Supporting Effective Collaborative Learning in a Computer Science Intelligent Tutoring System

by

RACHEL HARSLEY
B.S., Vanderbilt University, 2011

Thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2017

Chicago, Illinois

This thesis is dedicated to my grandpa, **Walter Peterson**, who believed I would become a doctor long before I could even imagine...

ACKNOWLEDGMENTS

I'd like to thank God for the strength and wisdom to complete this journey.

Barbara Di Eugenio, as my advisor and head of the Natural Language Processing Group at UIC, you have been a constant source of inspiration and counsel from the moment I first considered attending UIC. Since then, you have continued to help me to define my research topic and in my overall professional and personal development. You are the type of adviser any student would want. This dissertation would not be possible without your responsiveness, smarts, and kindness.

Thank you to Nick Green and Sabita Acharya for your help with in-class experiments. Nick, you have worked with me to think through classes and research since the moment we met at orientation. The remainder of the NLP Group is awesome as well. Thank you all for your ideas. Also, thank you to the additional members of my committee, Tom Moher, Susan Goldman, Cynthia Taylor, Davide Fossati, and Tiffany Barnes for your guidance on my research direction.

Thank you so much to my parents, Tommie and Vanassa, for your prayers and constant support. I love you all. Your example and persistence has made this journey possible for me. Thank you to my siblings, Robert (Ericka) and Thomas, and my best friends Jon and Janna. You all have been there for me for as long as I can remember. Also, to LaShonda, you have consistently upheld me with your love and encouragement. Thank you.

ACKNOWLEDGMENTS (Continued)

Finally, if you had any part in my progression and development, if you are family, a friend, or an associate, thank you. Special thanks to my grandparents for your continued prayers.

This work was supported by the Intel GEM Fellowship 2012–2013, Abraham Lincoln Fellowship 2015–2016 from the University of Illinois at Chicago, and Award NPRP 5–939–1–155 from the Qatar National Research Fund.

REH

CONTRIBUTION OF AUTHORS

Chapter 4 presents a published manuscript (Harsley et al., 2017) for which I was the primary author. Davide Fossati and Barbara Di Eugenio assisted in the editing and revision process. Nick Green collected and shared data for the individual ChiQat condition.

Chapter 5 presents two published manuscripts (Harsley et al., 2016; Harsley et al., 2017) for which I was the primary author and driver of the research. Barbara Di Eugenio contributed to discussions about system design and edited the manuscript. Davide Fossati assisted in the editing and revision process. Sabita Acharya and Nick Green helped facilitate the classroom experiment and data collection.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

ASR	Automatic Speech Recognition
CIT	Collaborative Intelligent Tutoring System
CS	Computer Science
CSCL	Computer Supported Collaborative Learning
CSE	Computer Science Education
GUI	Graphical User Interface
ITS	Intelligent Tutoring System

SUMMARY

The standard paradigm of an Intelligent Tutoring System (ITS) exclusively affords one-on-one learning between a student and a computer tutor. However, the benefits of collaborative learning are extensive and include learning for transfer and development of higher order thinking skills such as planning. With this research as motivation, my work shifts the ITS paradigm to accommodate multiple-student, computer-mediated learning. The primary research objective is to investigate how design choice within a Collaborative Intelligent Tutoring System (CIT) impacts student learning and interaction.

To begin, we synthesized research from Computer Supported Collaborative Learning and ITSs to develop a framework for design and evaluation of CITs. From there, we grounded our development and assessment of three versions of our CIT based on this framework. Over 200 students used the systems in the classroom and on average experienced significant learning gains. We compared our collaborative ITS to its one-on-one counterpart and found that students learn equivalently while being more efficient in timing, example use, and code correctness. Additionally, the evaluations showed that design choice influences student interaction such as symmetry, planning, and on-topic discussion. Finally, we discovered that dialogue-based analysis can provide significant correlations to measures of group cognition.

Overall, my current research is motivated by my personal experience as an underrepresented student in the field with a strong desire to be connected to others as I learn CS.

CHAPTER 1

INTRODUCTION

Intelligent Tutoring System (ITS) research explores how to provide educational, tutorial services in user adaptive ways to benefit learning. In examining its counterpart, human tutoring, we understand that the effectiveness of the tutor often coincides with their ability to accurately assess a student's knowledge on an ongoing basis and adaptively (both proactively and reactively) share their expertise, all in real-time. Much is to be said of the great strides Intelligent Tutoring Systems (ITSs) have made in replicating this non-trivial human tutoring process. Some ITSs have even reached and surpassed student learning gains achieved in comparison to human-tutoring interactions (Fletcher and Morrison, 2012); (Fossati et al., 2009); (Kulik and Fletcher, 2015); (Ritter et al., 2007).

ITSs have traditionally been geared toward one-to-one student learning. Originally a constraint due to physical hardware ability, this feature of the one-to-one model is baked into the framework of ITS literature (Carbonell, 1970); (Paviotti et al., 2012). Moreover, given the support of the education research community which demonstrates the superiority of one-to-one student teaching in comparison to one-to-many and other models, the goal of individual support for learning has become widespread (Bloom, 1984); (Kaptelinin, 1999).

Yet, in recent years, the paradigm has begun to shift and researchers have started to design and evaluate ITSs that support collaborative learning. The distinct but related field of Computer Supported Collaborative Learning (CSCL) explores how students learn in collaborative settings and how technology can support this collaboration. Benefits of collaborative learning include increased group performance as well as individual performance. Moreover, collaborative problem solving is consistently associated with higher order thinking skills including planning, reflection, and metacognition (Kaptelinin, 1999).

Domains of intervention in this newly synthesized area of Collaborative Intelligent Tutoring Systems (CITSs) include math, science, and even Computer Science education. A CITS aims to capitalize on the same power of human ability as stated above. However, it extends its scope and allows students, as a team of tutees, to mutually assess their knowledge on a continuous basis and adaptively share expertise in real time along with the tutor. The tutor thus aims to ameliorate this natural interaction in addition to promoting domain learning just as a standard, non-collaborative ITS.

This dissertation aims to address the following research questions:

1. How can we support collaborative learning in an ITS in the domain of computer science?
2. What is the effect of collaborative support on student learning and interaction?
3. What are the salient aspects of student interaction that correlate to learning?

In the remainder of this chapter, I will provide additional background to motivate the research, an outline of my approach to answering the questions at hand, and an overview of remaining chapters.

1.1 Supporting Collaboration

With the benefits of collaboration in mind, it would seem fine to simply place two students together and allow them to work with the ITS just as if they were engaging with the system one on one. However, for both practical and theoretical reasons this setup would not be ideal. Practically, systems created for individual use do not always lend themselves to collaboration without designer adaption of the system. Moreover, theoretically, collaborative learning and CSCL literature shows that collaboration in and of itself does not guarantee more effective learning (Dillenbourg et al., 2009); (Kaptelinin, 1999). Instead, the tutoring system must provide activities which trigger learning and cognitive mechanisms (Dillenbourg, 1999).

1.1.1 Interaction

Given that effective collaboration does not necessarily follow simply by placing students in groups, CSCL research has focused on understanding and promoting collaborative interactions that yield meaningful learning outcomes for students. These outcomes can be measured across the particular learning domain of the CSCL exercise (i.e. math, or CS education) and even the student's collaborative behaviors themselves.

Thus, in a CITS, the role of the tutor extends to include an analogous human role of a group facilitator, in addition to the adaptive domain-centered feedback that drives most ITSs. Moreover, the learners' social context and interactions become an object of analysis and design for the CITS (Israel and Aiken, 2007); (Kaptelinin, 1999). In this regard, the tutor becomes responsible for ensuring group dynamics are such that domain learning is optimal along with collaborative skill practice.

1.1.2 Structure

The role of the CITS in serving as a facilitator lies on a scale as depicted in Figure 1. On the far left, the CITS is hands off in terms of collaborative interaction management, or guidance.

In this capacity, the CITS focuses on domain learning just as a standard ITS. Though students may work as collaborators while using the system, the CITS does not aid them in collaborative activities. However, on the other end of the scale, the CITS fully manages both domain learning and the state of collaboration. The CITS's role is then to ensure student's 1) collaborate effectively 2) learn domain content 3) learn better collaborative behaviors.

As CITS research is moderately new, the question arises as to wherein on the spectrum students benefit the most. Certainly, in the research field of CSCL, it has been established that students gain from the system's collaboration guidance (Dillenbourg et al., 2009). However, this may not be the case for a CITS. Perhaps the guided and adaptive nature already provided by necessity and as part of standard ITS design in itself provides enough structure for collaborators to succeed.

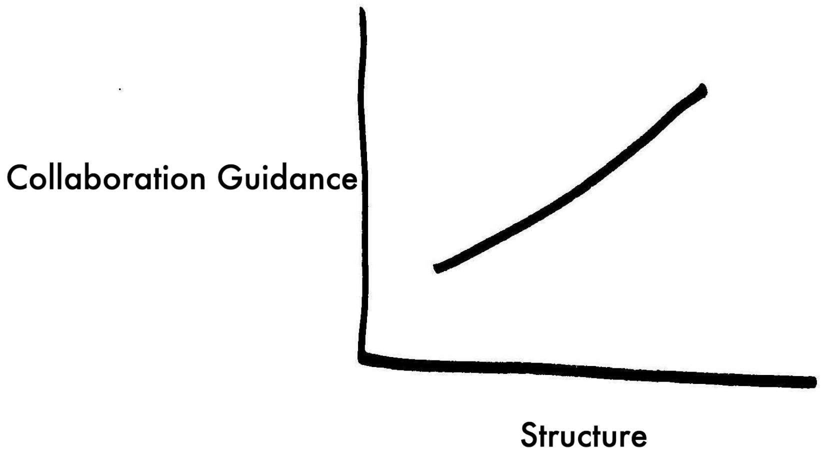


Figure 1: Structure vs Collaboration Guidance in role of the CIT

1.1.3 Modelling

A key part of the ITS is its student modeling component. In plain terms, this is equivalent to a human tutor's ability to gather whether or not a student understands a given topic. For example, if the topic were exponents, a human tutor may make sure the student has a knowledge of multiplication and division as necessary building blocks. In the ITS community, the student model accomplishes this same task. It is the system's computational representation of a student's knowledge, even down to the particular problems the student may have solved or left incomplete.

Depending on the nature of the ITS task, accommodating more than one student may be simple. For example, a system where students work individually on problem sets could yield a complementary CITS implementation through the addition of a shared communication portal for collaboration. Even systems where students participate in different, assigned roles in distributed settings afford more straightforward crossover from ITS to CITS in terms of modelling because student actions are easily delineated and tracked (Walker et al., 2009b). Unfortunately, difficulty arises once students are in a collocated space and sharing the same desktop state. Unlike a human tutor, the standard ITS has no innate method of determining who is performing what actions and thus assessing and aligning the proper knowledge level to a given collaborator.

1.2 Learning on Multiple Fronts

It is clear that the role of the CITS includes assisting students in domain specific learning and in their ability to collaborate. In some sense, the CITS can be seen as a coach for collaboration, especially as the amount of structure increases. In many fields of study, the knowledge of how to collaborate is just as essential as the domain knowledge itself. Though viewed as a soft skill, the implications for computer scientists with high and low collaborative abilities are far reaching (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013). It is essential for programmers to learn how to effectively collaborate both in large and small scale projects. Even more, a growing industry standard of “pair programming” motivates the need for addressing collaborative skill building. In this practice, two users share the same computer, keyboard and mouse. One user serves as the *driver* while the other serves as the *navigator*. Both users work to solve a given problem or programming task. The driver's role is to control keyboard and mouse, to physically type the code, and to reason about solutions. The navigator's role is to think about the direction of the code and help the pair avoid possible pitfalls and reach a solution. Benefits of pair programming are well documented and include increases in student enjoyment, confidence, learning, learning speed, and coding efficiency (Salleh et al., 2011a) (Porter et al., 2013) (Pears, 2010). According to a survey of industry pair programmers, the most commonly recognized traits of a successful pair programming partnership include communication and respect (Ally et al., 2005).

1.3 Research Contributions

My research objective is to investigate and understand which design choices for a collaborative ITS for CS Education better support learning of domain content and promote effective collaborative behaviors. In keeping with its collaborative objective and the system from which it has been extended, it is named Collab-ChiQat. My research questions are the following:

1. How can we support collaborative learning in a Computer Science ITS?
2. What is the effect of collaborative support on student learning and interaction?
3. What are the salient aspects of student interaction that correlate to learning?

In order to address question (1) I have developed a framework which sets forth three paradigms for supporting collaboration in an ITS (Harsley, 2014). As the area of CITs research is relatively new it combines important themes in CSCL and ITS including: modeling of the student and domain, group dynamics, collaboration and pedagogical guidance, and technology. Further, in addressing question (1), I created two versions of the Collab-ChiQat system, *unstructured* and *semistructured*, each aligned with a CITS paradigm. These conditions were developed in parallel.

To address question (2), I conducted experimental studies in the classroom in order to compare the standard non-collaborative intervention with the collaborative system and to compare the two methods of collaboration structuring themselves. Over 150 students participated in these studies. In each experiment, students faced the same domain-learning task, a foundational CS education concept. Students were introduced to the concept of pair programming through a brief video before the intervention. Students were then able to choose and switch their roles for engaging with the system throughout the intervention. The means of outcome measurement for the system encompassed individual learning via pretest and posttest as well as analysis of the collaborative process and perceptions.

Finally, the data from these studies was used for interaction analysis to understand both the collaborative and learning process and derive a model of effective learning and collaboration. This analysis necessitated logging of all peer-to-peer and student-tutor interactions. This includes, but is not limited to, speech interaction; turn taking behavior, interface interactions, and problem solving steps. This analysis and subsequent model of learning addresses question (3). Moreover, it motivated the design decisions and features instrumented in a subsequent system system iteration labeled *hybrid*. I evaluated the *hybrid* system with a final classroom experiment where an additional 100 students used the system and experienced significant learning gains.

Both students and the research community have benefited from this work. Notable contributions include the following:

- I developed a framework for designing and classifying CITSs based on established ITS and CSCL research
- I reconceptualized and extended an existing tutoring system to create Collab-ChiQat which supports collaborators as they learn a foundational CS topic
- I designed and implemented multiple strategies for facilitating collaboration through tutoring system feedback based on CITSs paradigms.
- I evaluated three different versions of the collaborative system which allowed over 200 students to learn about linked lists with significant gains equivalent to those working individually
- I collected and analyzed log data in order to model important features leading to student learning and effective collaboration.
- I collected, transcribed, annotated, and analyzed a corpus of student dialogue as they solve problems with the CITS
- I operationalized CSCL concepts of planning and symmetry as actionable features for tutor feedback generation.
- I validated the role of initiative shifts as proxies for detecting knowledge co-construction and subsequent student learning

- I analyzed knowledge convergence and lexical entrainment as operationalized measures of group cognition, a primary CSCL collaborative learning outcome

1.4 Outline

This dissertation is organized as follows:

- Chapter 2 provides a description of the research areas conjoined in the synthesis of this work, specifically CSCL, ITS, and Computer Science Education (CSE). Moreover, we introduce the standard ChiQat tutoring system along with non-traditional CITs.
- Chapter 3 describes a framework for collaborative ITS as well as the design of Collab-ChiQat in application of two structuring paradigms
- Chapter 4 offers perspective on the effect of collaboration through a contrastive analysis of learning and system interaction outcomes in comparison of Collab-ChiQat users and users of the standard, non-collaborative system.
- Chapter 5 compares the effects of design choice on student's collaborative learning process and outcome. It then describes a subsequent model of important features correlated to learning in our CIT.
- Chapter 6 presents results on the effectiveness of the collaboration from the perspective of group cognition. It then offers the design and evaluation of a *hybrid* version of the system which aims to automatically improve collaborative learning.
- Chapter 7 gives conclusions as well as directions for future study and long-term research.

CHAPTER 2

RELATED WORK

2.1 Intelligent Tutoring Systems

With the demand for CS talent at an all time high and growing body of work in computing education, there has been an increase of interest in teaching computing in engaging, automated, and scalable methods (Monge et al., 2015). One such method is the use of an Intelligent Tutoring System. Intelligent Tutoring Systems mark the forefront of research in applied artificial intelligence to education. The origins of this interdisciplinary field date back to the early 1970s (Nwana, 1990). For at least two decades, the term “ITS” has been used to replace the phrase “Intelligent Computer-Aided Instruction” (ICAI) of the same meaning. Traditional ITSs aim to provide user-adapted support during problem-solving processes in a manner resembling of a human tutor (Tchounikine et al., 2010). The intelligent tutor compares student actions to a domain model and uses the subsequent contextual awareness to tailor instructional activities and offer relevant help (Walker et al., 2009a). Although ITSs have not mastered the same level of effectiveness achieved by expert human tutors, especially in regards to cross-domain expertise and pedagogical strategies, multiple research studies show the impact of these systems to significantly increase learning (Fletcher and Morrison, 2012); (Fossati et al., 2009); (Koedinger et al., 1997); (Ritter et al., 2007)

Intelligent Tutoring Systems provide adaptive, user-centric feedback to students as they learn a given topic. The tutors are trained experts in domain content. Corbett, Koedinger, and Anderson (1997) explain the role of the tutoring system this way,

> The goal of intelligent tutoring systems (ITSs) would be to engage the students in sustained reasoning activity and to interact with the student based on a deep understanding of the student's behavior. If such systems realize even half the impact of human tutors, the payoff for society promised to be substantial. (Corbett et al., 1997)

Much of the work of an Intelligent Tutoring System designer lies in shaping the tutor to interact with the student in a way that meets and hopefully transcends the abilities of a human tutor. Intelligent tutoring systems provide several benefits to enhancing student learning including: 1) avoid the one-to-many student teacher model and free teachers to more effectively triage student needs 2) provide students with individual, customized support tailored to their level of proficiency 3) allow students to reach common proficiency goals in an efficient and verifiable manner.

Classical definitions such as the one provided by Conati (2009) often limit the scope of the tutor to interactions and modeling of an individual learner:

> Intelligent Tutoring Systems (ITS) is the interdisciplinary field that investigates how to devise educational systems that provide instruction tailored to the needs of *individual* learners, as many good teachers do. (Conati, 2009)

Research in this field has provided both techniques and systems that adaptively support students as they learn and problem solve in a plethora of domains (Conati, 2009). Moreover, the focus of ITS on an individual learner (depicted in Figure 3) is an important distinction between ITS and Computer Supported Collaborative Learning (CSCL).

2.1.1 Structure of ITSs

Early work in ITS research led Self to argue that computer aided instruction needed a representation of "what is being taught, who is being taught, and how to teach him/her" (Self, 1974). This led to the acceptance of a three-module architecture for ITS. The current standard for ITS construction now includes a fourth module, "user interface" and is depicted in Figure 4 (Nkambou et al., 2010); (Nwana, 1990). Methods of ITS implementation vary greatly while they remain consistent to this baseline architectural framework.

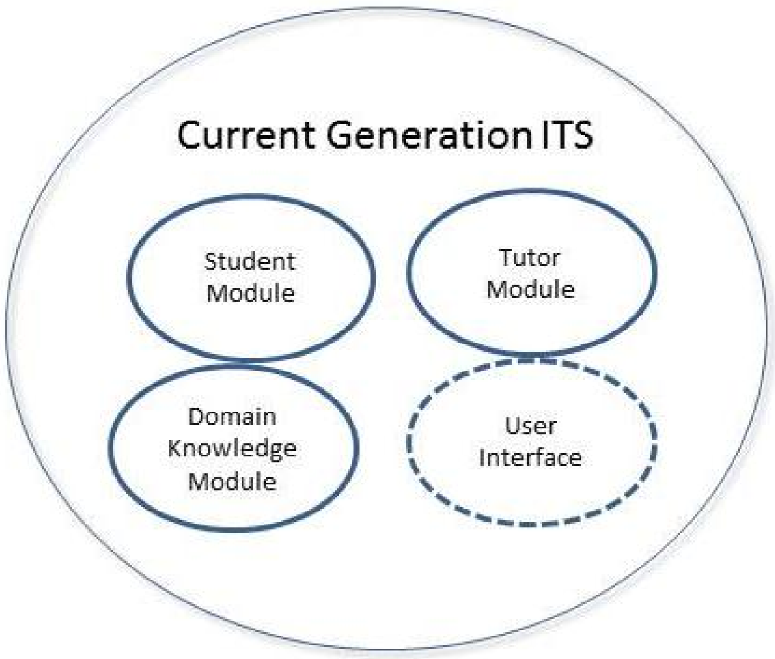


Figure 2: View of the four-module ITS architecture. This representation is standard for current ITS after the fourth module, User Interface became recognized as a basic component.

The domain knowledge module, also known as the expert knowledge module, allows the tutor to represent and reason regarding the learning domain to be conveyed. Three standard types of knowledge models frequently used are rule-based, constraint based, and expert system (Paviotti et al., 2012). Historically, much work has gone into discovering and building adequate domain knowledge representation. However, recent research efforts have begun to explore how this knowledge could be automatically sourced through machine learning, such as extraction from the work of previously tutored students (Floryan and Woolf, 2013). The knowledge module is also used to assess the tutee's overall progress via comparison to the knowledge module.

The student model module dynamically represents the knowledge and skills of the student. The model works in conjunction with the knowledge module to infer progress in problem solving as well as estimations of knowledge acquisition (Kersey et al., 2009); (Nwana, 1990, p. 10). Self's list, as summarized by Nwana (1990), of a student model functions, though non-exhaustive, is helpful for understanding the model's potential roles:

(1) Corrective: to help eradicate bugs in the student's knowledge.

(2) Elaborative: to help correct 'incomplete' student knowledge.

(3) Strategic: to help initiate significant changes in the tutorial strategy other than the tactical decisions of 1 and 2 above.

(4) Diagnostic: to help diagnose bugs in the student's knowledge.

(5) Predictive: to help determine the student's likely response to tutorial actions.

(6) Evaluative: to help assess the student or the ITS.

Over the last twenty years, several studies have focused on the design and testing of user profiles as means to model the student. These profiles contain information regarding the characteristics and habits of the students such as personality, behavior, learning style and affective states (Paviotti et al., 2012). Much research emphasis is placed on inferring characteristics that are not directly explicated beyond the student knowledge level. As a whole, accurate student modelling is essential for adaptable and supportive ITS.

The tutoring module both encompasses and enacts the instructional design paradigm of the ITS. Linkage with the student module enables it to decide when, how, and what pedagogical activities to present. These cover a wide range from hints to explanations, to changes in activities or tasks. Thus, the tutor module is the ultimate source for pedagogic interventions (Nwana, 1990). Just as with the other components, subtle changes to this module can drastically affect tutoring outcomes. For example, in some situations, it may be best to allow the student extended time in solving a problem before interrupting, while in others a student may become lost without guided and proactive assistance.

The user interface module controls interaction between the student and system. It bi-directionally connects the ITS's internal representation to communicate with the student. Current ITSs provide a variety of interface mechanisms through graphics, text-entry, key-board, and mouse-driven events and menus (Ahuja and Sille, 2013).

2.2 Computer Supported Collaborative Learning

The field of Computer Supported Collaborative Learning focuses on how students learn in collaborative settings and how technology can enhance collaborative peer interaction and work (Dillenbourg et al., 2009; Tchounikine et al., 2010; Magnisalis et al., 2011). Longstanding research has shown that both cooperative and collaborative interactions among students are beneficial to learning (Lehtinen et al., 1999). In fact, the learning gains are often more profound than those achieved by the best of individual learners (Kaptelinin, 1999). Students also benefit from development of higher order thinking skills such as planning, reflection, and metacognition, and they experience learning for transfer (Kaptelinin, 1999; Walker et al., 2009b). According to Woolf (Woolf, 2010), Collaborative strategies emphasize "positive interdependence (of task, identity, role and goals), individual and group accountability (challenging and motivating every student), and authentic interaction (brainstorming, planning, social and team building skills, and solidarity)" . This definition is shared by several CSCL and learning scholars (Dillenbourg, 1999; Soller, 2001; Strijbos, 2011).

However, assigning students to a group and charging them with a task does not ensure that students will engage in effective collaborative learning behavior (Kobbe et al., 2007; Soller, 2001). It is not uncommon for groups to struggle with dysfunction that often stems from an unbalance of participation, lack of leadership, understanding, and encouragement (Soller, 2001). Thus, CSCL requires careful construction of the collaboration so that interactions benefit the individual and group (Dillenbourg, 1999).

Research in CSCL accounts for the social and construction elements of the learning process in its application of technology (Nkambou et al., 2010). Soller's work to understand the social interactions that support learning shows that students must ask questions, explain and justify opinions, articulate reasoning, and elaborate and reflect on knowledge (Soller, 2001). Further, Dillenbourg states that "knowledge generative" interactions such as giving explanations, engaging in argumentation, negation, conflict resolution, or mutual regulation lead to positive learning outcomes (Dillenbourg et al., 2009). Studies also suggest these kinds of interactions lead to deep learning, or the process of learning for transfer (Walker et al., 2009a). As these interactions do not necessarily emerge without guidance, CSCL applications structure group activity in order to promote these behaviors (Magnisalis et al., 2011); (Tchounikine et al., 2010).

2.2.1 Structure of CSCL

The classical approach to providing support in collaborative settings has been through macro-scripts, or pedagogical scripts (Tchounikine et al., 2010). The scripts introduce structure and constraint, and guide the collaborative interaction between students mediated by a computer-based system, whether in part or whole (Tchounikine et al., 2010). They track students' progress with the script sequences, prompt engagement in activities, and offer additional resources as needed (Kobbe et al., 2007). Thus, the work of the CSCL system rests in the design and implementation of situation, interaction, and processes that promote collaborative learning (Dillenbourg, 1999)

Kobbe's work synthesizes this objective in a concise framework intended to increase the reusability of scripts among researchers and practitioners (Kobbe et al., 2007). The work describes a thorough description of the components and mechanisms of a script as seen in Figure 5.

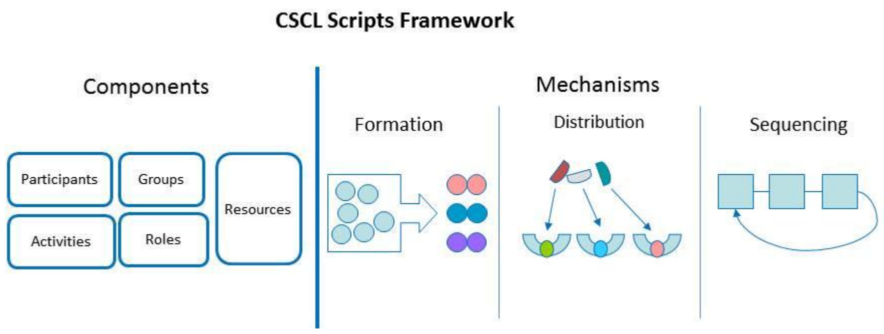


Figure 3: Scripts components and mechanisms.

Script components consist of participants, “the activities that they engage in, the roles they assume, the resources that they make use of, and the groups they form” (Kobbe et al., 2007). Script mechanisms include group formation (the distribution of participants over groups), component distribution (the distribution of components over participants) and sequencing (the distribution of components and groups over time (Kobbe et al., 2007).

Until recently, traditional script support has been fixed in that the same level of support is consistent across students regardless of ability or interaction (Walker et al., 2009a). Due to the nature of fixed support, the same script may provide insufficient support for poor collaborators, unnecessary support for experienced collaborators, and on-target support for others (Walker et al., 2009a). Recent developments in CSCL research have shifted their focus to developing systems that offer dynamic adaption to student interaction conditions. Magnisalis describes this type of CSCL system as “a helpful experienced partner who intervenes unobtrusively and just in time to support group learners in achieving a productive level of interaction and therefore in accomplishing their task” (Magnisalis et al., 2011).

2.3 Collaborative Intelligent Tutoring Systems

Both the CSCL and ITS community shape the current context of technology-enhanced learning. CSCL shows that students learn effectively in groups and computational environments can support collaboration. Further, ITS purposes to offer adaptive learner support that models an individual user, the learning domain, and the tutoring strategy. Tchounikine (2010) soundly juxtaposes the fields as follows:

> Whereas Intelligent Tutoring Systems address issues such as the analysis and understanding of learners' activity and production, problem solving or interaction control, classical CSCL systems have not addressed these issues at all. Whereas ITS research has, since its inception, leveraged Artificial Intelligence techniques, CSCL research has instead focused on HCI issues related to providing students with a good experience of communicating with their fellow students. (p. 459)

Recently, research efforts have focused on merging the affordances of both areas to capitalize on the benefits of group learning and adaptive support (Magnisalis et al., 2011). Several researchers in the CSCL community are exploring how adaptivity, automated analysis, and feedback integrate into CSCL approaches (Tchounikine et al., 2010). Similarly, ITS researchers are extending their individual use ITS systems to accommodate collaborative support through tools such as topic detection and feedback in group chat (Kumar and al, 2007; Walker et al., 2009b; Olsen et al., 2014; Olsen et al., 2015). The CSCL community's work on shaping effective collaboration must guide the creation of ITS systems and their integration of collaborative capabilities. In general, the creation of these Collaborative Intelligent Tutoring Systems (CITS) is considered to be more laborious than a typical ITS because it accounts for group dynamics and social relations in addition to pedagogical considerations (Magnisalis et al., 2011). In any case, researchers are motivated by the established benefits of collaborative learning along side adaptive support.

2.4 CS Education

The push for widespread computer science education (CSE) has received much publicity in recent years and has coincided with initiatives from the White House down (Richtel, 2014). The demand for workers with Computer Science (CS) skills is at an all-time high. Tech leaders across the industry, including Facebook CEO Mark Zuckerberg, have been quite vocal and supportive of the need for CSE beginning even at the K-12 level (Richtel, 2014). Cities such have Chicago have embraced this push and launched programs with the support of the National Science Foundation to prepare their students with computer science skills (Foundation, 2016). At the same time, coding “bootcamps” have emerged for older CS learners that place them in immersive, extensive, and expedited learning experiences (Gonzalez, 2015).

Researchers have implemented several automated and adaptable approaches to supporting student CS learners. Early work on ITSs for CS education focused on dialogue-based interaction to discuss a novice’s program design (Lane and VanLehn, 2004). More recent ITSs such as JavaTutor help students learn introductory CS through feedback on their problem solving as well as affective states (Ezen-Can and Boyer, 2014; Wiggins et al., 2015). While other focus on tutoring for SWL and database design through use of worked out examples and eye tracking (Najar and Mitrovic, 2013).

Additionally, the growing model of CS Education has shifted to an emphasis on collaborative work which has been correlated to both higher retention rates of underrepresented students and better learning (Porter and Simon, 2013; Porter et al., 2013). Specifically, CS educators have adopted the practice of pair-programming for the classroom (Porter et al., 2013; Salleh et al., 2011b) which is discussed more extensively in Section 2.4.2. Peer instruction has also become another successful methods of student support (Porter et al., 2011; Porter et al., 2016). In this pedagogy, students are active participants in the course through individual consideration and small group discussion of conceptual questions.

At the university level, CS has traditionally suffered from high attrition rates (Porter et al., 2013). This is especially true for underrepresented groups including women and minorities (Washington et al., 2015). A factor that has contributed to low retention is the student difficulty in grasping foundational CS concepts. This, in turn, leads to low performance and subsequently low motivation in pursuing the discipline (Beaubouef and Mason, 2005; Howles, 2009; Lewis, 2010). The most fundamental data structures and the algorithms used to manipulate them include linked lists, binary search trees, and recursion. Moreover, the joint task force of the two major CS professional societies, ACM and IEEE, have both noted the difficulty of these topics for students (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013).

2.4.1 Linked Lists

The domain of focus for my intervention is the linked list data structure. Linked lists allow for data to be stored in individual nodes and connected to each other via pointers. Access to the data must occur through pointer reference and requires sequential traversal of the data to perform common operations including search, insertion, and deletion.

The list is often depicted in a graphical representation as boxes (nodes of data) and arrows (pointers). This representation aids in disambiguating the complexity of the data structure, especially for novice CS learners. An example list is shown in Figure 2.

