When Two Heads are Better Than One: A Critical Review of Four Collaborative Intelligent Tutoring Systems

Rachel Harsley
Department of Computer Science, University of Illinois at Chicago, Chicago, IL USA
rharsl2@uic.edu

Abstract. This paper critically reviews four recently published approaches to the emergent scientific literature on Collaborative Intelligent Tutoring Systems (CITS). The focus is threefold: 1) provide an overview of the fields of Intelligent Tutoring Systems and Computer Supported Collaborative Learning 2) review the four works and evaluate them in an organized way emphasizing main features, results, and contributions, 3) highlight gaps for possible research. Systematic analysis of the four approaches and other relevant literature led me to propose a CITS classification scheme with five dimensions: modeling, collaboration cues, group dynamics, pedagogical guidance, and technology. The three emergent classifications of CITS are unstructured, semi-structured, and fully structured. This scheme guides the evaluation of the writings. The reviewed literature suggests that all three types of CITS offer opportunities to improve student learning gains. However, the full extent to which these gains may be achieved is subject to future research.

Keywords: intelligent tutoring systems, collaborative environments, computer supported learning

1. Introduction

The purpose of this paper is to critically review the approach of four intelligent tutoring systems developed to support collaborative learning. Both Intelligent Tutoring Systems (ITS) and Computer Supported Collaborative Learning (CSCL) have a long history within learning technologies [21, 26]. However, the combination of both approaches as a single intervention is a relatively new area of research. The affordances of both industries, specifically higher learning gains through structured collaboration and adaptive support, motivates the merge. In this paper, I use the acronym "CITS" as a general term to denote intelligent tutoring systems that support collaboration.

Because education and instruction design are also pivotal to CITS implementation, as a whole CITS research spans the three primary research areas shown in Figure 1. There are important implications resulting from this synergy including differences in theoretical frameworks and research goals. Therefore, a uniform classification scheme for analysis of the research designs is essential.

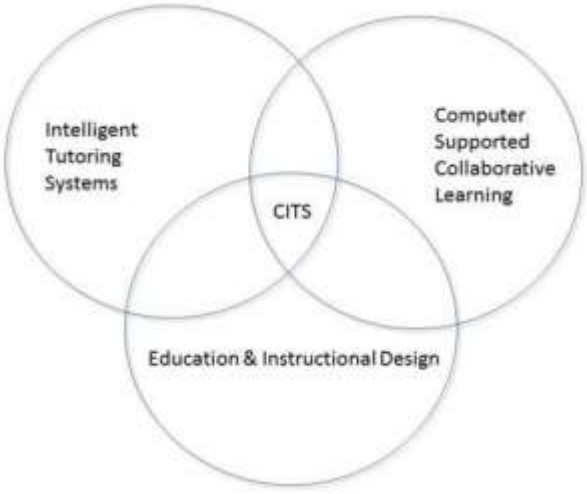

Figure 1 CITS domains.

After providing some background information on the research areas of ITS and CSCL, I will review the four studies under consideration. I will then present the classification scheme used for evaluation of the selected works. The classification scheme:

1. provides a framework for evaluating CITS and emphasizing main features, results, and contributions
2. highlights gaps for future research

Three of the reviewed works align with the classification scheme. However, a final study, a CITS authoring tool, accommodates implementations of all classifications. I conclude with a discussion of open research questions for the emergent field.

2. Intelligent Tutoring Systems Overview

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 [20]. The term "ITS" is synonymous with the phrase "Intelligent Computer-Aided Instruction" (ICAI). Traditional ITS aim to provide user-adapted support during problem-solving processes in a manner resemblant of a human tutor [27]. The intelligent tutor compares student actions to a domain model and uses the subsequent contextual awareness to tailor instructional activities and offer relevant help [28]. Although ITS have not yet mastered the same level of effectiveness achieved by expert human tutors, multiple research studies show the impact of these systems to significantly increase learning [14, 9].

Classical definitions such as the one provided by Conati 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. Research in this field has successfully delivered techniques and systems that provide adaptive support for student problem solving in a variety of domains. There are, however, other educational activities that can benefit from *individualized* computer-based support, such as studying examples, exploring interactive simulations and playing educational games [3].

The focus of ITS on an individual learner (depicted in Figure 2) is an important distinction between ITS and CSCL.

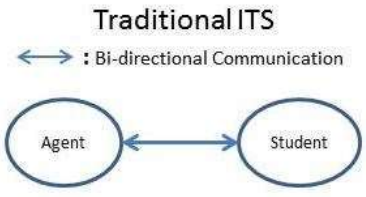

Figure 2 Agent tutors student one-on-one.

Structure of ITS

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" [24]. This led to the acceptance of a three-model architecture for ITS. The current standard for ITS construction now includes a fourth component, "user interface" and is depicted in Figure 3 [19, 20]. Methods of ITS implementation vary greatly while they remain consistent to this baseline architectural framework.

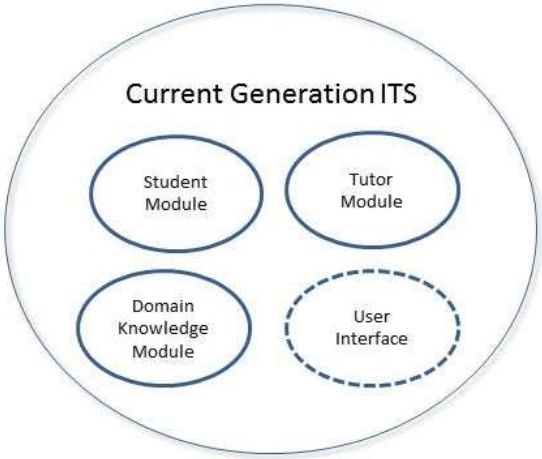

Figure 3 View of the four-model ITS architecture.

Domain Knowledge

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 through knowledge models. The three standard types of knowledge models frequently used are rule-based, constraint-based, and expert system[21]. Historically, 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 a derivation from the work of previous students [7]. The knowledge module is also used to assess the student's overall progress via comparison to the knowledge module.

Student

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 [12, 20]. Self's list of student model functions, though non-exhaustive, is helpful for understanding the models' 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. [23]

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 [21]. Much research emphasis is placed on inferring characteristics that are not directly explicated via the student knowledge level, for examples student learning style. As a whole, accurate student modelling is essential for adaptable and supportive ITS.

Tutoring

The tutoring module both encompasses and enacts the instructional design paradigm of the ITS. Linkage with the student module enables 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 [20]. 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.

User Interface
The user interface module controls interaction between the student and system. It bi-directionally connects the ITS's internal representation to communication with the student. Current ITS provide a variety of interface mechanisms through graphics, text-entry, key-board, and mouse-driven events and menus [1].

Existing ITS vary greatly in terms of learning domain as well as architectures. Example applications include the following:

- **Andes (Physics):** Considered to be the ancestor of the modern ITS. Offers adaptive help on over 500 online physics problems with proven learning gains [10].
- **ASSISTments(Mathematics):** Wide user base and problem set of over 60,000 questions. Research has shown student learning gains of half a standard deviation[22].
- **iList (Computer Science):** Facilitates learning of linked lists, a fundamental CS data structure. Equipped with dialog interface for sophisticated feedback regarding student programming in C++ and Java [8]. As depicted in Figure 4.
- **Tactical Language and Culture Training Systems (Language Learning and Professional Training):** Helps people quickly acquire communicative skills in foreign languages and cultures. More than 40,000 learners worldwide have used TLCTS courses [11]. As depicted in Figure 4.

Figure 4 Illustration of the TLCTS interface (left). Screenshot of the iList interface (left).

3. 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 [18, 27].Longstanding research has shown that both cooperative and collaborative interactions among students are beneficial to learning [17]. In fact, the learning gains are often more profound than those achieved by the best of individual learners [2]. 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)" [2].

However, assigning students to a group and charging them with a task does not ensure that students will engage in effective collaborative learning behavior [13, 25]. It is not uncommon for groups to struggle with dysfunction that often stems from an unbalance of participation, lack of leadership, understanding, and encouragement [25]. Thus, CSCL requires careful construction of the collaboration so that interactions benefit the individual and group [28].

