the manipulation of a single variable and the use of fixed procedures to test a set of hypotheses. In contrast, like design-based research, an in vivo experiment takes place in real-world contexts that involve social interaction, and characterizes the relationships between multiple process variables and outcome variables. The studies in *Phase 1, Phase 2, Phase 3,* and to a certain extent, *Phase 4*, are examples of this paradigm: Experimental variables are manipulated in real-world contexts, and control over the experimental procedure is traded for ecological validity. It is possible that there were effects of adaptivity on learning in *Phase 4* because that study was the most controlled of all four studies, occurring after school with paid participants. However, each study contributed to the overall set of empirical results presented in this dissertation.

In addition, I would argue that for *in vivo* experimentation to be successful it can be helpful to incorporate further elements of design based research outside those used in a single *in vivo* experiment: the use of participant co-design and analysis to develop a profile of what is occurring and inform flexible, iterative design revisions. *Iterated in vivo experimentation,* where we use a design-based research process to create an intervention, deploy the intervention using an *in vivo* experiment, and then interpret the effects through a design-based lens, may be a more effective way of theory building than executing an *in vivo* experiment in isolation. *Development 2* is a good example of this philosophy, where we broke from the traditional study format to conduct a principled exploration of the design space for adaptive support. In addition, the use of thinkalouds and qualitative analysis of study data to inform future iterations has added a richness to the data analysis (for example, inspiring the accountability manipulation in *Phase 4*). Given that *in vivo* experiments are high stakes experiments, in that they require a lot of time and investment to set up and construct, there are large benefits to low-cost design work and piloting early in the process, so that the choices made in designing and executing *in vivo* experiments are well informed.

To successfully iterate upon a program of research in this manner, it is important have a component-based design for the system that can be incrementally built upon, and that is where *CTRL*, developed in *Development 1*, played such a vital role. Refactoring the *CTA* so that it was suitable for collaboration was a huge up-front development effort, but the multiple iterations we have been able to execute on the system has demonstrated the effort to be worthwhile. The use of the *CTA* in *APTA* allowed the exploration at length of the benefits of including existing domain components in collaborative models.

It may be that in order to create a platform for experimentation in a new technology, a large upfront development effort is required.

There are many interesting future iterations that could be done based on the results presented in this dissertation. For one, the model tracing and knowledge tracing of help-giving support has not been properly evaluated, and that would be an important step, both for demonstrating validity and for highlighting areas for improvement. While this dissertation began the exploration of a design space for adaptive support, it is still not clear how the dimensions of explicitness and directness affect student feelings of accountability, efficacy, and relevance, and how that interacts with other qualities of support. Finally, the empirical results in *Phase 4* only lead to more research questions. What aspects of student interaction lead to the learning improvements in the adaptive condition? Did students in all conditions perceive the support as equally adaptive, or were the measures of perceived adaptivity inadequate for exploring student mental models of support? What effects did actual and perceived adaptivity have on student interaction? The answers to these questions will shed more light on the potential benefits of adaptive support for collaboration. To conclude, adaptive support for collaboration, while currently time consuming to develop, appears to be a promising advancement in computer-supported collaborative learning technology.

References

Aleven, V., & Koedinger, K.R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science*, *26* (2), 147-179.

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2004). Toward tutoring help seeking: Applying cognitive modeling to meta-cognitive skills. In J. C. Lester, R. M. Vicario, & F. Paraguaçu (Eds.), *Proceedings of 7th International Conference on Intelligent Tutoring Systems* (pp. 227-239).

Anderson, J. R., & Pelletier, R. (1991). A development system for model–tracing tutors. In Proceedings of the International Conference of the Learning Sciences (pp. 1–8). Evanston, IL.

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, R.S.J.d., Corbett, A.T., Aleven, V. (2008). More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 406-415.

Beck, J. E., Mostow, J., & Bey, J. (2004). Can automated questions scaffold children's reading comprehension? In: J. C. Lester, R. M. Vicari, F. Paraguacu (eds.): *Proceedings of 7th International Conference on Intelligent Tutoring Systems.* Berlin: Springer Verlag, pp. 478-490.

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.

Bernsen, N., Dybkjær, H., and Dybkjær, L. (1997). What Should Your Speech System Say?. *Computer 30* (12), 25-31.

Biswas, G., Leelawong, K., Schwartz, D., Vye, N. & The Teachable Agents Group at Vanderbilt (2005). Learning By Teaching: A New Agent Paradigm for Educational Software. *Applied Artificial Intelligence, 19*, 363-392.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13*, 3-16

Booth, J.L., & Koedinger, K.R. (2008). Key misconceptions in algebraic problem solving. In B.C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Cognitive Science Society* (pp. 571-576). Austin, TX: Cognitive Science Society.

Bransford, J. D. & Schwartz, D. (1999). Rethinking transfer: A simple proposal with multiple implications. In A. Iran-Nejad & P. D. Pearson (Eds.), *Review of research in education* (Vol. 24, pp. 61-100). Washington, DC: American Educational Research Association.