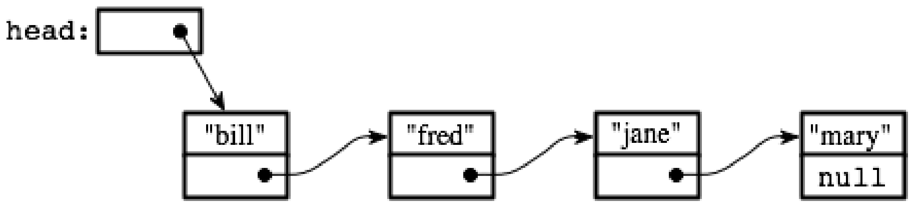


Figure 4: Example of a linked list.

2.4.2 Pair Programming

Pair programming is a practice that situates two coders at a single workstation. The coders assume the roles of *driver* and *navigator*. The driver types the code while the navigator checks for errors and plans the overall coding strategy. The practice began as an extreme programming, agile software development methodology within industry, and has subsequently found success in the classroom.

Many reports set forth the benefits and drawbacks of pair programming along and its usage within both industry and education (Maguire et al., 2014; Rodriguez and Boyer, 2015; Williams and Kessler, 2000; Walle and Hannay, 2009; Braught et al., 2011). These benefits include increases in student enjoyment, confidence, learning, learning speed, and coding efficiency (Salleh et al., 2011a; Porter et al., 2013; Pears, 2010). However, the benefits of pair programming are often met with apprehension on the part of instructors and some students. Traditionally viewed as an individual task, coding has a stigma of solitude that influences instructors to believe learning can only occur while designing, coding, debugging and testing individually. In fact, working with others on a program is often labeled “cheating” (Van Toll et al., 2007). Moreover, instructors often fear that if two student work together, only one student will do all of the work leaving the other with little engagement and learning (Chigona and Pollock, 2008).

Williams served as the catalyst for exploring the role of pair programming in education after its introduction as an extreme programming, agile software development methodology (Williams et al., 2000). Since then, preparing students to effectively collaborate and work in teams has become a core component of CS curriculum and accreditation requirements (Commission, 2014) (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013).

Within industry, an earlier study of pair programming paired programmers randomly and measured programmers productivity in terms of code produced over time and code error rate (Jensen, 2003). A development team consisting of 10 individuals, five pairs, experienced a 127 percent gain in code production per month and a decline of three orders of magnitude in coding errors.

In terms of pair programming use in the classroom, its benefits have also concerned increases in coding quality and speed. In Williams’ foundational work on classroom pair programming, both students’ self reported time spent on assignments and coding quality (based on the percentage of test cases which passed) were assessed (Williams and Upchurch, 2001). A more recent study compared individual versus pair programmers in terms of duration and effort in solving two coding problems in two sessions (Gmez et al., 2013). Duration captured the total time spent coding while effort denoted the amount of person-hours, thus *duration*2* for pairs. The study examined seven pairs and seven individuals and found that pair programmers had significantly less duration while individual programmers had significantly less effort.

Pair programming has also been shown to improve student confidence and enjoyment in CS courses. In McDowell's work, students were assigned to a programming lab that either held individual or pair programming exercises for the duration of the semester. The study showed that (McDowell et al., 2006) pair programmers had higher program quality (measured via scores on programming assignments), course enjoyment levels, and significantly higher confidence in their programming solutions. With these types of outcomes, use of pair programming has helped to spur retention of CS students in introductory and subsequent courses (Porter and Simon, 2013) (Porter et al., 2013). In terms of benefits to individual students participating in classroom pair programming, Braught has shown that individual programming skills of lower SAT students improve, and pair programmers are more confident in their work and are more likely to successfully complete the course (Braught et al., 2010).

2.5 Building on Our Previous Work

2.5.1 Standard ChiQat Tutoring System

Standard, non-collaborative ChiQat is an ITS for CS Education developed by the NLP Group at UIC. It aims to improve a student's practical learning of the linked list data structure among other foundational CS data structures and algorithms. In the linked list lesson, a problem is presented to a student in both textual and graphical representation. The student is then able to programmatically solve the problem. Moreover, the system provides relevant positive and negative feedback to the student in a manner analogous to the one-on-one human tutoring experience from which the system was derived. Example problem types involve linked list node insertion and removal in addition to other more complicated operations such as searching for a node in the list. The architecture of standard ChiQat is depicted in Figure 5 and is comprised of six major components as follows: graphical user interface, problem model, constraint evaluator, feedback manager, procedural knowledge model, and student model which are each described further in Section 3.3. There is abundant reference detailing the architecture, tutoring strategy, and subsequent learning outcomes of students using the system (Fossati et al., 2008; Fossati et al., 2009; Fossati, 2013; Green et al., 2015; AlZoubi et al., 2015; Harsley et al., 2016a)

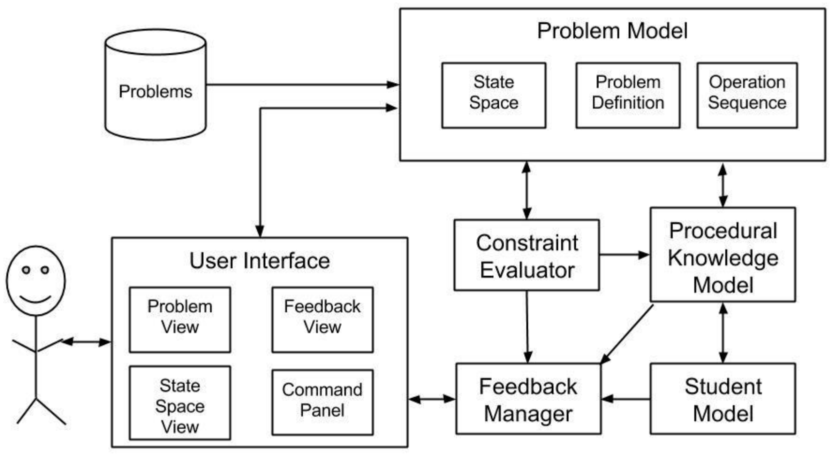


Figure 5: Standard, non-collaborative ChiQat Architecture

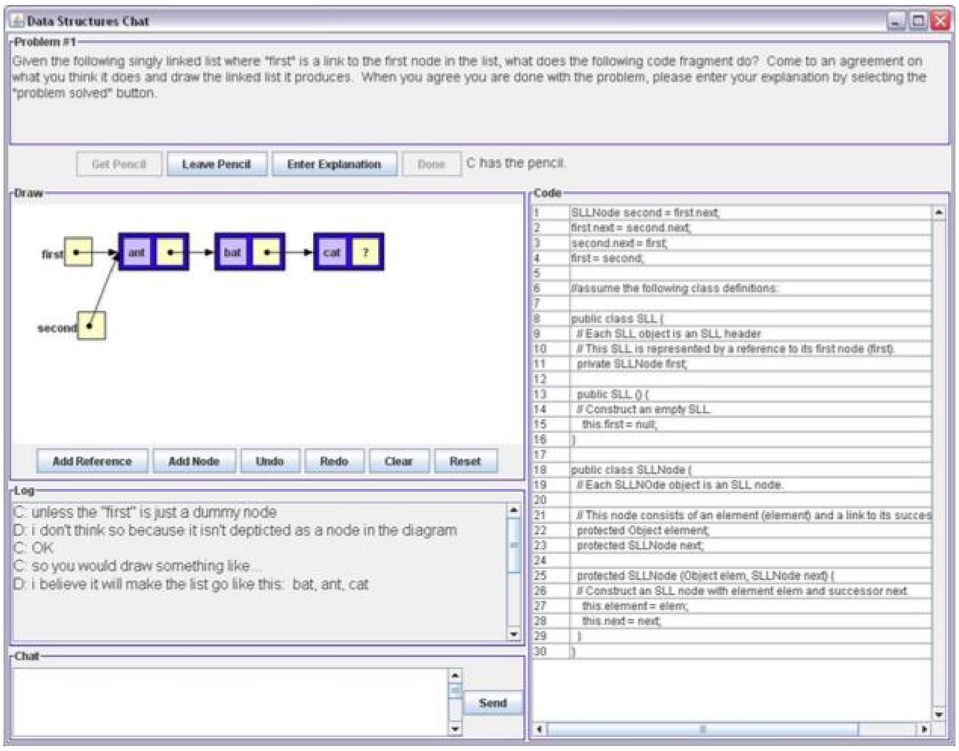
2.5.2 A Baseline Dataset from KSC-PaL

Prior work in our research lab evaluated a collaborative tutoring system through the design of a *simulated* peer learning agent. KSC-Pal, the thesis work of Cynthia Kersey, situated students with a virtual peer as they learn foundational CS topics including linked lists, stacks, and binary search trees. Thus, my research serves as the second examination of collaborative learning for the UIC NLP group. My work differs from Kersey in that I do not aim to simulate a peer. Instead, the CITS will facilitate more productive human-human pair interaction and still function as an expert tutor for domain content. Nonetheless, Kersey's work provides a baseline data source for analyzing the features of collaborative interaction that lead to student learning.

Kersey's overarching goal was to create a virtual peer which would act in lieu of one of the students. In order to design the agent, Kersey first examined human-human pair interaction all captured through the KSC-Pal interface (see Figure 6). In KSC-PaL's initial data collection phase, students were situated in separate physical locations and communicated through the system's chat interface as they worked to solve programming problems. Her corpus consisted of 15 pairs. For each problem, students either corrected or synthesized code as a pair. GUI changes and chats were automatically logged for analysis. The subsequent analysis of the human-human interaction then motivated the design of the peer learning agent.

Kersey examined the role of two types of initiative shifts. Dialogue-initiative tracks the conversation leader through exchange type (i.e. assertions, commands, questions, and prompts) while task initiative tracks leadership in problem solving (Kersey-Howard et al., 2015). Task initiative encompasses "any action by a participant to either achieve a goal directly, decompose a goal or reformulate a goal" (Kersey-Howard et al., 2015). Kersey hypothesized that initiative shifts are correlated to knowledge co-construction, or episodes in which students build a shared meaning of a concept during problem solving (Kersey-Howard et al., 2015). KCC episodes are known as a pivotal markers of overall learning during collaborative exercise. She coded the corpus for dialog initiative, task initiative, and knowledge co-construction episodes.

Ultimately, Kersey's work found that task initiative shifts were indeed indicative of KCC episodes and that both dialogue and task initiative were significant correlates to learning. Thus, she was able to design a virtual peer that monitored the number of initiative shifts in the chat dialogue and proactively encouraged task initiative if the level of initiative shifts fell below a given threshold (Kersey-Howard et al., 2015). In doing this, the virtual peer was created in such a way to encourage KCC.



Given the following singly linked list where "first" is a link to the first node in the list, what does the following code fragment do? Come to an agreement on what you think it does and draw the linked list it produces. When you agree you are done with the problem, please enter your explanation by selecting the "problem solved" button.

C has the pencil.

```
SLLNode second = first.next;
first.next = second.next;
second.next = first;
first = second;

//assume the following class definitions:

public class SLL {
 // Each SLL object is an SLL header
 // This SLL is represented by a reference to its first node (first).
 private SLLNode first;

 public SLL () {
 // Construct an empty SLL.
   this.first = null;
}

public class SLLNode {
 // Each SLLNOde object is an SLL node.

 // This node consists of an element (element) and a link to its succes
 protected Object element;
 protected SLLNode next;

 protected SLLNode (Object elem, SLLNode next) {
 // Construct an SLL node with element elem and successor next.
   this.element = elem;
   this.next = next;
 }
}
```

```
C: unless the "first" is just a dummy node
D: i don't think so because it isn't depticted as a node in the diagram
C: OK
C: so you would draw something like...
D: i believe it will make the list go like this:  bat, ant, cat
```

Figure 6: The KSC-PaL interface. Distributed, student pairs worked on coding and communicated via a chat interface.

CHAPTER 3

COLLAB-CHIQAT: APPLICATION OF A FRAMEWORK FOR COLLABORATIVE INTELLIGENT TUTORING SYSTEMS

(This chapter includes and expands on portions of my paper previously published in *Harsley, Rachel. 2014. "Towards a Collaborative Intelligent Tutoring System Classification Scheme." In **Proceedings Of The 11th International Conference On Cognition And Exploratory Learning In The Digital Age (Celda 2014)***).

3.1 Summary

In order to critically review and analyze any work effectively, a method of evaluation is necessary. This is especially true when the work spans multiple, broad areas of research as do Collaborative Intelligent Tutoring Systems (CITs). Thus, a primary contribution of this thesis was a comparative analysis between multiple studies in this domain resulting in a pioneering classification scheme for CITs that blends ITS and CSCL frameworks (Harsley, 2014). This analysis motivated the structuring of the subsequent investigation of how design choices in a CS CIT influence learning and collaborative interaction. The framework and our investigation's experimental design choices and motivation are presented in this chapter.

3.2 CITs Framework

The scheme consists of three categories: unstructured, semi-structured, and fully structured. These categories serve as points of reference across a spectrum of design choices from far left CITs with limited structuring to far right CITs with expansive structuring. Moreover, we acknowledge that while some interventions fall on a point outside of the three categories, many of the design choices that define the intervention are captured in the scheme dimensions nonetheless. Thus, the classification scheme is intended to support the design and analysis of CITs. [1] Overall, a CIT can be defined and analyzed via the following dimensions:

1. *Modeling:* how the system models learners and whether the learning domain includes collaborative behavior skills.
2. *Group Dynamics:* how groups and roles are determined
3. *Collaboration Cues:* the impetus of the initial collaboration and the timing of ongoing communication.
4. *Pedagogical Guidance*: how the topic is determined and how feedback and activity facilitation is implemented
5. *Technology:* the tools of the system for interaction

This classification scheme encompasses the main operational dimensions of a CIT. A CIT is a technological, collaborative learning tool (technology) that provides systematic support to learners (pedagogical guidance) working within groups (group formation) either assigned or at-will (collaboration cues) and maintained in computational representations as they work

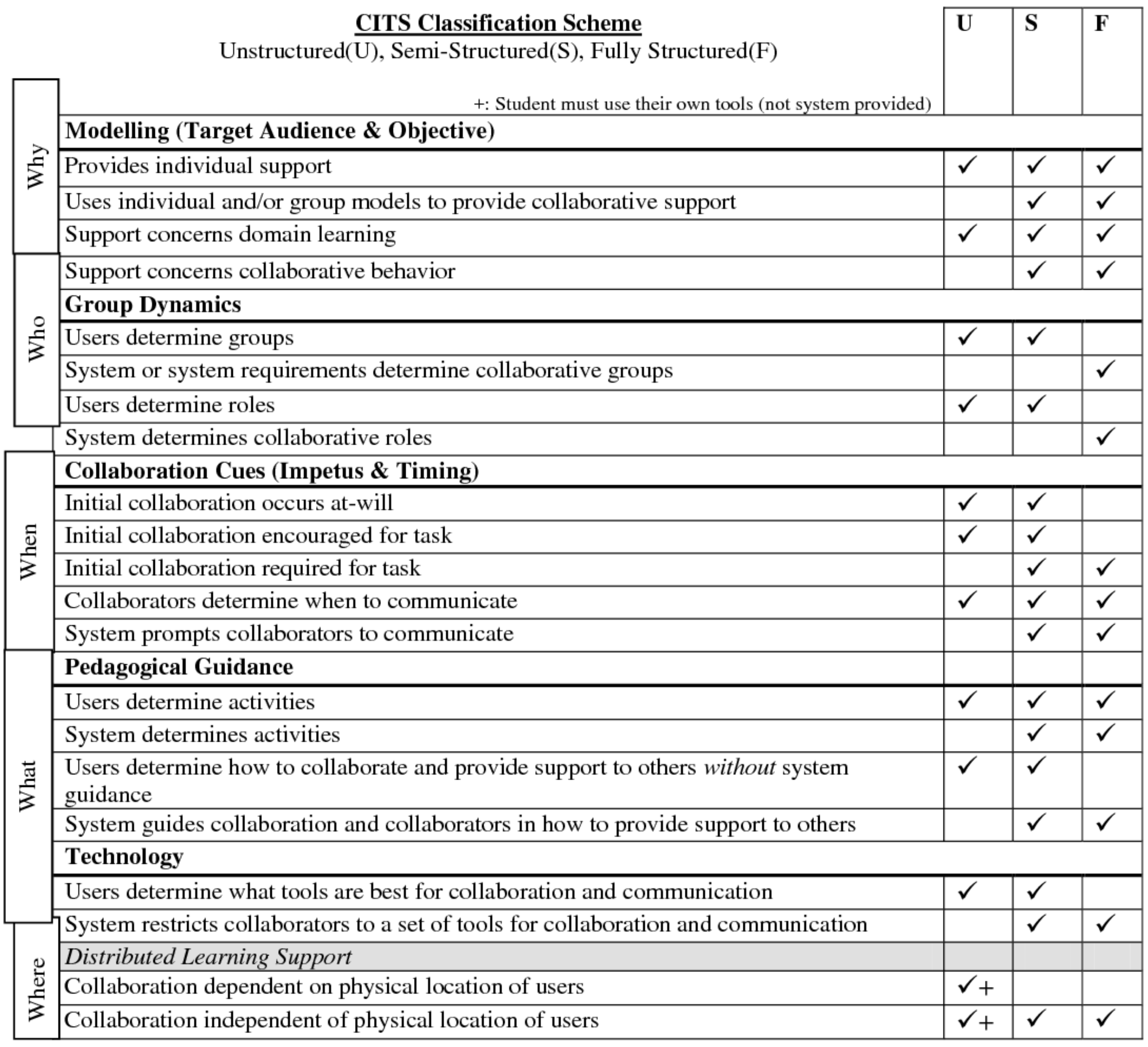
[1]We define the scope of CITs to collaboration to human-human collaborators who work with a tutoring system, not the simulation of a virtual peer(s) with an individual.

to achieve a learning goal (modeling). Table I provides detailed criteria for the classification dimensions along with a mapping to classification classes.

The modeling dimension encompasses the student and domain modules which are components of the basic ITS architecture (as discussed in 2.1.1). The group dynamics dimension accounts for the participants, groups, and roles components and the group formation mechanism characteristic of CSCL architecture. The collaboration cues dimension deals with the sequencing mechanisms of CSCL architecture. The pedagogical guidance dimension concerns both the tutoring module of the basic ITS Architecture and its counterpart within CSCL design, the activities component of scripts. This additional emphasis beyond the modeling dimension is due to the added significance of tutor modeling for collaborative interaction. Finally, the technological guidance dimension entails the resources component and distribution mechanisms of CSCL architecture along with the user interface component of basic ITS architecture.

3.2.1 Unstructured CIT

In the unstructured CIT, like a standard ITS, the system's focus still remains on student grasp of domain knowledge. However, the system accommodates for collaborative work. There is no modeling of the group or pedagogical directives in relation to assisting collaborative behaviors. In this type of CIT, group formation and role forming is also left to the students. Moreover, the students determine the nature of their collaboration in terms of when they communicate and the tools to facilitate their collaboration. For example, while working with a Math CIT, the students may elect to speak with others during each step of problem solving or after an incorrect solution notification. Finally, the pedagogical strategies of the system do not focus on bettering collaboration. Overall, the students must determine how to collaborate with each other without the guidance of the system. Examples of unstructured collaboration interventions include allowing casual collaboration as students work individually with a mathematics tutor and even the addition of a social network with chat and user profiles in another math tutoring system (Arroyo et al., 2012; Virvou and Sidiropoulos, 2013).



	CITS Classification Scheme Unstructured(U), Semi-Structured(S), Fully Structured(F) +: Student must use their own tools (not system provided)	**U**	**S**	**F**
Why	**Modelling (Target Audience & Objective)**			
	Provides individual support	✓	✓	✓
	Uses individual and/or group models to provide collaborative support		✓	✓
	Support concerns domain learning	✓	✓	✓
Who	Support concerns collaborative behavior		✓	✓
	Group Dynamics			
	Users determine groups	✓	✓	
	System or system requirements determine collaborative groups			✓
	Users determine roles	✓	✓	
	System determines collaborative roles			✓
When	**Collaboration Cues (Impetus & Timing)**			
	Initial collaboration occurs at-will	✓	✓	
	Initial collaboration encouraged for task	✓	✓	
	Initial collaboration required for task		✓	✓
	Collaborators determine when to communicate	✓	✓	✓
	System prompts collaborators to communicate		✓	✓
What	**Pedagogical Guidance**			
	Users determine activities	✓	✓	✓
	System determines activities		✓	✓
	Users determine how to collaborate and provide support to others *without* system guidance	✓	✓	
	System guides collaboration and collaborators in how to provide support to others		✓	✓
	Technology			
	Users determine what tools are best for collaboration and communication	✓	✓	
	System restricts collaborators to a set of tools for collaboration and communication		✓	✓
Where	*Distributed Learning Support*			
	Collaboration dependent on physical location of users	✓+		
	Collaboration independent of physical location of users	✓+	✓	✓

TABLE I: DETAILED CLASSIFICATION SCHEME FOR CITS WHICH INCLUDES CRITERIA FOR THE FIVE DIMENSIONS.

3.2.2 Semi-structured CIT

In semi-structured collaboration, the system offers a mix between assisting students in their domain learning and their collaborative interaction. In this class, the modeling uses a combination of individual and group metrics to support collaboration. Moreover, both measures are also used to support domain learning. The group formation and role forming is still left to the students. However, the collaboration cues are such that the system prompts students when to collaborate in addition to allowing for students' at-will interaction. The impetus for the collaboration can follow a range from student's voluntary grouping to the system mandating groups. Finally, the system guides the students on how to provide support to others, and can offer activities to guide the collaboration as well as the tools for the collaboration. In the biology tutoring system, Rashi, students are guided in a semi-structured manner with system guidance on more and when to collaborate (Dragon et al., 2010). Moreover, another example of semi-structured collaboration includes the COMET tutoring system which emulates the human-tutored medical PBL session which typically occurs in group settings on a whiteboard (Suebnukarn and Haddawy, 2007)

3.2.3 Fully Structured CIT

In the most structured CIT type, fully structured, the system treats collaborative skill learning as a full learning objective in addition to domain learning. The system models both the individual and the group in order to provide domain learning support and improve collaborative behaviors. Further, the system requirements determine the groups as well as functional roles within the group. For example, the system may intend for users of the same prior knowledge level work together and they each take turns acting in a pre-defined role of peer tutor and tutee as in the collaborative version of the well-known Cognitive Tutor Algebra (Walker et al., 2009b).

In the fully structured design, collaboration is required for the system task completion. Additionally, the system will prompt the students in regards to when to communicate. The system offers activities, normally in relation to the assigned roles, and the system guides the students on how to provide support to their co-collaborators. Finally, the system restricts students to a set of tools for their collaboration (i.e. chat interface). These tools are provided for system tracking of collaboration and domain learning status. The COLLECT-UML tutor, another example of the fully-structured collaboration class, offers students guidance on both effective collaboration behavior and learning UML modeling (Baghaei et al., 2007).

3.3 Collab-ChiQat Overview

The CIT classification analysis work shows that CITs exist across a spectrum of modelling options. Thus, the CIT classification scheme served as a necessary tool to guide design choices in the investigation of their effect on learning and collaboration. The newly adapted system, Collab-ChiQat, was initially evaluated across two different types of collaboration structuring, unstructured and semi-structured. After discovering statistically equivalent learning gains results from these version, we elected to not to pursue the fully-structured intervention in this thesis work. Instead, we designed a third, *hybrid*, system which lies on the middle of the spectrum between unstructured and semistructured.

Our choice not to implement the fully-structured version is primary due to our emphasis on “real-world”, classroom ready tutoring support. Both logistical concerns and our current findings cause reservation for pursuing this version. While many of the challenges of the fully structured system are addressed in the semistructured system (ie aspects of modeling group interaction), certain aspects of the extension of Collab-ChiQat to this level of collaboration are restricted by technical and logistical constraints. Namely, multiple experiments attested to the poor quality of the automatic speech recognition in the classroom setting. Moreover, the necessary ability for the system to guide group assignments was infeasible given unplanned student absences as this was a one-time intervention. In any case, our current work provide insight as to the possible outcomes in the fully-structured system from a theoretical perspective.

Firstly, we created Collab-ChiQat to support learning between pairs of students. The design choice to facilitate pair collaboration naturally extended from the positive body of research in support of pair programming. Two students share a single computer workstation as seen in Figure 7. Each of the students in the pair is equipped with a microphone headset. Before students begin using Collab-ChiQat, they are introduced to pair programming via a short video describing pair programming (see Appendix B).

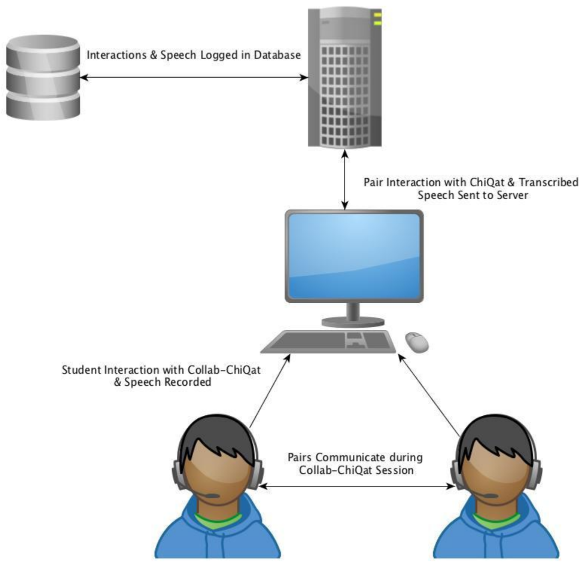

Interactions & Speech Logged in Database
Pair Interaction with ChiQat & Transcribed Speech Sent to Server
Student Interaction with Collab-ChiQat & Speech Recorded
Pairs Communicate during Collab-ChiQat Session

Figure 7: Collab-ChiQat pair programming setup. The pair share a single workstation and switch between navigator and driver roles.

In terms of architectural design decisions, Collab-ChiQat maintains all of the major components present in standard ChiQat. However, the collaborative system differs from the standard version in its design of the graphical user interface, student model, and feedback management. In Collab-ChiQat, a joint student model, collaboration panel, collaboration feedback manager and collaborative interaction model are introduced to the system architecture. These modifications are necessitated by Collab-ChiQat's added objective which is to measure, evaluate, and provide feedback regarding collaborative behavior in addition to supporting domain learning. The architecture is depicted in Figure 8.

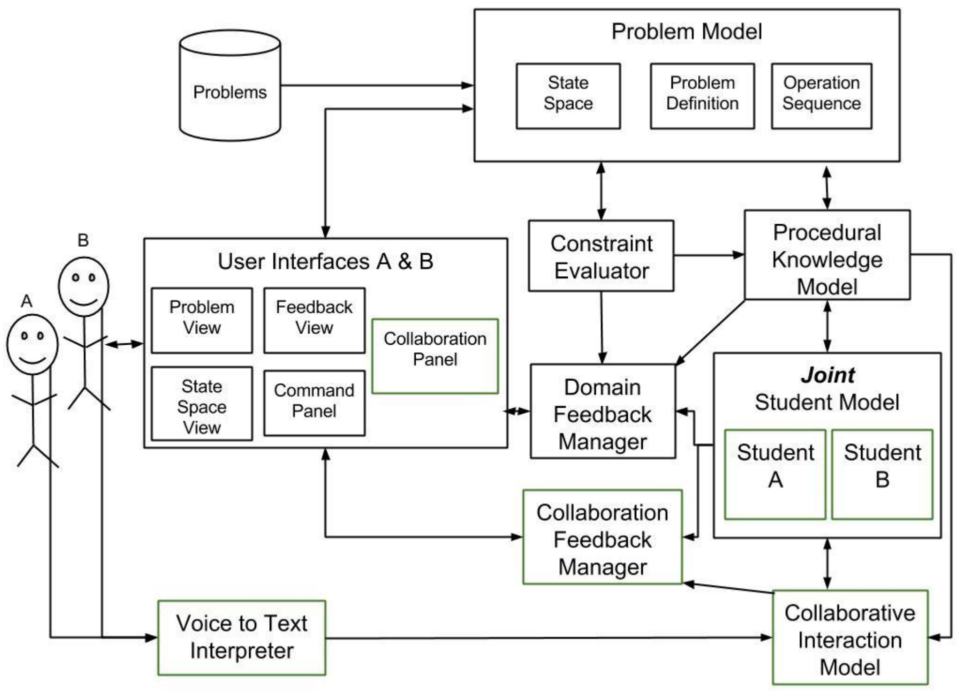


Figure 8: Collab-ChiQat architecture with newly added components highlighted.

The extent to which collaboration is facilitated by the system varies across the collaborative system: unstructured or semi-structured. Thus, the system's use of the collaborative components also differs across these versions. More detailed description of the Collab-ChiQat system versions is given in Section 3.4.

3.4 Architectural Design Choices in Collab-ChiQat

In unstructured Collab-ChiQat, students focus on CS domain learning with no system-provided support for collaborative interaction. However, in the semi-structured version of

Collab-ChiQat, students focus on CS domain learning and have feedback regarding their collaboration. While the system relies on several key, standard ChiQat components, distinct design decisions required for the collaborative implementation are described below.

3.4.1 Voice to Text Interpreter

The Voice to Text Interpreter translates each student's spoken utterances to text in real-time. The model is driven by the Google Speech API (Google, 2013). The textual representation is accompanied by time-stamps which allows for the student-to-student dialogue to be reconstructed by the system. The text corpus of peer dialogue interaction is then used by the collaboration interaction model to estimate the student's progress toward effective collaboration.

3.4.2 Student Model

The student model works as the storehouse of information pertaining to a student's problem solving behavior. The system monitors all GUI interactions as well as the timing of these actions. For example, the system tracks the time between code-submission attempts, coding error and success rates, and automated estimates of the number of spoken utterances. The collection of information available in both the joint and individual student models is used by both the domain and collaboration feedback manager to synthesize relevant and properly timed feedback.

In both conditions, the system equates individual knowledge of domain content with the group knowledge. The modelling for unstructured ChiQat closely emulates standard ChiQat.

However, there are several adjustments to accommodate for the student pair which are outlined below.

Joint Student Model: In Collab-ChiQat, the Joint Student Model is equivalent to the individual student model in standard ChiQat. In effect, the JSM treats the student pair as a single student in terms of the following behaviors:

1. history of students' actions (i.e. GUI clicks and code submission steps)
2. timing of students' actions (i.e. between GUI interactions, between code submissions)
3. feedback (i.e. # of syntax feedback, # positive/negative proactive/reactive feedback)
4. "undo/redo" behavior
5. # problem attempts and problems solved

Individual Student Models: In the case of unstructured Collab-ChiQat, the individual student model serves as the source of data regarding students' individual actions. This includes code ownership and code compilation error and success rates and spoken interactions. In particular, a rough estimation of the number of speaking turns (utterances) is derived automatically from the transcribed speech.

3.4.3 Collaborative Interaction Model and Feedback Manager

The collaborative interaction model maintains information about the health of the pair's collaborative behavior. The collaboration interaction model uses data from the student model and voice to text interpreter in order to create meaningful metrics for monitoring collaboration. In turn, these metrics inform the collaboration feedback manager which provides feedback to the student relative to their collaborative behavior. Metrics used to monitor the collaboration include code ownership rates, spoken utterance rates, and received peer bonuses. The unstructured condition does not provide collaboration feedback.