Research in CSCL accounts for the social and construction elements of the learning process in its application of technology [19]. 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 [22]. 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 [24]. Studies also suggest these kinds of interactions lead to deep learning, the process of learning for transfer [5]. As these interactions do not necessarily emerge spontaneously, CSCL applications structure group activity in order to promote these behaviors [4], [19].

Structure of CSCL

The classical approach to providing support in collaborative settings has been through macro-scripts, also known as pedagogical scripts [27]. Dillenbourg defines the script as follows:

> A script describes the way students have to collaborate: task distribution or roles, turn taking rules, work phases, deliverables, etc. This contract may be conveyed through initial instructions or encompassed in the learning environment. [5]

The scripts introduce structure and constraint, and partly or wholly guide the collaborative interaction between students facilitated by a computer-based system [27]. They track students' progress with the script sequences, prompt engagement in activities, and offer additional resources as needed [13].

Tchounikine's analysis of CSCL systems shows that common scripts:

1. describe the task to be achieved by students
2. define how the task is to be divided into subtasks and the sequencing of these subtasks
3. define the role of each student
4. set constraints or rules for the interaction
5. prescribe the features or tools of the computer-based system to be used by the students [27]

Kobbe's work is the first to synthesize these features in a concise framework intended to increase the reusability of scripts among researchers and practitioners [13].The work gives a thorough description of the components and mechanisms of a script as seen in Figure 5.

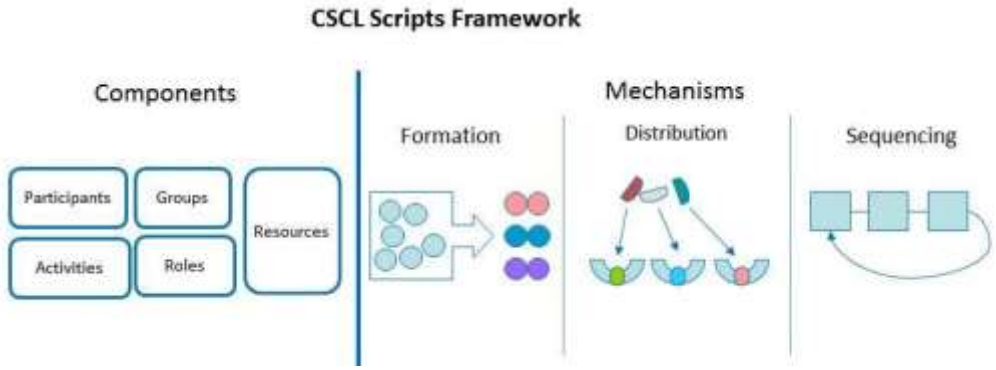

Figure 5 Scripts components and mechanisms.

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. 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)[13] .

Kobbe's framework serves as "a conceptual basis for the computational formalization of CSCL scripts" [13]. The framework enables the use of computer-based tools for modelling and design of the scripts as

well as the interpretation and execution of the scripts in CSCL environments [29]. Weinberger's lifecycle of a script is shown in Figure 6 [29].

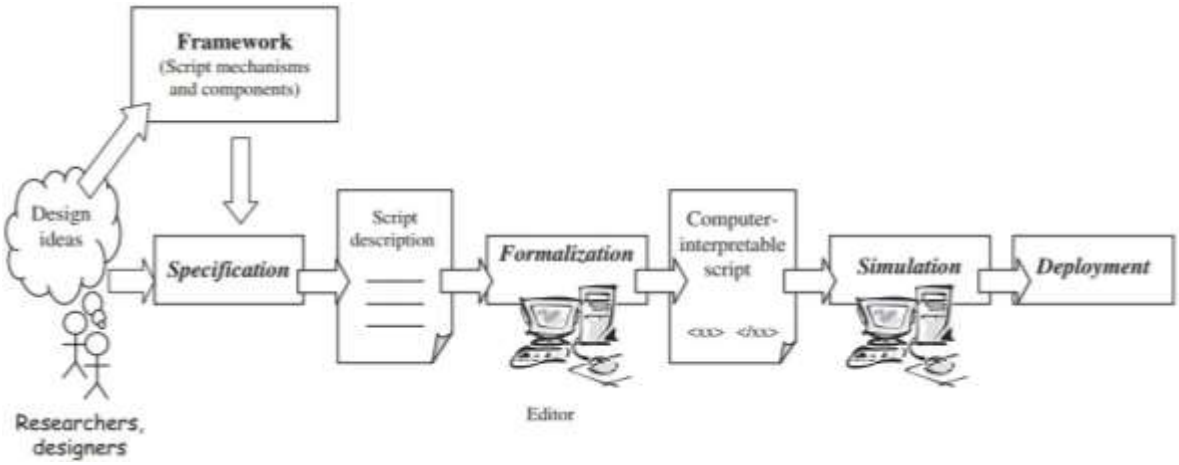

Figure 6 Lifecycle of CSCL script that includes computational formalization and development.

Until recently, traditional script support has been *fixed* in that the level of support is consistent across students regardless of ability or interaction [28]. Due to the nature of fixed support, the same script may provide insufficient support for poor collaborators, unnecessary support for experienced collaborators, or on-target support for others [28]. 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" [18].

An Example CSCL Script

ArgueGraph is a macro-script intended to promote argumentation amongst pairs and engage learners through individual, group, and class activities [4]. The five phases of the script are described below and depicted in Figure 7 [4]:

Phase One: Each student takes an on-line multiple choice questionnaire and justifies their choices in a text-entry field. There are no correct or incorrect answers.

Phase Two: The system plots all students according to their answers on a simple graph displayed to the class. The teacher discusses the graph while students react to the social dynamics. The system connects pairs of students separated by the greatest distance on the graph.

Phase Three: Pairs work together to answer the identical questionnaire phase one including the text entry argument. They can read their individual previous answer.

Phase Four: The system aggregates the pair questionnaire answers and arguments. The teacher thus engages the class in an interactive lecture based on student submissions. A theoretical framework is also introduced to structure the arguments.

Phase Five: Student individually writes a synthesis of all elements collected for a specific question using the theoretical framework introduced above.

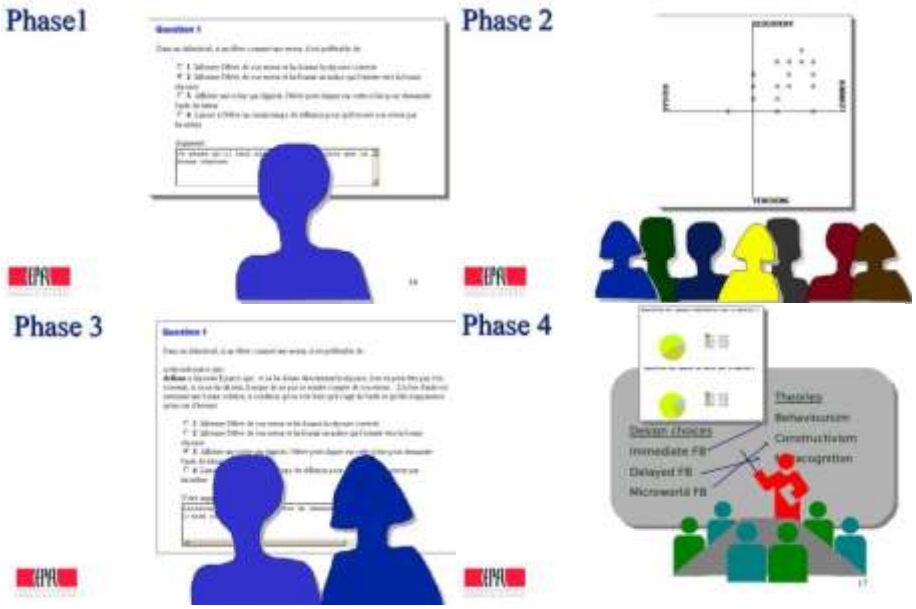

Figure 7 Four of the five phases of the collaborative script for ArgueGraph.

