To appear in print version of *International Journal of Artificial Intelligence in Education* 13.
Presently available online at http://www.cogs.susx.ac.uk/ijaied/abstracts/Vol_13/constantino.html

Coaching Web-based Collaborative Learning based on Problem Solution Differences and Participation

MARIA DE LOS ANGELES CONSTANTINO-GONZÁLEZ
Dept. of Computer Sciences
ITESM Campus Laguna
Paseo del Tecnológico #751,
Col. Ampliación la Rosita
Torreón, Coah. 27250, México
Voice: 52.871.729.6363
Fax: 52.871.729.6317
aconstan@campus.lag.itesm.mx

DANIEL D. SUTHERS
Dept. of Information and Computer Sciences
University of Hawai`i at Manoa
1680 East West Road, POST 309
Honolulu, HI 96822, USA
Voice: 1.808.956.3890
Fax: 1.808.956.3548
suthers@hawaii.edu
http://lilt.ics.hawaii.edu/

JOSÉ G. ESCAMILLA DE LOS SANTOS
Educational Technology and Information Department
ITESM - Virtual University
Av. Eugenio Garza Sada 2501 sur,
Edificio Cedes Semisótano 1,
64849 Monterrey N.L., México
Voice: 52.81.83.28.44.66
Fax: 52.81.83.28.40.55
jescamil@campus.ruv.itesm.mx

Abstract. This paper describes the design and evaluation of a coach that helps students collaborate while solving Entity Relationship modeling problems in a computer-mediated learning environment (COLER). Unlike previous work generally emphasizing dialogue analysis or expert models, this work evaluates a new approach to supporting collaboration that identifies learning opportunities based on differences between problem solutions and tracking levels of participation. The contribution made by these and other knowledge sources in the generation of collaboration advice was evaluated by comparing expert rankings of advice to the software coach's rankings, and by identifying the advice that would be lost if each respective knowledge source were removed. Results show that good quality advice can be obtained through these knowledge sources, although other knowledge sources may fill in gaps relative to the expert's performance. This work demonstrates how intelligent agents can produce reasonable collaboration advice in domains for which structured problem solutions exist by using a few basic knowledge sources, and illustrates several methods of evaluating the knowledge and reasoning of complex knowledge-based systems.

INTRODUCTION

Computer-mediated collaborative distance learning environments are receiving increasing attention due to the development of the World Wide Web (WWW) and the reported positive outcomes of collaborative learning studies in the classroom (Gokhale, 1995; Slavin, 1995; Johnson & Johnson, 1994). The WWW enables the development of these applications since Web-based systems can be accessed easily from any computer in the world. However, applying collaborative learning techniques in synchronous distance learning environments brings difficulties to facilitators, who have to monitor and guide participants in the application of collaborative skills. It is very hard for a human facilitator to track many teams working at different times with members located in different places. Therefore, there is growing interest in developing intelligent systems that support this facilitation process. Our work seeks to provide this support through the design and implementation of a computer coach that facilitates effective collaborative learning interactions.

The main problems a collaboration coach has to solve are similar to the ones for individual coaching: when to intervene and what to say. Yet designing a coach that supports students' collaboration is a new challenge, since most prior work on coaching has focused on expert and student modeling (Katz & O'Donell, 1999). In contrast, a collaboration coach has to monitor not only one student's activities, but also the teammates' activities, and should encourage interactions that influence individual learning and the development of collaborative skills, such as giving and receiving help and feedback, and identifying and resolving conflicts or disagreements (Dillenbourg et al, 1996; Johnson et al, 1991; Webb & Palincsar, 1996).

Several computer-mediated collaborative learning environments have been developed to support collaborative interaction. Jermann et al. (2001) present a brief overview of many of these systems. Some systems, such as DEGREE (Barros & Verdejo, 2000), offer collaboration advice in asynchronous learning environments by asking students to select a type of contribution from a list and later rate the collaboration in several dimensions in order to provide students with hints to improve their interaction. Some systems have been designed to encourage participation and facilitate group discussion with intelligent support, such as C-CHENE (Baker & Lund, 1996), the Group Leader Tutor (McManus and Aiken, 1995), iDCLE's Expert System Coordinator (Okamoto, Inaba & Hasaba, 1995), and BetterBlether (Robertson *et al*, 1998). All of these systems use restricted menu-driven or sentence-opener interfaces in order to understand students' interaction, and give guidance based on an ideal model of dialogue. Dialogue-based support provides several advantages, such as potential applicability to any subject matter area, automated interpretation of students' interactions, and restriction of discussion moves and learning interactions to those believed to be productive for learning. However, systems that require use of devices such as sentence openers present some disadvantages such as restricting the type of communicative acts, slowing the communication process, and misinterpreting students' dialogue when students use the interface buttons incorrectly. It would be advantageous to increase the repertoire of ways to provide automated support. One example is using action-based collaboration analysis (Mühlenbrock & Hoppe, 1999), which monitors and analyzes moves of multiple users within a shared workspace. Another example is GRACILE (Ayala & Yano, 1996) which gives help based on Vygostky's concept of the zone of proximal development.