The design choice for feedback in the semi-structured condition is grounded by both CSCL and ITS research. One such method of feedback is via the display of group and individual performance (Phielix et al., 2010; Phielix et al., 2011; Jrvel et al., 2015). These finding inspired the design of the coding compilation performance bar graph. Further, visualization of individual participation and peer feedback have also led to higher signs of engagement and improved performance (Janssen et al., 2007; Phielix et al., 2011; Janssen et al., 2011; Gabelica et al., 2014). Specifically, through Dillenbourg's work, we recognize that the role of our collaborative technology tools is not simply to mimic face-to-face conversation, but to instead avoid the imitation game and conversely augment the experience of collaborators (Dillenbourg, 2005). This work described how both archival tools (message history) and GUI mirroring of interactions (graphs of individual opinions and system interactions) could augment collaborator experience. Thus we adapted a similar rationale in our design of the participation pie chart, performance overview graph, and peer bonus tool. Specifically, our work archives student success in coding over time and we plot participation (of code and speech) as a mirror into group dynamics. Finally, CSCL literature has also shown that promotive interdependence, when students understand individual success is tied to a common group goal, is a central tenant of effective collaboration (Dillenbourg, 1999; Soller, 2001; Strijbos, 2011). Thus, we provide students with an overall rating of the collaborative performance.

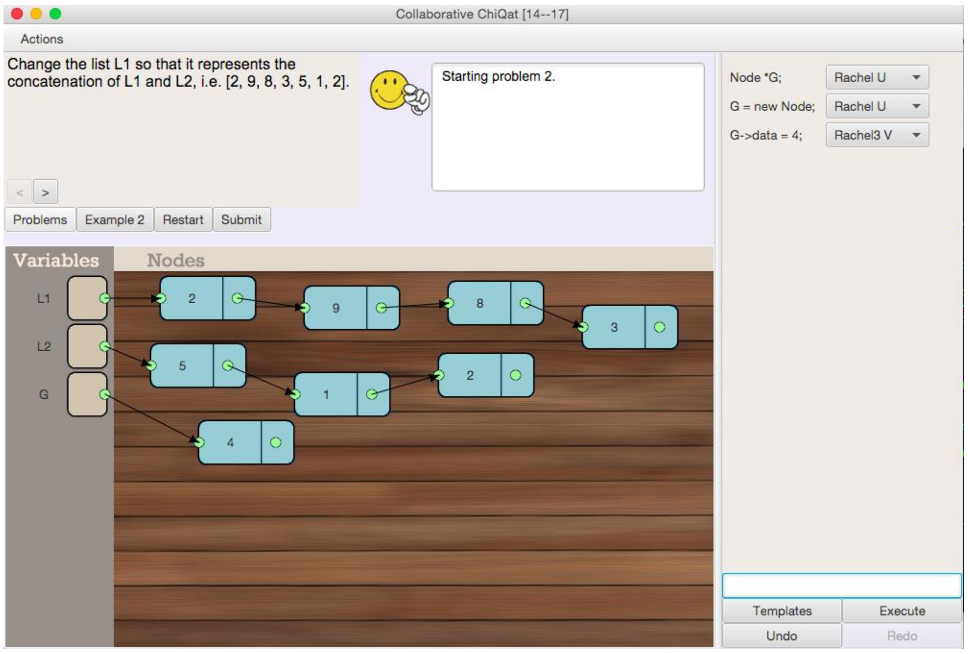
Furthermore, our design choices are also motivated by analysis of the KSC-Pal data-set. This data was intended to inform the design of KSC-PaL, a simulated peer learning agent, however, it is applicable to Collab-ChiQat as it consists of log and chat data from two collaborators working with an a ITS for CS Education. Overall analysis and evaluation of the KSC-PaL collaboration data confirmed that tracking partner contribution in the form of number of spoken utterances, code submissions, and GUI interactions is a valid metric for modeling learning within our domain. This result was also confirmed in another study regarding learning gains and pair programming process (Rodrguez et al., 2017). Additionally, the data showed that balance in pairs significantly correlated to learning. The list of manager supported feedback and rationale is given in Table II below.

Feedback Type	**Explanation**	**Collab-ChiQat Component**	**Description**
Student Partici-pation	The manager tries to promote equal sharing of ideas among collaborators. If one student has not spoken for some time or a student is monopolizing the conversation, the manager will present appropriate feedback. This can be seen as a load balancing task with the goal of symmetry.	Pie Chart	Compares # of spoken utterance and code submissions between collaborators. Intended to showcase individual performance contribution and promote balance.
Group Process-ing	The manager encourages group processing along with group and self assessment of performance by providing individual and group evaluation.	Bar Graph	Displays # of coding compile errors vs # successful compilations. Visualizes group performance per problem.
		Sentence Openers	Text input area for brief explanation of partner's achievement using sentence openers i.e.: _____ encouraged me by _____ _____ explained _____ _____ improved our solution by _____ Encourages individual assessment and reflection on self-performance.
Promotive Interde-pendence	The manager connects individual actions to the overall success of the group. Thus a team member can only perform highly if their partner does as well.	Collaboration Score	Offers linear metric based on system-detected collaboration events.

TABLE II: TYPE OF COLLABORATION FEEDBACK SUPPORTED IN COLLAB-CHIQAT ALONG WITH RATIONALE.

3.4.4 Graphical User Interface

The Collab-ChiQat graphical user interfaces (GUI) handle student interaction with the underlying system. It is composed of the problem's textual and graphical representations. It also contains the tutor feedback panel as well as an area for the students to type and submit code in order to solve the problem. It is shown in Figure 9.



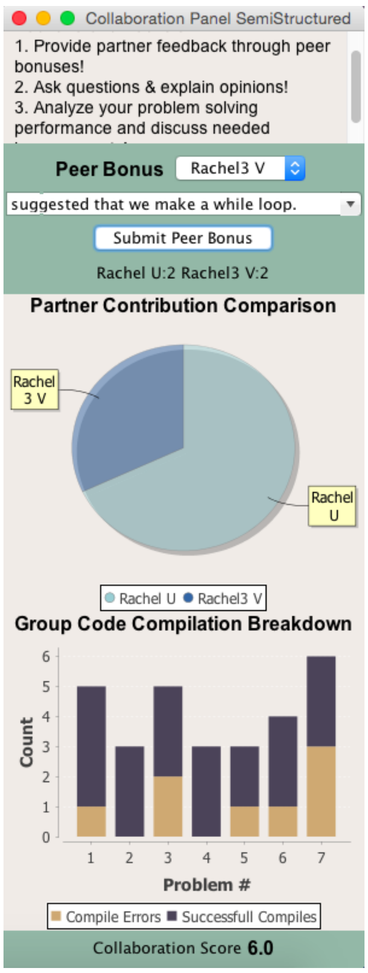
Change the list L1 so that it represents the concatenation of L1 and L2, i.e. [2, 9, 8, 3, 5, 1, 2].

Starting problem 2.

Node *G; Rachel U
G = new Node; Rachel U
G->data = 4; Rachel3 V

Figure 9: Collab-ChiQat graphical interface with feedback, problem panel, graphical state representation, and coding area, including code owner.

In Collab-ChiQat, a secondary GUI, the collaboration panel, is introduced. The panel serves as the view for collaboration feedback. The panel is shown in Figure 10.


Collaboration Panel SemiStructured

1. Provide partner feedback through peer bonuses!
2. Ask questions & explain opinions!
3. Analyze your problem solving performance and discuss needed

Peer Bonus Rachel3 V

suggested that we make a while loop.

Rachel U:2 Rachel3 V:2

Collaboration Score 6.0
Submit Peer Bonus
Partner Contribution Comparison
Rachel 3 V
Rachel U
● Rachel U ● Rachel3 V
Group Code Compilation Breakdown
Count
0 1 2 3 4 5 6
1 2 3 4 5 6 7
Problem #
■ Compile Errors ■ Successfull Compiles

Figure 10: The collaboration panel introduced in Collab-ChiQat.

In unstructured Collab-ChiQat, the main GUI remains the same as standard ChiQat with the addition of code owner selection for each submission. The collaboration panel is introduced in semi-structured Collab-ChiQat. The collaboration panel serves as a visualization of individual participation and group performance. The panel showcases the components listed in Table II.

CHAPTER 4

MEASURING THE EFFECT OF COLLABORATION ON STUDENT LEARNING AND INTERACTION

(This chapter includes and expands on portions of my paper previously published in *Harsley, Rachel, Davide Fossati, Barbara Di Eugenio, and Nick Green. 2017. "Interactions of Individual and Pair Programmers with an Intelligent Tutoring System for Computer Science." In* ***Proceedings of the 48th ACM Technical Symposium on Computing Science Education.***)

4.1 Summary

A primary contribution of this thesis is the evaluation of a collaborative ITS intervention in terms of the benefits and drawbacks as compared to interventions intended for individual, one-on-one use. In this chapter we compare the interactions of student programmers working as individuals and as pairs to solve coding problems using either standard ChiQat or Collab-ChiQat. Chapter 5 examines the effect of design choices within the pair programming condition in comparison of unstructured and semistructured Collab-ChiQat. [1]. Our analysis encompassed over 53,000 log events from 116 students. To our knowledge this study is the first of its kind to explore on a large scale pair and individual programming interaction in fine-grained detail,

[1]As will be discussed in Chapter 5, there were no significant difference in learning gains between the unstructured and semistructured conditions. Thus, for this comparative analysis, we combine the two conditions into a single condition labeled 'pair programming'.

including timing, example usage, compile success, and coding efficiency. We used unpaired t-tests to discover statistically significant differences in interactions between individuals and pairs. We discovered that the collaborative learners used fewer system-provided examples, completed problems faster and more efficiently, and showed higher indications of engagement with the task. Also, the majority of students in each condition reported satisfaction with the activity. Just as importantly, in both conditions, students experienced significant learning gains.

4.2 Evaluating Collaboration in CS

As described in Chapter 2, interest in collaborative student engagement continues to grow as Computer Science institutions and instructors in higher education and beyond look for ways to improve student retention. One such medium of collaboration has been pair programming. This method has many documented benefits, however, it is still often met with apprehension on the part of instructors and some students.

Our study puts both the benefits and concerns of pair programming to the test, and shows the advantages of collaborative learning at a fine-grained system interaction level. We compared individual programmers to pair programmers as they solved coding problems using our ChiQat-Tutor Intelligent Tutoring System (Green et al., 2015). Over 53,000 interactions were captured as 116 students participated. We assessed students' learning gain, problem solving time, reliance on system examples, coding efficiency, and signs of engagement within the activity. Finally, we compared students' perceptions of the system gathered from surveys. We discovered that while both groups exhibited significant learning gains, pair programmers completed problems more quickly, relied less heavily on system-provided examples, coded more efficiently, and showed higher signs of engagement. Moreover, students in both conditions found the system equally helpful.

Of the studies which examine pair programming, none are both large scale and fine-grained allowing for the ability to analyze line by line code and system interaction. With the exhaustive trace of interactions we gather while students work with the tutoring system, we explore details such as error rate, coding redo/undo behavior, copy and pastes, time to task completion, and signs of on-task engagement. Furthermore, our outcomes concern a one-time intervention. To our knowledge, this is the first study of its kind.

4.3 Experiment Design

Our study took place during two single interventions of an undergraduate level introductory Computer Science programming lab. The individual programming condition occurred in the spring of 2015, while the pair programming condition occurred in the fall of 2015 of the same course. The course was taught by a single professor with adherence to identical curriculum and pedagogical strategy across both semesters. The course focuses on introducing students to software development tools and practices. Additionally, it focuses on implementing foundational data structures and algorithms such as stacks, linked list, and depth first search to solve real-world problems. For example, one week, students are given a project to implement a linked list in C++ with support for insert, delete, and find operations. The following week they are required to implement a program which uses the linked list to store a waiting list for a restaurant.

In both Spring and Fall, our experiments ran over four different sessions of the course. In both conditions, students were given 12 minutes to individually complete identical pre and post tests regarding linked lists. The test covered key concepts such as node deletion and insertion and debugging errors. Students also completed a brief survey regarding their overall experience with the system. Students chose partners at the onset of the collaborative activity. In both cases, students used the system for a total of 40 minutes. Seven coding problems were available for students to solve.

Prior to beginning the pre-test, students in the collaborative condition collectively watched a three-minute video on pair programming (AgileAcademyAus, 2011). This video was shown in lieu of an additional pre-test on collaboration knowledge in order to avoid pre-test effect (Little and Bjork, 2012); (Richland et al., 2009). In the collaborative condition, students chose their own partners and each pair was stationed at a single workstation. Each student was also equipped with a headset as seen in Figure 11. Upon completing post-tests, students were given an exit survey regarding their perception of Collab-ChiQat and their abilities, their attitudes towards CS and the course, and their understanding of successful pair programming traits (see Appendix C). As the survey in itself introduces collaboration terms, this survey could not be used as a pre-test measure. The survey was derived from Braught's survey administered to students after a semester of pair programming (Braught et al., 2011).

All student interactions with the systems were exhaustively logged for both the individual and collaborative conditions. This includes a time-stamped trace of student activity such as clicks and keyboard events allowing us to later retrace the students' activity on a fine grained

Figure 11: Student pairs using Collab-ChiQat.

level. The two conditions shared a common feature set of 106 items, many of which are typical for all ITSs, such as time spent on a problem and number of errors. In total, 53,585 events were mined for our current analysis. Table III outlines key features collected in both conditions. There were a total of 116 students analyzed in the final data set with 75 in the individual condition and 41 in the paired condition. In the paired condition, we restricted our dataset to students who had consented to the study and who were new to the system along with their partner (n=41).

TABLE III: SELECT SHARED INTERACTION DATA SET FEATURES BETWEEN INDIVIDUAL AND PAIR CONDITIONS.

Problem Usage	**Coding**	**Example Usage**
Correct solution submitted	Operation executed	Example started
Incorrect solution submitted	Operation pasted	Example completed
Problem duration	Operation undone	Example duration
Problem restarted	Operation redone	Example step taken
Problem solved	Syntax help requested	

Navigation and GUI Interaction	**Tutor Interaction**	
New problem selected	Feedback given	
Problem list requested	Correct answer response to feedback	
Scratch pad next requested	Incorrect answer response to feedback	
Scratch pad prev requested	Tutor feedback question given	
Widget moved	Feedback acknowledged	
Window maximized		
Window minimized		

The goal of this study is to present a comparative analysis of how individuals and pairs interact with the system, engage in programming, and subsequently learn. In order to gain insight on similarities and differences between conditions, we performed unpaired t-test between each of the logged features including those listed in Table III. In our findings, we analyze those features with significant and notable distinctions between the conditions.

4.4 Findings

In this section, we describe the differences in condition that arose, arranged by theme.

4.4.1 Learning

The primary goal of our system is to help students learn computer science. We use our prior study of human tutoring data as a comparison baseline for students' learning gain and adopt the following measure of learning gain:

$$gain = postTestScore - preTestScore \tag{4.1}$$

In the human tutoring work, students achieved an average learning gain of .14. Results of the current study showed student learning gains to be .10 and .11 for the individual and pair conditions respectively. Notably, the difference in pre and post test was significant in both cases though shy of the human tutoring baseline. There was no significant difference in the learning gain between conditions. Additionally, we note that the difference in pre-test scores between conditions was not significant. This allows us to attribute the results to factors beyond the incoming knowledge level of students. Details of student test scores are given in Table IV.

Tutor	**N**	**P**re-test		**P**ost-test		**G**ain	
		μ	σ	μ	σ	μ	σ
Human	54	.40	.26	.54	.26	.14	.25
Individual	75	.49	.20	.59	.20	**.10**	.18
Pair	41	.51	.20	.62	.20	**.11**	.17

TABLE IV: INDIVIDUAL AND PAIR LEARNING GAINS

With outcome showing significant learning gain, it is safe to establish that students knowledge of linked lists benefited from the exercise in both the individual and pair conditions.

4.4.2 Example Use

We discovered that as a whole, pairs requested significantly less ($p < .01$, t = 4.79) system-provided examples than individuals. However, of the examples that were started, individuals and pairs completed the examples at a similar rate of around 96% (Note: equal completion rate does not signify equal time to complete as discussed below in Section 4.4.3). As students in both conditions solve the same number of problems, the lack of example usage among pairs shows that they are able to reason about the problem on their own. In other words, the pairs do not suffer from the absence of examples. This finding should serve as motivation for instructors fearful of implementing pair programming. In this paradigm, the teacher (or intelligent tutoring system) is not the sole source of technical information. Instead, students can rely on their partner and collective reasoning.

4.4.3 Timing

We examined both the time students spent on solving problems and the time spent interacting with system-provided examples. On average, students in both conditions successfully completed the first four problems leaving the remaining three problems incomplete. This is shown in Figure 12.

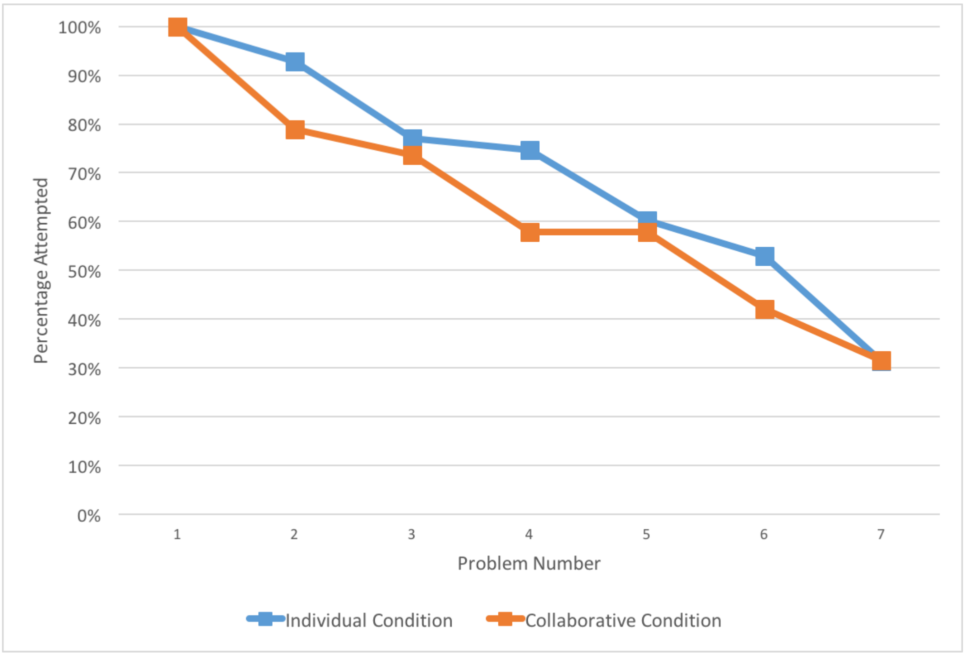


Figure 12: Comparison of problem attempts between individual and collaborative conditions.

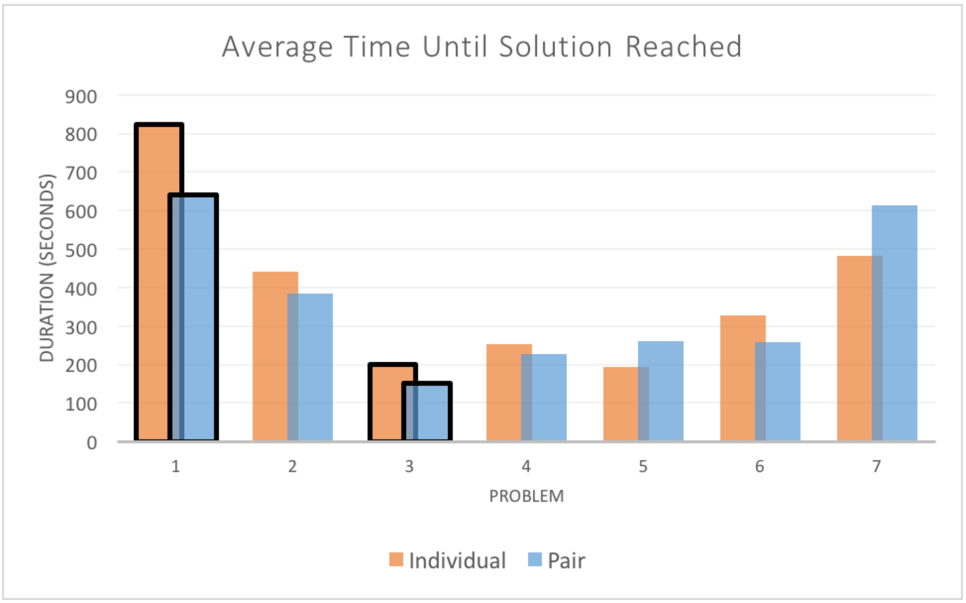


Figure 13: Comparison of problem duration between individual and pair programming conditions for those who solved the problem. Statistically significant differences in bold.

In five out of the seven problems, the pair programmers successfully completed the problem in less time than individuals. The difference was significant in two cases (Problem 1: $p < .01$, t = 2.85; Problem 3: $p < .05$, t = 2.33). In particular, in the first problem, there is a significant difference of 131% in the time to solve the problem. The difference shows that the pair is able to drastically decrease ramp up time. They accomplish both system acclimation and problem solving. Yet, if this difference solely allude to the pairs' ability to problem-solve through an unknown user interface more quickly, the result is nonetheless important.

Notably, the two cases in which individuals spend less time than pairs are problem five and seven. Though the differences in timing are not significant, both of these problems offer a shift in the problem type and skills required to solve. For example, problem five combines multiple skills from the prior problems (node creation, list traversal, node deletion) in addition to a new skill (node value comparison). In problem five, none of the pairs elected to use the example, while only 6% of the individual users did. Given that example use did not seem to be the cause of the individuals' timing advantage, this result suggests that while pairs were able to initially ramp up to the overall task more quickly than individuals, they also reached a plateau. Thus, the individuals were able to adapt and display the new skills required to solve problem five at a rate similar to the pairs. The pairs once again translated the knowledge from the prior problem more quickly than individuals as evidenced in problem six. Finally, in problem seven, the most difficult of the set, a total of only 22 students completed the problem (12 in individual condition, and 10 in pairs condition). Of this small portion of students, individuals once again completed the problem faster (though not significantly). Figure 13 shows the time spent on each problem.

Timing is also a factor in regards to students' use of system provided examples. In each of the seven problems, individuals use examples for more time than pairs. This difference is significant in four out of the seven examples as shown in Table VI . We infer that pairs used examples as a catalyst for discussion or hint for problem approach. Thus, engagement with examples was short, and minimal (as shown in Section 4.4.2). On the other hand, individuals took time to more thoroughly examine examples, and they relied on examples to a greater degree for each problem. The graphical comparison of example usage duration is shown in Figure 14. Our future work, will examine the discussion of pairs to gain a better understanding of their example use and timing.

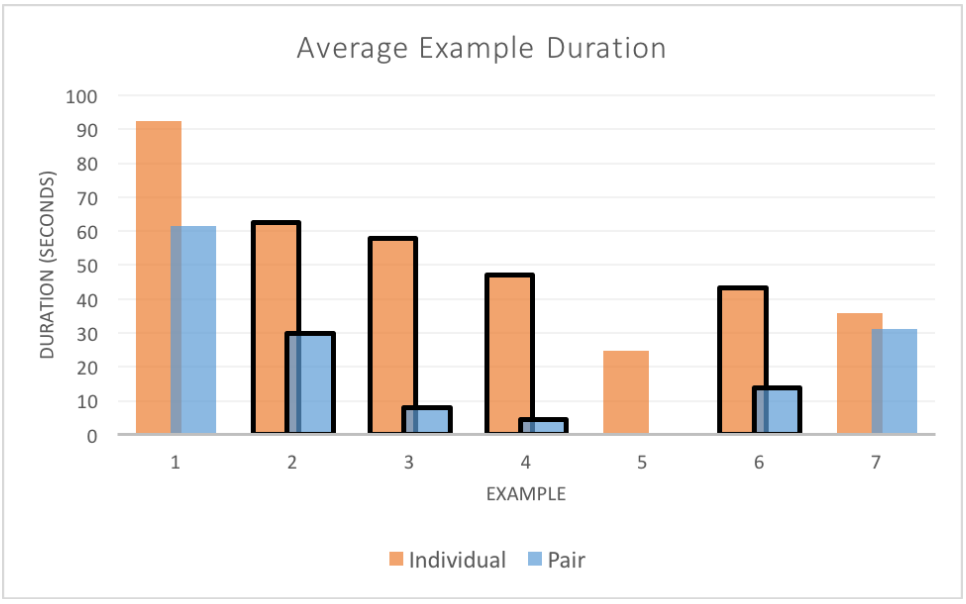


Figure 14: Comparison of example usage duration between individual and pair programming conditions for those who used examples. Statistically significant differences in bold.

4.4.4 Coding Efficiency

We measured coding efficiency by examining the number of operations required to reach a successful solution, the number of undo/redo operations, the number of problem restarts,

Example Number	**Individual μ**	**Pair μ**	**p-value**	**t-value**
One	92.28	61.55	p = 0.07	t = 1.76
Two	62.65	29.83	p = 0.00	t = 3.04
Three	57.87	8.00	p = 8.36e-06	t = 6.60
Four	47.00	4.50	p = 0.00	t = 4.05
Five	*n/a*	*n/a*	*n/a*	*n/a*
Six	43.28	13.92	p = 0.00	t = 4.22
Seven	35.75	31.27	p = 0.62	t = 0.50

TABLE V: T-TEST AVERAGE EXAMPLE DURATION (SECONDS) BETWEEN THE INDIVIDUAL AND PAIRED CONDITION

the number of programming errors, and the number of bad submissions. Pair programmers were significantly ($p < .01$, t = 4.63) more efficient in terms of the number of operations per solution. While individuals required around 23 operations per solution, pairs required only 13, a difference of 174%. Along the same line, individuals performed around four times more undos and redos than pairs, a significant difference ($p < .01$, t = 6.41; $p < .05$, t = 2.26). This suggests that individuals were more uncertain of their solution than pairs. In contrast, pairs had the ability of the navigator to plan and strategize over the direction of the solutions, causing fewer undos, redos, and overall operations to result.

Further, we analyzed the difference in problem restarts, coding errors, and bad submissions in order to further quantify coding efficiency. Students submitted their code to verify that their overall solution to solve the problem was correct. Additionally, once a student entered each individual line of code, it was compiled. In each problem, individuals had more compile errors than pairs. This difference was significant ($p < .01$) in all but problems five and seven. Moreover, across all of the problems, individuals submitted more incorrect overall solutions than pairs. This difference was significant in problems two, three, four, and six. Each of these problems were similar to the prior problem with a slight increase in difficulty. While pairs were able to transpose the prior knowledge into the new problem, individuals struggled. Finally, individuals requested significantly more problem restarts than individuals ($p < .01$, t = 3.35). This also signifies the lack of certainty in direction and solution strategy for individuals.

Problem Number	**Individual μ**	**Pair μ**	**p-value**	**t-value**
One	8.07	3.16	p = 3.04e-06	t = 4.90
Two	3.91	2.06	p = 0.00	t = 3.23
Three	2.14	0.65	p = 8.36e-06	t = 4.64
Four	1.33	0.54	p = 0.00	t = 3.23
Five	0.49	0.32	p = 0.17	t = 1.36
Six	3.07	0.36	p = 0.00	t = 3.25
Seven	1.72	1.33	p = 0.45	t = 0.75

TABLE VI: T-TEST AVERAGE NUMBER OF COMPILE ERRORS BETWEEN THE INDIVIDUAL AND PAIRED CONDITION

4.4.5 Engagement

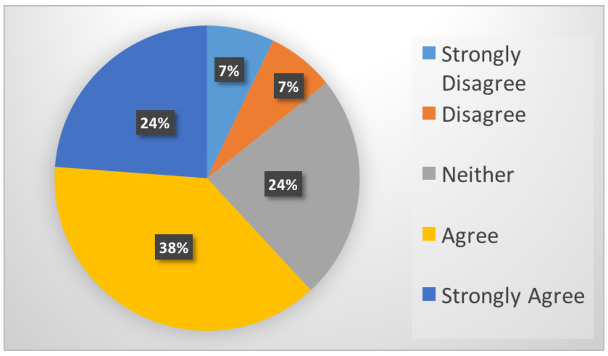
As signals to quantify user engagement, we explored the students' use of the lesson tutorial and resilience to working through a problem. The lesson tutorial was the same for both conditions and walked students through the features of the system including how the command line editor related to the graphical list representation. Students were invited to launch the lesson tutorial at the onset of the activity. Pairs elected to proceed through the lesson tutorial for significantly longer than individuals ($p < .01$). We believe this is a valid signal to suggest that pairs were 1) more interested in knowing how to use the system and 2) thus more engaged in the activity. In general, individuals requested the problem list (the primary interface used to switch problems) significantly more ($p < .01$) than pairs. Individuals acted on this impulse to switch problems significantly more as well ($p < .01$). They switched problems an average of 10 times, though only four problems were solved on average. However, pairs chose a new problem seven times. Once faced with difficulty in solving a problem, pairs continued to work through the problem while individuals switched problems.

4.4.6 Satisfaction

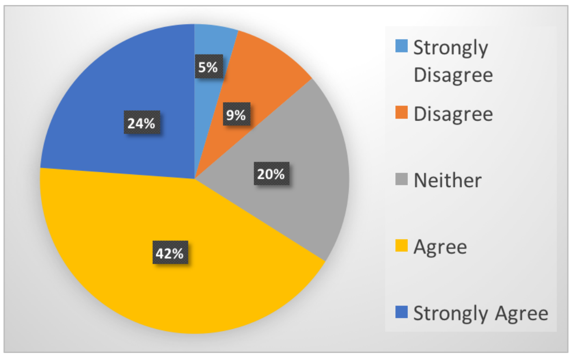
Finally, we outline students reported perceptions of the systems based on survey feedback. Both conditions expressed satisfaction with their experience using the system. The majority of students in both conditions found the system helpful as shown in Figure 15. There was also no significant differences in whether students found working with the system to be interesting Figure 16b. We find this results surprising given the reported research which typically links pair programming with enhanced enjoyment. We believe this difference may be found in time with repeated use. Our study was a one-time intervention in a lab session without pair programming activities and it may have been an uneasy adjustment for students.

4.5 Conclusion

In this chapter, we presented a quantitative study comparing the fine-grained interactions of individual programmers versus pair collaborators as they work to solve coding problems using standard ChiQat and Collab-ChiQat respectively. We collected data from over 115 students resulting in more than 53,000 log events. We discovered that while both individual and pair

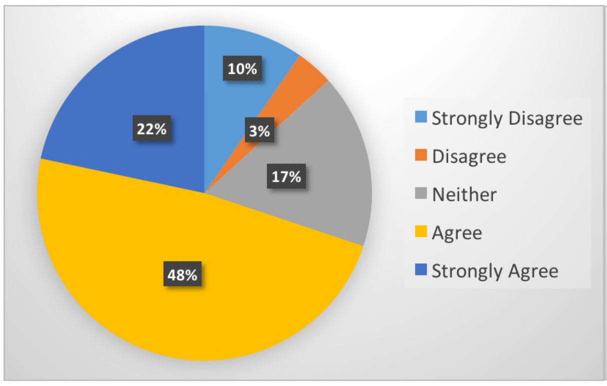


(a) Individuals

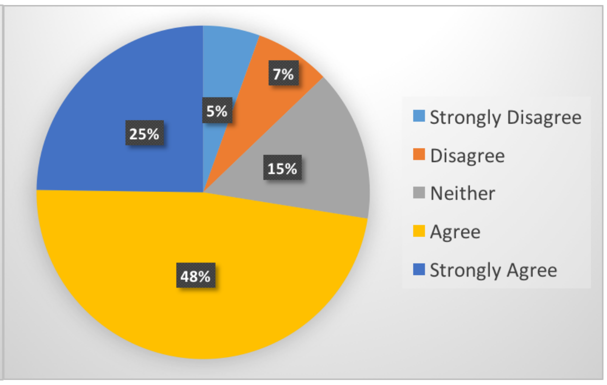


(b) Pair Programmers

Figure 15: Breakdown of student survey responses to question: *"Do you feel that (Collab)ChiQat helped you learn about linked lists?"*



(a) Individuals



(b) Pair Programmers

Figure 16: Breakdown of student survey responses to question: *"Do you feel that working with (Collab)ChiQat was interesting?"*

programmers had equivalent learning gains, students in the collaborative condition took significantly less time on most problems, consulted fewer examples, coded more efficiently, and showed more signs of engagement. Signals such as coding efficiency seem to attest to the superiority of pair programming. However, we acknowledge that this result may also simply signify that individuals learn through trial-and-error, a "best" approach when lacking a co-programmer. Thus, though the individual approach is less efficient, trial-and-error can be a meaningful strategy in and of itself for those individuals. Lastly, we found that the majority of students in both conditions found the system to be helpful and interesting.