4. Collaborative Intelligent Tutoring Systems

Both the CSCL and ITS communities 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.

Recently, research efforts have focused on merging the affordances of both industries to capitalize on the benefits of group learning and adaptive support [18]. Several researchers in the CSCL community are exploring how adaptivity, automated analysis, and feedback integrate into CSCL approaches [27]. 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 [15]. The CSCL community's work on shaping effective collaboration must guide the creation of ITS systems and their integration of collaborative capabilities. Tchounikine 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. [27]

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 [18]. For the purpose of this critique, I have reviewed four CITS studies. Brief synopses of the systems are given below. These synopses will be helpful for the analysis that follows using the dimensions of the forthcoming classification scheme.

Cognitive Tutor Algebra with Peer Tutoring

Cognitive Tutor Algebra (CTA) is a traditional ITS for high school algebra learning. The system works on an individual basis with a given student. CTA works by comparing the student's actions with a model of good and bad problem solving steps, and provides corresponding feedback and instructions to the student. Though the impact of the system falls short of the effects achieved by human tutors, the system

still increases learning gains by approximately one standard deviation above classroom instruction [14]. Over 2600 classrooms across the United States use the system [28].The current study on CTA implements an extension to the original system in order to allow for collaboration via peer tutoring [28].

The current work on the CTA Peer Tutor explores two research questions. First, Walker investigates how collaborative learning with an ITS may yield improved domain learning in comparison to individual learning with an ITS. The study's motivation comes from previous work in CSCL that shows collaboration has a positive effect on individual and group learning gains and promotes deep learning. Secondly, the study investigates whether adaptive support for collaboration is more effective than providing fixed support. The study yields inaugural work in learning technologies due to the method of empirical evaluations that combine data regarding student processes (dialog and problem-solving actions) as well as learning outcomes. The three conditions of the experiment produce no difference in learning outcomes. However, the student processes attribute noteworthy conclusions regarding the benefits of peer tutoring and adaptive assistance.

The control condition focuses on individual tutoring, where students perform algebra equation solving in a step by step manner allowing them to perform one mathematical operation at a time as depicted in Figure 8 [28]. The cognitive tutor provides feedback after student operations and hints upon request along with a skill mastery meter.

In a second condition, the fixed collaboration condition, students work in peer tutor-tutee pairs. The condition has two phases: preparation and collaboration. In the preparation phase, students use the CTA to solve the same problems they will later tutor. The preparation phase is similar to the individual tutoring condition. However, after each solved problem, students answer a reflection question to prepare them for tutoring (e.g. "A good question asks why something is done, or what would happen if the problem was solved a certain way. What is a good question to ask about the problem?")[28]. Students do not receive feedback on their reflections.

In the collaboration phase, students work at different computers in the same classroom and take turns being tutors and tutees. The interface for peer tutoring allows for chat, and an enlarged skill meter as shown in Figure 8 [28]. Acting in the role of the cognitive tutor, tutors can see their tutees work and mark actions right or wrong. However, they cannot solve the problems themselves. Peers can communicate freely over chat to ask questions and give explanations and hints. Finally, peer tutors receive fixed support that consists of solutions to the problems. Once the peer tutee and tutor agree a problem is complete, the tutee can move to the next problem. This holds true even if the tutee did not correctly solve the problem.

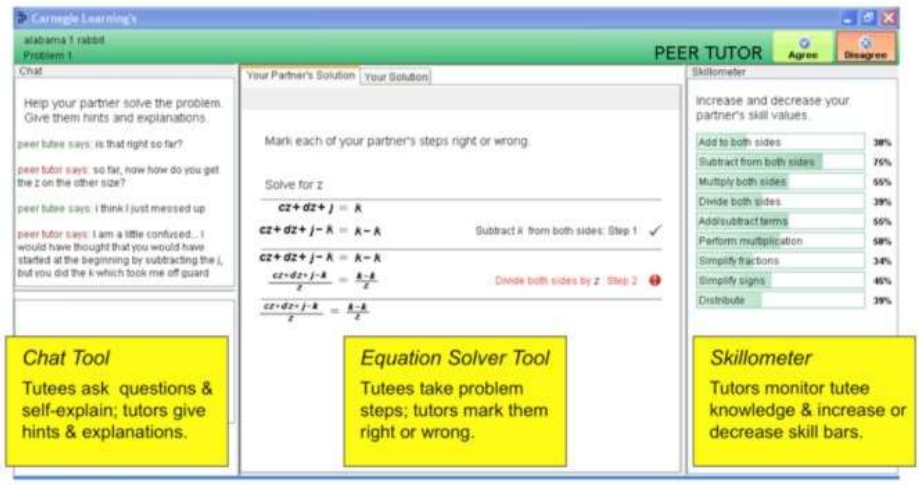

Figure 8 CTA Interface with peer tutor(right). Students use menu-based interaction to solve problems.

The final condition, the adaptive collaboration condition, is nearly identical to the fixed collaboration condition. However, the condition adds intelligent tutoring support to assist the tutor in two ways. First, the peer tutor can ask the CTA for a hint and then relay this information to the tutee. Second, if the peer tutor incorrectly marks a tutee answer, the intelligent tutor highlights this mistake and presents the peer tutor with an error. The error consists of a prompt to help the tutee again and the domain assistance the peer tutor would have received if they were completing the problem individually. Finally, in this condition, the pairs cannot move forward unless problems are completed successfully.

The study consisted of 62 student participants from a vocational high school algebra course. Pretests, posttests, and delayed tests helped to shape the result set along with collected data on student dialog and problem-solving actions. Results showed that students learned equally across all conditions. However, the paths to the learning differed based on the conditions. Key results include the following:

- Students in collaborative conditions completed far fewer problems to achieve the same learning gains as the individual condition (though this may be due to the control for time)
- Students made a parallel number of errors and asked for help at the same rate across all conditions.
- Tutor feedback that consisted of elaborated help (explanation of a step beyond yes-no feedback) and that was in response to a help request strongly correlated to tutor learning
- Viewing tutee errors and problem solving impasses also correlated to tutor learning
- Moving on without correctly solving a problem correlated to negative learning gains for both tutor and tutee
- Tutor withholding of feedback given by the cognitive tutor negatively correlated with tutee learning while feedback communicated was positively correlated with tutee learning gains.
- Tutee receipt of help when requested positively correlated to tutee learning.

This study integrates multiple sources, namely, records of collaborative dialogs and fine-grained problem solving data, to analyze the effects of adaptive collaborative learning.. The rarity and richness of this data contribute to literature in learning because the empirical results themselves show how tutors and tutees learn. Due to the complexity of the data, which includes multiple phases of learning (individual and collaborative) and multiple roles (tutor and tutee), several layers of analysis present themselves as depicted in Figure 9 [28]. These layers correspond to analyses found in both CSCL and ITS approaches. The technique of combining the parallel data sources also sets a precedent for data analysis in CITS [28].

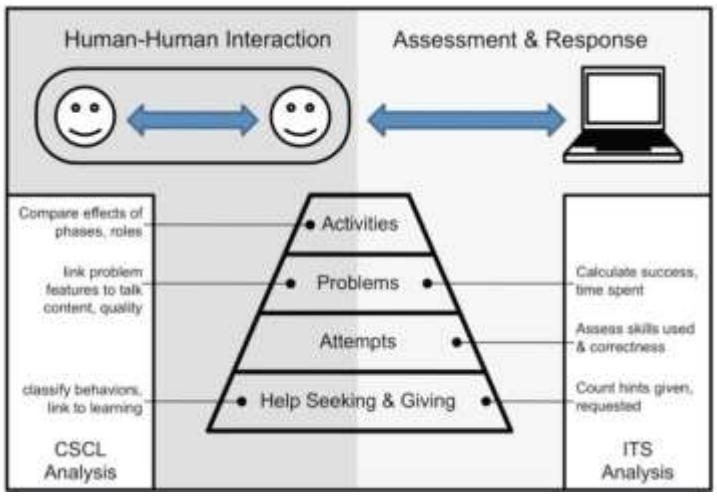

Figure 9 Analysis of possible interactions using CTA Peer Tutor. Both CSCL and ITS approaches allow for different levels of examination.