Brusilovsky, P.: (2001), Adaptive Hypermedia. *User Modeling and User-Adapted Interaction, 11* (1/2), 111-127.

Carmien, S., Kollar, I., Fischer, G. & Fischer, F. (2006). The interplay of internal and external scripts. In F. Fischer, I. Kollar, H. Mandl &, J. Haake, Scripting computer-supported communication of knowledge. *Cognitive, computational, and educational perspectives* (pp. 289-311). New York: Springer.

Chan, C. (2001). Peer collaboration and discourse patterns in learning from incompatible information. *Instructional Science, 29*, 443–479.

Chan, T.-W. & Chou, C.-Y. (1997). Exploring the design of computer supports for reciprocal tutoring. *International Journal of Artificial Intelligence in Education, 8*(1), 1–29.

Chaudhuri, S., Kumar, R., Howley, I., Rosè, C. P. (2009). Engaging Collaborative Learners with Helping Agents, (2009). In V. Dimitrova, R. Mizoguchi, B. du Bulay, A. Graesser (Eds.), *Proceedings of the 14th Intl. Conf. on Artificial Intelligence in Education (AIED 2009)* (pp. 365–372), Amsterdam: IOS Press.

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

Clark, D, (2007). Symposium # 3: Orchestrating Learning Activities on the Social and the Cognitive Level to Foster CSCL. *Computer Supported Collaborative Learning Conference.*

Collins, A. (1999). The changing infrastructure of education research. In E. C. Lagemann, & L. S. Shulman (Eds.) *Issues in education research: Problems and possibilities.* (pp. 289–298). San Francisco: Jossey-Bass Publishers.

Constantino-González, M. A., Suthers, D., & Escamilla de los Santos, J. (2003). Coaching web-based collaborative learning based on problem solution differences and participation. *International Journal of Artificial Intelligence in Education*, 13(2–4), 263–299.

Corbett, A.T., Anderson, J.R. (1995) Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, *4*, 253-278.

Davidoff, S., Lee, M.K., Day, A.K., Zimmerman, J. (2007). Rapidly exploring application design through Speed Dating. *Proc. Ubicomp*, 429-446.

Dey, A.K. (2009) Modeling and intelligibility in ambient environments. *Journal of Ambient Intelligence and Smart Environments, 1*(1), pp. 47-62.

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., Blaye, A., & O'Malley, C. (1995), The evolution of research on collaborative learning. In: H. Spada & P. Reimann (eds.): *Learning in Humans and Machine: Towards an Interdisciplinary Learning Science.* Oxford: Elsevier, pp. 189-211.

Dillenbourg, P., & Jermann, J. (2007). Designing integrative scripts. In Scripting Computer-Supported Collaborative Learning - Cognitive, Computational, and Educational Perspectives, Computer-Supported Collaborative Learning Series, pages 275-301. Springer, New York, 2007.

Diziol, D., Rummel, N., Kahrimanis, G., Guevara, T., Holz, J., Spada, H., Fiotakis, G. (2008). Using contrasting cases to better understand the relationship between students' interactions and their learning outcome. In G. Kanselaar, V. Jonker, P.A. Kirschner, & F. Prins, (Eds.), International perspectives of the learning sciences: Cre8ing a learning world. *Proceedings of the Eighth International Conference of the Learning Sciences (ICLS 2008), Vol 3* (pp. 348-349). International Society of the Learning Sciences, Inc. ISSN 1573-4552.

Diziol, D., Walker, E., Rummel, N., & Koedinger, K. R. (2010). Using Intelligent Tutor Technology to Implement Adaptive Support for Student Collaboration. *Educational Psychology Review, 22*(1), 89-102.

Dybowski, R., Laskey, K. B., Myers, J. W., & Parsons, S. (2003). Introduction to the Special Issue on the Fusion of Domain Knowledge and Data for Decision Support. *Journal of Machine Research. 4*. 293-294.

Elliot, A. J., & McGregor, H. A. (2001). A 2 x 2 achievement goal framework. Journal of *Personality and Social Psychology, 80*, 501-519.

Fantuzzo, J. W., King, J., & Heller, L. (1992). Effects of reciprocal peer tutoring on mathematics and school adjustment: A componential analysis. *Journal of Educational Psychology, 84*, 331-339.

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.

Finney, S. J., Pieper, S. L., & Barron, K. E. (2004). Examining the psychometric properties of the Achievement Goal Questionnaire in a general academic context. *Educational and Psychological Measurement, 64*, 365-382.

Fischer, F., Kollar, I., Mandl, H., & Haake, J. (2007). Scripting Computer-Supported Collaborative Learning – Cognitive, Computational, and Educational Perspectives. *Computer-Supported Collaborative Learning Series*, New York: Springer.

Fuchs, L., Fuchs, D., Hamlett, C., Phillips, N., Karns, K., & Dutka, S. (1997). Enhancing students' helping behavior during peer-mediated instruction with conceptual mathematical explanations. *The Elementary School Journal, 97*(3), 223-249.