CHAPTER 5

THE ROLE OF CIT DESIGN ON LEARNING AND COLLABORATIVE INTERACTION

(This chapter includes and expands on portions of my papers previously published in *Harsley, Rachel, Barbara Di Eugenio, Nick Green, Davide Fossati, and Sabita Acharya. 2016. "Integrating Support for Collaboration in a Computer Science Intelligent Tutoring System." In* ***Proceedings of the 13th International Conference on Intelligent Tutoring Systems.*** and *Harsley, Rachel, Barbara Di Eugenio, Davide Fossati and Nick Green. 2017. "Collaborative Intelligent Tutoring Systems: Comparing Learner Outcomes Across Varying Collaboration Feedback Strategies." In* ***Proceedings of the Computer Supported Collaborative Learning (CSCL) Conference.***)

5.1 Summary

In the last chapter, we examined the differences in students' collaborative use of an ITS in comparison to individual use. In this chapter, we aim to understand how different methods of structuring the collaborative condition using the CITs design framework affects student learning and interaction. While the unstructured Collab-ChiQat does not provide students with feedback on their collaboration, the semistructured version offers a visualization of group performance over time, partner contribution comparison and feedback, and general tips on collaboration. In this portion of the thesis, we first present a contrastive analysis of the student learning and interaction outcomes between unstructured and semistructured Collab-ChiQat. We then describe interactions that were statistically correlated to student learning gains both within and across conditions. Finally, we explore students reported perceptions of both systems. We found that students in both conditions have significant learning gains and equivalent efficiency in coding and limited reliance on system examples. However, unstructured users are more on-topic in their conversational dialogue while semistructured users exhibit better planning skills as problem difficulty increases. We discovered that several behaviors consistent with CSCL literature on effective collaboration also correlate to learning in our intervention. These behaviors include planning and team symmetry.

5.2 Evaluating CIT Design Choice

It is well accepted that effective collaboration and student learning does not necessarily follow simply by placing students in groups. Instead, broadly, much CSCL research has examined how collaborative activities can be designed and structured in order to facilitate the most desirable outcomes. Moreover, we recognize that the role of the tutor in structuring collaboration can widely range from limited structure with no collaboration feedback to high structuring with role definitions, group formation, and even timing of communication (Harsley, 2014).

Our study contrasts two distinct methods of supporting collaboration through an ITS. In one condition, unstructured, students do not receive feedback on their collaboration, while in the second condition, the tutor provides automated feedback regarding performance and participation. 102 students used the system to solve computer science coding problems and we collected over 39,000 interactions. Analysis of student learning gain and behaviors allowed us to further understand the effect of CIT design choice. Students in both conditions experienced significant learning gains, however, there were also significant differences in their example and tutorial use, time to start problems, and dialogue-based activity. Furthermore, we examined features with significant correlations to learning and found that students' practice of planning and symmetry, known characteristics of effective collaboration within CSCL, was significantly correlated to learning. Finally, students in both conditions found the system to be helpful and interesting.

5.3 Experimental Design

As described in Chapter 4, an experiment involving student participants was conducted in Fall of 2015 in a second year Computer Science programming course. At the onset of the experiment, students chose their own partners. Our decision to allow student choice in partners follows the specification of the CIT framework for unstructured and semistructured systems. Then, the pair was randomly assigned to either the unstructured or semistructured condition. Each pair was stationed at a single workstation and individually equipped with a headset. They were given 40 minutes to work with the system. Student interaction with the system was continually logged. Students were given an exit survey regarding their perception of Collab-ChiQat, their abilities, their attitudes towards CS and the course, and their understanding of successful pair programming traits. Students were allowed 12 minutes to perform the pre and post tests individually. We used the same measure of learning gain presented previously to assist in our analysis of learning.

A total of 102 students used Collab-ChiQat during the study. However, given the extensive use of ChiQat across several courses over the last two years, a large portion of the students had used the non-collaborative ChiQat system previously. Their data, along with their partner (totaling n=60), is held out from our analysis and reserved for future work. Thus, we narrowed our current analysis to students who who were new to the system along with their partner and who had consented to the study (n=41).

5.4 Learning Outcomes

Of foremost importance in evaluating the effectiveness of the intervention is the answer to the question of whether students learned. In answer to this question, the students did learn in both of the conditions. Overall, student post test scores were significantly better than pre-test scores (p=.0003). Moreover, the significant difference held across each conditions individually. Students achieved learning gains averaging .12 in the unstructured condition (p=.007) and .11 in the semistructured condition (p=.018). It is worth noting, the learning gains approach our prior results for human tutoring[1] as shown in Table VII, despite higher average student pre-test scores.

[1]The human tutoring condition measured learning gains of students after one 40-minute session of working with a human computer science tutor.

Tutor	N	Pre-test		Post-test		Gain	
		μ	σ	μ	σ	μ	σ
Human	54	.40	.26	.54	.26	.14	.25
Unstructured Collab-ChiQat	22	.46	.20	.58	.21	**.12**	.19
Semi-Structured Collab-ChiQat	19	.57	.23	.67	.24	.11	.28

TABLE VII: COMPARISON OF LEARNING GAINS BETWEEN Collab-CHIQAT VERSIONS AND HUMAN TUTORS.

5.5 Dataset

Subsequent to the question of whether students learned, we explored the questions of 1) how did students interactions compare across conditions and 2) what factors contributed to their learning. Overall, we are able to take into account several distinct features in regards to the disposition of the student and their interaction with the system in answer to these questions. Figure 17 depicts our representation of the student from a known data perspective. These data points are used as the basis for comparison and correlation analysis.

5.5.1 System Log Features

Collab-ChiQat exhaustively logs all student interaction. This includes time-stamped traces of student clicks, keyboard events, time to start problems, undo operations, and even number of lines coded before switching driver, in addition to proactive and reactive tutor feedback. This

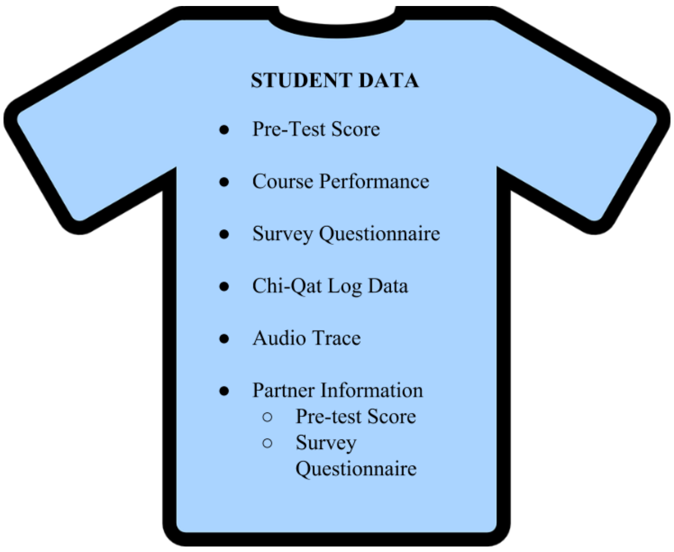


Figure 17: Data available for each student

allows for later recreation of the students activity on a fine grained level. Post-intervention, exhaustively logged data provides another point to measure differences in student interaction and discover predictors of learning.

The standard, processed log features[1] common to Collab-ChiQat and non-collaborative ChiQat include raw counts of undo/redo, error rate, and example usage. However, the standard features do not account for interactions unique to *pairs*. Thus this feature set was enhanced in order to accommodate collaborative learning. The enhanced collaboration feature set is shown in Table VIII and includes metrics such as coding turn length and time to start a problem.

[1]The full listing of standard metrics is given in Appendix D.

Collaborative Features by Type
Coding Turn Taking
Partner’s Total Turn Length
Difference Between Partner’s Total Turns
Sum of Turns Between Partners
Number of Turns of Length N
Speech
Number of Total Utterances
Number of Partner’s Total Utterance
Difference Between Partner’s Total Utterances
Sum of Utterances Between Partners
Percentage of Utterance Contribution
Domain-related Words
Utterance Types
Initiative Shifts
Timing
Time to Start Problem N
Action Density for Problem N

TABLE VIII: ADDED FEATURES TO MEASURE COLLABORATIVE INTERACTION

5.5.2 Audio Discourse Features

The discourse data available consists of an audio recording of a student pair during their 40-minute session using Collab-ChiQat. Collab-ChiQat provides automated, real-time transcription of dialogue via the Google Speech API (Google, 2013). However, after discovering poor transcription reliability of the automated process, an undergraduate researcher and I manually transcribed the audio data. Manual transcription resulted in over 9,000 spoken utterances and over 40,000 words.

Following this, we generated transcription-based features including counts of domain-related words and number of utterances. We distinguished a set of words as domain-related words for our analysis. These words were extracted based on those syntactically required for code (ie *Node* and *link*) and those presented in the worked out examples (ie "connect the node" and "insert into the list"). We believed they were representative of the overall coding task and could serve as a general meter for assessing whether students were off-topic as shown in (Friedberg et al., 2012). Moreover, we automatically labeled each transcribed utterance as either 1) question 2) command 3) prompt or 4) assertion following Walker and Whitakers utterance based control rules (Walker and Whittaker, 1990). These labels were then used to track shifts in linguistically-based initiative. In particular, we examine two types of initiative as follows the work of KCC-PaL (Kersey-Howard et al., 2015). Dialogue initiative tracks the leader in conversation by changes in utterance types (assertion, command, prompt and question) (Kersey-Howard et al., 2015). Dialogue initiative occurs when a speaker contributes new content (including questions) to the conversation that are not in response to the other participant. On the other hand, task initiative tracks the leader in problem solving (Kersey-Howard et al., 2015). Thus, both forms of initiative are methods of tracking which speaker is leading the conversation, its focus, and problem solving efforts.

5.6 Contrastive Analysis

In this section we present findings from our contrastive analysis between the unstructured and semistructured conditions. We performed unpaired t-tests between every feature across conditions. For example, we compared the time to start problem one in the unstructured condition versus the semistructured condition. This analysis allowed us to establish how different methods of structuring collaboration guidance affect the pair's interaction between themselves and the system. Overall, we found that the conditions were similar in terms of example use and coding efficiency. However there were distinct differences in student dialogue, tutorial use, and time taken to start writing code. Specifically, unstructured users had more on-topic conversation and used the lesson tutorial more while students using the semistructured system showed some signs of competitiveness and urgency in submitting code. Furthermore, semistructured users also took time to plan before beginning more difficult problems. We attribute these observed differences to the alterations in the system structuring which effect collaborative interaction.

5.6.1 Dialogue

There were several key distinctions between the conditions in terms of linguistic features. To begin, students in the unstructured condition spoke significantly more than the semistructured students and averaged around 2,700 words per session. Also, t-test comparison revealed that students using the unstructured version used both a significantly higher volume and proportion of domain related words. This suggests that students in the semistructured condition were more off-topic, however, as shown, this was not detrimental to their learning.

Further analysis revealed which specific domain words were used more significantly. These words include *node*, *list*, *link*, and *new* as shown in Table IX. However, one key domain-related word semistructured used significantly more was *submit* (p <.05). *Submit* was used in reference to code execution. We believe semistructured students were much more conscientious of the role of submissions due the its effect on the collaboration panel feedback. Students' submissions affected their participation comparison chart and overall group performance breakdown.

Feature	Unstructured μ	Semistructured μ	p-value	t-value
node	30.30	19.85	p = 0.02	t = 2.36
list	7.95	4.64	p = 0.00	t = 2.97
link	36.15	21.64	p = 0.01	t = 2.51
new	10.7	5.0	p = 0.01	t = 1.81
submit	1.30	2.83	p = 0.00	t = -2.96

TABLE IX: COMPARISON OF NUMBER OF OCCURENCES OF DOMAIN WORDS BETWEEN UNSTRUCTURED AND SEMISTRUCTURED SCORERS

Students in the unstructured condition asked significantly more questions than those in the semistructured condition. Examples of observed student question types are given in Table X. We used the Pearson Coefficient test to find correlations between our feature set and the number of questions that pairs asked. The data shows several viable explanations for this phenomena. Firstly, we discovered that the number of domain words and the frequency of domain words is significantly correlated to the number of questions. This held true when examining the semistructured condition (R=0.9233029, p<.001) and all combined data (R=0.7594092, p<.001). This correlation clearly demonstrates that students who are on-topic, are asking questions. Thus, because students in the semistructured condition were off-topic more frequently than students in the unstructured condition, they did not ask as many questions.

The analysis also showed that the number of questions asked in the semistructured condition is strongly negatively correlated to code owner changes (R= -0.8669467, p<.001). Code owner changes occur when students revisit a line of code and change the driver to be the other partner. These alterations suggest that students who asked fewer questions were also concerned about having balanced participation levels displayed in the collaboration feedback panel. Moreover, these same students were motivated to "learn by doing" because each action improved their performance rating in the collaboration panel. Thus, we believe their inquiry was through actions as opposed to simply question and answering between pairs. This hypothesis is supported by the fact that semistructured students not only discussed and considered code submissions more frequently but also committed more overall interface actions in problems four and five (the most difficult problems that the majority of students reached). In both of these problems,

Student Question	**Type**
You get it?	Check for partner understanding
Did you connect?	Coding navigation
We dont remove it?	Coding navigation
This should get here right?	Elicit confirmation
What did I do?	Express confusion
Wait, what?	Express confusion
How would you do that?	Seek help
Can you send the execute command?	Request action
Can you draw it?	Request action

TABLE X: EXAMPLE QUESTIONS FROM PAIR DIALOGUE

semistructured students had significantly more action density (summation of node clicks, code hints, code executions) than unstructured students (Problem Four: p = 0.02, t = -2.29; Problem Five: p=0.05, t = -2.02).

Finally, we found that the number of questions asked is negatively significantly correlated to the number of correct answers to the tutor's proactive question prompts in problem one (R= -0.8945217, p<.001). An example of proactive tutor question prompt is shown in Figure 18. The number of questions asked is also significantly negatively correlated to the amount of positive tutor feedback [1] in problem one (R= -0.768932, p<.001). Though the meaning of these findings must be further validated, it may suggest that after students do well on problem one, they are less inclined to reach out to each other for help as evidenced through questioning. Alternatively, they may simply be more knowledgeable about the lesson and subsequently have fewer questions.

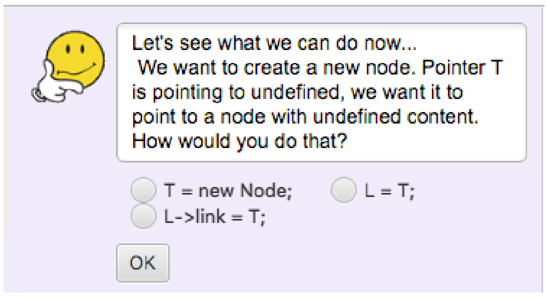

Let's see what we can do now...
We want to create a new node. Pointer T is pointing to undefined, we want it to point to a node with undefined content. How would you do that?

Figure 18: Tutor proactive question to aid student in node declaration.

5.6.2 Example Use and Coding Efficiency

There was not a significant difference between conditions on time spent with worked out examples and neither was there a difference in the amount of examples requested. Furthermore, our measures of coding efficiency showed no significant differences. These included the number of operations required to reach a successful solution, the number of undo/redo operations,

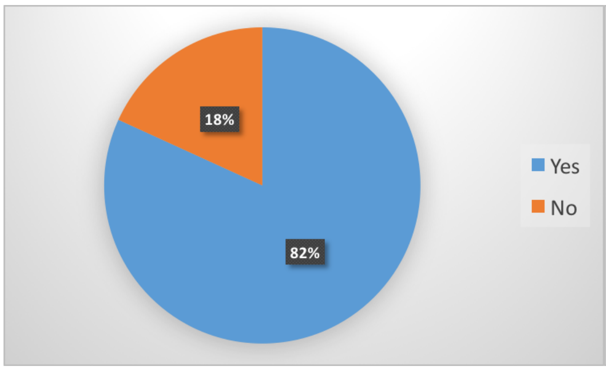
[1]Positive tutor feedback results when a student performs a correct action.

the number of problem restarts, the number of programming errors, and the number of bad submissions. Overall, both conditions allowed pairs to benefit from having an external metacognizer in terms of avoiding errors and relying less on the system for guidance.

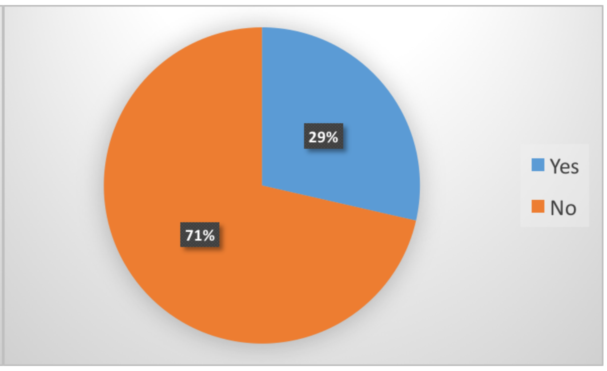
5.6.3 Tutorial Use

Students in the unstructured condition viewed the lesson tutorial significantly more than students using the semistructured version as shown in Figure 19. Once students first begin the lesson, the tutoring system invites them to do the tutorial. The tutorial provides students with information on navigating the interface and the relationship between their code and its visual representation. Our hypothesis is that the semistructured students felt more pressure to perform (in terms of code submissions and other actions) once presented with the collaboration panel also at the onset of the exercise. The panel feedback remains fully visible throughout the session without window activation [1] Conversely, unstructured students felt more relaxed and able to explore the system. In support of this hypothesis, we found that, in spite of continued window visibility, semistructured users still activated the collaboration panel an average of three times throughout the lesson. Though we were unable to track students full attention to the panel via measures such as eye tracking, this simple measure gives credence to the fact that students did consider the panel. Finally, analysis of the student dialogue also supports the notion that semistructured students heightened sense of urgency to participate may be the cause of their neglect of the tutorial.

[1]Activation occurs once the window is clicked and subsequently brought into focus.



(a) Unstructured



(b) Semistructured

Figure 19: Student lesson tutorial use.

5.6.4 Time to Start

The difference in the total time to start all problems and the average time to start per problem was trending towards significance with unstructured students taking more time. The time to start a problem is the time spent between when the problem is introduced and when students submit their first line of code. We use the time to start as an intuitive approximation for planning time. Moreover, given our audio analysis, this time was indeed most often spent discussing the meaning of the problem and proposing a solution approach. Students in the unstructured condition did take significantly more time to start problem one. In general, students took a similar amount of time to start problems two and three. However, in problems four and five, semistructured students took significantly more time before they began to code. As a reminder, each problem was given in order of difficulty and students were required to solve each problem in order to advance. Students from both conditions averaged completion of around four problems. Notably, semistructured students began an attempt to solve problems four and five double the amount of times as the unstructured condition, a significant difference. Multiple attempts signify that either the problem was explicitly restarted or a prior problem was revisited before returning. We believe these findings suggest that not only did semistructured students spend ample amount of time planning their approach to harder problems, but that the collaboration feedback served as their motivation for restarting (or revisiting) these problems more frequently. This is because the problem restarts also reset the visualized group performance metric of coding error rate for the particular problem. These findings suggest that students wanted to maintain a favorable group performance rating and accomplished this through resetting the problem or visiting a prior problem to reflect on the solution. As will be discussed in section 5.7.2.1, this planning time paid off in terms of learning gains.

5.7 Modeling Learning

With the premises established that the system is effective in helping students to learn and that structure influences learning and student interaction, our focus shifts to understanding how learning occurs. As standard for ITS research, we aim to model learning in order to design the system's student and tutor modules. Because Collab-ChiQat is derived from non-collaborative ChiQat, much of its internal modeling and tutoring strategy is informed based on prior modeling of *one-on-one* student use of the system. In this paradigm, the student works individually with the tutoring system and learning results as a correlated side-effect of problem solving. Instead, in a CIT, the tutoring system must provide activities which trigger learning

Feature	Unstructured μ	Semistructured μ	p-value	t-value
Tutorial Views	1.27	.37	p = 0.00	t = 2.93
Time to Start P1 (ms)	215639.00	56669.35	p = 0.02	t = 2.51
Time to Start P4 (ms)	19819.05	40758.31	p = 0.04	t = -2.08
Time to Start P5 (ms)	18357.36	37657.75	p = 0.04	t = -2.15

TABLE XI: T-TEST COMPARISON OF FEATURES BETWEEN UNSTRUCTURED AND SEMISTRUCTURED SCORERS

and cognitive mechanisms and not simply rely on the group setting to innately produce effective collaboration (Dillenbourg, 1999).

Our choices in both activities and model features for unstructured and semistructured Collab-ChiQat are grounded in research from both the CSCL and ITS communities. However, future system iterations must be derived based on our corpus of data as well as established literature. Namely our model of learning through collaborative interaction plays an essential role in influencing the system design.

In order to model learning, we used multiple linear regression with the post-test score as the dependent variable. Linear regression allows us to find statistically verifiable correlations between what students do and how much they learn. In phase one, we first used feature selection to narrow down our number of model attributes from our entire set of available attributes. We then found the best fit (highest Adjusted-R^2) model for learning using (a subset of) these features. The goal of this phase was to establish whether our feature set was sufficiently explanatory for our data [1]. In phase two, we created a model with each feature along with a students pre-test score as a co-variate. Phase two was needed because the feature selection in phase one does not allow for all possibly significant features to be explored due to our dataset size. Thus, this step allowed us to test individual significance of features. Overall, our aim in this two-part analysis was to establish which features were significant correlates to student learning and thus be empowered to use this insight to inform future system design.

5.7.1 Phase One Modeling

Our available feature set consists of the standard features used for log analysis in the single user version of ChiQat along with the additional features for collaboration. However, the number of measured features exceeds the number of examples per condition (n=22 & n=19). Therefore, we needed to find a smaller set of features in order to create a multiple regression model. In order to determine important features to comprise this smaller set, we created a regression tree for learning gain. The regression tree was derived using recursive partitioning (Therneau et al., 2015). From this point, we derived a linear additive model using the smaller subset of important features capable of being added to the model. This yielded a standard multiple regression model. We show and discuss best fit models resulting in the highest adjusted

[1] In the proposal, we showed that the feature set needed to be enhanced given its low Adjusted-R^2 values. Use of our standard model features in the collaborative case results in insufficient model fit for both methods of structuring the collaboration (Adjusted-R^2=.37 for unstructured, Adjusted-R^2=.14 for semistructured.)

R^2 after completing this process for our available feature sets. For example, in the case of the standard analysis feature set, there are over 130 features. We narrowed this set to n features using the regression tree, and all possible model combinations using these n features were compared for best fit. The size of n depends on whether we examine all data or only the unstructured or semistructured condition. In the case of all data, the top 20 most important features from all categories after obtaining the regression tree are shown in Table XII.

I began by performing modeling with each individual feature along with the pre-test as a measure of prior knowledge. I also performed the model fit using student course performance. This is a student's calculated grade in the course up to that point in the semester on a scale of 0-100. At this point, there is no automated grading of pre-tests, thus the course performance could serve as valid input to the system in future implementations. Finally, the standard analysis and collaboration features were individually modeled then added to these prior knowledge features to compare fits. The models which combine the semistructured and collaborative features obtained the best fits for both conditions as seen in Table XIII and Table XIV. Moreover, these models were just as effective without use of students' prior knowledge. The full models can be found in Appendix E.

Features
Pre Test
P7 Number of Turns
Total Turn Length
P5 Number of Turns
P6 Number of Turns
Max Turn Length
P5 Difference in Number of Turns
Problem One Duration
create (Spoken Word)
Operation Undone
Undo Operation Requested
P1 Undos
Action Density P1
Partnership Total Utterances Difference
Percentage Utterance Contribution
Example Completed
Example Started
Template Help Requested
Problem List Requested

TABLE XII: MOST IMPORTANT FEATURES FROM REGRESSION TREE WHEN MODELLING ALL DATA USING ALL POSSIBLE FEATURES

Model	**Adjusted-R^2**
Pre-test	.37
Course Performance	-.06
Standard Features	.44
Collaboration Features	.73
Standard + Pre-test + Course Performance	.54
Collaboration Features + Pre-test + Course Performance	.56
Standard Features + Collaboration Features	.94
Standard Features + Collaboration Features + Pre-test + Course Performance	.94

TABLE XIII: POST-TEST SCORE MODELS USING PRE-TEST, STANDARD FEATURES, COLLABORATIVE FEATURES, AND COURSE PERFORMANCE FOR THE UNSTRUCTURED CONDITION.

Model	**Adjusted-R^2**
Pre-test	.57
Course Performance	-.08
Standard Features	.29
Collaboration Features	.83
Standard + Pre-test + Course Performance	.61
Collaboration Features + Pre-test + Course Performance	.80
Standard Features + Collaboration Features	.95
Standard Features + Collaboration Features + Pre-test + Course Performance	.92

TABLE XIV: POST-TEST SCORE MODELS USING PRE-TEST, STANDARD FEATURES, COLLABORATIVE FEATURES, AND COURSE PERFORMANCE FOR THE SEMI-STRUCTURED CONDITION.

Overall the best fit models showed that our collaborative feature set drastically improves on features that originated from one-on-one tutoring. Given the tight fit of the models, we believed our feature sets were sufficiently explanatory for understanding the mechanisms influencing student learning with Collab-ChiQat.

5.7.2 Phase Two Modeling

5.7.2.1 Symmetry and Planning

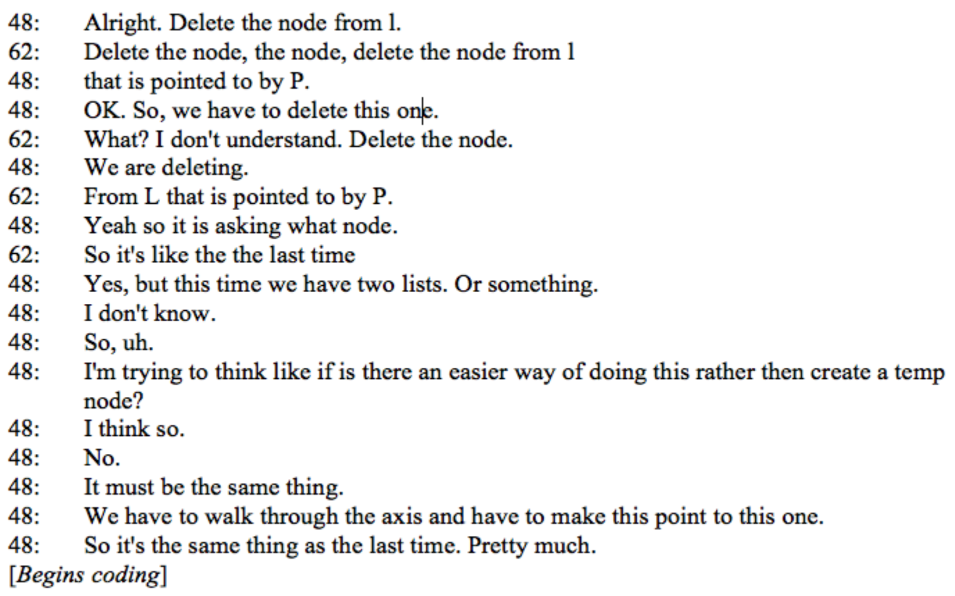
In phase two, we modeled post-test performance using each individual feature along with pre-test score as co-variate. We examined all possible features, not just those considered important from the regression tree creation. Our analysis revealed that *symmetry* plays an important role in student learning. We measured symmetry in terms of balance in initiative taken in student dialogue and in coding turns. Overall, the number of times a student took dialogue initiative was significantly positively correlated to learning ($p = 0.00$, Adjusted-R^2 =0.59). However, in the semistructured condition, the amount of times a *partner* took initiative was negatively significantly correlated to learning. This implies that the semistructured condition did not promote a balanced, or symmetric relationship, one in which learning was not inhibited by a partner's initiative. Intuitively, as students work to solve the problem together, the initiative should shift between students. As expected, learning occurs as a student takes initiative. However, in the case of the semistructured condition, learning suffered as their partner took initiative. This lack of symmetry was confirmed with analysis of student turn taking behavior while writing code. In the semistructured condition, the difference in coding turns between partners is significantly positively correlated to learning ($p = 0.04$, Adjusted-$R^2 = 0.73$). This means that as one student had a larger amount of coding turns than their partner, or one student dominated, their learning improved. This was not the case for the unstructured condition.

We also discovered that planning played a key role in modeling learning. The time to start problems four through seven was positively significantly correlated to learning as described in

Table XV. Given the significant difference between unstructured and semistructured conditions in the time to start problems four and five described in Section 5.6.4, these findings suggest that students in the semistructured condition engaged in more planning as problem difficulty increased. Examination of student transcription provides evidence of this phenomenon as shown in Figure 20. We hypothesize that students in the semistructured condition were motivated by the group performance feedback to keep their coding error rates minimal. Thus, they took time to plan. Further, as the problems increased in difficulty, the majority of students did not complete problems six and seven, thus we find no significant difference between conditions in these problems.

Model	**Adjusted-R^2**	**p-value**
Time to Start P4	0.63	p = 0.000
Time to Start P5	0.60	p = 0.002
Time to Start P6	0.60	p = 0.003
Time to Start P7	0.59	p = 0.003

TABLE XV: POST-TEST SCORE MODELS USING TIME TO START AS CO-VARIATE



48: Alright. Delete the node from l.
62: Delete the node, the node, delete the node from l
48: that is pointed to by P.
48: OK. So, we have to delete this one.
62: What? I don't understand. Delete the node.
48: We are deleting.
62: From L that is pointed to by P.
48: Yeah so it is asking what node.
62: So it's like the the last time
48: Yes, but this time we have two lists. Or something.
48: I don't know.
48: So, uh.
48: I'm trying to think like if is there an easier way of doing this rather then create a temp node?
48: I think so.
48: No.
48: It must be the same thing.
48: We have to walk through the axis and have to make this point to this one.
48: So it's the same thing as the last time. Pretty much.
[*Begins coding*]

Figure 20: Excerpt of student dialogue at the beginning of problem four in the semistructured condition.

5.7.2.2 Simple Speech Signals

Due to the fact that students use Collab-ChiQat in "real-world", classroom settings in order to learn, we aim to explore tractable solutions to automating our feedback in real-time. The use of automatic speech recognition (ASR) signals from students as they work in pairs is one such method. Thus, our regression with collaboration features also includes assessments of the quality of dialogue features for modeling learning. We used raw counts of our domain-related keywords as features. Analysis revealed that use of the words *loop* and *while* were significantly correlated to learning in the unstructured condition (shown in Table XVI). As described in Section 5.6.1, the unstructured condition had the higher volume of domain word usage. These words with significant correlations to learning are intuitively important as they are essential for understanding how to solve the problems with increasing difficulty. Furthermore, pair use of the word *undo* was significantly negatively correlated to learning ($p < .05$). We believe a pairs' use of the word *undo* showed that they were confused or uncertain about the direction that the code should take.