Rashi

The Rashi inquiry learning system provides an open-ended space for students to explore real world problems [6]. The focus of the current research lies in the implementation of Rashi as the Human Biology Tutor. Equipped with interactive images, interview interfaces, video, tools for inquiry and information organization, students evaluate and hypothesize the medical condition of patients (Figure 10).

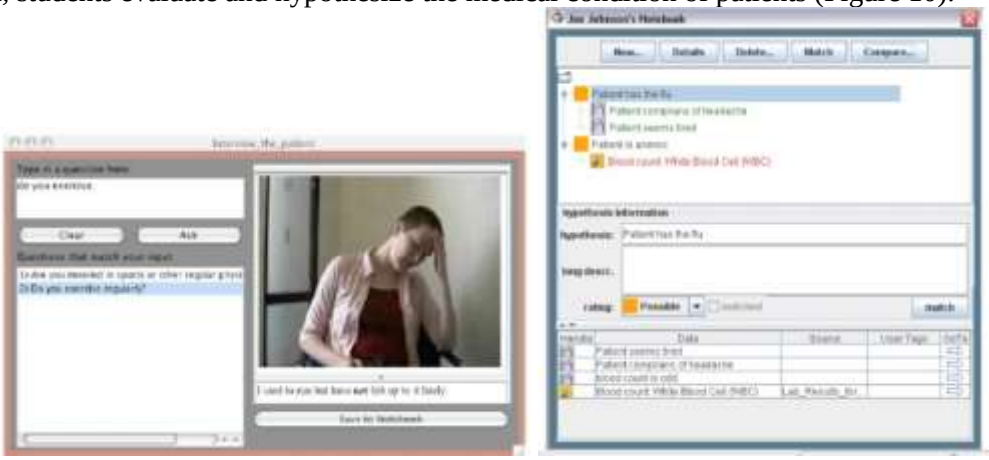

Figure 10 Rashi system allows students to interview patients (left) and record their diagnosis and supporting or refuting evidence (right).

Rashi's major research contribution substantiates the use of a knowledge base for recognition of student content in an open-ended dialog space. Previous research has used sentence openers and data mining of previous student work to recognize the content of chat [6]. The solution to the problem of content recognition for text discussion may help to address a wider CITS issue of helping students to become better collaborators. Because a system that identifies student content focus, can then become more effective in helping to maintain student focus.

Rashi promotes collaborative interaction between students through data tracking and chat. First, students keep track of data and hypotheses in individual notebooks. Students are able to drag and drop information from other student notebooks into their own. Collaboration can involve individuals that fulfill specific roles within a group or a student may work individually but use the sharing mechanisms when caught in an impasse. Secondly, Rashi also provides student chat functionality that includes subject labelling. The subjects allow for easier filtering of prior conversations and linkage of subjects to notebook items.

The introduction of collaborative features in earlier versions of Rashi used in high school and college classrooms produced an increased volume of student work. However, the increase in the amount of work was in no way indicative of performance in final projects and overall grade. This resolved Rashi researchers to the question of quantity versus quality. Students engaged in lots of "work" within the system; however the work was often "off-task, repetitive, or tangential to the problem at hand". Thus the current exploration of Rashi involves focusing student collaborative work through the automatic recognition of student content.

In order to combat the off-task collaboration, Rashi currently offers a critique-rebuttal feature for student-to-student notebook correspondence. Each item of a notebook can be critiqued by another user and the notebook owner has the ability to offer a rebuttal. The critique and corresponding rebuttals can continue as forum-style threads with the overarching goal for students to engage in constructive criticism and discussion around content. The recognition system takes shape within the critique-rebuttal feature of the notebook, overall notebook interactions, and student chat messaging.

An expert knowledge base powers Rashi's recognition system. Domain experts across varying fields including biology, forestry, and geology, helped to create the knowledge base. Thus, the system can provide expertise over many different subjects. The expert knowledge base is a directed, acyclic graph connecting supporting and refuting evidence to specific domain concepts as seen in Figure 11.

Figure 11 Expert Knowledge Base for three diagnosis and the supporting and refuting evidence.

Rashi uses the Lucene search engine library (lucene.apache.org) to match student text-entry to knowledge base items. Lucene indexes the knowledge base elements along with their keywords and the resulting search engine set. Scores are then matched against student statements.

Essentially, Rashi aims to help students help each other much like a coach. Thus, in lieu of fielding individual student support requests, an ideal collaborative coaching agent automatically reasons about all student work concurrently and thus equips students to *help each other* at the right times. However, the current state of the work requires the individual recognition scheme to be extended to accommodate more precise and constant updating of matches to the knowledge base.

The study takes a step forward in achieving the ideal collaborative coaching agent. It examines Rashi's ability to accurately classify 1) if chat messages refer to domain content and 2) Rashi's ability to identify the content of discussion. Rashi's performance was analyzed in two classrooms with a total of 796 chat messages. Three datasets form the analysis, "Summer 09 w/ enhanced KB", "Fall 09 w/ enhanced KB", "Fall 09 w/o enhanced KB". Rashi agreed with an independent judge in 88% percent of the cases, identifying messages containing domain content across three trials. Rashi had an average success rate of 70% in identifying the domain content of chat messages across the same three trials [6].

Though the study provides no baseline for accuracy achieved using other methods, the results show that an ITS can link an infinite space of student input to a set of hundreds of elements within a knowledge base. This is more significant than exclusively determining if student input relates to domain content. The ability for an ITS to accurately determine domain content of student discussion yields numerous opportunities for enhanced coaching. In particular, these results are helpful for the enhancement of intelligent tutoring in ill-defined domains.

Wayang Outpost

Wayang Outpost is an intelligent tutoring system for K-12 mathematics education [2]. The system utilizes an animated character, Jake, as the embodiment of the tutoring agent. The system also incorporates multimedia and animated adventures into the process of problem solving. Students may solve both word problems and problems in the format of standardized test such as the SAT. Wayang Outpost has been shown to improve average student scores on standardized tests by an average of 20% after 4-5 hours of using the system to solve problems from Massachusetts standardized tests. The interface is shown in Figure 12.

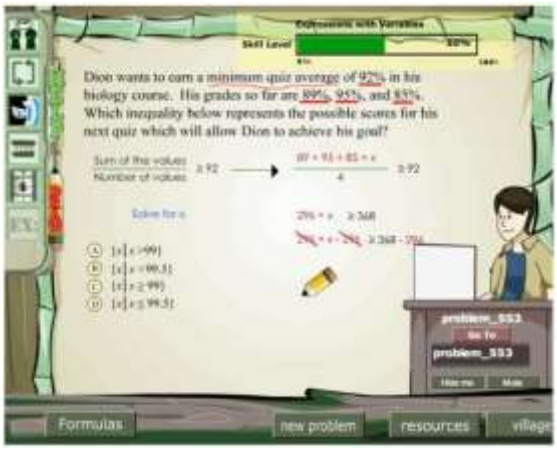

Figure 12 The tutoring interface for Wayang Outpost where students interact with an animated partner..

The primary contribution of this study is the exploration of "natural collaborations" for tutoring systems. Natural collaborations are those which emerge when students work in a one-to-one learning environment, however, they work in some way with a collocated partner. Thus, the fact that the tutoring system was developed for a "private" one-to-one session is overruled by the students' interactions. The research study examines engagement behaviors of collaborators versus non-collaborators. The results of the study point to possible applications of collaborative learning to discourage gaming of tutoring systems.

The Wayang Outpost system promotes natural collaboration by allowing students to navigate directly to a problem via a "Go To" button. This type of navigation differs from Wayang's standard implementation which has mandatory problem progression enforced by the system. The remainder of the functionality does not differ from the individual ITS. The system provides help via hints, videos, and animated annotated examples. Wayang adapts its presentation and problem selection based on student skills via its Adaptive Problem Selection algorithm [2]. The algorithm uses "individual problem difficulty as well as student effort, mastery, and emotions" to construct its model of interaction. With the use of a continuum of math skill sets and correlated problems, Wayang accommodates students of varying abilities from grade four math to developmental math for college [2].