Genesereth, M. R. (1997). An agent-based framework for interoperability. In: J. M. Bradshaw (ed.), *Software Agents*. Menlo Park, California: AAAI Press/MIT Press, pp. 317-345.

Gweon, G., Rose, C., Carey, R. and Zaiss, Z. (2006). Providing support for adaptive scripting in an on-line collaborative learning environment. In *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems* (pp. 251-260). ACM Press.

Hake, R.R. (1998). Interactive-engagement versus traditional methods: a six-thousand- student survey of mechanics test data for introductory physics courses. *American Journal of Physics, 66*, 64 – 74.

Hall, M. Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). *The WEKA Data Mining Software: An Update*; SIGKDD Explorations, Volume 11, Issue 1.

Hmelo-Silver, C. E. (2004) Problem-based learning: What and how do students learn? *Educational Psychology Review, 16*(3), 235–266.

Hoppe, H. U. (1995). Using multiple student modeling to parameterize group learning. In *Proc. of AI-ED '95*, 234--241, Washington D.C., August 1995: AACE.

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.

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., 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.

Koedinger, K. R., & Aleven V. (2007). Exploring the assistance dilemma in experiments with Cognitive Tutors. *Educational Psychology Review*, 19(3), 239-264.

Koedinger, K. R., Aleven, V., Roll, I., & Baker, R. (2009). In vivo experiments on whether supporting metacognition in intelligent tutoring systems yields robust learning. In D. J. Hacker, J. Dunlosky, & A. C. Graesser (Eds.), *Handbook of Metacognition in Education* (pp. 897-964). *The Educational Psychology Series*. New York: Routledge.

Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. (in press) A Data Repository for the EDM commuity: The PSLCDataShop. To appear in Romero, C., Ventura, S., Pechenizkiy, M., Baker, R.S.J.d. (Eds.) *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press.

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.), The next 10 years*! Proceedings of the International Conference on Computer Support for Collaborative Learning* 2005 (pp. 331-340). Mahwah, NJ: Lawrence Erlbaum Associates.

Krueger, C. W. (1992). Software reuse. *ACM Computing Surveys, 24*(2), 131-183.

Kumar, V., McCalla, G., & Greer, J. (1999). Helping the peer helper. *Proceedings of the 9th World Conference on Artificial Intelligence in Education (AI-ED '99)*, LeMans, France, 325-332.

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, & J. Greer (Eds.) *Proceedings of Artificial Intelligence in Education* (pp. 383-390). IOS Press.

Kumar, R., Ai, H., Beuth, J. L., Rosé, C. P. (2010). Socially-capable Conversational Tutors can be Effective in Collaborative-Learning situations. *Intl. Conf. on Intelligent Tutoring Systems,* Pittsburgh, PA.

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

Leelawong, K., & Biswas, G. (2008). Designing learning by teaching agents: The Betty's Brain System. *International Journal of Artificial Intelligence in Education, 18*(3), 181–208.

Lou, Y., Abrami, P. C., d'Apollonia S. (2001). Small group and individual learning with technology: A meta-analysis. *Review of Educational Research, 71*(3), 449-521.

Medway, F. & Baron, R. Locus of control and tutors' instructional style. *Contemporary Educational Psychology, 2*, 298-310 (1997).

Meier, A., Spada, H., & Rummel, N. (2007). A rating scheme for assessing the quality of computer-supported collaboration processes. *International Journal of Computer-Supported Collaborative Learning, 2* (1), 63-86.

Michaels, S., O'Connor, C., & Resnick, L. B. (2008). Deliberative discourse idealized and realized: Accountable talk in the classroom and in civic life. *Studies in the Philosophy of Education, 27*(4), 283-297.

Mitrovic, A., Koedinger, K. R., & Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modelling. 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-Verlag.

Mitrovic, A., Weerasinghe, A. (2009) Revisiting ill-definedness and the consequences for ITSs. In: *Proceeding of the 2009 conference on Artificial Intelligence in Education*, Amsterdam, The Netherlands, The Netherlands, IOS Press, 375–382.

Mostow, J., Aist, G. (2001). Evaluating tutors that listen: An overview of Project LISTEN. In K. Forbus & P. Feltovich (Eds.), *Smart Machines in Education* (pp. 169-234). Menlo Park, CA: MIT/AAAI Press.

Mühlenbrock, M. (2004). Shared Workspaces: Analyzing User Activity and Group Interaction. In: H. U. Hoppe, M. Ikeda, & H. Ogata (eds.): *New Technologies for Collaborative Learning*, Dordrecht: Kluwer Academic Publishers.

Mühlenbrock, M., Tewissen, F. & Hoppe, H. U. (1998). A framework system for intelligent support in open distributed learning environments. *International Journal of Artificial Intelligence in Education*, 9, 256-274.

Nicol, D. J. & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education, 31*(2), 199-218.

Palincsar, A.S., & Brown, A.L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction, 1*(2), 117-175.