Model	**Adjusted-R^2**	**p-value**
loop	0.555	$p = 0.00$
while	0.55	$p = 0.00$
undo	0.43	$p = 0.05$

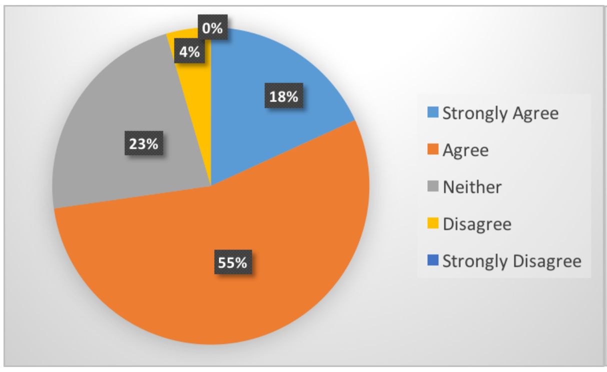
TABLE XVI: POST-TEST SCORE MODELS USING DOMAIN WORD COUNTS AS CO-VARIATE

5.8 Student Perceptions from Survey

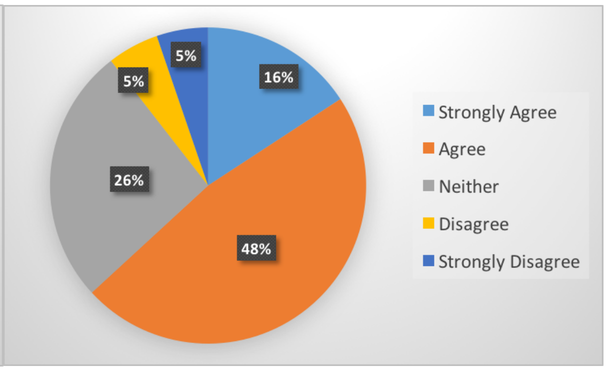
Finally, we examined the student responses to survey questions to see their perceptions of collaborative coursework, themselves as practitioners of computing, and Collab-ChiQat. We also performed the exit survey in order to capture students' understanding of the skills needed for effective collaboration. The complete survey questionnaire is given in Appendix C.

Survey results from students revealed that roughly the same proportion of students found the system helpful and interesting across conditions as depicted in Figure 21 . However, students in each condition had distinct perspectives on the attributes of a good pair programming team. Students were asked to rank a set of attributes of a good pair programming team in order of most to least important. These attributes were based on a prior survey conducted with Microsoft developers and engineering managers (Begel and Nagappan, 2008). Students in the unstructured condition ranked *Common Goals*, *Good Communication*, and *Fast and Efficient* to be the top three attributes. Contrastingly, students in the semistructured condition ranked *No Ego*, *Common Goals*, and *Flexibility* as top attributes. This suggests that the semistructured attributes place importance on individual acts that may be perceived as deference to their partner (*No Ego* and *Flexibility*). On the other hand, unstructured users focus on group process and outcomes. Both groups share an understanding of the importance of *Common Goals*. A visualization of student responses is given in Figure 22.

This same trend persists in students' free responses to top attributes. Students in the semistructured condition used words such as *understanding*, *code*, and *confidence* which do not appear at all or as frequently in unstructured student responses. Instead, the unstructured responses such as *respect* and *trust*, which do not appear in semistructured, allude to more natural group symmetry. These free responses are visualized in Figure 23 and Figure 24. We believe the semistructured students were much more cognizant of individual performance and

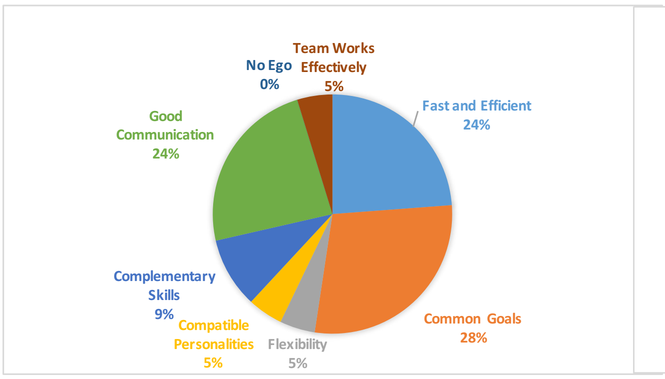


(a) Unstructured

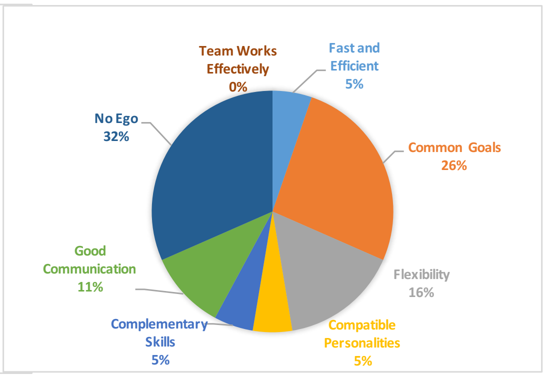


(b) Semistructured

Figure 21: Student responses to survey question: *"Do you feel that (Collab)ChiQat helped you learn about linked lists?"*



(a) Unstructured



(b) Semistructured

Figure 22: Unstructured (left) and semistructured (right) most important attribute of a pair programming team.

contribution to the group due to the collaboration feedback. Thus, their responses reflect much more of an attitude of self-sacrifice and endurance for the sake of the team.

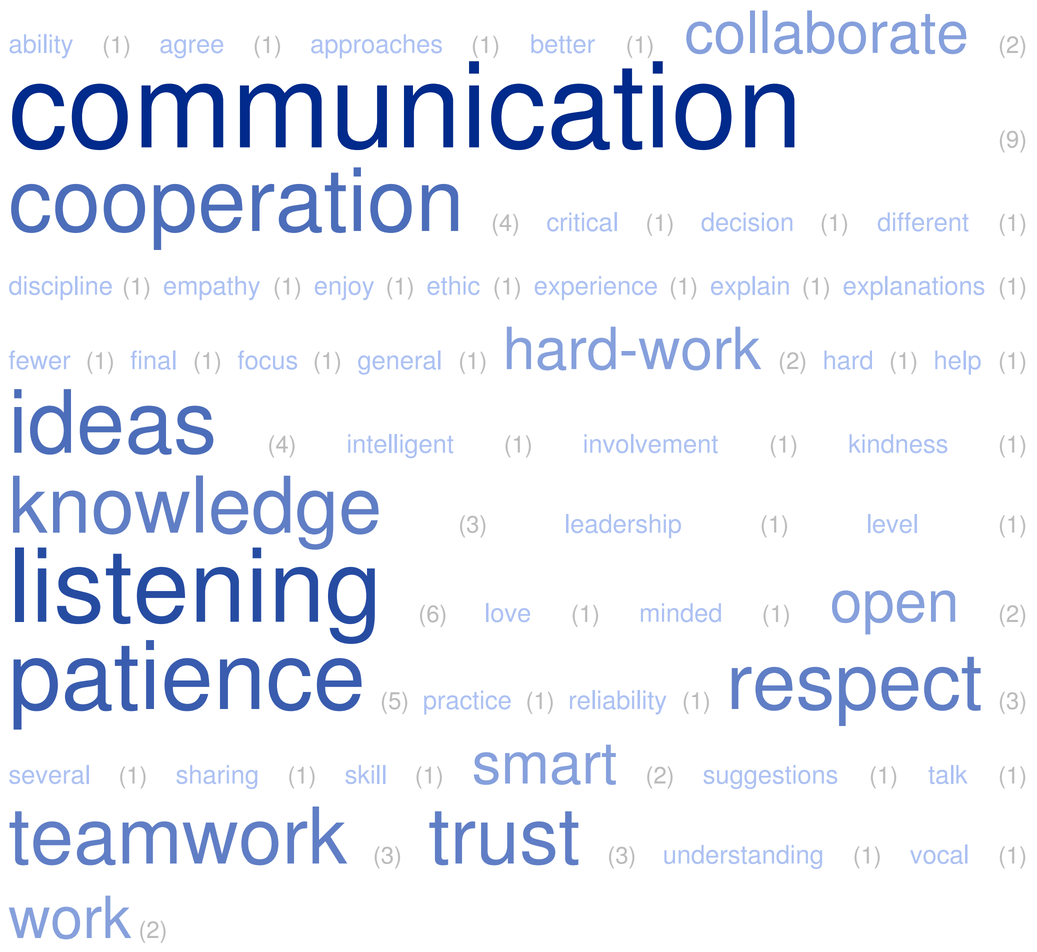


Figure 23: Visualization of student responses to traits of good programming pairs for the Unstructured condition.

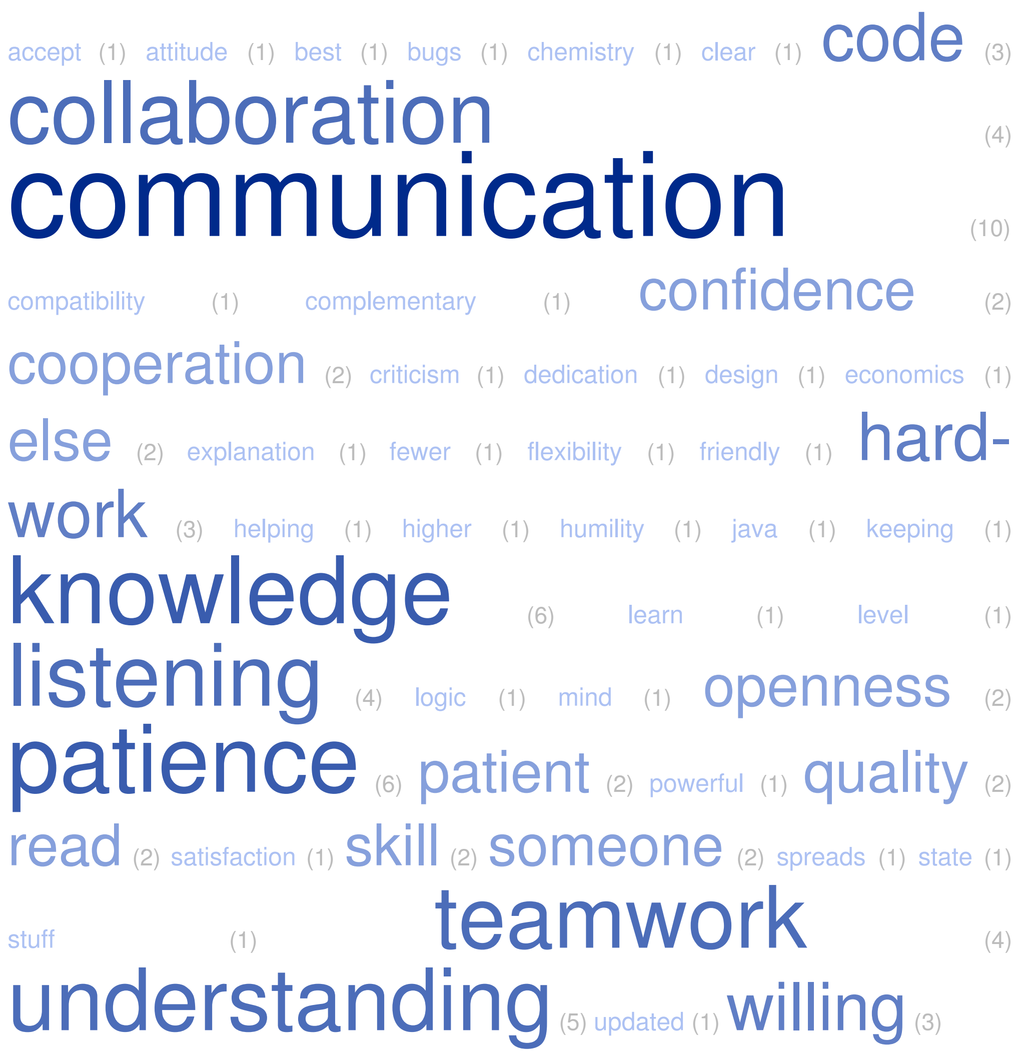


Figure 24: Visualization of student responses to traits of good programming pairs for the Semi-structured condition.

We also examined the relationship between students' self reported survey confidence and their learning using the Pearson Correlation Analysis. There was a trending correlation (R=0.29, p <.10) between answer to the question of whether working with groups gave the student added confidence and learning gain. Moreover, there was a significant correlation (R=.29, p <.05) between confidence in ChiQat code submission correctness and post test performance. There was also significant correlation between confidence in post test performance and actual post test scores. Overall, student confidence plays a major role in their learning experience.

There was a significant correlation (R=0.3996805, p <.01) between student's perception of ChiQat's helpfulness and their post test scores. Additionally, the correlation between student's perception of the benefit of taking time to help others and their post test results was trending toward significance (R=-0.25, p <.10). This trend shows that students who find spending time to help others to be useful, also scored higher on the post test.

Lastly, we explored student's perception of Collab-ChiQat in relation to their sentiment in regards to working with partners in general. As intuition would suggest, there was a significant correlation (R=.44, p<.001) between students who valued spending time helping others and those who found Collab-ChiQat to be helpful. Along the same line, there was a significant negative correlation (R=-.30, p<.05) between agreement that it takes more time to work with a partner than to work alone and those who found working with Collab-ChiQat to be helpful. In short, if students felt working with a partner was more time consuming, they also tended not to find Collab-ChiQat to be helpful.

5.9 Conclusion

In this chapter, we compared the effects of collaborative intelligent tutoring system design choice on students' interaction and learning process. Over 100 students used the unstructured and semistructured Collab-ChiQat systems in the classroom and achieved significant learning gains. We discovered that added collaborative feedback can promote planning before problem solving. Moreover, in comparing the effects of the two collaborative systems, there was a significant difference in students' dialogue features such as the number of questions asked and the frequency of domain-related word use. While we attribute the added collaborative feedback to differences in features such as planning time and symmetry there is still more work to be done in correlating which exact features of the collaboration panel produce these effects. For example, both the compile error and success rate graph and the collaboration score may lead students in the semistructured condition to take more time to start, and the relative effective of either is currently not distinguished. Though intuition and prior research has led our analysis to this point, ongoing work would benefit from deeper exploration.

Finally, the majority of students in both conditions found the intervention to be helpful and interesting, especially those who favored working in teams. However, these distinctions did not result in a significant differences in learning gains between the versions.

CHAPTER 6

USING DATA TO DESIGN FOR EFFECTIVE COLLABORATIVE LEARNING

6.1 Summary

In this chapter we explore how data can be used to design for effective collaborative learning. We begin by presenting our analysis of modeling effective collaboration. We borrow from the literature on learning and discussions on what constitutes relevant outcomes for collaborative learning and thus narrow our focus to group cognition (Stahl, 2005; Strijbos, 2011). Moreover, we operationalize this paradigm through the metric of knowledge similarity, which we measure via convergence of partner scores in pre and post-test. We also connect it to the the linguistic phenomenon of entrainment. We present outcomes for both metrics. Following this discussion, we then present our design of *hybrid* Collab-ChiQat, which incorporates our models of learning presented in Chapter 5 and our understanding of interactions leading to collaborative learning.

6.2 Measuring Collaborative Learning

While the goal of the system is ultimately that students achieve the highest possible individual learning gain, the goal of the system is also to serve as a facilitator of effective collaboration through guidance of situation, interactions, and processes (Dillenbourg et al., 2009). These goals are not necessarily tangential, but instead may align with each other as evidenced in the modeling of student learning in Chapter 5 where we showed that planning and symmetry of action lead to higher learning gains. Thus, we believe, and CSCL literature supports, that as the system is better able to facilitate collaboration, irrespective of the degree of structuring, the students individual learning should benefit.

There are several commonly accepted criteria for assessing collaborative learning outcomes. They span three major perspectives; motivational, social (cohesion), and cognition (Strijbos, 2011). We focus our current outcome analysis on group cognition. The construct of group cognition concerns itself with the act of collective meaning making in group settings (Stahl, 2006; Akkerman et al., 2007). It's evaluation spans multiple psychological and educational frameworks and thus, the spectrum of theoretical positions on the proper unit of assessment vary from individual(cognitivist) to group (socio-cultural) units (Stahl, 2006; Akkerman et al., 2007). In particular, we examine group knowledge convergence (also known as *consensus, mutuality, equivalence, similarity* in the literature), dialogue-based initiative shifts, and lexical entrainment (alignment, adaption).

6.2.1 Knowledge Convergence Outcomes

Knowledge convergence is a measure of the amount of knowledge students have in common following a collaborative learning exercise. While complete convergence (students have the same individual knowledge) is often the goal, it is also unlikely to occur (Strijbos, 2011). Thus, knowledge similarity, or the knowledge level becoming more similar between partners, is more likely to occur, while, knowledge divergence is also a possible outcome of interaction. We measure knowledge convergence as the difference between partner *S1* and partner *S2* post-test scores as below:

$$conv = |postTestScore_{S1} - postTestScore_{S2}| \tag{6.1}$$

where *conv = 0* is perfect convergence.

Given the role of group cognition as an accepted outcome for assessing the effectiveness of student collaboration within the CSCL and learning community, knowledge convergence is a reasonable measure. In order to establish a lower bound for knowledge convergence which results from the collaborative learning exercise, we calculated the convergence score for both the pre-test and post-test. We also introduced an additional metric, convergence change, which is the difference in pre-test and post tests convergence scores. Convergence change allows us to establish whether students' shared knowledge improved. The formula for convergence change, *conv'*, is given below.

$$conv' = |postTestScore_{S1} - postTestScore_{S2}| - |preTestScore_{S1} - preTestScore_{S2}| \tag{6.2}$$

Paired t-test analysis revealed that there was no significant difference between the pre and post test convergence scores when considering the conditions both individually and combined. However, using conv', we noted that 50% of students increased their level of convergence between the pre and post test, 10% remained the same, and 40% decreased convergence as depicted in 25a. Moreover, because the problem difficulty increases as students progress in both the tests and tutoring problems, we believed it would be worthwhile to examine convergence per individual test problem. In fact, students did have significant changes (μ=.65, p=0.003) when individually examining convergence scores for the first of the three test questions as detailed in Table XVII and Figure 25. As most students did not complete all tutoring problems, they may be unable to establish significant changes in convergence beyond the first problem which more closely emulates the problems they encountered during the tutoring. As a final measure, we tested for by-problem convergence in the unstructured and semistructured conditions separately. Results showed that there was a significant, positive change in convergence for test problem one in both the unstructured and semistructured conditions. This means both methods of collaboration structuring allowed students to build shared knowledge of certain concepts.

Further exploration would benefit from explicit definition and assessment of student mastery regarding specific knowledge components. Our current pre/post test is a sound method for evaluating student's working knowledge of linked lists from a cumulative, applied-learning perspective. However, we do not explicitly track knowledge components, which are smaller units of cognitive structures and processes that a student uses. For example, knowledge components for our domain might include the following:

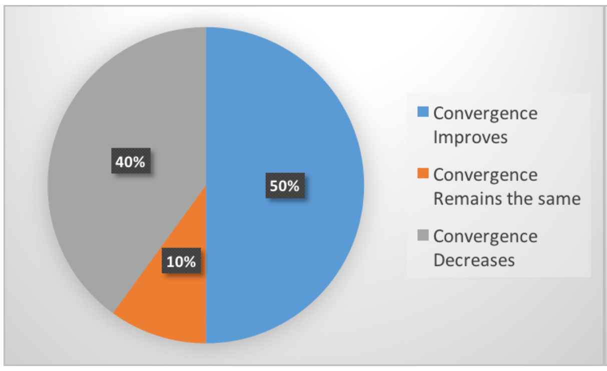
- Facts such as the structure of a list consists of nodes which have links and data.
- Procedures such as list traversal through use of links.
- Rules such as in-order node insertion.

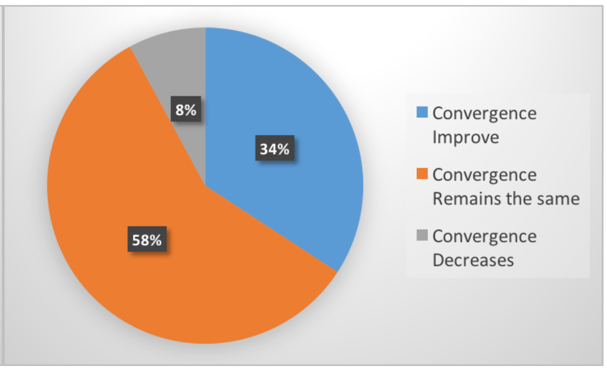
Evaluation of student knowledge component mastery would enable us to more clearly assess knowledge convergence at a micro-level, one which is more in line with mental models. In any case, the current analysis provides compelling rationale for further exploration of changes in group cognition resulting from the intervention.

We also aimed to determine if students converged in the proper direction. This would allow us to determine the strength of knowledge convergence as a metric in terms of improvements to student learning. In order to explore this, we examined, among those students whose level of convergence improved, was there any negative learning gain. Our analysis revealed that in only two groups where convergence improved, a student in the pair had negative learning gains. Moreover, in both cases, this student had a perfect pre-test score. Thus, essentially the student that was most knowledgeable about the topic may have became misinformed through their interaction with their partner and thus did not perform as well. Alternatively, these students may have tired on the post-test and opted not to fully recite information they had previously shared in the pre-test without the benefit of the tutoring. Ultimately, this exploration provided further support for knowledge convergence as a valuable metric for assessing the strenght of collaboration.

Method	**Pre-Test Problem One**		**Post-Test Problem One**		**p-value**	**t-value**
	μ	σ	μ	σ		
Unstructured	0.95	1.17	0.50	.80	p = 0.02	t = 2.48
Semistructured	1.88	1.59	.94	1.18	p = 0.05	t = 2.11
All	1.34	1.42	.68	.99	p = 0.00	t = 3.07

TABLE XVII: T-TEST COMPARISON RESULTS OF KNOWLEDGE CONVERGENCE BETWEEN PROBLEM ONE OF PRE AND POST TESTS.





(a) All Problems

(b) Problem One

Figure 25: Percentage of students whose converges increase, remains the same, or decreases based on total test performance (left) and problem one (right).

6.2.2 Dialogue Analysis

Given the critical nature of linguistic cues as building blocks of cognition in a collaborative problem solving task, we hypothesized that dialogue features can be indicators of group level cognition, our measure of effective collaboration. This section investigates both individual and socio-cultural measures of group cognition through analysis of dialogue during collaborative problem solving. In particular, we examine lexical entrainment (alignment, adaption) and dialogue-based initiative shifts. While lexical entrainment measures the subconscious alignment of speech signals between conversational partners, dialogue initiative tracks the leader in determining the focus of the conversation (Chu-Carroll and Brown, 1997; Jordan and Di Eugenio, 1997; Nenkova et al., 2008). Both of these are socio-cultural measures. However, we also analyze their relationship to group knowledge measures.

Our analysis presented in Chapter 5 revealed that initiative shifts play an important role in individual learning. Further, we discovered that the correlation between the difference in total initiative shifts between partners halfway through the session and the difference in final learning gains between partners is trending toward significance ($p < .10$, $t = 1.772$). Moreover, the difference in initiative between partners at halfway point is also significantly correlated to the difference in pre-test scores between partners ($p < .05$, $t = 2.1448$). Thus, tracking initiative shifts is a valid metric for assessing group dynamics such as prior knowledge level and the progression of knowledge gain.

In our analysis we also examined lexical entrainment, which measures the alignment of speech signals between conversational partners. In effect, lexical entrainment captures how the words that interlocutors use throughout a conversation become more similar with time. Research has shown that interlocutors converge on multiple levels including amplitude, syntactic and semantic constructions, and even sentence complexity when conversing (Nenkova et al., 2008; Xu and Reitter, 2016). Moreover, researchers have linked lexical entrainment to task success in game scores and semester-long team performance scores for students (Nenkova et al., 2008; Friedberg et al., 2012). Ultimately, the level of lexical entrainment may serve as an indicator of the extent to which a group shares a mental modal of a concept, in this case, their mental model, or internal representation, of linked lists (Friedberg et al., 2012). Thus, we believe lexical entrainment is another method of assessing group cognition outcomes in collaborative learning.

In this analysis, we aimed to assess if entrainment correlated to a group's post-test performance, knowledge convergence, and knowledge convergence change scores. Our measure of entrainment follows the work of (Nenkova et al., 2008). It is the negation of the absolute value of the difference in the proportion that speaker one *S1* uses word w subtracted from speaker two's (*S2)* proportional use. The formula for entrainment, *entr(w)*, is as follows:

$$entr(w) = -\left| \frac{count_{S1}(w)}{ALL_{s1})} - \frac{count_{S2}(w)}{ALL_{S2}} \right| \tag{6.3}$$

We measured entrainment scores for two equal halves of the intervention and calculated our entrainment score using only the domain-related words (ie *node, click, link*) described in Chapter 5 as done in related work (Friedberg et al., 2012). From this, we derived an additional metric, entrainment change, as the difference in entrainment scores between the final half and first half of the session. This metric allows us to understand the change in entrainment over time similar to our measure of knowledge convergence change above.

We first used a t-test to discover if there were significant difference in entrainment scores between the high and low group performers. The group performance was calculated based on the sum of post-test scores. High performers achieved more than the median sum of post-test scores while and low performance did not. Following this analysis, we calculated the Pearson Coefficient to measure the correlation between entrainment and group performance.

The average entrainment in the first and second half of the intervention for both high and low post-test performers is shown in Figure 26.

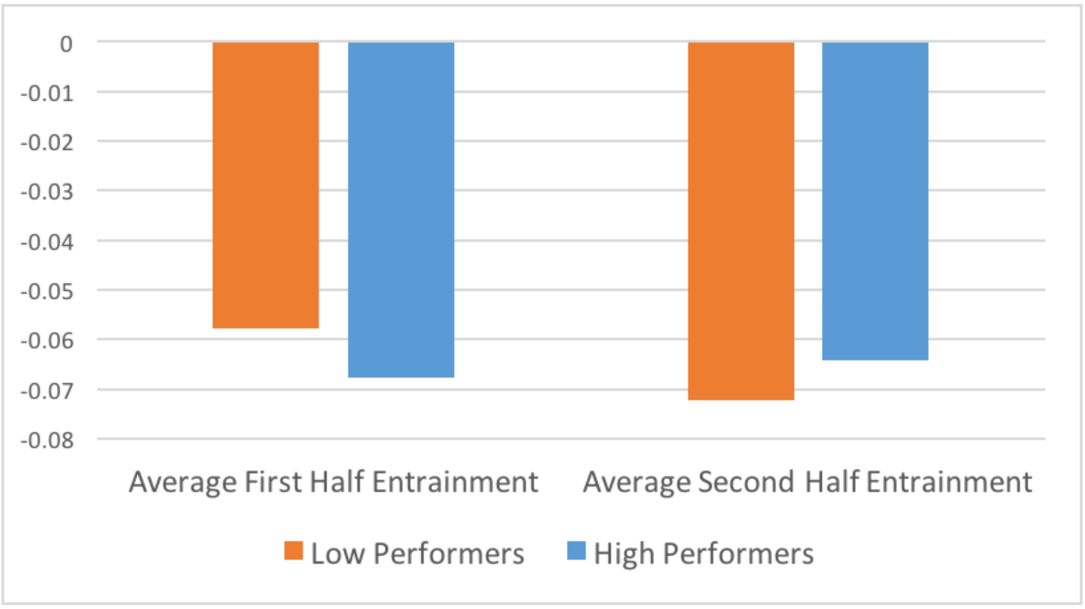


Figure 26: Comparison of entrainment in first and second half of session between high and low performers.

There was not a significant difference in the final entrainment score between high and low performers. However, there was a significant difference in the entrainment change score between these groups as shows in Table XVIII. In order to understand the nature of this result, we performed an additional paired t-test analysis to compare the first and second half entrainment scores within conditions. This test revealed that there was a significant difference ($p < .05$) between halves for low performing students. Specifically, the level of entrainment for low-performing students decreased. Thus, these results signify that low performing teams do not improve on their shared mental model as evidenced through entrainment. Instead, their lexical alignment significantly decreases as their use of the tutoring system progresses. Because we do not know the causal relationship between entrainment and performance, this result may suggest that either students are low performing because they are not entraining, or that they lack entrainment because they are low performing.

Method	**Score for Low Team**		**Score for High Team**		**p-value**	**t-value**
	μ	σ	μ	σ		
Half Entrainment Score	-.057	.013	-.067	.023	p=0.248	t = 1.20
Final Entrainment Score	-.072	.019	-.064	.023	p=0.457	t = 0.76
Entrainment Change	-.014	.016	-0.003	.011	p=0.028	t = 2.43

TABLE XVIII: T-TEST COMPARISON OF ENTRAINMENT SCORES AND ENTRAINMENT CHANGE BETWEEN HIGH AND LOW SCORERS

A Pearson Correlation test showed no significant correlation between the group performance and the final entrainment score though the correlation was trending toward significance ($p < .10$). On the other hand, there was a significant correlation between a group's post test performance and their entrainment change ($R = .75$, $p < .01$). This further evidenced the notion that lower performing teams entrain less and to such an extent that their entrainment is an indicator of their performance. This result is shown in Table XIX. These findings are consistent with prior literature which found entrainment could predict task success and that higher performing teams entrain more than their lower performing counterparts (Reitter et al., 2006; Friedberg et al., 2012).

In order to better understand the relationship between entrainment and group performance, our work would benefit from exploration of *how* the teams entrain. This includes analysis of the actual domain words used and the context in which they are used. For example, in a low performing team, one student may ask many questions and these questions may be laced with domain words. While, on the other hand, the partner may only respond with uncertainty of their own such as "I'm not sure". This pattern of interaction would decrease their level of entrainment over time. Thus, this example demonstrates that the context in which the groups uses the domain words and subsequently entrain (or not) is a valuable area to explore. It will allow us to better understand where the differences in the pair's mental models occur and how the system can help to support students.

Method	Pearson Correlation Coefficient	p-value
Final Entrainment	.439	p=0.088
Entrainment Change	.757	p=0.000

TABLE XIX: PEARSON CORRELATION RESULTS FOR ENTRAINMENT AND GROUP POST-TEST PERFORMANCE

6.3 Hybrid Collab-ChiQat

Our investigation of effective collaborative learning outcomes in addition to analysis presented in prior chapters led us to the design of an additional Collab-ChiQat version, *hybrid*. The hybrid version, as its name suggests, is the intermediary between unstructured and semistructured Collab-ChiQat, which purposes to benefit from the strengths of each. Whereas students in the unstructured condition were on-topic and asked more questions, students in the semistructured condition took time to plan before coding. Moreover, both groups experienced significant knowledge convergence in relation to key concepts. Thus, the hybrid system design was created in order to integrate our findings in a new system and further investigate the effects on collaboration and learning.

In terms of updates, the hybrid system no longer has the collaboration panel interface. We discovered that the panel served two primary purposes for students, 1) increase awareness of individual participation and turns 2) motivate planning in order to improve group performance.

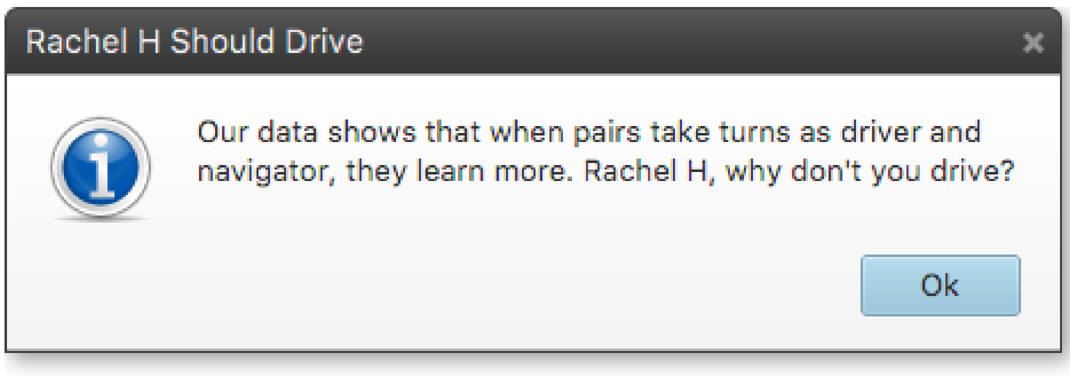
We believed that we could accomplish these things without the collaboration panel in itself. We also saw that while the panel may have motivated students students in the semistructured condition to engage in problem solving, their gains were not significantly higher or different than the unstructured condition. Thus, we follow the guidance of Occams Razor and remove the panel in favor of a simpler interface. We describe further changes in design choice for hybrid Collab-ChiQat below with reference to supporting rationale.