The current study takes place in two middle school math classes (N=20 students and N=23 students) in a public school in Massachusetts. The students used the tutoring systems for at least two 45 minute sessions over a period of two months. Data analysis from the sessions takes place in three phases. First, student-

activity interactions and student-problem engagement states are marked within the raw data. The key for the interactions is depicted in Table 1 [2].The study labels NOTR, GIVEUP, and GUESS as disengagement student-problem engagement while SOF, ATT, and SHINT are productive behaviors. BOTT does not retain an engagement label as it could indicate students are searching for the right answer due to engagement or they are learning through examples.

State	Description
NOTR	Not enough time to read the problem before an action is taken.
SOF	Solved on first attempt without help.
BOTT	Getting answer from hints or help. Possibly seeing problem as example.
GIVEUP	Moved on before answering.
ATT	Made 1-2 incorrect attempts and self-corrected, without help.
SHINT	Answered with help, without guessing possible solved on first attempt.
GUESS	Guess answer after several attempts.

Table 1 Student-problem interaction states.

The second analysis phase infers the collaborative pairs from the data. The study uses an NxN matrix of simultaneous problem activities. Each element SP_{ij} of the matrix corresponds to the total number of problems that student s_i and s_j solved simultaneously. The mean number of collaborations for all students is used as a baseline to distinguish heavy collaborators from those who may have collaborated on a problem by chance. Within this study, the threshold to determine high collaborators is twice the average overlap of problems. The final phase analyzes the engagement behaviors of collaborative pairs in comparison to non-collaborators.

The results (Table 2) show the difference of likelihood of productive behaviors between collaborators and non-collaborators. They also show the difference in disengagement behaviors between the two groups. Collaborators have a mean advantage of 10% in productive behaviors with a 6% less likelihood of disengagement behaviors. Wayang Outpost attributes the productive behavior to the act of collaboration. However, it is unclear whether the groups may have been formed by more productive people at the onset.

SmdId	Mean Difference in Likelihood of Productive Behavior, for Collaborators minus Non-collaborators $\hat{PBC}_i - \hat{PBNC}_i$	Mean Difference in Likelihood of Disengagement Behaviors, for Collaborators minus Non-collaborators $\hat{DBC}_i - \hat{DBNC}_i$
9647	0.07	-0.07
9648		-0.02
9652	0.05	-0.05
9653		-0.02
9654	0.13	-0.12
9655	0.26	-0.26
9656		-0.10
9658	0.01	-0.03
9660	0.02	-0.01
9661		-0.02
9662	0.05	-0.05
9663		0.05
9664	-0.03	-0.03
9665	0.30	-0.12
MEAN	**0.10**	**-0.06**

Table 2 Comparison of likelihood of productive learning behaviors vs. disengagement behaviors among students acting as collaborators and non-collaborators.

Basilica

Basilica is a software architecture and toolkit intended to support the extension of traditional, individual tutoring dialog systems to accommodate collaborative learning. The system uses an object-oriented approach to represent Conversational Agents (CAs) as a connected system of components with varying

functionalities. The current study explores the architectural features of Basilica and three tutoring systems built using Basilica.

Basilica takes an innovative approach to CA development. A primary contribution of the Basilica architecture is that it allows for flexibility to create CAs that can implement interaction strategies for situation-specific, complex interactions. The majority of traditional CAs have been created and tested in a one-to-one tutoring system. Basilica addresses the assumptions of these state-based systems and plan-based approaches that do not extend to a collaborative setting. Typical approaches to CAs for a single user rely on the assumption that only two participants will interact [16]. Based on this premise, CA developers are able to assume two important design factors. First, the two parties will participate relatively evenly (as in turn-taking). Second, the addressee of conversational dialogue is known, namely the addressee is always the participant who is not the speaker. The typical interaction of a single user and multiparty users are depicted in Figure 13.

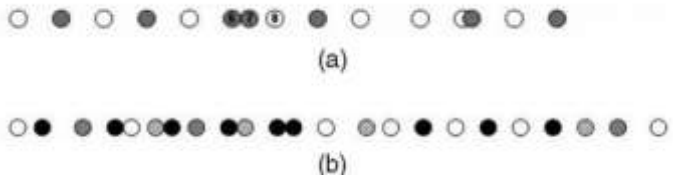

Figure 13 Failure of the even participation assumption with multi-party interaction. White dots are the agent. Gray, Striped, and Black dots are other participants. (a) is a single-user interaction while (b) is a multi-party interaction.

The CSCL community has applied several strategies for addressing these assumptions within individual systems including error recovery strategies to cope with misunderstandings and assignment of roles amongst participants and to focus the tutor support on a particular role [16]. Basilica aims to offer a more flexible and less error-prone solution. Two additional factors motivate the design of the Basilica system - the affordance of the ability to represent complex interaction behaviors and the decrease of development effort required for CA implementations. The final architecture is able to address these requirements.

Three exemplified Basilica agents suggest the architecture can be a valuable tool for building collaborative learning environments. Basilica's ability to define interactive behaviors and extend current systems to meet the needs of a collaborative tutoring environment showcases the system's promise for CITS authoring.

The Basilica interaction architecture is built as a model-based reflex agent. The agent receives stimuli from the environment. The agent then processes the stimuli by the trigger of relevant behavior. The relevant behavior may generate internal events and/or send a response back to the environment. The internal component states receive the generated events and the internal component states may be updated (Figure 14).

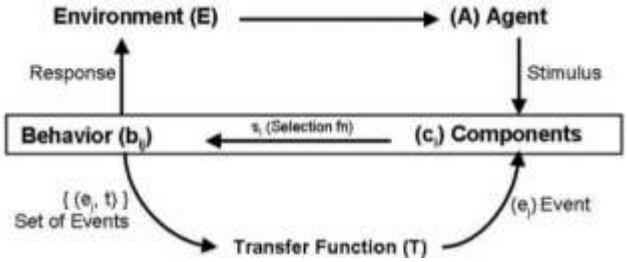

Figure 14 Model of the conversational agent.

Basilica agents are implemented as a collection of "behavioral components" [16]. These are computational capabilities such as "perception, action, cognition, affect, and memory" [16]. The object-oriented programming approach to agent creation allows the agents to be decomposed into smaller,

loosely-coupled components that abstract behaviors [16]. A selection function of each component consists of a one-to-one mapping between event/stimulus and behavior. The transfer function propagates all relevant events to a given component via a broadcast to all connected components or developers may selectively transmit data.

Basilica provides the foundational abstract classes required for definition of agents, components, connections, and events (in Java). Default component implementation allows for asynchronous, non-blocking behavior execution. Basilica also provides a generic implementation of the selection function where events are buffered for processing, and a generic memory class.

Kumar reviews several agents built using the Basilica architecture [16]. These examples include abbreviated looks at TuTalk and the Second Life tutor. However, the primary examples involve CycleTalk Tutor, WrenchTalk Tutor, and PsychChallenge Peer. In the CycleTalk tutor agent, undergraduate engineering students work in pairs to learn principles of thermodynamics cycles. The collaboration environment consists of text-based chat. As students work on a design challenge, the CycleTalk tutor offers instructional support via tutoring scripts. The TuTalk state-based dialog engine powers the scripts. The scripts define "directed lines of reasoning composed of a sequence of steps that implement an Initiation-Response-Feedback interaction pattern with the goal of leading a student to construct a correct explanation for a complex concept as independently as possible" [16].

In an effort to engage students and combat the scenario in which students either ignore or abuse the tutor, the study implements two interaction strategies, 1) *Attention Grabbing* and 2) *Ask when Ready*. In the *Attention Grabbing* strategy, the tutor requests the students pay attention before beginning instructional dialog. In the *Ask when Ready* strategy, the tutor simply asks students to notify the tutor when they are ready to discuss a relevant instructional topic (Figure 15).