Our work seeks to facilitate effective collaborative learning interactions with minimal reliance on restricted communication devices such as sentence openers. We focus particularly on helping students to recognize and resolve conflicts between their problem solutions, because (as we discuss later) these kinds of collaborative interactions are expected to lead to learning.

In this paper, we evaluate the feasibility of generating advice based primarily on comparing students' individual and group solutions and on tracking student participation (contributions to the group diagram). The approach taken is close in spirit to the action-based analysis of Mühlenbrock & Hoppe (1999). Our approach differs in that we monitor individual work in private workspaces as well as the shared workspace to identify conflicts. Other studies have

used automated coaches to give advice when a student's solution differs from an expert's solution (Burton & Brown, 1982; Paolucci, et al., 1996). In contrast, our work evaluates the possibility of giving advice without comparing student work with an expert solution. We excluded discourse models and expert solutions as a research strategy, in order to evaluate the value of the knowledge sources on which we focus. This strategy should not be interpreted as a denial of the importance of these other knowledge sources.

The remainder of the paper is organized as follows. First we introduce the domain of Entity Relationship modeling and COLER, the learning environment within which the studies were undertaken. Then we describe the architecture of COLER's coach, and detail its algorithms for recognizing differences between solutions, monitoring participation, and generating and selecting advice. The remainder of the paper focuses on our extensive evaluations of the quality of COLER's advice and the roles of the knowledge sources in generating this advice.

COLER

Entity-Relationship (ER) Modeling (Chen, 1976) is one of the most commonly used data modeling formalisms for conceptual database design, a collaborative task in which analysts and database users participate to produce a conceptual schema (Batini, et al., 1992; Gordon & Hall, 1998). The performance of the final database system depends highly on the correct design of the conceptual schema, yet data modeling is a difficult task for novices (Batra & Antony, 1994; Shanks, 1996). The ER model is a diagram composed of a set of basic elements called entities, attributes and relationships. An example of a simple ER diagram, in the original Chen notation, that keeps track of employees and projects is shown in Figure 1. In this example, the Employee and Project entities are related by a many to many relationship called "works-on," and have the attributes indicated in ellipses. The ER diagram notation used in this research is essentially based on the Chen formalism (only binary relationships are permitted). However, IDEF1X notation (Bruce, 1992) is used to represent entities' attributes in more compact diagrams.

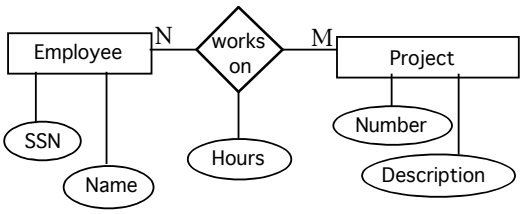

Figure 1. Example Entity-Relationship Diagram using Chen Notation

COLER (COllaborative Learning environment for Entity-Relationship modeling) is a Web-based collaborative learning environment in which students can solve ER modeling problems while working synchronously in small groups at a distance. A personal coach was implemented and included for each student within COLER, as shown in Figure 2. Each coach analyzes the state of interaction and guides the participant by recommending actions student might take to improve their interaction. COLER has been classified as a coaching or advising system according to the categorization presented by Jermann et al. (2001).

Figure 2. COLER Personal Agents

COLER provides four different modes of operation according to the type of user (student or professor) and the selected type of session (individual or group). Before students can work in a COLER collaborative session, the instructor must perform several setup functions, such as defining

teams (name and members) and creating the HTML pages of the database problems to solve. Additionally, the instructor must define the glossary of words for each problem, as well as review/update the value (weight) of typical differences between ER diagrams to indicate which ones are worth discussing. After the instructor has finished all setup activities, students can launch COLER and log into the system. During the collaborative session, instructors can observe students' individual and group progress and give students private or public advice.