6.4 Design Changes and Rationale

6.4.1 Turn Taking

Hybrid Collab-ChiQats collaboration feedback aims to promote symmetry of participation. In particular, the system monitors code turn taking patterns. Group symmetry is not only a theoretical sign of effective collaboration, but it is also a correlate to learning as evidenced through our experimentation. Our model of learning shows that when students in both conditions took coding turns of length four, the students learned ($p < .01$, Adjusted-$R^2 = 0.59$). This is also supported by the well-accepted paradigm of role-switching within CSCL literature. Role switching promotes turn-taking behavior that contributes to social grounding (Inaba and Mizoguchi, 2004; Soller, 2001)). Notably, role-switching often coincides with activities such as asking questions, clarifying views, and commenting on peer work.

With this motivation for implementing turn taking feedback, we updated the system in order to monitor student turns. At the launch of each problem, if one student has fallen more than four coding turns behind another student, then the system prompts students to switch roles as driver and navigator. The tutoring feedback is displayed in a modal popup as shown in Figure 27.

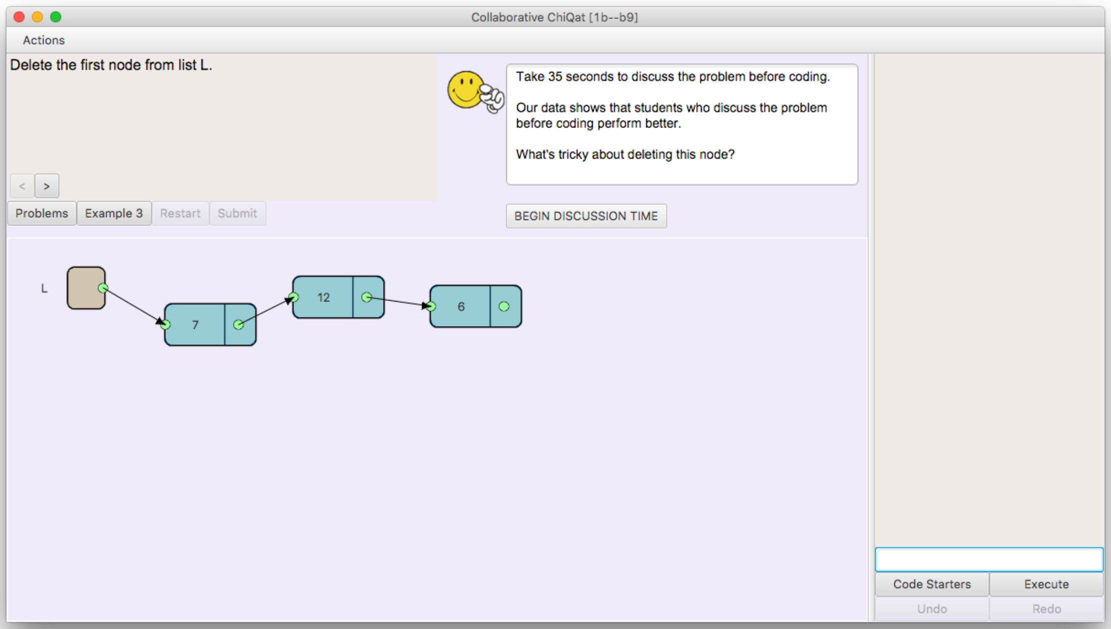

Rachel H Should Drive

Our data shows that when pairs take turns as driver and navigator, they learn more. Rachel H, why don't you drive?

Figure 27: Tutor feedback to promote role-switching after imbalance detected.

6.4.2 Time to Start

Our prior analysis of learning revealed that taking time before coding in problems four through seven was significantly positively correlated to learning (Section 5.7.2.1. Moreover, this analysis also showed that the amount of time students take to begin problem one is trending towards a significant negative correlation to learning. Given the literature which endorses planning as an indicator of effective collaboration and these findings as substantiation, we implemented tutoring feedback to promote planning. First, we calculated average time to start for problems one and four through seven. Then, the tutoring strategy was updated to suggest that students take an allotted amount of time before coding to discuss their approach. The time was based on the calculated averages. The tutor also provides students with a discussion starter question which are relevant to the particular problem. A sample tutor prompt for this strategy is shown in Figure 28.


Delete the first node from list L.

Take 35 seconds to discuss the problem before coding.

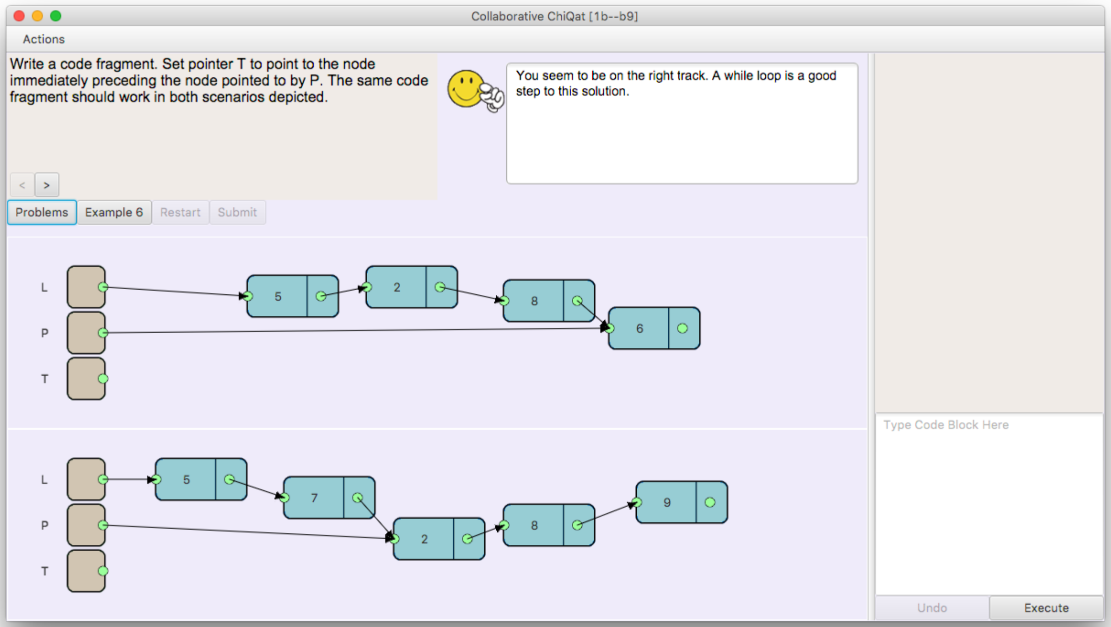
Our data shows that students who discuss the problem before coding perform better.

What's tricky about deleting this node?

Figure 28: Tutor prompt for student planning of solution approach at onset of problem.

6.4.3 Keyword Detection

Though automatic speech recognition is not without flaws, we believe a moderate approach to its future use could be the detection of keywords. Collab-ChiQat, and indeed ASR technology, has not reached the point to which we can implement full-scale natural language understanding of the pairs dialogue. However, our model of learning allowed us to understand ways to support students through simple keyword detection. In particular, we focused on detecting the words *while*, *loop*, and *undo*. Within the contexts of students problem solving activity, we used detection of these words to provide relevant feedback. For example, if the system detects repeated use (n=2 within same problem) of the word *undo*, it triggers proactive feedback. Further, when given time to discuss solutions for problems six and seven, if a student does not mention the word *while* the tutor reminds them that the while-loop conditional structure could help with list traversal. Conversely, if they do mention the structure, they are provided positive feedback in regards to the while-loops applicability for the problem. An example intervention prompt is shown in Figure 29



Write a code fragment. Set pointer T to point to the node immediately preceding the node pointed to by P. The same code fragment should work in both scenarios depicted.

You seem to be on the right track. A while loop is a good step to this solution.

Figure 29: Tutor response after keyword *loop* detected during discussion time.

6.5 Experimental Design

We evaluated the hybrid Collab-ChiQat system in an experiment that took place in Fall of 2016. Our goals was to investigate the effect of design choice and enhanced modeling in the hybrid Collab-ChiQat system. The setting was the same second year Computer Science course as in prior experiments. However, the study took place later in the semester than in the prior experiments. At the time of this study, students had already been introduced to linked lists via an in-class lab exercises and a take-home project.

There were two conditions in the experiment, the unstructured and hybrid versions of the system. The unstructured version was the same as in the prior experiment. Additionally, other aspects of the experimental setup remained the same as in prior experiments. Students chose their partners and used the system for 40 minutes. Students chose partners, individually performed pre-test, post-test, and an exit survey, and they watched the informational pair programming video (detailed in Appendix B) at the onset of the class. In total, 114 students elected to participate in the study and share their data. 58 students were in the unstructured condition while 56 while were in the hybrid condition. Regrettably, due to server errors, we were unable to capture the extensive log data which allowed us to retrace students time with the system.

6.6 Outcomes

6.6.1 Learning

We began by understanding the impact of structure on student learning. We evaluated learning gain based on the difference between post and pre test as previously.

$$gain = postTestScore - preTestScore \tag{6.4}$$

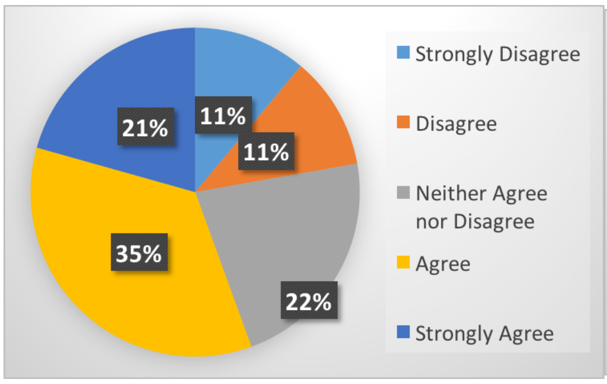
The learning gain results showed that both the unstructured and hybrid groups experienced significant learning gains with no significant difference in learning gains between the two conditions. As expected, due to the experiment's timing, the pre-test scores were significantly higher than in our prior experiments (p <.05). The full learning gain results are listed in Table XX.

Tutor	**N**	**P**re-test		**P**ost-test		**G**ain	
		μ	σ	μ	σ	μ	σ
Unstructured Collab-ChiQat	58	.64	.22	.77	.16	**.13**	.14
Hybrid Collab-ChiQat	56	.56	.25	.66	.21	**.10**	.12

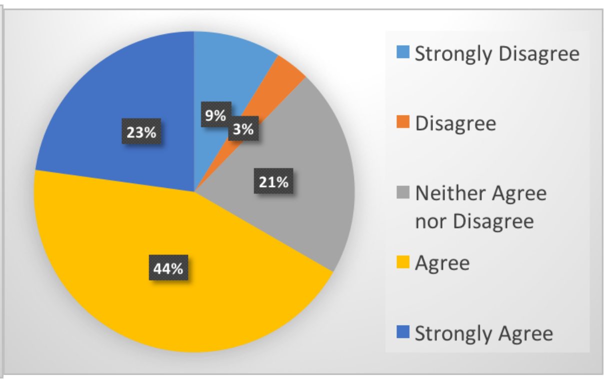
TABLE XX: COMPARISON OF LEARNING GAINS BETWEEN UNSTRUCTURED AND HYBRID COLLAB-CHIQAT.

6.6.2 User Satisfaction

The exit surveys showed that the majority of students in each condition found the system to be both helpful and interesting as shown in Figure 30. Unsurprisingly, students' perceptions of the system's helpfulness was significantly correlated to their perceptions of the benefits of working in teams. Specifically, students who believed they work more thoroughly on their own (Response 11, R= -.20, p=.02) and who find that working with a partner takes more time (Response 13, R= -.19, p=.03), also did not think the system was as helpful.



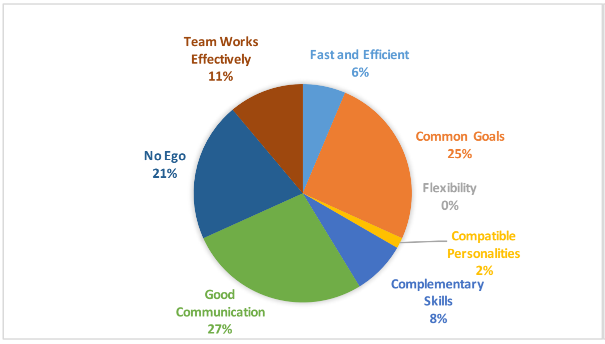
(a) Unstructured



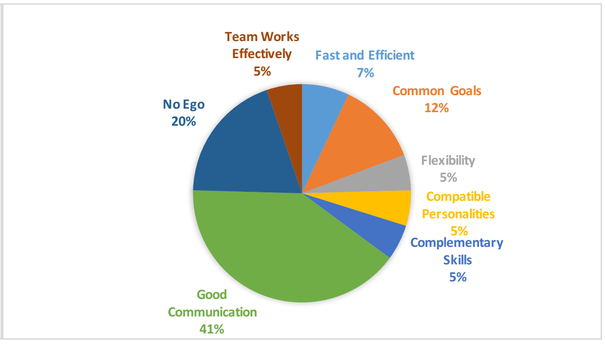
(b) Hybrid

Figure 30: Student responses to survey question: *"Do you feel that (Collab)ChiQat helped you learn about linked lists?"*

Additional exit survey analysis revealed that 47% of hybrid system users choose *Good Communication* as top trait for successful pair programmers as opposed to 27% in the unstructured condition (shown in Figure 31). In the prior experiment, 24% of students in the unstrucutred condition and 11% in the semistructured condition chose *Good Communication* as the top trait. We believe this difference is indicative of the influence of the "Time to Start" intervention which emphasized student discussion. Notably, this choice of communication also coincides with an industry survey of Microsoft developers and managers in which they also chose "Good Communication" as the top trait (Begel and Nagappan, 2008).



(a) Unstructured



(b) Hybrid

Figure 31: Unstructured (left) and hybrid (right) most important attribute of a pair programming team.

6.7 Audio Analysis

The goal of our audio analysis was to further assess the quality of the automatic transcription provided through the Google Speech API. We first began with a comparison of the most frequently occurring words between the auto-transcription of the Fall 2016 session and the manual transcription of the Fall 2015 session. Though we expected differences, we believed that evidence of a high quality automatic transcription would yield more similarities between the two experiments. The results of the comparison of the top 15 most frequently occurring domain words are in Table XXI. Words in **bold** were in the top most-frequent manual transcription, but not in the automatic transcription.

The analysis revealed that the auto-transcription recognizes some domain-related words, however, it may miss a large portion of them. Words included in the most frequent Fall 2015 manual transcription such as *node*, *null*, and *link* did not appear in the Fall 2016 list until much later in order of frequency. However, for example, in the case of *link*, there were several words present in the auto-transcription that may have been a misinterpretation of *link* such as *linkedin*, *linkin*, *linkange*, *xlinked*,*slink* and *outlink*. Furthermore, the automatic transcription had difficulty with the tutoring system's variable names such as *L*, *L1* and *L2*. Though they were occasionally recognized, their frequency did not match the expected amount of student's usage based on the prior experiment.

Overall, a t-test revealed an extremely significant difference (p <0.0001, '15 μ = 0.103 σ =0.026; '16 μ=0.049 σ= 0.015) between the domain word frequencies between Fall 2015 and Fall 2016. However, because the system was able to recognize a large portion of words, though not as consistently, we are encouraged by the results of the auto-transcription. Further, we recognize that future work would benefit from informing the ASR with a language model that is more trained to programmatic speech. We can also consider similar words, i.e. *link* and *slink*, to be equivalent in our real-time system processing.

Fall 2015		**Fall 2016**	
Manually-Transcribed Word	Frequency	Auto-Transcribed Word	Frequency
node	0.79%	next	0.49%
link	0.73%	equals	0.42%
L	0.55%	equal	0.40%
equals	0.54%	point	0.36%
equal	0.47%	if	0.20%
point	0.47%	while	0.20%
next	0.37%	node	0.17%
null	0.30%	L	0.16%
if	0.25%	delete	0.12%
new	0.25%	list	0.15%
delete	0.22%	pointer	0.11%
pointer	0.20%	new	0.09%

TABLE XXI: TOP 12 MOST FREQUENT DOMAIN WORDS BETWEEN FALL 2015 MANUAL TRANSCRIPTION AND FALL 2016 AUTOMATIC TRANSCRIPTION.

6.8 Conclusion

In this chapter we addressed the issue of modeling of collaborators during joint problem solving through the metrics of knowledge convergence, dialogue initiative, and lexical alignment. Our analysis revealed that students converged on some knowledge topics. Moreover, our analysis showed that shifts in initiative correlate to cognitive measures of group knowledge assessed both prior to and following the tutoring intervention. Moreover, we discovered that lexical entrainment of interlocutors has distinct changes between halves and is significantly correlated to group knowledge. Finally, the results of this analysis and those presented in prior chapters motivated the design of the hybrid Collab-ChiQat system which over 100 students used and experienced significant learning gains.

CHAPTER 7

CONCLUSIONS & FUTURE WORK

7.1 Contributions

This dissertation offers contributions to multiple research communities including CSCL, ITS, and CS Education. This work provides clear validation of the effectiveness of collaborative learning, in particular as supported through an ITS. It advances the depth of knowledge in regards to CITs, an emergent field that combines the benefits of CSCL and ITS for enhanced student learning.

In order to arrive at our collaborative ITS reconceptualization, I provided a practical design and evaluation framework for collaborative tutoring systems. It resulted from an extensive review and synthesis of literature of CSCL and ITS research. Thus, the dissertation investigated multiple, grounded methods for supporting collaboration based on this framework, each with their own influence on student interaction.

I originally developed two collaborative versions of our ITS for CS Education. The semistructured version provides automated feedback on student collaboration through visualization of group performance, individual contribution, and peer review, while the unstructured version allows students to work in pairs and does not offer feedback on their collaboration. Over 100 students used the systems in the classroom and achieved significant learning gains while having higher coding efficiency and speed with fewer worked out examples use as compared to individual learners.

We discovered that two of the CSCL and learning community's markers of effective collaboration, group symmetry and planning, also significantly correlate to learning in our intervention. Moreover, we discovered that added collaborative feedback can promote planning, and that keywords in student dialogue such as *undo* and *while* can also indicate a students' level of learning. In comparing the effects of the two collaborative systems, there was a significant difference in students' dialogue features such as the number of questions asked and the frequency of domain-related words. However, these distinctions did not result in a significant differences in learning gains between the versions. We also explicitly analyzed the effect of design choice on group cognition, an accepted outcome of collaborative learning. We operationalized this notion through the metrics of knowledge convergence and lexical entrainment.

With these findings as motivation, we then developed a third system version, hybrid. This system promoted symmetry in turn taking, use of planning time, and it implemented keyword detection for feedback triggering. Over 100 students used this system in the classroom as well and experienced significant learning gains. Further, the majority of students using either of the three versions found the system to be both helpful and interesting.

Overall, researchers across our areas of synthesis have already found value in this work. Findings discussed in this dissertation have also been published in notable peer-reviewed venues (Harsley, 2014; Harsley, 2015; Harsley et al., 2016c; Harsley et al., 2016; Harsley et al., 2016b; Harsley et al., 2017; Harsley et al., 2017). Lastly, it is my hope that this work will continue to help students to learn, remain in, and contribute to the CS discipline, especially those looking to connect with others during their studies.

7.2 Future Research Directions

There are several noteworthy avenues open for furthering this work. This following list provides an overview of promising possibilities:

- Student transcription analysis would greatly benefit from time-synchronization to student problem-solving activity including knowledge of current student roles. This can be accomplished through a smaller-scale study in which data is also video recorded. Additionally, as ASR technology improves, the quality of the system real-time transcription would also provide exact time-stamps correlated to system time. A time and system-action synced dialogue corpus would allow us to further analyze the roles, turn taking behavior, and much more in regards to student interaction.
- A key next-step for the investigation of the role of design choice in collaborative learning is the implementation of additional collaboration-centered feedback. This requires modeling for early-detection of collaboration issues as well the creation of further "goodness" of collaboration metrics just as we monitor progress in student code. This real-time model will be informed by our current work, such as the operationalization of mental models through lexical entrainment, and must also take further advantage of additional dialogue components of the interaction.
- Group formation is an important topic in collaborative learning. In order to account for group dynamics in the system, there are two approaches. 1) Develop an ability to suggest

group or group types to professors using ChiQat based on a students' course performance or 2) develop automatic pre-test grading to detect group dynamics. The second option is more viable given the large amount of student-test data we have collected. The data we have collected to date can serve as the basis for a machine learning approach to automatic grading. Finally, analysis of group dynamics in our current data can also inform development of tutor strategy.

7.3 Long Term Research Directions

There are also multiple areas of consideration for the long term research direction of this work. We would be remiss if not to explore affective aspects of student interaction, including motivation. In performing the manual transcription of student dialogue, I noticed a plethora of dialogue cues that may relate to the affective state of the students, including excitement, frustration, and anger. These may not all be automatically ascertained through currently available sentiment analysis methods. Annotation and analysis of these features would be a necessary step for gaining a better understanding of the nature of student collaboration and how it relates to learning.

An additional area for further work is an investigation of long-term CIT use. To date, all of our Collab-ChiQat interventions have lasted a single class period of no more than 1.5 hours including pre and post tests along with exit survey. An extended study is needed in order to tie our CIT use to larger metrics of student retention and perception of themselves as practitioners. With this mandates the creation of additional problem sets and topics beyond the lessons we currently support.

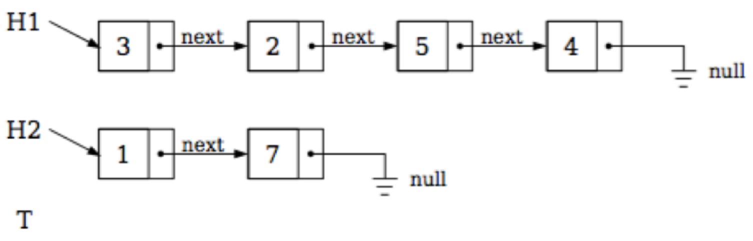
Finally, with the emergence of online CS learning tools, the future of this work may rest in retooling the system to be an extension for such systems. In this pattern, Collab-ChiQat's functionality would be serviced through an API with optional GUI components. To begin, the other CS learning tool would simply pass its dataset of problems and prior student interactions to the Collab-ChiQat API. From there, Collab-ChiQat would adapt and be able to provide feedback to students through either notifications or subsequent API calls. This approach would rely heavily on both data mining and machine learning in addition to our foundations in learning theory. Moreover, as is the nature for Machine Learning based approaches, the quality of feedback and performance would improve with increased use among the learning tool providers.

APPENDICES

Appendix A

PRE/POST TEST

You have the following two *linked lists*, starting from the head pointers H1 and H2. You also have a temporary pointer T.



1. Look at the following procedure. The procedure is written in pseudo C/C++/Java, but don't worry about programming details such as declarations etc. What is the status of the data structures after its execution? Draw a picture representing them.

```
T = H2;
while (T.next != null) {
    T = T.next;
}
T.next = H1;
```

2. Consider the following "variation" of the same procedure. Why doesnt it work?

```
T = H2;
while (T != null) {
    T = T.next;
}
T.next = H1;
```

3. Write a sequence of operations, in pseudo-code or in a programming language of your choice, that moves the first node of the original list H1 to the end of the list.

Appendix B

PAIR PROGRAMMING VIDEO TRANSCRIPT

"Agile in Practice: Pair Programming" (AgileAcademyAus, 2011)

Hi and welcome to the Agile Academy in this agile in practice talk we'd like to show you how developers on an actual team use peer programming tool or defects help deliver faster and produce higher quality work. Let's have a look at how this practice works. The way developers are collaborating is a bit like driving a car on a rally. One of you is driving while the other is navigating reading a map while the driver is keeping their eyes on the road. The navigator is focusing on the destination and how to get there.

Regardless of the kind of work to do you may want to try out collaborating in the same way. Especially when you're dealing with a difficult task. It works better if both people are peers sharing the same role and having a similar style and experience level.

Here are Bill and Liz, two developers from the agile team who delivering a new employee performance review system for the HR department. They'll be working together side by side at the scene computer solving the same problem. They take turns using the keyboard and mouse.

At the moment, Liz is the driver. While she is writing the code and performs test on that code Bill as a navigator observes her work considering the strategic direction, the next steps, and potential pitfalls. He also considers is this the simplest design possible and comes up with ideas for improvement.

Appendix B (Continued)

After a period of time, they swap roles and Liz navigates while Bill drives. Because they are constantly communicating and reviewing together, they produce higher quality work at a faster pace than either of them could do on their own it. Also ensures that more than just one developer knows each part of the system. Pairs swap places frequently throughout the day and also new pairs are formed to promote sharing of knowledge throughout the team.

At the same time a uniform style of planning is carried out which makes working together easier. It also creates an ownership and commitment to the collective code.

Despite the benefits, pair programming is not always easy. As a driver you have someone observing you constantly and when you're navigating you need to refrain from rushing to point out details like missing semicolons or typos. Instead you need to get the driver time to write on rewrite the work while you focus on the direction the work is going.

Also you should consider that peer programming is is suitable for all tasks or situation. But if you choose to do it you need to respect each other, communicate, and make it easy on yourself by using a computer with a big screen.

Enjoy the ride!

Appendix C

EXIT SURVEY

1. Major (if anticipated, mark *Anticipated*):

---

2. Gender:

---

3. Today is the first time I've used the Linked List lesson in ChiQat.
True
False
4. Do you feel that ChiQat-Tutor helped you learn about linked lists?

1. Strongly Agree
2. Agree
3. Neither Agree nor Disagree
4. Disagree
5. Strongly Disagree

5. Do you feel that working with ChiQat-Tutor was interesting?

1. Strongly Agree
2. Agree
3. Neither Agree nor Disagree
4. Disagree
5. Strongly Disagree

6. When I submitted code for ChiQat, I was confident it was correct:

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

Appendix C (Continued)

7. I am confident I did well on the post-test exam:

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

8. Right now my plan is to:

1. Take another CS Course
2. Take several CS Courses
3. Minor in CS
4. Major in CS
5. I'm Unsure
6. Not take any more CS

9. I enjoy this class.

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

10. I am good with computers

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

Appendix C (Continued)

11. When I encounter difficulties with an assignment I always find a way to work though them.

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

12. I enjoy hearing a variety of ideas for how to solve a problem that I am working on.

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

13. My work is more thorough when I work alone than when I work with a partner or group.

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

Appendix C (Continued)

14. Working with a partner or a group gives me more confidence in the correctness of our work than when I work alone.

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

15. Taking extra time to explain course material to someone else is worthwhile because it helps me to learn the material more thoroughly.

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

16. For me, completing a project with a partner takes more time than working alone.

1. Strongly Agree
2. Agree
3. Mildly Agree
4. Mildly Disagree
5. Disagree
6. Strongly Disagree

Appendix C (Continued)

17. Fill in Multiple Blanks: Three attributes of a good pair programming partnership are [X], [Y], and [Z].
X:
Y:
Z:
18. Rank the attributes of a pair programming team in order of importance.

1. Team Works Effectively
2. Complementary Skills
3. Good Communication
4. Compatible Personalities
5. No Ego
6. Fast and Efficient
7. Flexibility
8. Common Goals

Appendix D

STANDARD FEATURE SET DESCRIPTIONS

Feature	**Description**
Collection_Duration	The time, in milliseconds, that the user used the system (from log in to logout)
Example_Completed	The user has completed viewing of the example
Example_Requested	The user requests an example from the system
Example_Started	An example has started to be played to the user
Executing_Example_Step	The user has clicked the 'next' button during an example, moving the example onto the next step
Feedback_Given	Feedback has been given to the user
Final_Pre	The pre-test score
Node_Clicked	The user clicked on a node within the graphical problem solving area
Operation_Execution_Submitted	The user has submitted an operation, such as a line of code, for a problem
Operation_Redone	The user elects to redo a step that was previously undone
Operation_Undone	The user elects to undo a previously submitted step

Continued on next page

Table XXII – continued from previous page

Feature	**Description**
Question_Answer_Response	The intelligent tutor has responded to an answer that the user has given to a posed question
Question_Given	A question has been posed to the user from the intelligent tutor
Solution_Submitted	A problem solution has been submitted for validation by the user to the intelligent tutor
Solved_Problems	The total number of problems the user has solved
Template_Help_Item_Selected	The user clicked on a code template item in order to help them write code for problem solving
Template_Help_Requested	The user has requested to see all commands that can be used in solving the related problem
Tutorial_Duration	The total duration that the user viewed the lesson tutorial
User_Acknowledge	User clicks on the intelligent tutor's 'ok' button, signifying that they have acknowledged any feedback

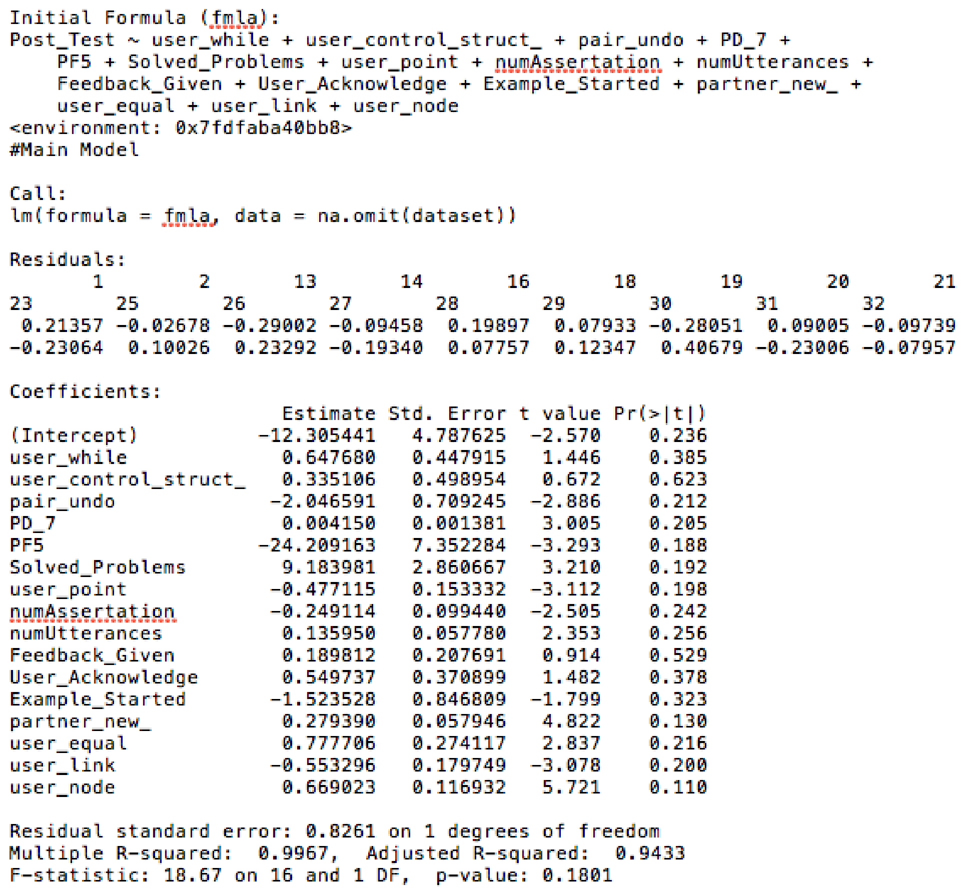
TABLE XXII: LIST OF STANDARD FEATURES USED IN MULTIPLE LINEAR REGRESSION MODELS ALONG WITH THEIR DESCRIPTIONS.