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). Elsevier Science Publishers.

Prichard, J. S., Stratford, R. J., & Bizo, L. A. (2006). Team-skills training enhances collaborative learning. *Learning and Instruction, 16*(3), 256-265.

Resnick, L., O'Connor, C., and Michaels, S. (2007). Classroom Discourse, Mathematical Rigor, and Student Reasoning: An Accountable Talk Literature Review.

Ritter, S., Blessing, S. B., & Hadley, W. S. (2002). SBIR Phase I Final Report 2002. Department of Education. Department of Education RFP ED: 84-305S.

Ritter, S. and Koedinger, K.R. (1996). An architecture for plug-in tutor agents. *International Journal of Artificial Intelligence in Education, 7*(3/4), 315-347.

Rittle-Johnson, B., & Alibali, M. W. (1999). Conceptual and procedural knowledge of mathematics: Does one lead to the other? *Journal of Educational Psychology, 91*, 175–189.

Robinson, D., Schofield, J., & Steers-Wentzell, K. (2005). Peer and cross-age tutoring in math: Outcomes and their design implications. *Educational Psychology Review, 17*(4), 327-362.

Roll, I., Aleven, V., McLaren, B. M., & Koedinger, K. R. (2007). Can help seeking be tutored? Searching for the secret sauce of metacognitive tutoring. In: R. Luckin, K. Koedinger, & J. Greer (eds.), *Proceedings of the 13th International Conference on Artificial Intelligence in Education AIED 2007*, pp. 203-10.

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. P., Wang, Y.C., Cui, Y., Arguello, J., Stegmann, K., Weinberger, A., Fischer, F. (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., & Spada, H. (2007) Can people learn computer-mediated collaboration by following a script? In F. Fischer, I. Kollar, H. Mandl, & J. Haake. *Scripting computer-supported communication of knowledge. Cognitive, computational, and educational perspectives.* (pp. 47-63). New York: Springer.

Rummel, N. & Weinberger, A. (2008). New challenges in CSCL: Towards adaptive script support. In G. Kanselaar, V. Jonker, P.A. Kirschner, & F. Prins, (Eds.), International perspectives of the learning sciences: Cre8ing a learning world. *Proceedings of the Eighth International Conference of the Learning Sciences (ICLS 2008), Vol 3* (pp. 338-345). International Society of the Learning Sciences.

Saab, N., Van Joolingen, W. R., & Van Hout-Wolters, B. (2007). Supporting communication in a collaborative discovery learning environment: The effect of instruction. *Instructional Science*, 35, 73-98.

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.

Scott, L. A., & Reif. F. (1999). Teaching Scientific Thinking Skills: Students and Computers Coaching Each Other. In *Proceedings of AI-ED 99 World Conference on Artificial Intelligence in Education*, Le Mans, France, 285–293.

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. (2004). Computational Modeling and Analysis of Knowledge Sharing in Collaborative Distance Learning. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research, 14* (4), 351-381.

Soller, A., Martinez, A., Jermann, P., and Mühlenbrock, M. (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.

Stahl, G. (2000). A Model of Collaborative Knowledge-Building. In B. Fishman & S. O'Connor-Divelbiss (Eds.), *Fourth International Conference of the Learning Sciences* (pp. 70-77). Mahwah, NJ: Erlbaum.

Strijbos, J. W., Martens , R. L., & Jochems, W. M. G. (2004). Designing for interaction: Six steps to designing computer-supported group-based learning. *Computers & Education*, 42(4), 403-424 .

Suebnukarn S and 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.

Suthers, D. (2001). Architectures for Computer Supported Collaborative Learning. In: T. Okamoto, R. Hartley, Kinshuk, J.P. Klus (eds.): *Proceedings of the IEEE International Conference on*

Advanced Learning Technology: Issues, Achievements and Challenges. Los Alamitos, CA: IEEE Computer Society, pp. 25-28.

Tabachnick, B.G. & Fidell, L.S. (1996). *Using multivariate statistics* (3rd edition). New York: Harper Collins College Publishers.

Teasley, S., & Fischer, F. (2008). Cognitive convergence in collaborative learning. In G. Kanselaar, V. Jonker, P.A. Kirschner, & F. Prins, (Eds.), International perspectives of the learning sciences: Cre8ing a learning world. *Proceedings of the Eighth International Conference of the Learning Sciences* (ICLS 2008), Vol 3 (pp. 360-368). International Society of the Learning Sciences, Inc. ISSN 1573-4552

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.

Uresti, J. A. R. (2000). Should I Teach my Computer Peer? Some Issues in Teaching a Learning Companion. In G. Gauthier, C. Frasson, & K. Vanlehn (Eds*.). Intelligent tutoring Systems. Fifth International Conference, ITS'2000*, Vol. 1839 of Lectures Notes of Computer Science, Springer-Verlag, 103–112.

Van den Bossche, P., Gijselaers, W., Segers, M., & Kirschner, P. (2006). Social and cognitive factors driving teamwork in collaborative learning environments. *Small Group Research*, 37, 490-521.