59	09:35:07	**Student2**	the pressure stays constant through the boiler?
60	09:35:47	**Student1**	Yeah
61	09:35:55	**Tutor**	Now might be a good time for some reflection.
62	09:36:00	**Tutor**	Consider the effect of increasing Qin. What happens to power out when Qin is increased?
64	09:36:46	**Student2**	i am pretty sure we want high Qin

40	08:52:24	**Student5**	and then Power out vs. the same things
41	08:52:26	**Tutor**	Lets review the effect of changing P-max on the cycle.
42	08:52:27	**Tutor**	Type: HELP WITH PMAXKCD if you want to discuss it with me
47	08:54:08	**Student7**	HELP WITH PMAXKCD
48	08:54:14	**Tutor**	When P-max increases, is the need to reject heat from the cycle increased or decreased?
49	08:54:51	**Student5**	Decreased

Figure 15 Excerpt of showing the Attention Grabbing strategy (left) and Ask when Ready (right).

There are six types of components, *Listeners, Actors, Filters, Detectors, Coordinators,* and *Managers*. *Listeners* listen to stimuli from the environment and translate them into events internal to the agent. *Actors* perform actions, which may be directly observable by other participants in the environment. *Filters* process information that events carry and propagate them further based on their programmed conditions. *Detectors* are special kinds of Filter, which detect specific semantic concepts/phrases and send out a detection event. *Coordinators* control the flow of events between related components to achieve coordinated behavior. *Manager* components exhibit a variety of behavior like planning, execution, and control. The component network for CycleTalk employs 13 components and 21 connections as seen in Figure 16.

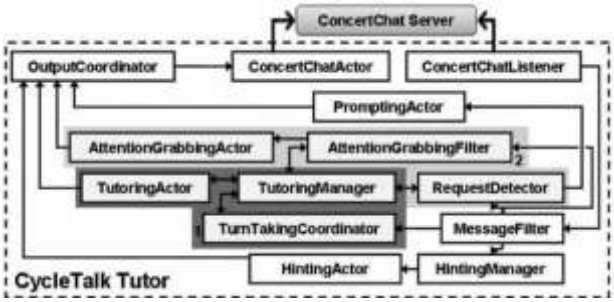

Figure 16 Component Network of CycleTalk

The WrenchTalk tutor helps freshman engineering students, who work in groups of three to five students, learn concepts of force, moment, and stress in a wrench design class. The interface contains text-based chat and a shared whiteboard. The WrenchTalk tutor provides information regarding the learning task and instructional content via hints and TuTalk scripts in a manner similar to the CycleTalk tutor. The WrenchTalk tutor integrates eleven social interaction strategies using a social behavior component, *SocialController*. The table of social interactions along with the component network is depicted in Figure 17.

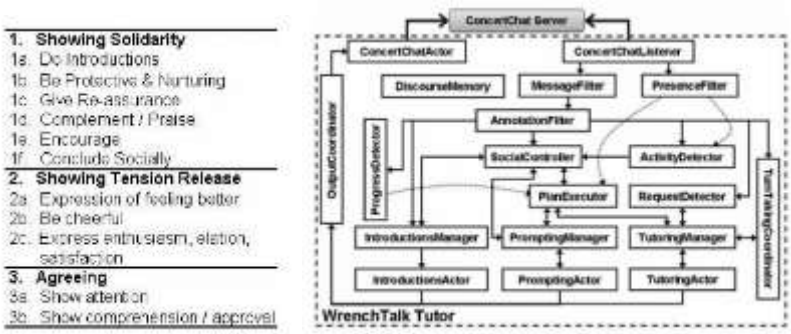

Figure 17 Social Interaction Strategies and Component Network of the WrenchTalkTutor

The final example of the Basilica architecture is the PsychChallenge Peer. The significance of the PsychChallenge peer is that it showcases the ability to build agents that are not tutors using the Basilica architecture. As the name suggests, the agent plays the role of a peer. Students play a vocabulary game and can connect with either networked peers or the agent to serve as a learning companion. The groups play a game to guess vocabulary terms given hints from a single player. The agent can play the role of g*uesser* or *giver*. However, the agent works to promote participation from all players.

Results of the three Basilica architecture agents - CycleTalk, WrenchTalk, and PsychChallenge Peer - show Basilica's ability to support a variety of collaborative learning environments. In CycleTalk, student pairs working with a tutor learned 1.24σ more than students working without a partner or conversational agent. Individual students working with a conversational agent learned 1.06σ more while students working with a partner learned .9σ more compared to the control of no partner or conversational agent [16].

In the WrenchTalk study, the small talk interaction strategy led to marginal learning effect of .55σ [16]. However, students perceived themselves and their partners to be more helpful. Further experiments also showed the same intervention led to significant improvements in student attitudes towards tutors. Basilica studies on behavioral components for social interaction strategies show that systematically-designed social interactions led to a significant learning effect of .71σ. Further, students prefer the agents that showcase social capabilities [16]. Lastly, the PsychChallenge Peer emphasizes Basilica's ability to support other collaborative learning environments in addition to tutoring.

5. Classification and Critical Review

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 CITS. Thus, the comparative analysis between the selected works resulted in a pioneering classification scheme for CITS that blends ITS and CSCL paradigms. The scheme consists of three categories: unstructured, semi-structured, and fully structured. The classification scheme is intended to support the analysis of design principles used within the emergent class of CITS. I propose that CITS can be classified and analyzed via the following dimensions:

1. *Modeling:* how the system models learners and whether the learning domain includes collaborative behavior.
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 of communication is determined and how feedback and activity facilitation is implemented
5. *Technology:* the system tools for interaction

This classification scheme encompasses the main operational dimensions of CITS. CITS are a collaborative learning environments that uses technological tools (technology) to provide 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 to achieve a learning goal (modelling). Table 3 provides detailed criteria for the classification dimensions along with a mapping to classification types.

	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	✓	✓	✓
	Support concerns collaborative behavior		✓	✓
Who	**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 3 Detailed classification scheme for CITS which includes criteria for the five dimensions.

Modelling

The modelling dimension encompasses the student and domain modules which were discussed earlier as components of the basic ITS architecture. The CTA Tutor models the individual tutor and tutee as well as their collective interactions. This is in stark contrast to the approach of Wayang Outpost which only models individuals and their one-to-one interaction with the system. Rashi also follows suit after the CTA Tutor and models individual and collective participation.

Rashi and CTA Tutor's approach to model collaborative activity is sound . The collaborative efforts in Wayang Outpost can only be traced offline in a seemingly hacked method of finding collaborating pairs. A simple improvement to the system would allow students to notify the system if they work with a partner. This would be unobtrusive to Arroyo's exploration of natural collaborations. With the lack of an accurate group model that at least identifies collaborating participants, data results and discussions are less impactful.

Each approach has reasonable methods of modelling domain knowledge. This is validated by improved students learning gains after using the tutoring systems. The divergence in the systems involves whether the domain knowledge includes the objective to teach effective collaborative behavior. In the case of Wayang Outpost, the tutor does not aim to model or teach collaborative behavior. In CTA Tutor, there is a limited model of effective collaborative behavior. The tutor showcases this model when it prompts the peer tutor to correct their mistakes with their tutee after incorrectly marking a response. The model is limited because it only entails the bare minimum of when a collaborative action is mandated.

A more holistic model would incorporate known strategies regarding tutor timing, affective states of tutees, type of feedback most helpful in a given situation, and other traits of effective collaboration. A primary goal of Rashi is to incorporate this type of holistic model of collaboration. The model would rely heavily on computational formalization of CSCL literature on good collaborative behaviors. Both the CTA Tutor and Wayang Outpost, ground their related work in CSCL literature, however, their collaboration models do not adhere to the literature's consensus on how to breed effective collaboration. .

Group Dynamics

The group dynamics dimension accounts for the participants, groups, roles components, and the group formation mechanism of CSCL architecture. In Rashi, group formation is left to the teacher's discretion. Rashi supports students working in tightly knit group with strict role assignments as well as activities where students work mostly independently with no assigned group role. In CTA Tutor, teachers also assign pairs, however, with the explicit goal of aligning students of similar abilities. Group size consists of two students and roles within the group are strictly defined as an alternation between tutor and tutee.