Appendix E

BEST FIT REGRESSION MODELS

Unstructured Model: Standard Features + Collaboration Features

R^2=.94

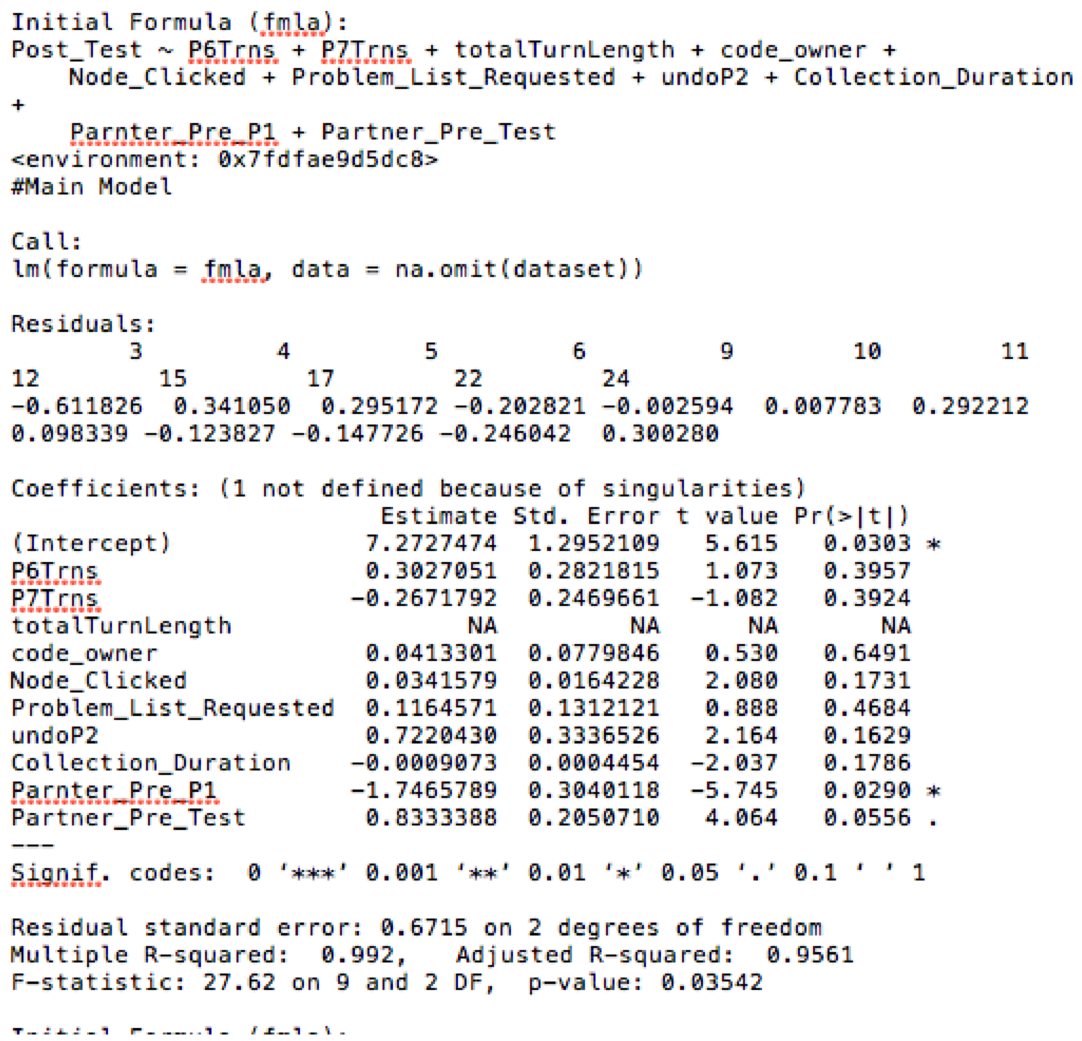


```
Initial Formula (fmla):
Post_Test ~ user_while + user_control_struct_ + pair_undo + PD_7 +
    PF5 + Solved_Problems + user_point + numAssertation + numUtterances +
    Feedback_Given + User_Acknowledge + Example_Started + partner_new_ +
    user_equal + user_link + user_node
<environment: 0x7fdfaba40bb8>
#Main Model

Call:
lm(formula = fmla, data = na.omit(dataset))

Residuals:
       1        2       13       14       16       18       19       20       21 
23       25       26       27       28       29       30       31       32 
 0.21357 -0.02678 -0.29002 -0.09458  0.19897  0.07933 -0.28051  0.09005 -0.09739 
-0.23064  0.10026  0.23292 -0.19340  0.07757  0.12347  0.40679 -0.23006 -0.07957 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)         -12.305441   4.787625  -2.570    0.236
user_while            0.647680   0.447915   1.446    0.385
user_control_struct_  0.335106   0.498954   0.672    0.623
pair_undo            -2.046591   0.709245  -2.886    0.212
PD_7                  0.004150   0.001381   3.005    0.205
PF5                 -24.209163   7.352284  -3.293    0.188
Solved_Problems       9.183981   2.860667   3.210    0.192
user_point           -0.477115   0.153332  -3.112    0.198
numAssertation       -0.249114   0.099440  -2.505    0.242
numUtterances         0.135950   0.057780   2.353    0.256
Feedback_Given        0.189812   0.207691   0.914    0.529
User_Acknowledge      0.549737   0.370899   1.482    0.378
Example_Started      -1.523528   0.846809  -1.799    0.323
partner_new_          0.279390   0.057946   4.822    0.130
user_equal            0.777706   0.274117   2.837    0.216
user_link            -0.553296   0.179749  -3.078    0.200
user_node             0.669023   0.116932   5.721    0.110

Residual standard error: 0.8261 on 1 degrees of freedom
Multiple R-squared:  0.9967,	Adjusted R-squared:  0.9433 
F-statistic: 18.67 on 16 and 1 DF,  p-value: 0.1801
```

Appendix E (Continued)

Semi-structured Model: Standard Features + Collaboration Features

R^2=.95



```
Initial Formula (fmla):
Post_Test ~ P6Trns + P7Trns + totalTurnLength + code_owner +
    Node_Clicked + Problem_List_Requested + undoP2 + Collection_Duration
+
    Parnter_Pre_P1 + Partner_Pre_Test
<environment: 0x7fdfae9d5dc8>
#Main Model

Call:
lm(formula = fmla, data = na.omit(dataset))

Residuals:
        3         4         5         6         9        10        11
12        15        17        22        24
-0.611826  0.341050  0.295172 -0.202821 -0.002594  0.007783  0.292212
0.098339 -0.123827 -0.147726 -0.246042  0.300280

Coefficients: (1 not defined because of singularities)
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)            7.2727474  1.2952109   5.615   0.0303 *
P6Trns                 0.3027051  0.2821815   1.073   0.3957
P7Trns                -0.2671792  0.2469661  -1.082   0.3924
totalTurnLength               NA         NA      NA       NA
code_owner             0.0413301  0.0779846   0.530   0.6491
Node_Clicked           0.0341579  0.0164228   2.080   0.1731
Problem_List_Requested 0.1164571  0.1312121   0.888   0.4684
undoP2                 0.7220430  0.3336526   2.164   0.1629
Collection_Duration   -0.0009073  0.0004454  -2.037   0.1786
Parnter_Pre_P1        -1.7465789  0.3040118  -5.745   0.0290 *
Partner_Pre_Test       0.8333388  0.2050710   4.064   0.0556 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6715 on 2 degrees of freedom
Multiple R-squared:  0.992,   Adjusted R-squared:  0.9561
F-statistic: 27.62 on 9 and 2 DF,  p-value: 0.03542

Initial Formula (fmla):
```

Appendix F

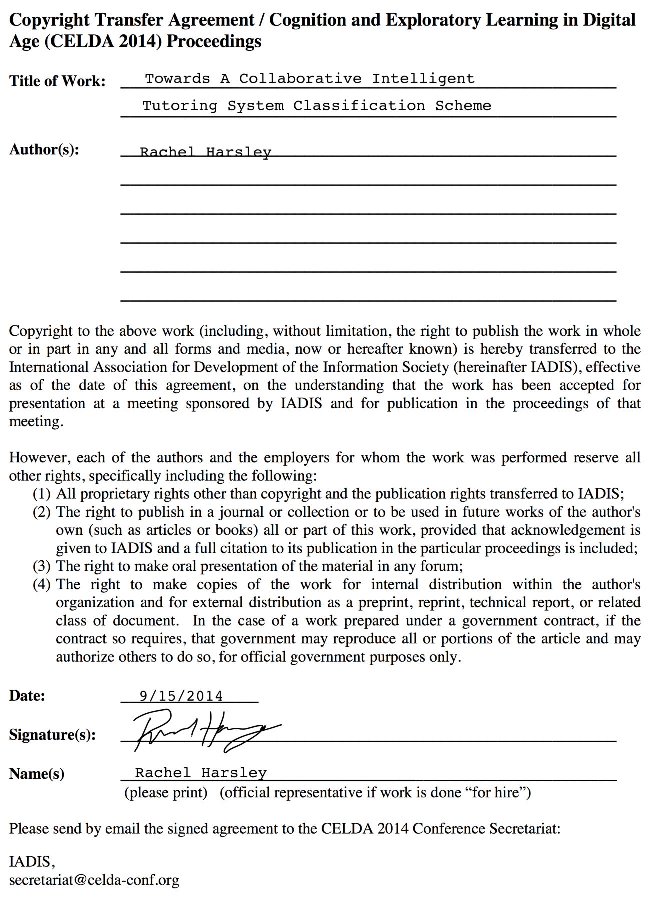
ALL EXPERIMENT LEARNING GAINS

Tutor	**N**	**P**re-test		**P**ost-test		**G**ain	
		μ	σ	μ	σ	μ	σ
Human	54	.40	.26	.54	.26	.14	.25
Unstructured Collab-ChiQat F'15	22	.46	.20	.58	.21	**.12**	.19
Semi-Structured Collab-ChiQat F'15	19	.57	.23	.67	.24	**.11**	.28
Unstructured Collab-ChiQat F'16	58	.64	.22	.77	.16	**.13**	.14
Hybrid Collab-ChiQat F'16	56	.56	.25	.66	.21	**.10**	.12

TABLE XXIII: COMPARISON OF LEARNING GAINS BETWEEN ALL EXPERIMENTS.

Appendix G

COPYRIGHTS



Copyright Transfer Agreement / Cognition and Exploratory Learning in Digital Age (CELDA 2014) Proceedings

Title of Work:	Towards A Collaborative Intelligent Tutoring System Classification Scheme
Author(s):	Rachel Harsley

Copyright to the above work (including, without limitation, the right to publish the work in whole or in part in any and all forms and media, now or hereafter known) is hereby transferred to the International Association for Development of the Information Society (hereinafter IADIS), effective as of the date of this agreement, on the understanding that the work has been accepted for presentation at a meeting sponsored by IADIS and for publication in the proceedings of that meeting.

However, each of the authors and the employers for whom the work was performed reserve all other rights, specifically including the following:

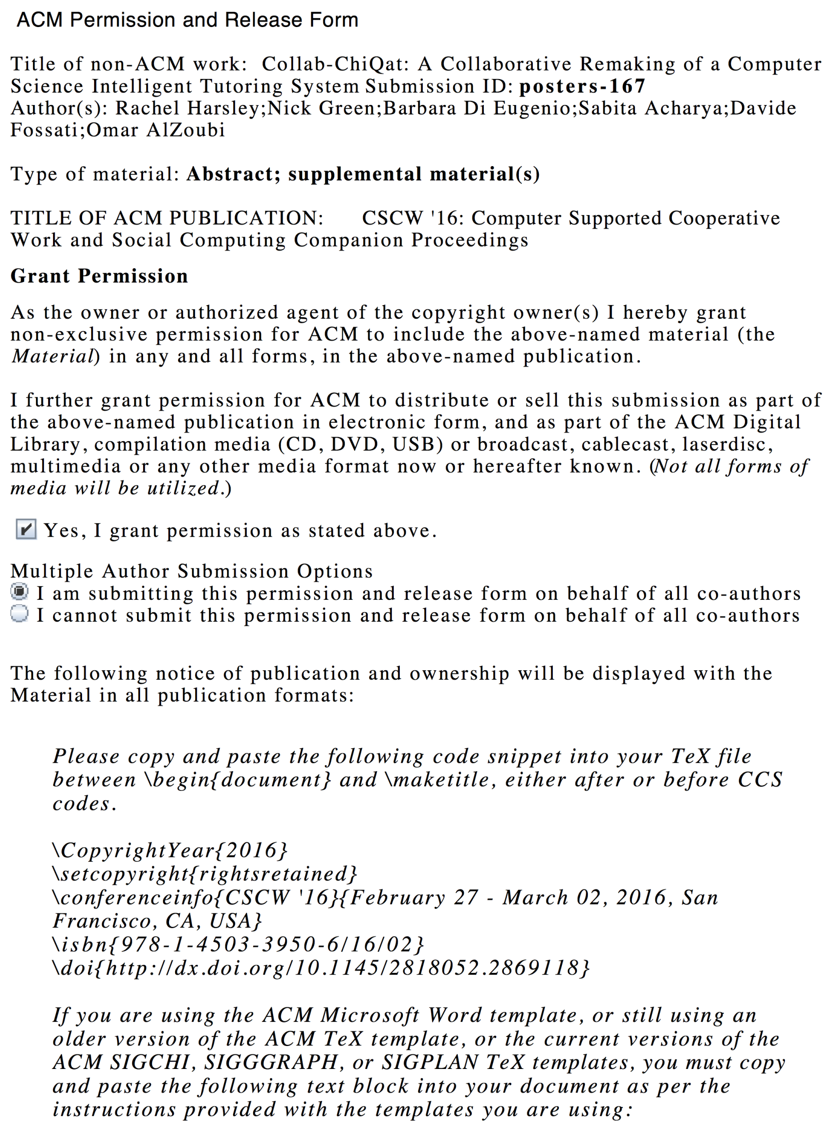
(1) All proprietary rights other than copyright and the publication rights transferred to IADIS;
(2) The right to publish in a journal or collection or to be used in future works of the author's own (such as articles or books) all or part of this work, provided that acknowledgement is given to IADIS and a full citation to its publication in the particular proceedings is included;
(3) The right to make oral presentation of the material in any forum;
(4) The right to make copies of the work for internal distribution within the author's organization and for external distribution as a preprint, reprint, technical report, or related class of document. In the case of a work prepared under a government contract, if the contract so requires, that government may reproduce all or portions of the article and may authorize others to do so, for official government purposes only.

Date:	9/15/2014
Signature(s):	[illegible]
Name(s)	Rachel Harsley
	(please print) (official representative if work is done "for hire")

Please send by email the signed agreement to the CELDA 2014 Conference Secretariat:

IADIS,
secretariat@celda-conf.org

Appendix G (Continued)



ACM Permission and Release Form

Title of non-ACM work: Collab-ChiQat: A Collaborative Remaking of a Computer Science Intelligent Tutoring System Submission ID: **posters-167**
Author(s): Rachel Harsley;Nick Green;Barbara Di Eugenio;Sabita Acharya;Davide Fossati;Omar AlZoubi

Type of material: **Abstract; supplemental material(s)**

TITLE OF ACM PUBLICATION: CSCW '16: Computer Supported Cooperative Work and Social Computing Companion Proceedings

Grant Permission

As the owner or authorized agent of the copyright owner(s) I hereby grant non-exclusive permission for ACM to include the above-named material (the *Material*) in any and all forms, in the above-named publication.

I further grant permission for ACM to distribute or sell this submission as part of the above-named publication in electronic form, and as part of the ACM Digital Library, compilation media (CD, DVD, USB) or broadcast, cablecast, laserdisc, multimedia or any other media format now or hereafter known. (*Not all forms of media will be utilized.*)

[✓] Yes, I grant permission as stated above.

Multiple Author Submission Options
● I am submitting this permission and release form on behalf of all co-authors
○ I cannot submit this permission and release form on behalf of all co-authors

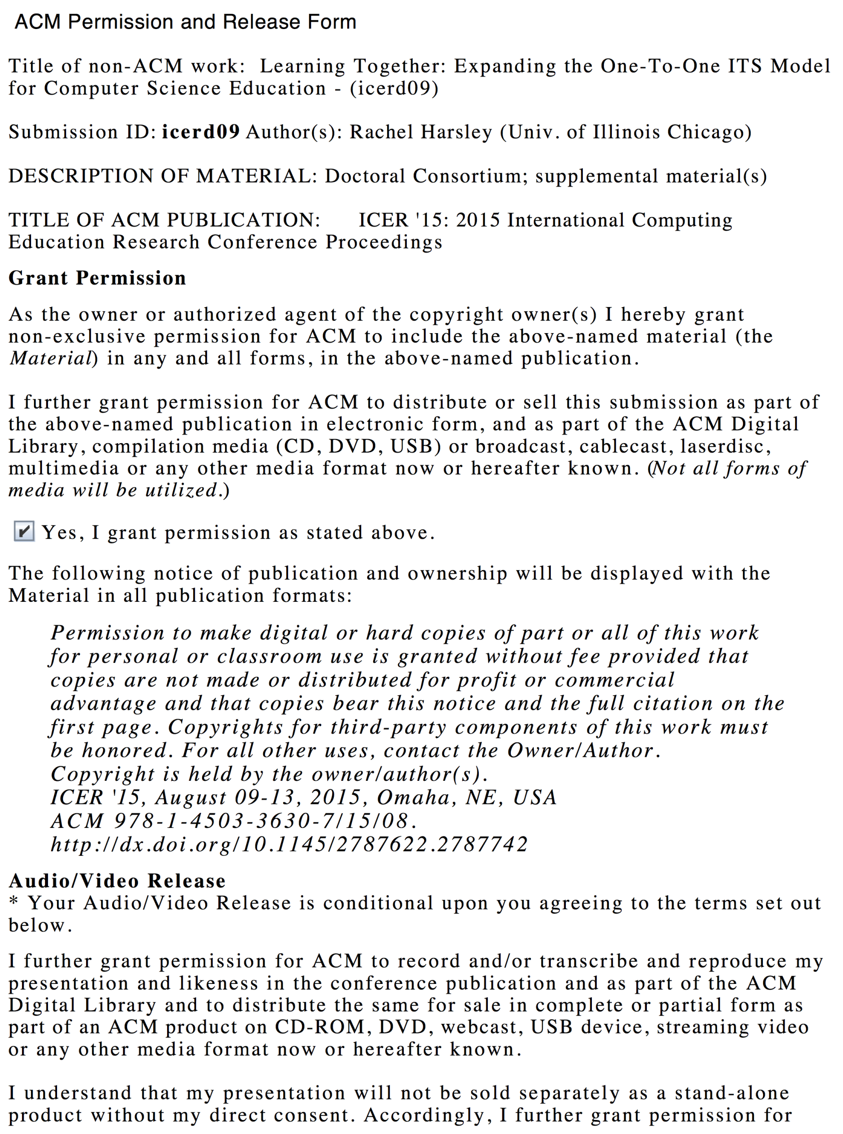
The following notice of publication and ownership will be displayed with the Material in all publication formats:

Please copy and paste the following code snippet into your TeX file between \begin{document} and \maketitle, either after or before CCS codes.

\CopyrightYear{2016}
\setcopyright{rightsretained}
\conferenceinfo{CSCW '16}{February 27 - March 02, 2016, San Francisco, CA, USA}
\isbn{978-1-4503-3950-6/16/02}
\doi{http://dx.doi.org/10.1145/2818052.2869118}

If you are using the ACM Microsoft Word template, or still using an older version of the ACM TeX template, or the current versions of the ACM SIGCHI, SIGGGRAPH, or SIGPLAN TeX templates, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

Appendix G (Continued)



ACM Permission and Release Form

Title of non-ACM work: Learning Together: Expanding the One-To-One ITS Model for Computer Science Education - (icerd09)

Submission ID: **icerd09** Author(s): Rachel Harsley (Univ. of Illinois Chicago)

DESCRIPTION OF MATERIAL: Doctoral Consortium; supplemental material(s)

TITLE OF ACM PUBLICATION: ICER '15: 2015 International Computing Education Research Conference Proceedings

Grant Permission

As the owner or authorized agent of the copyright owner(s) I hereby grant non-exclusive permission for ACM to include the above-named material (the *Material*) in any and all forms, in the above-named publication.

I further grant permission for ACM to distribute or sell this submission as part of the above-named publication in electronic form, and as part of the ACM Digital Library, compilation media (CD, DVD, USB) or broadcast, cablecast, laserdisc, multimedia or any other media format now or hereafter known. (*Not all forms of media will be utilized.*)

[✓] Yes, I grant permission as stated above.

The following notice of publication and ownership will be displayed with the Material in all publication formats:

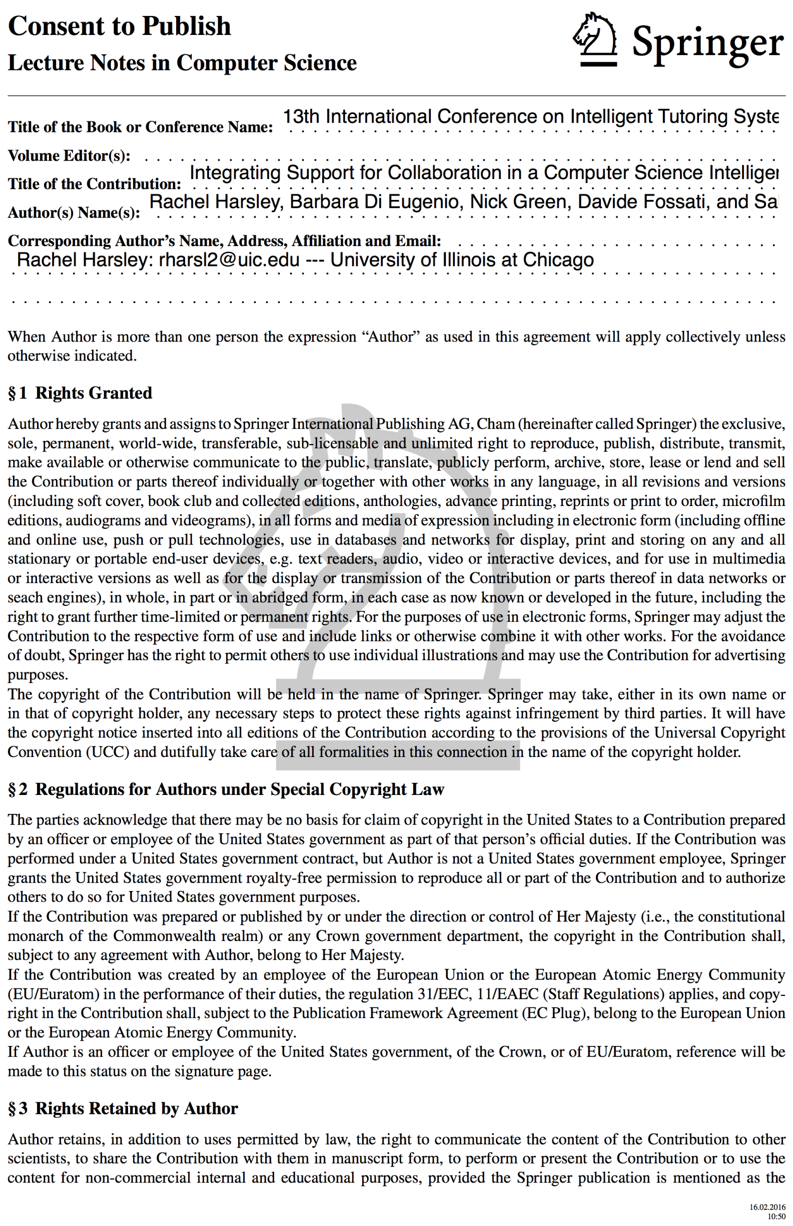
> *Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.*
> *Copyright is held by the owner/author(s).*
> *ICER '15, August 09-13, 2015, Omaha, NE, USA*
> *ACM 978-1-4503-3630-7/15/08.*
> *http://dx.doi.org/10.1145/2787622.2787742*

Audio/Video Release
* Your Audio/Video Release is conditional upon you agreeing to the terms set out below.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation and likeness in the conference publication and as part of the ACM Digital Library and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I further grant permission for

Appendix G (Continued)



Consent to Publish
Lecture Notes in Computer Science

Springer

Title of the Book or Conference Name:	13th International Conference on Intelligent Tutoring Syste
Volume Editor(s):	
Title of the Contribution:	Integrating Support for Collaboration in a Computer Science Intelligei
Author(s) Name(s):	Rachel Harsley, Barbara Di Eugenio, Nick Green, Davide Fossati, and Sa
Corresponding Author's Name, Address, Affiliation and Email:	Rachel Harsley: rharsl2@uic.edu --- University of Illinois at Chicago

When Author is more than one person the expression "Author" as used in this agreement will apply collectively unless otherwise indicated.

§ 1 Rights Granted

Author hereby grants and assigns to Springer International Publishing AG, Cham (hereinafter called Springer) the exclusive, sole, permanent, world-wide, transferable, sub-licensable and unlimited right to reproduce, publish, distribute, transmit, make available or otherwise communicate to the public, translate, publicly perform, archive, store, lease or lend and sell the Contribution or parts thereof individually or together with other works in any language, in all revisions and versions (including soft cover, book club and collected editions, anthologies, advance printing, reprints or print to order, microfilm editions, audiograms and videograms), in all forms and media of expression including in electronic form (including offline and online use, push or pull technologies, use in databases and networks for display, print and storing on any and all stationary or portable end-user devices, e.g. text readers, audio, video or interactive devices, and for use in multimedia or interactive versions as well as for the display or transmission of the Contribution or parts thereof in data networks or seach engines), in whole, in part or in abridged form, in each case as now known or developed in the future, including the right to grant further time-limited or permanent rights. For the purposes of use in electronic forms, Springer may adjust the Contribution to the respective form of use and include links or otherwise combine it with other works. For the avoidance of doubt, Springer has the right to permit others to use individual illustrations and may use the Contribution for advertising purposes.
The copyright of the Contribution will be held in the name of Springer. Springer may take, either in its own name or in that of copyright holder, any necessary steps to protect these rights against infringement by third parties. It will have the copyright notice inserted into all editions of the Contribution according to the provisions of the Universal Copyright Convention (UCC) and dutifully take care of all formalities in this connection in the name of the copyright holder.

§ 2 Regulations for Authors under Special Copyright Law

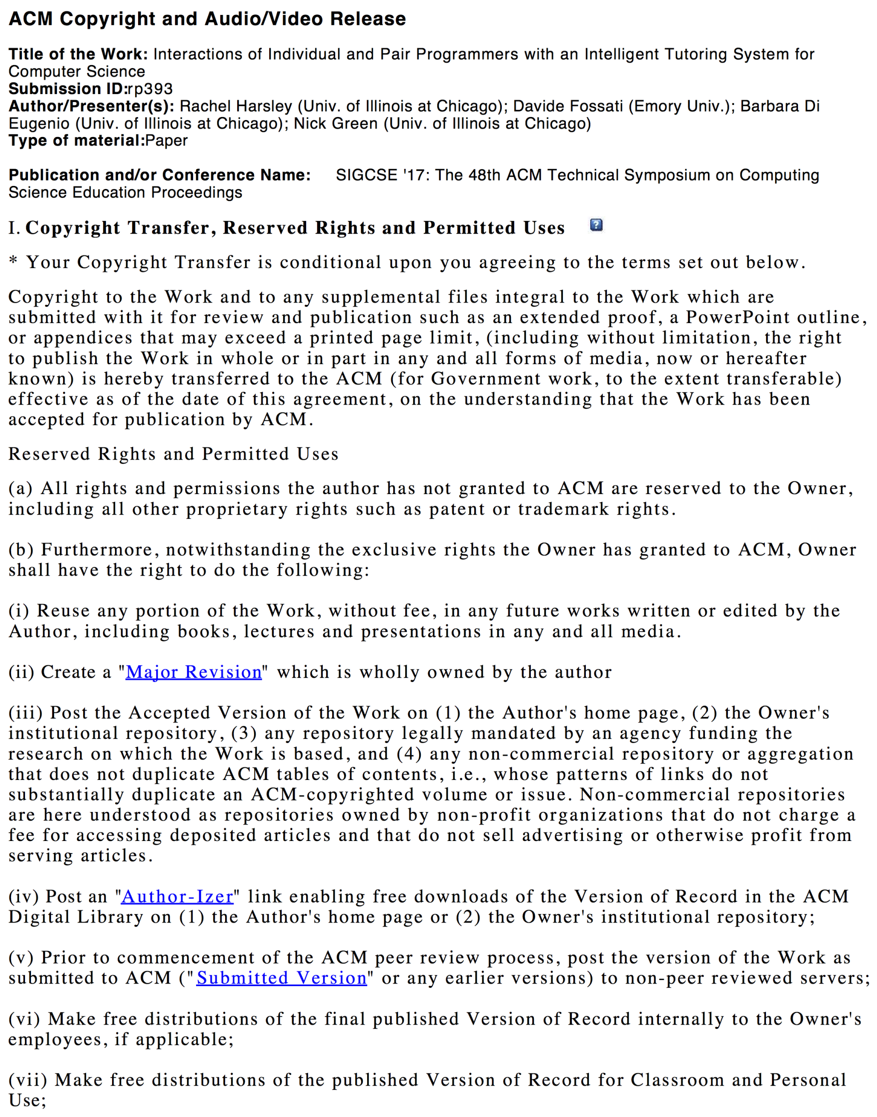
The parties acknowledge that there may be no basis for claim of copyright in the United States to a Contribution prepared by an officer or employee of the United States government as part of that person's official duties. If the Contribution was performed under a United States government contract, but Author is not a United States government employee, Springer grants the United States government royalty-free permission to reproduce all or part of the Contribution and to authorize others to do so for United States government purposes.
If the Contribution was prepared or published by or under the direction or control of Her Majesty (i.e., the constitutional monarch of the Commonwealth realm) or any Crown government department, the copyright in the Contribution shall, subject to any agreement with Author, belong to Her Majesty.
If the Contribution was created by an employee of the European Union or the European Atomic Energy Community (EU/Euratom) in the performance of their duties, the regulation 31/EEC, 11/EAEC (Staff Regulations) applies, and copyright in the Contribution shall, subject to the Publication Framework Agreement (EC Plug), belong to the European Union or the European Atomic Energy Community.
If Author is an officer or employee of the United States government, of the Crown, or of EU/Euratom, reference will be made to this status on the signature page.

§ 3 Rights Retained by Author

Author retains, in addition to uses permitted by law, the right to communicate the content of the Contribution to other scientists, to share the Contribution with them in manuscript form, to perform or present the Contribution or to use the content for non-commercial internal and educational purposes, provided the Springer publication is mentioned as the

16.02.2016
10:50

Appendix G (Continued)



ACM Copyright and Audio/Video Release

Title of the Work: Interactions of Individual and Pair Programmers with an Intelligent Tutoring System for Computer Science
Submission ID:rp393
Author/Presenter(s): Rachel Harsley (Univ. of Illinois at Chicago); Davide Fossati (Emory Univ.); Barbara Di Eugenio (Univ. of Illinois at Chicago); Nick Green (Univ. of Illinois at Chicago)
Type of material: Paper

Publication and/or Conference Name: SIGCSE '17: The 48th ACM Technical Symposium on Computing Science Education Proceedings

I. **Copyright Transfer, Reserved Rights and Permitted Uses**

* Your Copyright Transfer is conditional upon you agreeing to the terms set out below.

Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM.

Reserved Rights and Permitted Uses

(a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

CITED LITERATURE

ACM/IEEE-CS Joint Task Force on Computing Curricula: Computer Science Curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, December 2013.

AgileAcademyAus: Agile in Practice: Pair Programming, May 2011.

Ahuja, N. J. and Sille, R.: A Critical Review of Development of Intelligent Tutoring Systems: Retrospect, Present and Prospect. International Journal of Computer Science Issues (IJCSI), 10(4), 2013.

Akkerman, S., van den Bossche, P., Admiraal, W., Gijselaers, W., Segers, M., Simons, R. J., and Kirschner, P. A.: Reconsidering group cognition: From conceptual confusion to a boundary area between cognitive and socio-cultural perspectives?, 2007.

Ally, M., Darroch, F., and Toleman, M.: A Framework for Understanding the Factors Influencing Pair Programming Success. In Extreme Programming and Agile Processes in Software Engineering, eds. H. Baumeister, M. Marchesi, and M. Holcombe, number 3556 in Lecture Notes in Computer Science, pages 82–91. Springer Berlin Heidelberg, 2005.