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education, 16*(3), 227-265.

VanLehn, K., Siler, S., Murray, C., Yamauchi, T., & Baggett, W. (2003). Why do only some events cause learning during human tutoring? *Cognition and Instruction, 21*(3), 209-249.

VanLehn, K., Koedinger, K. R., Skogsholm, A., Nwaigwe, A., Hausmann, R. G. M., Weinstein, A., Billings, B. (2007). What's in a step? Toward general, abstract representations of tutoring system log data. In: C. Conati, K. F. McCoy, G. Paliouras (eds.)*: User Modeling* 2007, 11th International Conference. Springer, pp. 455-459.

Vassileva, J., McCalla, G., and Greer, J. (2003). Multi-Agent Multi-User Modeling in I-Help. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research, 13*, 179-210, DOI: 10.1023/A:1024072706526.

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

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.* Berlin: Springer-Verlag, pp. 262-271.

Walker, E. Mutual Peer Tutoring: A Collaborative Addition to the Cognitive Tutor Algebra-1 (2005). Accepted as a Young Researcher's Track paper at the *International Conference on Artificial Intelligence and Education* (AIED-05).

Walker, E., Koedinger, K. R., McLaren, B. M. and Rummel, N. (2006). Cognitive Tutors as Research Platforms: Extending an Established Tutoring System for Collaborative and Metacognitive Experimentation (2006). In Ikeda, M., Ashely, K. D., & Chan, T.-W. (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (pp. 207-216). Berlin: Springer.

Walker, E., Rummel, N., McLaren, B. M. & Koedinger, K. R. The Student Becomes the Master: Integrating Peer Tutoring with Cognitive Tutoring (2007a). In Chinn, C., Erkins, G., Puntambekar, S. (Eds.), Proceedings of the 8th International Conference on Computer Supported Collaborative Learning (pp. 751-753). Mahwah, NJ: Lawrence Erlbaum Associates.Wang, Y. C., Joshi, M., Rosé, C. P., Fischer, F., Weinberger, A., Stegmann, K. 2007. Context Based Classification for Automatic Collaborative Learning Process Analysis. Poster in: the 13th International Conference on Artificial Intelligence in Education (AIED 2007).

Walker, E., McLaren, B. M., Rummel, N., and Koedinger, K. R. Who Says Three's a Crowd? Using a Cognitive Tutor to Support Peer Tutoring (2007b). In Luckin, R., Koedinger, K.R., & Greer, J., Proceedings of the 13th International Conference on Artificial Intelligence and Education (pp. 399-406). Amsterdam: IOS Press.

Walker, E., Rummel, N., and Koedinger, K. R. To Tutor the Tutor: Adaptive Domain Support for Peer Tutoring (2008). In Woolf, B., Aimeur, E., Nkambou, R., & Lajoie, S., Proceedings of the 9th Internationa Conference on Intelligent Tutoring Systems (pp. 626-635). Berlin: Springer.

Walker, E., Rummel, N., & Koedinger, K. R. (2009a). CTRL: A research framework for providing adaptive collaborative learning support. *User Modeling and User-Adapted Interaction, 19*(5), 387-431.

Walker, E., Rummel, N., & Koedinger, K. R. (2009b). Integrating collaboration and intelligent tutoring data in the evaluation of a reciprocal peer tutoring environment. *Research and Practice in Technology Enhanced Learning, 4*(3), 221-251.

Walker, E., Rummel, N., & Koedinger, K. R. (2009c) Beyond Explicit Feedback: New Directions in Adaptive Collaborative Learning Support. In O'Malley, C., Suthers, D., Reimann, P., & Dimitracopoulou, A. (Eds.), *Proceedings of the 9th International Conference on Computer Supported Collaborative Learning* (pp. 552-556). Mahwah, NJ: Lawrence Erlbaum Associates.

Walker, E., Rummel, N. & Koedinger, K. (2009d). Modeling helping behavior in an intelligent tutor for peer tutoring. In V. Dimitrova, R. Mizoguchi, B. du Boulay, & A. Graessar (Eds.), *Proceedings of the 14th International Conference on Artificial Intelligence in Education* (pp. 341- 349). Amsterdam: IOS Press.

Walker, E., Walker, S., Rummel, N., & Koedinger, K. (2010). Using Problem-Solving Context to Assess Help Quality in Computer-Mediated Peer Tutoring. *To appear in The 10th International Conference on Intelligent Tutoring Systems.*

Webb, N., Troper, J., & Fall, R. Constructive activity and learning in collaborative small groups. *Journal of Educational Psychology, 87*(3), 406-423 (1995).

Webb, N. M. (1989). Peer interaction and learning in small groups. *International Journal of Educational Research, 13*, 21–40.