The strength of the Rashi and CTA Tutor approaches to group dynamics depends on the context in which the tutoring system will be applied. Both approaches to group dynamics promote knowledge-generative activities such as explanation and conflict resolution. The CTA Tutor offers a viable explanation of its rationale behind group formation. Walker argues that peers of all skill levels should be afforded the opportunity to tutor and that prior work shows the importance of peers feeling like they could tutor their partner [28]. However, Rashi affords a greater scale of flexibility in group dynamics in order to accommodate a range of collaborative activities. Neither paper offers convincing learning gains to outweigh one approach over the other. However, the peer tutoring paradigm may benefit from added flexibility such as manual role selection and even tutoring of a virtual peer.

In Wayang Outpost, students form collaborative partnerships on their own, exterior to the teacher and system. Students choose the roles they wish to enact within their collaborative interactions. The system has no knowledge of these roles or the collaborative partnerships themselves. The system's lack of knowledge of these fundamental aspects of CSCL architecture put it at a distinct disadvantage for effective collaborative support. Though the results show derived collaborative partners have a higher rate of productive behaviors, these results may be by chance. Further, the study states the time spent on problems as a predictor of learning; however, it presents no evidence to show these behaviors lead to increased learning gains.

All three systems lack the automated functionality to suggest groups and roles that may be effective. Automated suggestions for group formation and roles based on skill assessment or prior student models

would benefit each system. The system could generate the information for the teacher or implement the partnerships and role assignments as part of the built-in activity enactment.

Collaboration Cues

The collaboration cues dimension deals with the sequencing mechanisms of the previously-outlined CSCL architecture. In Rashi, the initial collaboration can occur at-will, be encouraged for the task, or be required for the task. The collaborative activity is the determining factor for the impetus of collaboration. Some activities may require collaboration while others simply allow the participants to act with or without encouragement. Dragon accurately contends this flexibility allows Rashi to support a wider variety of collaborative activities. Further, Rashi allows students to determine when to collaborate once the initial collaboration occurs. Rashi can also prompt the users to communicate depending on the activity [6]. CTA Tutor attributes this same type of freedom in collaboration timing.

By contrast, CTA Tutor requires initial collaboration. The peer tutoring architecture both mandates and justifies this design choice. Finally, Wayang Outpost allows initial collaboration to occur at will. However, given CSCL literature and study results, a better approach to the design would encourage collaboration for the task. Given Wayang's emphasis on natural collaboration, it is obvious that the collaborators determine when to communicate.

Pedagogical Guidance

The pedagogical guidance dimension concerns both the tutoring module of the basic ITS Architecture and the activities component of scripts, its counterpart within CSCL design. Due to the added significance of tutor modelling for collaborative interaction, it deserves analysis outside of modelling dimension. In Wayang Outpost, students do not encounter any type of pedagogical guidance in relation to their collaborative behavior. Students determine their topic of communication and are free to engage in off-task conversations. Further, the system does not guide the students in their collaboration or on how to provide support to their fellow collaborators. Arroyo does not provide any support to suggest natural collaborations will yield improved student learning. Instead, he points to literature regarding the benefits of group learning [2]. Unsurprisingly, all of this literature revolves around structured collaborations as the CSCL work clearly shows that merely aligning students in groups does not guarantee favorable outcomes. Thus, Arroyo's insistence on natural collaborations seems counter-productive to the established literature. Yet, there is room for exploration given the added benefit of individual tutoring which is not accounted for in CSCL literature.

Rashi and CTA Tutor allow both the user to determine and the system to suggest topics of communication. This approach is the most balanced as it allows students the freedom to communicate and engage in the social aspect of learning. Yet, the system still encourages on-task communication throughout the interaction.

CTA Tutor and Rashi both rely on system guidance of collaborators' activities and communication. A primary purpose of Rashi's content recognition study is to apply the algorithm in an effort to support collaborators' communication. Yet, both systems still allow for participants to communicate and collaborate outside of the bounds of the system's guidance. Students are free to chat and offer help that the system does not offer. The importance of this feature for both systems is that it is supported by years of research in CSCL. The system guidance is the enactment of the activities of a CSCL script. Without this component, the script will not consistently produce the desired results of improved student learning gains given collaboration. For this reason, both systems guide students to give explanations, engage in argumentation, conflict resolution, and other proven knowledge generative interactions.

However, the systems are not without fault in their enactments. The CTA Tutor would be better served to provide guidance to peer tutors before they damage the learning gains of tutees. For example, the peer

tutor marking can pass through a filter before being submitted to the tutee. This would allow the peer tutors to receive feedback and still learn from their mistakes. Further, the CTA tutor could provide more proactive assistance regarding what makes a good tutor. At this point, the system can improve support of the peer tutors as they fulfill their roles.

Technology

The technological guidance dimension entails the resources component and distribution mechanism of CSCL architecture along with the user interface component of basic ITS architecture. In Wayang Outpost, the tool to accommodate collaboration is the "Go To" button for navigation to a particular problem. However, there are no dedicted tools to facilitate communication among collaborators. Wayang relies on the physical proximity of users in order for collaboration to occur. Collaborators could also communicate at a distance with use of their own tools such as cell phone or instant messaging. Wayang's exploration of collaboration may benefit from integration of additional collaboration tools into the system, specifically text-based chat. This would fully decouple collaboration from physical location. Wayang would not have to force communication through the chat medium in order to promote natural collaboration; however, it would allow support for dispersed peers. This tool would also afford more complete analysis of collaborative behavior such as seen in the CTA Tutor.

The CTA Tutor strictly enforces the tools for collaboration. Partners must communicate via the given interface through hint requests, chat, and answer marking in order to progress in the activity. Contrastingly, Rashi provides collaborative tools; but the student's activity progression is not contingent on their use. In theory, students could communicate by phone or e-mail and still complete the collaborative activity tasks. Dragon makes a strong case for why Rashi tools are best suited for effective collaboration. The notebook's critique-rebuttal format encourages both on-task communication in addition to argumentation and negation, which are both knowledge-generative interactions well documented in CSCL literature. Surprisingly, Walker does not justify the choice of interaction tools provided in CTA Tutor but she does explain the core functionality [28]. Exploration of the rationale for the interaction tool usage would improve the study.

CTA Tutor also supports collaboration among physically dispersed participants. In the same way, Rashi allows peers to be dispersed. Neither author comments on the benefit of this feature. However, its importance should not be diminished because with it comes the opportunity for larger networks of peer collaborators. The system could match peers to groups regardless of physical proximity. The importance of this feature may not be as apparent in classroom settings, however, tutoring systems for learning outside of the classroom would rely heavily on this feature.

Revisiting Basilica

The Basilica architecture lies exterior to the CITS classification scheme because it is intended to be a tool for CITS authoring. Basilica supports development of CITS from all of the three classification schemes. The Basilica paper shows how developers can model both individuals and groups via Basilica. Though none of the exemplified Basilica implementations showcase domain learning of collaborative behaviors, the architecture clearly identifies how this functionality can be implemented as an additional component via *Filters* and *Actors*.

Basilica also allows for developers to decide and implement rules for collaboration initiation and the system's role in encouraging communication among collaborators. Developers using Basilica may also instrument their own logic behind group and role assignments. The architecture does not limit itself to specific interfaces as demonstrated through varying usages from the graphics-heavy Second Life Tutor to CycleTalk with the text-based environment, ConcertChat. The tools for interaction may also have the ability to vary beyond chat-based if the *TutoringActor* includes tutoring scripts different from the TuTalk system. Though Kumar does not make this point clear, the architecture suggests that a change in this

manner would allow other tutoring scripts to be used that provide feedback better equipped for the learning domain.

Overall, the Basilica architecture boasts a thoughtful and well-constructed framework intended for rich object-oriented representations, complex multi-party interactions, and reuse. The architecture even boasts a visual debugging interface for CA developers. The lack of public release is the architecture's primary downfall. The author makes heavy emphasis on "mass development and deployment" of systems using Basilica. However, several years after the paper's publishing, the reality of mass usage is unfortunately unapparent.