AlZoubi, O., Fossati, D., Di Eugenio, B., Green, N., Alizadeh, M., and Harsley, R.: A Hybrid Model for Teaching Recursion. In Proceedings of the 16th annual ACM SIGITE conference on Information technology education, Chicago, IL, USA, October 2015.

Arroyo, I., Woolf, B. P., and Shanabrook, D.: Casual Collaborations while Learning Mathematics with an Intelligent Tutoring System. Collaborative Learning Environments Workshop at ITS 2012, 2012.

Baghaei, N., Mitrovic, A., and Irwin, W.: 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, September 2007.

Beaubouef, T. and Mason, J.: Why the high attrition rate for computer science students: some thoughts and observations. ACM SIGCSE Bulletin, 37(2):103–106, 2005.

Begel, A. and Nagappan, N.: Pair Programming: What's in It for Me? In Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, New York, NY, USA, 2008. ACM.

Bloom, B. S.: The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. Educational Researcher, 13(6):4–16, June 1984.

Braught, G., MacCormick, J., and Wahls, T.: The Benefits of Pairing by Ability. In Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE '10, pages 249–253, New York, NY, USA, 2010. ACM.

Braught, G., Wahls, T., and Eby, L. M.: The Case for Pair Programming in the Computer Science Classroom. Trans. Comput. Educ., 11(1):2:1–2:21, February 2011.

Carbonell, J.: AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction. IEEE Transactions on Man-Machine Systems, 11(4):190–202, December 1970.

Chigona, W. and Pollock, M.: Pair programming for information systems students new to programming: Students' experiences and teachers' challenges. In Portland International Conference on Management of Engineering Technology, 2008. PICMET 2008, pages 1587–1594, July 2008.

Chu-Carroll, J. and Brown, M. K.: Tracking Initiative in Collaborative Dialogue Interactions. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, ACL '98, pages 262–270, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

Commission, A. C. A.: Criteria for Accrediting Computing Programs, 2016-2017 | ABET. ABET, November 2014.

Conati, C.: Intelligent Tutoring Systems: New Challenges and Directions. In Proceedings of the 21st International Jont Conference on Artifical Intelligence, IJCAI'09, pages 2–7, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

Corbett, A. T., Koedinger, K. R., and Anderson, J. R.: Chapter 37 - Intelligent Tutoring Systems. In Handbook of Human-Computer Interaction (Second Edition), ed. M. G. H. K. L. V. Prabhu, pages 849–874. Amsterdam, North-Holland, 1997.

Dillenbourg, P.: What do you mean by collaborative learning. Collaborative-learning: Cognitive and computational approaches, 1:1–15, 1999.

Dillenbourg, P.: Designing Biases That Augment Socio-Cognitive Interactions. In Barriers and Biases in Computer-Mediated Knowledge Communication, eds. R. Bromme, F. W. Hesse, and H. Spada, number 5 in Computer-Supported Collaborative Learning Series, pages 243–264. Springer US, 2005. DOI: 10.1007/0-387-24319-4_11.

Dillenbourg, P., Jrvel, S., and Fischer, F.: The evolution of research on computer-supported collaborative learning. In Technology-enhanced learning, pages 3–19. Springer, 2009.

Dragon, T., Floryan, M., Woolf, B., and Murray, T.: Recognizing Dialogue Content in Student Collaborative Conversation. In Intelligent Tutoring Systems, eds. V. Aleven, J. Kay, and J. Mostow, number 6095 in Lecture Notes in Computer Science, pages 113–122. Springer Berlin Heidelberg, January 2010.

Ezen-Can, A. and Boyer, K. E.: Combining task and dialogue streams in unsupervised dialogue act models. In Proceedings of the 15th Annual SIGDIAL Meeting on Discourse and Dialogue, pages 113–122, 2014.

Fletcher, J. D. and Morrison, J. E.: DARPA Digital Tutor: Assessment Data (IDA Document D-4686). Alexandria, VA: Institute for Defense Analyses, 2012.

Floryan, M. and Woolf, B. P.: Authoring Expert Knowledge Bases for Intelligent Tutors through Crowdsourcing. In Artificial Intelligence in Education, eds. H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, number 7926 in Lecture Notes in Computer Science, pages 640–643. Springer Berlin Heidelberg, January 2013.

Fossati, D., Di Eugenio, B., Brown, C., Ohlsson, S., Cosejo, D., and Chen, L.: Supporting Computer Science Curriculum: Exploring and Learning Linked Lists with iList. IEEE Transactions on Learning Technologies, 2(2):107–120, April 2009.

Fossati, D.: ChiQat: An intelligent tutoring system for learning computer science. Qatar Foundation Annual Research Forum Proceedings, (2013):ICTP 020, November 2013.

Fossati, D., Di Eugenio, B., Brown, C., and Ohlsson, S.: Learning linked lists: Experiments with the iList system. In Intelligent tutoring systems, pages 80–89. Springer, 2008.

Fossati, D., Eugenio, B. D., Brown, C. W., Ohlsson, S., Cosejo, D. G., and Chen, L.: Supporting computer science curriculum: Exploring and learning linked lists with iList. Learning Technologies, IEEE Transactions on, 2(2):107–120, 2009.

Foundation, N. S.: NSF awards $25 million in new projects in support of the Computer Science for All Initiative, September 2016.

Friedberg, H., Litman, D., and Paletz, S. B.: Lexical entrainment and success in student engineering groups. In Spoken Language Technology Workshop (SLT), 2012 IEEE, pages 404–409. IEEE, 2012.

Gabelica, C., Van den Bossche, P., De Maeyer, S., Segers, M., and Gijselaers, W.: The effect of team feedback and guided reflexivity on team performance change. Learning and Instruction, 34:86–96, 2014.

Gonzalez, R.: Having Success With Code Bootcamps: A Guide For Employers And Bootcamp Grads, November 2015.

Google: Voice Driven Web Apps: Introduction to the Web Speech API |, 2013.

Green, N., AlZoubi, O., Alizadeh, M., Di Eugenio, B., Fossati, D., and Harsley, R.: A scalable intelligent tutoring system framework for computer science education. In CSEDU 2015 - 7th International Conference on Computer Supported Education, Proceedings, pages 372–379. SciTePress, 2015.

Gmez, O. S., Batn, J. L., and Aguilar, R. A.: Pair versus Solo Programming – An Experience Report from a Course on Design of Experiments in Software Engineering. SOURCEInternational Journal of Computer Science Issues (IJCSI), page p.734, January 2013. arXiv: 1306.4245.

Harsley, R.: Towards a Collaborative Intelligent Tutoring System Classification Scheme. In Proceedings Of The 11th International Conference On Cognition And Exploratory Learning In The Digital Age (Celda 2014), pages 290–291, Porto, Portugal, October 2014.

Harsley, R.: Learning Together: Expanding the One-To-One ITS Model for Computer Science Education. In Proceedings of the Eleventh Annual International Conference on International Computing Education Research, ICER '15, pages 263–264, New York, NY, USA, 2015. ACM.

Harsley, R., Di Eugenio, B., Fossati, D., and Green, N.: Collaborative Intelligent Tutoring Systems: Comparing Learner Outcomes Across Varying Collaboration Feedback Strategies. In Proceedings of the Computer Supported Collaborative Learning (CSCL) conference, volume 2, 2017.

Harsley, R., Di Eugenio, B., Green, N., Fossati, D., and Acharya, S.: Integrating Support for Collaboration in a Computer Science Intelligent Tutoring System. In Proceedings of the 13th International Conference on Intelligent Tutoring Systems, Croatia, 2016.

Harsley, R., Fossati, D., Di Eugenio, B., and Green, N.: Interactions of Individual and Pair Programmers with an Intelligent Tutoring System for Computer Science. In Proceedings of the 48th ACM Technical Symposium on Computing Science Education, Seattle, WA, USA, 2017. ACM.

Harsley, R., Green, N., Alizadeh, M., Acharya, S., Fossati, D., Di Eugenio, B., and AlZoubi, O.: Incorporating Analogies and Worked Out Examples as Pedagogical Strategies in a Computer Science Tutoring System. In Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE), Memphis, TN, USA, 2016.

Harsley, R., Green, N., Alizadeh, M., Acharya, S., Fossati, D., Di Eugenio, B., and AlZoubi, O.: Incorporating Analogies and Worked Out Examples as Pedagogical Strategies in a Computer Science Tutoring System. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pages 675–680, Memphis, TN, USA, 2016. ACM.

Harsley, R., Green, N., Di Eugenio, B., Acharya, S., Fossati, D., and AlZoubi, O.: Collab-ChiQat: A Collaborative Remaking of a Computer Science Intelligent Tutoring System. In CSCW '16: Computer Supported Cooperative Work and Social Computing Companion Proceedings, San Francisco, CA, USA, March 2016.

Howles, T.: A study of attrition and the use of student learning communities in the computer science introductory programming sequence. Computer science education, 19(1):1–13, 2009.

Inaba, A. and Mizoguchi, R.: Learners Roles and Predictable Educational Benefits in Collaborative Learning. In Intelligent Tutoring Systems, eds. J. C. Lester, R. M. Vicari, and F. Paraguau, number 3220 in Lecture Notes in Computer Science, pages 285–294. Springer Berlin Heidelberg, August 2004. DOI: 10.1007/978-3-540-30139-4_27.

Israel, J. and Aiken, R. M.: Supporting Collaborative Learning With An Intelligent Web-Based System. IJ Artificial Intelligence in Education, 17(1):3–40, 2007.

Janssen, J., Erkens, G., Kanselaar, G., and Jaspers, J.: Visualization of Participation: Does It Contribute to Successful Computer-supported Collaborative Learning? Comput. Educ., 49(4):1037–1065, December 2007.

Janssen, J., Erkens, G., and Kirschner, P. A.: Group awareness tools: Its what you do with it that matters. Computers in human behavior, 27(3):1046–1058, 2011.

Jensen, R. W.: A pair programming experience. CrossTalk, 16(3):22–24, 2003.

Jordan, P. W. and Di Eugenio, B.: Control and initiative in collaborative problem solving dialogues. In Working Notes of the AAAI Spring Symposium on Computational Models for Mixed Initiative, pages 81–84, 1997.

Jrvel, S., Kirschner, P. A., Panadero, E., Malmberg, J., Phielix, C., Jaspers, J., Koivuniemi, M., and Jrvenoja, H.: Enhancing socially shared regulation in collaborative learning groups: designing for CSCL regulation tools. Educational Technology Research and Development, 63(1):125–142, February 2015.

Kaptelinin, V.: Learning together: Educational benefits and prospects for computer support. Journal of the Learning Sciences, 8(3-4):499–508, 1999.

Kersey, C., Di Eugenio, B., Jordan, P., and Katz, S.: KSC-PaL: A Peer Learning Agent That Encourages Students to Take the Initiative. In Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications, EdAppsNLP '09, pages 55–63, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Kersey-Howard, C., Jordan, P., Di Eugenio, B., and Katz, S.: Shifting the Load: a Peer Dialogue Agent that Encourages its Human Collaborator to Contribute More to Problem Solving. International Journal of Artificial Intelligence in Education, pages 1–29, 2015.

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hmlinen, R., Hkkinen, P., and Fischer, F.: Specifying computer-supported collaboration scripts. International Journal of Computer-Supported Collaborative Learning, 2(2-3):211–224, September 2007.

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

Kulik, J. A. and Fletcher, J. D.: Effectiveness of Intelligent Tutoring Systems A Meta-Analytic Review. Review of Educational Research, page 0034654315581420, April 2015.

Kumar, R. and al, e.: Tutorial Dialogue as Adaptive Collaborative Learning Support. 2007.

Lane, H. C. and VanLehn, K.: A Dialogue-Based Tutoring System for Beginning Programming. In FLAIRS Conference, pages 449–454, 2004.

Lehtinen, E., Hakkarainen, K., Lipponen, L., Rahikainen, M., and Muukkonen, H.: Computer supported collaborative learning: A review. The JHGI Giesbers reports on education, 10, 1999.

Lewis, C.: Attrition in Introductory Computer Science at the University of California, Berkeley. Berkeley: University of California, Berkeley, 2010.

Little, J. L. and Bjork, E. L.: Pretesting with multiple-choice questions facilitates learning. In Proceedings of the annual meeting of the cognitive science society, 2012.

Magnisalis, I., Demetriadis, S., and Karakostas, A.: Adaptive and Intelligent Systems for Collaborative Learning Support: A Review of the Field. IEEE Transactions on Learning Technologies, 4(1):5–20, 2011.

Maguire, P., Maguire, R., Hyland, P., and Marshall, P.: Enhancing collaborative learning using pair programming: Who benefits? AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education, 6(2), June 2014.

McDowell, C., Werner, L., Bullock, H. E., and Fernald, J.: Pair Programming Improves Student Retention, Confidence, and Program Quality. Commun. ACM, 49(8):90–95, August 2006.

Monge, A. E., Fadjo, C. L., Quinn, B. A., and Barker, L. J.: EngageCSEdu: Engaging and Retaining CS1 and CS2 Students. ACM Inroads, 6(1):6–11, February 2015.

Najar, A. S. and Mitrovic, A.: Do novices and advanced students benefit differently from worked examples and ITS. In Proceedings of International Conference Computers in Education, pages 20–29, 2013.

Nenkova, A., Gravano, A., and Hirschberg, J.: High Frequency Word Entrainment in Spoken Dialogue. In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, HLT-Short '08, pages 169–172, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Nkambou, R., Bourdeau, J., and Mizoguchi, R.: Advances in intelligent tutoring systems, volume 308. Springer, 2010.

Nwana, H. S.: Intelligent tutoring systems: an overview. Artificial Intelligence Review, 4(4):251–277, December 1990.

Olsen, J. K., Aleven, V., and Rummel, N.: Adapting Collaboration Dialogue in Response to Intelligent Tutoring System Feedback. In Artificial Intelligence in Education, eds. C. Conati, N. Heffernan, A. Mitrovic, and M. F. Verdejo, Lecture Notes in Computer Science. Springer International Publishing, June 2015.

Olsen, J. K., Belenky, D. M., Aleven, V., and Rummel, N.: Using an Intelligent Tutoring System to Support Collaborative as well as Individual Learning. In Intelligent Tutoring Systems, eds. S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, number 8474 in Lecture Notes in Computer Science, pages 134–143. Springer International Publishing, June 2014.

Paviotti, G., Rossi, P. G., and Zarka, D.: Intelligent Tutoring Systems: an Overview. Pensa Multimedia, 2012.

Pears, A.: Enhancing student engagement in an introductory programming course. In 2010 IEEE Frontiers in Education Conference (FIE), pages F1E–1–F1E–2, October 2010.

Phielix, C., Prins, F. J., and Kirschner, P. A.: Awareness of Group Performance in a CSCL-environment: Effects of Peer Feedback and Reflection. Comput. Hum. Behav., 26(2):151–161, March 2010.

Phielix, C., Prins, F. J., Kirschner, P. A., Erkens, G., and Jaspers, J.: Group Awareness of Social and Cognitive Performance in a CSCL Environment: Effects of a Peer Feedback and Reflection Tool. Comput. Hum. Behav., 27(3):1087–1102, May 2011.

Porter, L., Bailey Lee, C., Simon, B., and Zingaro, D.: Peer Instruction: Do Students Really Learn from Peer Discussion in Computing? In Proceedings of the Seventh

International Workshop on Computing Education Research, ICER '11, pages 45–52, New York, NY, USA, 2011. ACM.

Porter, L., Bouvier, D., Cutts, Q., Grissom, S., Lee, C., McCartney, R., Zingaro, D., and Simon, B.: A Multi-institutional Study of Peer Instruction in Introductory Computing. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education, SIGCSE '16, pages 358–363, New York, NY, USA, 2016. ACM.

Porter, L., Guzdial, M., McDowell, C., and Simon, B.: Success in Introductory Programming: What Works? Commun. ACM, 56(8):34–36, August 2013.

Porter, L. and Simon, B.: Retaining Nearly One-third More Majors with a Trio of Instructional Best Practices in CS1. In Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13, pages 165–170, New York, NY, USA, 2013. ACM.

Reitter, D., Keller, F., and Moore, J. D.: Computational Modelling of Structural Priming in Dialogue. In Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, NAACL-Short '06, pages 121–124, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Richland, L. E., Kornell, N., and Kao, L. S.: The pretesting effect: Do unsuccessful retrieval attempts enhance learning? Journal of Experimental Psychology: Applied, 15(3):243–257, 2009.

Richtel, M.: Reading, Writing, Arithmetic, and Lately, Coding. The New York Times, May 2014.

Ritter, S., Kulikowich, J., Lei, P.-W., McGuire, C. L., and Morgan, P.: What evidence matters? A randomized field trial of Cognitive Tutor Algebra I. Frontiers in Artificial Intelligence and Applications, 162:13, 2007.

Rodriguez, F. J. and Boyer, K. E.: Discovering Individual and Collaborative Problem-Solving Modes with Hidden Markov Models. In Proceedings of the 17th International Conference on Artificial Intelligence in Education (AIED). 2015.

Rodrguez, F. J., Price, K. M., and Boyer, K. E.: Exploring the Pair Programming Process: Characteristics of Effective Collaboration. 2017.

Salleh, N., Mendes, E., and Grundy, J.: Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. IEEE Transactions on Software Engineering, 37(4):509–525, July 2011.

Salleh, N., Mendes, E., and Grundy, J.: Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. IEEE Transactions on Software Engineering, 37(4):509–525, July 2011.

Self, J. A.: Student models in computer-aided instruction. International Journal of Man-Machine Studies, 6(2):261–276, March 1974.

Soller, A.: Supporting social interaction in an intelligent collaborative learning system. International Journal of Artificial Intelligence in Education (IJAIED), 12:40–62, 2001.

Stahl, G.: Group cognition in computer-assisted collaborative learning. Journal of Computer Assisted Learning, 21(2):79–90, April 2005.

Stahl, G.: Group Cognition: Computer Support for Building Collaborative Knowledge (Acting with Technology). Cambridge and London, The MIT Press, April 2006.

Strijbos, J.-W.: Assessment of (Computer-Supported) Collaborative Learning. IEEE Transactions on Learning Technologies, 4(1):59–73, January 2011.

Suebnukarn, S. and Haddawy, P.: COMET: A Collaborative Tutoring System for Medical Problem-Based Learning. IEEE Intelligent Systems, 22(4):70–77, 2007.

Tchounikine, P., Rummel, N., and McLaren, B. M.: Computer Supported Collaborative Learning and Intelligent Tutoring Systems. In Advances in Intelligent Tutoring Systems, pages 447–463. Springer, 2010.

Therneau, T., Atkinson, B., and port), B. R. a. o. i. R.: rpart: Recursive Partitioning and Regression Trees, June 2015.

Van Toll, T., Lee, R., and Ahlswede, T.: Evaluating the Usefulness of Pair Programming in a Classroom Setting. In 6th IEEE/ACIS International Conference on Computer and Information Science, 2007. ICIS 2007, pages 302–308, July 2007.

Virvou, M. and Sidiropoulos, S.-C.: An Intelligent Tutoring System over a social network for mathematics learning. In 2013 Fourth International Conference on Information, Intelligence, Systems and Applications (IISA), pages 1–4, 2013.

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

Walker, E., Rummel, N., and Koedinger, K. R.: Integrating Collaboration And Intelligent Tutoring Data In The Evaluation Of A Reciprocal Peer Tutoring Environment. Research and Practice in Technology Enhanced Learning, 04(03):221–251, November 2009.

Walker, M. and Whittaker, S.: Mixed Initiative in Dialogue: An Investigation into Discourse Segmentation. In Proceedings of the 28th Annual Meeting on Association for Computational Linguistics, ACL '90, pages 70–78, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics.

Walle, T. and Hannay, J.: Personality and the nature of collaboration in pair programming. In 3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009, pages 203–213, October 2009.

Washington, A. N., Burge, L., Mejias, M., Jean-Pierre, K., and Knox, Q.: Improving Undergraduate Student Performance in Computer Science at Historically Black Colleges and Universities (HBCUs) Through Industry Partnerships. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE '15, pages 203–206, New York, NY, USA, 2015. ACM.

Wiggins, J. B., Boyer, K. E., Baikadi, A., Ezen-Can, A., Grafsgaard, J. F., Ha, E. Y., Lester, J. C., Mitchell, C. M., and Wiebe, E. N.: JavaTutor: An Intelligent Tutoring System That Adapts to Cognitive and Affective States During Computer Programming. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE '15, pages 599–599, New York, NY, USA, 2015. ACM.

Williams, L., Kessler, R. R., Cunningham, W., and Jeffries, R.: Strengthening the case for pair programming. IEEE Software, 17(4):19–25, August 2000.

Williams, L. and Upchurch, R. L.: In Support of Student Pair-programming. In Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education, SIGCSE '01, pages 327–331, New York, NY, USA, 2001. ACM.

Williams, L. A. and Kessler, R. R.: All I Really Need to Know About Pair Programming I Learned in Kindergarten. Commun. ACM, 43(5):108–114, May 2000.

Woolf, B. P.: Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning. Morgan Kaufmann, 2010.

Xu, Y. and Reitter, D.: Convergence of Syntactic Complexity in Conversation. In The 54th Annual Meeting of the Association for Computational Linguistics, page 443, 2016.

VITA

Rachel E. Harsley
rharsl2@uic.edu
850 South Morgan Avenue — Mobile: (314) 406-1577
Chicago, IL 60607 — Homepage: http://www.cs.uic.edu/~rharsley/

EDUCATION

University of Illinois at Chicago, Chicago, IL
Ph.D. Student in Computer Science
Expected Graduation May 2017;
Advisor: Dr. Barbara Di Eugenio

Vanderbilt University, Nashville, TN
B.S. in Computer Science, May 2011

Georgia Institute of Technology, Metz, FR
Study Abroad Participant, Spring 2011

RESEARCH INTERESTS

- Natural Language Processing
- Machine Learning
- Computer Science Education
- Collaborative Learning Environments

RESEARCH SUMMARY

My research applies natural language processing and machine learning techniques to the emergent field of Collaborative Intelligent Tutoring Systems (CITS). The work shifts the standard paradigm of an intelligent tutoring system (ITS) from a one-to-one learning model to multiple-student, computer-mediated learning. The research situates itself in the domain of Computer Science education and addresses widely applicable issues of defining, tracking, and adaptively promoting effective collaboration and learning. ChiQat-Tutor is a joint effort and product of this work and other related work. It is a desktop-based Intelligent Tutoring System for learning core Computer Science concepts including linked lists, binary search trees, and recursion.

REFEREED CONFERENCE PAPERS & PROCEEDINGS

1. **Harsley, R.**, Fossati, D., Di Eugenio, B., & Green, N. (2017). Interactions of Individual and Pair Programmers with an Intelligent Tutoring System for Computer Science. In *Proceedings of the 48th ACM Technical Symposium on Computing Science Education*. Seattle, WA, USA: ACM. *"In Press"*.
2. **Harsley, R.,** Gupta, B., Di Eugenio, B., & Li, H. (2016). Hit Songs' Sentiments Harness Public Mood & Predict Stock Market. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 17–25). San Diego, California: Association for Computational Linguistics
3. **Harsley, R.**, Di Eugenio, B., Green, N., Fossati, D., & Acharya, S. (2016). Integrating Support for Collaboration in a Computer Science Intelligent Tutoring System. In *Proceedings of the 13th International Conference on Intelligent Tutoring Systems*. Croatia.
4. Green, N., Di Eugenio, B., **Harsley, R.**, Fossati, D., & AlZoubi, O. (2016). Behavior and Learning of Students using Worked-out Examples in a Tutoring System. In *Proceedings of the 13th International Conference on Intelligent Tutoring Systems*. Croatia.
5. **Harsley, R.,** Green, N., Alizadeh, M., Acharya, S., Fossati, D., Di Eugenio, B., & AlZoubi, O. (2016). Incorporating Analogies and Worked Out Examples as Pedagogical Strategies in a Computer Science Tutoring System. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 675–680). Memphis, TN, USA: ACM.

6. **Harsley, R.,** Green, N., Di Eugenio, B., Aditya, S., Fossati, D., & Al Zoubi, O. (2016). Collab-ChiQat: A Collaborative Remaking of a Computer Science Intelligent Tutoring System. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion* (pp. 281–284). San Francisco, CA, USA: ACM.
7. AlZoubi, O., Fossati, D., Di Eugenio, B., Green, N., Alizadeh, M., & **Harsley, R**. (2015). A Hybrid Model for Teaching Recursion. In *Proceedings of the 16th annual ACM SIGITE conference on Information technology education*. Chicago, IL, USA.
8. Di Eugenio, B., Green, N., Alizadeh, M., **Harsley, R.,** & Fossati, D. (2015). Worked-out Examples in a Computer Science Intelligent Tutoring System. In *Proceedings of the 16th annual ACM SIGITE conference on Information technology education*. Chicago, IL, USA.
9. Green, N., Fossati, D., Di Eugenio, B., **Harsley, R**., AlZoubi, O., & Alizadeh, M. (2015). Student Behavior with Worked-out Examples in a Computer Science Intelligent Tutoring System. Presented at the 3rd International Conference on Educational Technologies, Florianópolis, Santa Catarina, Brazil.
10. **Harsley, R.** (2015). Learning Together: Expanding the One-To-One ITS Model for Computer Science Education. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 263–264). New York, NY, USA: ACM.
11. **Harsley, R**., Green, N., Alizadeh, M., Tandon, A., & Prabhu, A. (2015). Sick Kitty – Toward Promoting Deductive Reasoning through an Embodied Medical Diagnosis Game. In *Proceedings of the Games+Learning+Society (GLS) Conference*. Madison, WA: ETC Press.
12. Alizadeh, M., Di Eugenio, B., **Harsley, R**., Green, N., Fossati, D., & AlZoubi, O. (2015). A study of analogy in computer science tutorial dialogues. In *CSEDU 2015 - 7th International Conference on Computer Supported Education, Proceedings* (pp. 232–237). SciTePress.
13. Green, N., AlZoubi, O., Alizadeh, M., Di Eugenio, B., Fossati, D., & **Harsley, R.** (2015). A scalable intelligent tutoring system framework for computer science education. In *CSEDU 2015 - 7th International Conference on Computer Supported Education, Proceedings* (pp. 372–379). SciTePress.
14. **Harsley, R**. (2014). Towards a Collaborative Intelligent Tutoring System Classification Scheme,. In *Proceedings Of The 11th International Conference On Cognition And Exploratory Learning In The Digital Age (Celda 2014)* (pp. 290–291). Porto, Portugal.

<u>POSTERS & INVITED TALKS</u>

- Harsley, R. et al. (2015). Navigating Academia and the Technical Internship Process. Panel presentation at Google Chicago. Chicago, IL.
- Harsley, R. (2013). Leveraging Mobile Technology for Computer Science Learning in Low Socioeconomic and Minority Communities. Poster presentation at the CRA-W Graduate Cohort Workshop. Boston, Massachusetts.
- Harsley, R. et al. (2013). Voices from the Field. Panel presentation at the Illinois GEM Grad Lab. Chicago, IL.

<u>MEDIA & PRESS</u>

• **2016**	WinTrust Business Lunch Guest	*WGN Radio*
• **2016**	"10 Unique Black Women-Owned Tech Startups"	*Black Enterprise Magazine*
• **2016**	"A UIC PhD Student Made a Messaging App That Doesn't Leave a Digital Trail"	*ChicagoInno*
• **2014**	"Women in STEM"	*National Society of Black Engineers News*
• **2014**	University Promotional Video	*UIC Office of the Chancellor*
• **2014**	"Computer Science Students Among 50 for the Future"	*UIC News*
• **2014**	Featured Student	*UIC Engineering News Magazine*

SKILLS

Creative thinker and problem solver with ability to plan and implement;
Strong leadership and communication ability with valuable research and industry experience
C++, Java, Objective-C, C, C#, JavaScript, HTML, CSS, SQL, Python, Lisp;

TEACHING EXPERIENCE

University of Illinois at Chicago, Chicago, IL 2011-2012, 2014
Teaching Assistant, Department of Computer Science (4.61/5 Student Evaluation Rating Spring '14)

Northwestern University, Chicago IL November, 2013
Java Course Instructor, Center for Talent Development

University of Illinois at Chicago, Chicago, IL 2012-2013
Tutor & Student Coordinator of the African American Academic Network Learning Resource Center

HONORS & AWARDS

NSBE Graduate Student of the Year	2017
ACM-W Scholarship	2017
Chicago RECESS Pitch 1st Place	2016
UIC Startup Challenge 1st Place	2016
Chicago Urban League Leading Woman in Tech Award	2015
UIC Abraham Lincoln Fellowship	2015
Xerox Technical Minority Scholarship	2013, 2015
NSBE Region IV - Living the NSBE Mission Scholarship	2013
Tapia Scholarship Travel Grant Award	2013
Illinois Technology Foundation Fifty for the Future Award	2013
Microsoft NSBE Scholarship	2012
CRA-W Grad Cohort Travel Scholarship	2012
National GEM Consortium GEM Fellowship	2012-2013
UIC C.S. Department Grace Hopper Travel Scholarship	2011
Vanderbilt University Chancellor's Scholar Award	2007-2011

SERVICE & LEADERSHIP EXPERIENCE

STEM Education Advocate & Motivational Speaker	2015 - present
Christian Education Teacher; Grace New Covenant Church, St. Louis, MO	2011, 2015
Media Ministry Volunteer; Apostolic Faith Church, Chicago, IL	2013
Student Government Secretary; **Georgia Tech-Lorraine,** Metz, France	2011 (spring)
Residential Advisor; **Vanderbilt University,** Nashville, TN	2008-2011
Executive Board Member; **Vanderbilt University Chapter of NSBE,** Nashville, TN	2007-2011

Reviewer:

- Games + Learning + Society Conference (2015)
- International Conference on Intelligent Tutoring Systems (2014)
- International Conference on Artificial Intelligence in Education (2013, 2015);

PATENTS

- Harsley, R. (2015, May 8). System and Method for Stateless Communication Device Control. U.S. Provisional Patent Appl. 62/159,091

INDUSTRY EXPERIENCE

Maychild Technologies LLC, St. Louis, MO 2012-present
Founder and CEO
Developed Clean Slate Messenger, a messaging platform for private, instant chat. In three months since the soft-launch on iOS, the app gained grew to users in ten countries organically via word of mouth.
http://maychildtechnologies.com/

Google Incorporated 2014, 2015 (summer)
Software Engineering Intern, Machine Intelligence
Built an applied machine learning approach to predict Google Ad Exchange platform search ranking results.
https://www.doubleclickbygoogle.com/solutions/digital-marketing/ad-exchange/

Improved predictive neural network's visualizer by altering the underlying architecture to utilize Google Brain's deep learning technology. Resulted in enhanced traceability and debugability of approach.
http://research.google.com/pubs/ArtificialIntelligenceandMachineLearning.html

General Electric Global Research, San Ramon, CA 2013(summer)
R&D Intern, Mobility and Collaboration Lab
Wrote significant portions of Objective-C code included in official mobile application release.
http://www.geglobalresearch.com/innovation/predix-software-platform-industrial-internet

Intel Corporation, Bellevue, WA 2012(summer)
Software Intern Graduate Level
Developed Windows gadget application to alert software developers in real-time of status and updates concerning assigned bugs in Intel Clover Trail atom chip release.
http://ark.intel.com/products/codename/53606/Cloverview

AT&T Incorporated, St. Louis, MO 2011(summer)
Student Intern – Technical I
Created graphical interface for Java-based Application Resource Optimizer and demoed Agile accomplishments in bi-weekly retrospective for clients resulting in a $1.2 million budget increase for further enhancements.
https://developer.att.com/application-resource-optimizer

Grace New Covenant Church, St. Louis, MO 2010(summer)
Administrative Intern
Served as general technology advisor. Responsible for the creation of the church website which has seen over 6,000+ unique visitors and showcases a high quality community presence and brand.
http://www.gracenewcovenantchurch.org/

Save-A-lot Corporate Headquarters, St. Louis, MO 2008, 2009(summer)
IT Department Intern
Responsible for testing and debugging of Java applications for a wide variety of usages.