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

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

Appendix A: Domain Learning Measures for *Phase 4*

This appendix contains all forms of the domain learning measures in *Phase 4: Cognitive and Motivational Benefits of Adaptive Support.*

Questionnaire – Part 1 -- Form A

Study ID: _____

Date: _____

Type (circle one): Pre / Post

1. You are solving for z in the following equations. Write the coefficient for z in each equation. HINT: When you want to get the z alone, the coefficient is what you need to get rid of.

   a. $5z = c$ Coefficient: _____

   b. $z * 5a = c$ Coefficient: _____

2. What does x equal? Simplify your answer as much as possible.

   a. $\frac{abc}{x} = 1$ $x =$ _____

   b. $6cr + 4cr = x$ $x =$ _____

3. Factor y out of the following equations. HINT: Your answer should take the form of y multiplied by some combination of numbers and/or symbols.

   a. $-7y + ay = \quad y\,(\qquad\qquad)$

   b. $y + cy = \quad y\,(\qquad\qquad)$

   c. $\frac{y}{d} + bcy = y\,(\qquad\qquad)$

4. Remove all brackets from the following equations. Make sure your new equations are equal to the starting equations:

   a. $(a + x) * c =$

   b. $(c(a + b) + d) * f =$

5. You are solving for *b*. What is an good first step for solving the following equation:

   $-ab + cb + dc = xyz$

   a. Divide by c - a YES NO

   b. Subtract dc from both sides YES NO

6. Which of the following equations are equivalent to the following equation:

   $f(a + b) = ac$

   a. $fa + b = ac$ YES NO

   b. $f = \frac{ac}{a+b}$ YES NO

7. Dave is saving to buy a bicycle. He has *a* dollars. Each day he saves *b* dollars.

    a. How much money does Dave start out with? _____

    b. How much money does Dave save per day? _____

    c. If Dave saves money for *x* days, how much money has he saved (*y* = money saved)? Write an expression for *y* using *a*, *b*, and *x*:

        y = _____

    d. If Dave has saved *y* dollars, how many days has he saved for (*x* = number of days)? Write an expression for *x* using *a*, *b*, and *y*:

        x = _____

8. The solution to the following problem is incorrect. There are TWO mistakes. Circle each step that is a mistake. Then, explain what the person did wrong.

   Solve for a

   $$rz = \frac{a(-tg + bg)}{cn}$$

   $$rz = \frac{-atg + abg}{cn}$$

   $$rz + cn = \frac{-atg + abq}{cn} + cn$$

   $$rz + cn = -atg + abq$$

   $$rz + cn = a(tg - bq)$$

   $$\frac{rz + cn}{tg - bq} = a$$

Explanation:

Questionnaire -- Part 2 – Form A

Study ID: _____

Date: _____

Type (circle one): Pre / Post

1. Solve for k

$$-kr + bz = tw - 2m$$

2. Solve for w

$$zx - cf = zw + nw$$

3. Solve for u

$$bt - ud = 11u - bf$$

4. Solve for n

$$w + a = \frac{b}{n}$$

5. Solve for a

$$\sqrt{a-b} = c - df$$

6. You are tutoring a student on the following equation. Explain the problem solution to your partner so that they know how to solve the entire problem.

Solve for *r*

$$k(su+rg) = 11u - bf$$

$$ksu + krg = 11u - bf$$

$$krg = 11u - bf - ksu$$

$$r = \frac{11u - bf - ksu}{kg}$$

Explanation:

Questionnaire – Part 1 -- Form B

Study ID: _____

Date: _____

Type (circle one): Pre / Post

1. You are solving for z in the following equations. Write the coefficient for z in each equation. HINT: When you want to get the z alone, the coefficient is what you need to get rid of.

   a. $5az = c$ Coefficient: _____

   b. $\frac{z}{5} = c$ Coefficient: _____

2. What does x equal? Simplify your answer as much as possible.

   a. $rkf - x = 0$ $x =$ _____

   b. $\frac{x}{5d} = b$ $x =$ _____

3. Factor x out of the following equations. HINT: Your answer should take the form of x multiplied by some combination of numbers and/or symbols.

   a. $ax + 2x = x($ $)$

   b. $x - cx = x($ $)$

   c. $ax + bx + cx = x($ $)$

4. Remove all brackets from the following equations. Make sure your new equations are equal to the starting equations:

   a. $(a - b) - c =$

   b. $(a + b) / c =$

5. You are solving for *b*. What is an good first step for solving the following equation:

   $-ab + cb + dc = xyz$

   a. Factor b YES NO

   b. Add ab to both sides YES NO

6. Which of the following equations are equivalent to the equation:

   $f(a + b) = ac$

   a. $fa + fb = ac$ YES NO

   b. $f = \frac{c}{b}$ YES NO

7. Sarah, a rock climber, is currently *a* feet off the ground. She can climb an average of *b* feet per minute.

    a. How high up is Sarah right now? _____

    b. How fast can Sarah climb in feet per minute? _____

    c. If Sarah climbs for *x* minutes, how far off the ground will she be (*y* = distance)? Write an expression for *y* using *a*, *b*, and *x*:

        y = _____

    d. If Sarah is *y* feet off the ground, how many minutes has she climbed for (*x* = number of minutes)? Write an expression for *x* using *a*, *b*, and *y*:

        x = _____

8. The solution to the following problem is incorrect. There are TWO mistakes. Circle each step that is a mistake. Then, explain what the person did wrong.