6. Discussion and Conclusions

So far, this paper has accomplished two of the primary focus points: first, I provided an overview of both the ITS and CSCL fields. Second, I critically reviewed four approaches to CITS through use of a pioneering framework for evaluation. In the section below, I will summarize analysis results and comment on areas for possible research.

CITS and Learning

My analysis suggests that the alleged potential for CITS to improve student learning gains beyond the traditional ITS is yet to be proven. The CSCL literature provides strong support as to why extending current ITS systems to accommodate group learning may be beneficial. Further, the three CITS studies presented also offer promising results that merit further exploration despite clear correlation to learning gains. These results include the success of content recognition in free text entry, the promotion of productive behaviors through collaborative pairs, and understanding of tutor-tutee processes that lead to learning.

However, the CITS framework shows that a plethora of learning designs and capabilities exist and accommodate collaboration. The collaboration may be unstructured, semi-structured, or fully structured. All of these variations will have a different impact on learning. Thus, authoring tools with capabilities similar to the Basilica architecture will be valuable for cost-efficient, mass development of CITS.

The CITS framework provides a solid tool for analysis of a variety of CITS while emphasizing main features required from both ITS and CSCL research. As CITS research continues to mature, equilibrium may be reached among CITS framework features to create the most effective type of collaboration. However, much of the design is contingent on how teachers, practitioners, and individuals prefer to structure their learning and collaborative activities.

Research Opportunities

The reviewed articles suggest several promising areas for research as follows:

Virtual Peer Tutoring: The work of CTA Tutor suggests that implementing virtual peers as tutees may be a beneficial avenue for collaborative learning. Because the tutor often gained learning benefits from properties that are detrimental to an actual tutee (i.e. viewing tutee errors, responding to hint requests, correcting tutor mistakes), a virtual peer would be an ideal tutee. In this way, the tutor can still gain from these experiences.

Passive Support: Rashi's results suggest further work may be beneficial in implementing passive tutor support. As the tutor cannot guarantee accuracy in recognizing content, future work may explore the implementation of passive support such as a "Suggested Topic" list alongside group interactions. This

would follow Magnisalis's notion of a CSCL system as a helpful, unobtrusive peer, a model different from traditional ITS in which the tutor is focused exclusively on active support.

Group Formation: Wayang Outpost promotes the notion of exploration of which patterns (or groups) will create the most effective collaboration. Moreover, the study addresses one means of measuring effectiveness via productive behavior. However, future research may explore other characteristics as measures of efficiency.

CITS Authoring Tools: There is a strong need for public releases of CITS authoring tools and architectural frameworks. As production hours for systems heavily outweigh hours of product, authoring tools have cost efficiency benefits for CITS designers. Secondly, common authoring tools will help to promote best practice approaches to CITS development as well as reusability of common components.

References

1. Ahuja, N.J., Sille, R.: A Critical Review of Development of Intelligent Tutoring Systems: Retrospect, Present and Prospect. Int. J. Comput. Sci. Issues IJCSI. 10, 4, (2013).
2. Arroyo, I. et al.: Casual Collaborations while Learning Mathematics with an Intelligent Tutoring System.
3. Conati, C.: Intelligent Tutoring Systems: New Challenges and Directions. Proceedings of the 21st International Jont Conference on Artifical Intelligence. pp. 2–7 Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009).
4. Dillenbourg, P., Crivelli, Z.: A model of collaborative learning scripts instantiated with mobile technologies. Comb. E-Learn. M-Learn. New Appl. Blended Educ. Resour. 56 (2011).
5. Dillenbourg, P., Jermann, P.: Designing Integrative Scripts. In: Fischer, F. et al. (eds.) Scripting Computer-Supported Collaborative Learning. pp. 275–301 Springer US (2007).
6. Dragon, T. et al.: Recognizing Dialogue Content in Student Collaborative Conversation. In: Aleven, V. et al. (eds.) Intelligent Tutoring Systems. pp. 113–122 Springer Berlin Heidelberg (2010).
7. Floryan, M., Woolf, B.P.: Authoring Expert Knowledge Bases for Intelligent Tutors through Crowdsourcing. In: Lane, H.C. et al. (eds.) Artificial Intelligence in Education. pp. 640–643 Springer Berlin Heidelberg (2013).
8. Fossati, D. et al.: Learning linked lists: Experiments with the iList system. Intelligent tutoring systems. pp. 80–89 Springer (2008).
9. Fossati, D. et al.: Supporting Computer Science Curriculum: Exploring and Learning Linked Lists with iList. IEEE Trans. Learn. Technol. 2, 2, 107–120 (2009).
10. Gertner, A.S., VanLehn, K.: Andes: A coached problem solving environment for physics. Intelligent Tutoring Systems. pp. 133–142 Springer (2000).
11. Johnson, W.L. et al.: Tactical language training system: Supporting the rapid acquisition of foreign language and cultural skills. InSTIL/ICALL Symposium 2004. (2004).
12. Kersey, C. et al.: KSC-PaL: A Peer Learning Agent. In: Aleven, V. et al. (eds.) Intelligent Tutoring Systems. pp. 72–81 Springer Berlin Heidelberg (2010).
13. Kobbe, L. et al.: Specifying computer-supported collaboration scripts. Int. J. Comput.-Support. Collab. Learn. 2, 2-3, 211–224 (2007).
14. Koedinger, K.R. et al.: Intelligent tutoring goes to school in the big city. Int. J. Artif. Intell. Educ. IJAIED. 8, 30–43 (1997).
15. Kumar, R. et al.: Tutorial dialogue as adaptive collaborative learning support. Front. Artif. Intell. Appl. 158, 383 (2007).
16. Kumar, R., Rosé, C.P.: Architecture for Building Conversational Agents that Support Collaborative Learning. IEEE Trans. Learn. Technol. 4, 1, 21–34 (2011).

17. Lehtinen, E. et al.: Computer supported collaborative learning: A review. JHGI Giesbers Rep. Educ. 10, (1999).
18. Magnisalis, I. et al.: Adaptive and Intelligent Systems for Collaborative Learning Support: A Review of the Field. IEEE Trans. Learn. Technol. 4, 1, 5–20 (2011).
19. Nkambou, R. et al.: Advances in intelligent tutoring systems. Springer (2010).
20. Nwana, H.S.: Intelligent tutoring systems: an overview. Artif. Intell. Rev. 4, 4, 251–277 (1990).
21. Paviotti, G. et al.: Intelligent Tutoring Systems: an Overview. Pensa Multimed. (2012).
22. Razzaq, L. et al.: The ASSISTment Builder: Supporting the life cycle of tutoring system content creation. Learn. Technol. IEEE Trans. On. 2, 2, 157–166 (2009).
23. Self, J.: Artificial intelligence and human learning. Chapman & Hall London (1988).
24. Self, J.A.: Student models in computer-aided instruction. Int. J. Man-Mach. Stud. 6, 2, 261–276 (1974).
25. Soller, A.: Supporting social interaction in an intelligent collaborative learning system. Int. J. Artif. Intell. Educ. IJAIED. 12, 40–62 (2001).
26. Strijbos, J.-W.: Assessment of (Computer-Supported) Collaborative Learning. IEEE Trans. Learn. Technol. 4, 1, 59–73 (2011).
27. Tchounikine, P. et al.: Computer Supported Collaborative Learning and Intelligent Tutoring Systems. Advances in Intelligent Tutoring Systems. pp. 447–463 Springer (2010).
28. Walker, E. et al.: Integrating Collaboration and Intelligent Tutoring Data In The Evaluation Of A Reciprocal Peer Tutoring Environment. Res. Pract. Technol. Enhanc. Learn. 04, 03, 221–251 (2009).
29. Weinberger, A. et al.: Computer-Supported Collaboration Scripts. In: Balacheff, D.N. et al. (eds.) Technology-Enhanced Learning. pp. 155–173 Springer Netherlands (2009).