   Solve for d

   $$lf = \frac{a(td - bk)}{cn}$$

   $$lfcn = \frac{a(td - bk)}{cn} * cn$$

   $$lfcn = atd - bk$$

   $$lfcn - bk = atd$$

   $$d = \frac{lfcn - bk}{at}$$

Explanation:

Questionnaire -- Part 2 – Form B

Study ID: _____

Date: _____

Type (circle one): Pre / Post

1. Solve for *g*

   $abg - 2ch = zxd - cfa$

2. Solve for *d*

   $-bd + rd = vc - gm$

3. Solve for z

$$cn + bz = az + rk$$

4. Solve for k

$$f + g = \frac{h}{k}$$

5. Solve for *a*

$$a^2 - b^2 = c^2 - 2cb$$

6. You are tutoring a student on the following equation. Explain the problem solution to your partner so that they know how to solve the entire problem.

Solve for *c*

$$\frac{c}{k} - cdb = \frac{lf}{m}$$

$$c\left(\frac{1}{k} - db\right) = \frac{lf}{m}$$

$$c\left(1 - dbk\right) = \frac{lfk}{m}$$

$$c = \frac{lfk}{m(1 - dbk)}$$

Explanation:

Appendix B: Reflective Prompts in *Phase 4*

This appendix contains the xml markup for the reflective prompts peer tutors received in *Phase 4: Cognitive and Motivational Benefits of Support.* The <name> tag represents the name of the production rule that fired based on the peer tutor action. The <text> tag denotes a potential feedback message that could be triggered should the rule file.

```
<messages>
    <message-set>
        <name>helpAfterRequest</name>
        <message>
            <text>Keep at it! When your partner asks for help, it's a good chance to explain
            how to solve the problem.</text>
            <text>Keep going! It's important to talk about the problem with your
            partner.</text>
            <text>You're doing well! When your partner gets help when they need it, they will
            learn more.</text>
        </message>
    </message-set>
    <message-set>
        <name>helpAfterIncorrect</name>
        <message>
            <text>Keep at it! When your partner makes a mistake, it's a good opportunity to
            help them understand what to do.</text>
            <text>Keep going! It's important to talk about the problem, especially when your
            partner is getting steps wrong.</text>
            <text>You're doing well! When your partner gets help after errors, they will learn
            more.</text>
        </message>
    </message-set>
    <message-set>
        <name>noHelpAfterRequestLong</name>
        <message>
            <text>[Tutor], if you don't know how to help your partner ask the computer for a
            hint.</text>
        </message>
    </message-set>
    <message-set>
        <name>noHelpAfterRequestShort</name>
        <message>
            <text>[Tutor], does your partner know what to do? Check to see if they are doing
            the right thing.</text>
            <text>[Tutor], look at the last step. Is it right? Get your partner to explain why
            they took it.</text>
            <text>[Tutor], there may be something your partner doesn't understand. Do you
            know what it is?</text>
            <text>[Tutor], do you know what your partner should do? Try asking the computer
            for a hint.</text>
            <text>[Tutor], is your partner taking the right steps? To see, try marking the steps
            right or wrong using the checkmark or the x.</text>
        </message>
    </message-set>
    <message-set>
        <name>noHelpAfterIncorrect</name>
        <message>
            <text>[Tutor], is your partner taking the right steps? Make sure both sides of the
            equation still equal each other.</text>
            <text>[Tutor], are those steps right? To check, try substituting numbers for
            letters.</text>
            <text>[Tutor], are those steps right? To see, try marking them right or wrong using
            the checkmark or the x.</text>
            <text>[Tutor], do you know what your partner should do? Try asking the computer
            for a hint.</text>
```

```
                <text>[Tutor], is your partner taking the right steps? To see, try marking the steps
                right or wrong using the checkmark or the x.</text>
        </message>
</message-set>
<message-set>
        <name>helpAfterCorrect</name>
        <message>
                <text>[Tutor], after you help the [Tutee], ask the [Tutee] if they can explain what
                you just said in their own words.</text>
                <text>[Tutor], after you give help, check to see if your partner understands.</text>
                <text>[Tutor], think about what you last said. Were you telling your partner the
                answer? Giving your partner a hint?</text>
                <text>[Tutor], make sure the tutee needs help before you give it.</text>
                <text>[Tutor], when giving help, remember to make sure your partner understands
                why to do things.</text>
                <text>[Tutor], remember to wait until your partner has asked for help or tried the
                step before you jump in.</text>
        </message>
</message-set>
<message-set>
        <name>noPromptAfterMisconception</name>
        <message>
                <text>[Tutor], before you help your student on the next step, you may want to ask
                to them about their previous step.</text>
                <text>[Tutor], do you know why your partner took the step they did?</text>
                <text>[Tutor], is there anything your partner doesn't understand right now about
                the problem?</text>
                <text>If your partner has made a mistake, ask them to explain what they did (using
                the "Ask Why" button).</text>
        </message>
</message-set>
<message-set>
        <name>noPromptAfterMisconceptionRequest</name>
        <message>
                <text>[Tutor], before you help your student on the next step, you may want to ask
                to them about their previous step.</text>
                <text>[Tutor], do you know why your partner took the step they did?</text>
                <text>[Tutor], is there anything your partner doesn't understand right now about
                the problem?</text>
        </message>
</message-set>
<message-set>
        <name>noErrorFeedbackAfterMisconception</name>
        <message>
                <text>[Tutor], do you know if your partner has made a mistake?</text>
                <text>[Tutor], can you explain your partner's mistake?</text>
                <text>[Tutor], is there anything your partner doesn't understand right now about
                the problem?</text>
                <text>If your partner has made a mistake, help them to figure out what they don't
                understand (using the "Explain Why Wrong" button).</text>
        </message>
</message-set>
<message-set>
        <name>noErrorFeedbackAfterMisconceptionRequest</name>
        <message>
                <text>[Tutor], do you know if your partner has made a mistake?</text>
```

```
                <text>[Tutor], can you explain your partner's mistake?</text>
                <text>[Tutor], is there anything your partner doesn't understand right now about
                the problem?</text>
        </message>
</message-set>
<message-set>
        <name>PromptAfterIncorrect</name>
        <message>
                <text>Good work! Remember, asking your partner to explain a step can help them
                learn how to solve the problem.</text>
                <text>You two are doing well. Now do you have a better sense of what your
                partner was thinking?</text>
        </message>
</message-set>
<message-set>
        <name>ErrorFeedbackAfterIncorrect</name>
        <message>
                <text>Good work! Remember, exploring what your partner is doing wrong can
                help them not make the same mistake on future problems.</text>
                <text>Well done! Remember, it's important to tell your partner what they did
                wrong on top of explaining what to do next.</text>
        </message>
</message-set>


<message-set>
        <name>PromptAfterErrorRelatedRequest</name>
        <message>
                <text>Good work! Remember, asking your partner to explain a step can help them
                learn how to solve the problem.</text>
                <text>You two are doing well. Now do you have a better sense of what your
                partner was thinking?</text>
        </message>
</message-set>
        <message-set>
        <name>ErrorFeedbackAfterErrorRelatedRequest</name>
        <message>
                <text>Keep at it! Remember, exploring what your partner is doing wrong can help
                them not make the same mistake on future problems.</text>
                <text>Good going! It's important to tell your partner what they did wrong on top
                of explaining what to do next.</text>
        </message>
</message-set>
<message-set>
        <name>highLevelHelp</name>
        <message>
                <text>Good work! Hinting or explaining the reason for a step can help your
                partner learn how to do the step.</text>
                <text>Nicely done! Explaining why to do a step will prepare your partner to solve
                future problems.</text>
                <text>Keep it up! Talking about concepts behind the problems can help you to
                understand them better.</text>
        </message>
</message-set>
<message-set>
        <name>lowLevelHelp</name>
```

```
        <message>
            <text>[Tutor], think about the last help you gave. Why did you say that? Can you
            explain more?</text>
            <text>[Tutor], when you explain a step to your partner tell them why they should
            be doing the step.</text>
            <text>[Tutor], after giving help check to see your partner understands what to do
            and why.</text>
            <text>[Tutor], when giving a hint, get your partner to figure out the next step for
            themselves.</text>
            <text>[Tutor], to help your partner understand, explain how the equation can relate
            to the real world.</text>
            <text>[Tutor], when explaining something, you can pretend the letters in the
            problem are numbers.</text>
            <text>[Tutor], when helping, use examples or facts your partner already
            understands.</text>
        </message>
    </message-set>
    <message-set>
        <name>noStartersWithHelp</name>
        <message>
            <text>[Tutor], remember to use the buttons "ask why", "explain why wrong",
            "hint", or "explain next step" when you help your partner.</text>
            <text>[Tutor], the buttons underneath the chat let your partner know how you're
            trying to help them.</text>
            <text>[Tutor], use the buttons underneath the chat before you write your message
            to plan how to help.</text>
            <text>[Tutor], think about whether "ask why", "explain why wrong", "hint", or
            "explain next step" best describes what you last said.</text>
            <text>[Tutor], when you give help next, think about which button describes what
            you want to say.</text>
        </message>
    </message-set>
    <message-set>
        <name>startersWithNoHelp</name>
        <message>
            <text>[Tutor], if you're not giving help, use the "other" button to label your
            chat.</text>
            <text>[Tutor], think about whether the button you just used to describe your chat
            made sense.</text>
            <text>[Tutor], when you give help next, think about which button describes what
            you want to say.</text>
        </message>
    </message-set>
</messages>
```