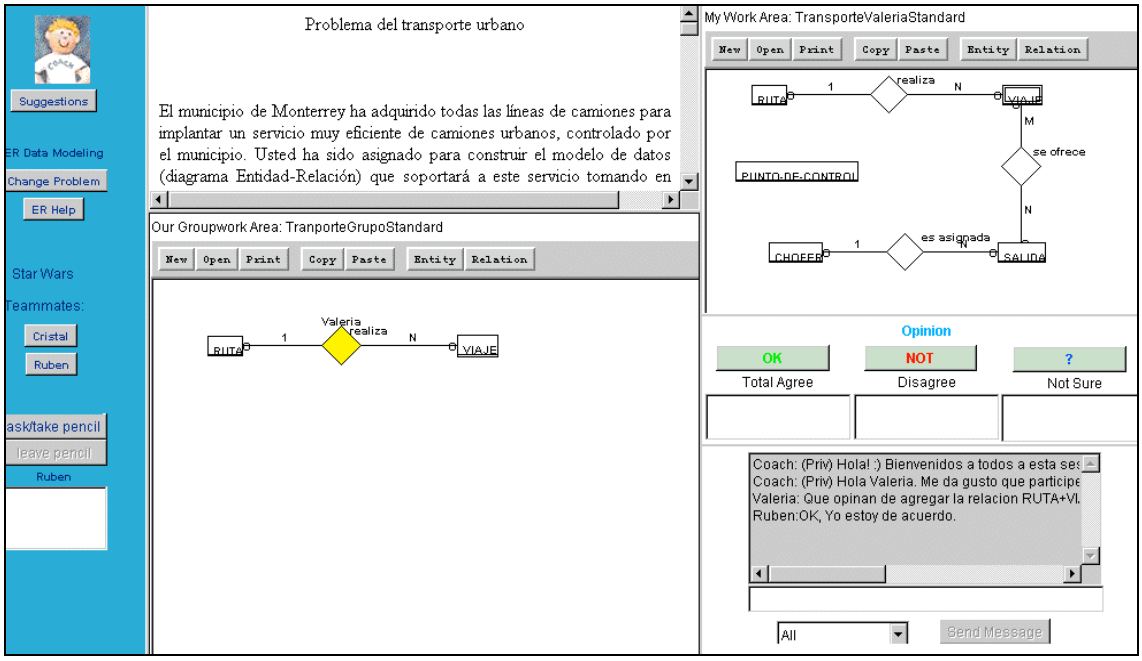

Figure 3: COLER Collaborative Student Interface

COLER's student group interface is shown in Figure 3. The *problem description window* (upper center) presents an entity-relationship modeling problem. Students construct their individual solutions in the *private workspace* (upper right). They use the *shared workspace* (lower center) to collaboratively construct ER diagrams while communicating largely via the *chat window* (lower right). They can use a **HELP** *button* (upper left) to get information about Entity-Relationship Modeling. A *team panel* (middle left) shows the teammates already connected. Only one student, the one who has the pencil, can update the shared workspace at a given time. The *floor control panel* (bottom left) provides two buttons to control this workspace: **ASK/TAKE PENCIL** and **LEAVE PENCIL**. Additionally, this panel shows the name of the student who has the control of this area and the students waiting for a turn. An *opinion panel* (middle right) contains two areas. The upper area contains three buttons that enable students express their opinion regarding the last object added to the shared area: **OK**: Total Agreement, **NOT**: Total or Partial Disagreement, and **?**: Not sure, Uncertainty. When a button is selected, students have the option of annotating their selection with a justification. Opinion button selections are displayed in the chat area (along with any optional justifications) in order to correlate these opinion-expressing actions with the chronology of the chat discourse. The bottom area within this panel shows a persistent summary of the teammates' opinions on a current issue by showing the teammates' name in the box below the opinion button the teammate selected (OK, NOT, ?). A *personal coach* (upper left) gives advice in the chat area based on students' participation and group diagram construction. Although several suggestions may be computed at a certain time, only one is shown in the chat area. The others may be obtained by pressing the **SUGGESTIONS** button, which is disabled if the coach does not have any advice to offer.

COLER is designed for sessions in which students first solve problems individually and then join into small groups to develop group solutions. The private workspace enables students to try solutions without feeling they are being watched. When all of the students have indicated readiness to work in the group, the shared workspace is activated, and they can begin to place components of their solutions in the workspace. This may be done either with **COPY/PASTE** from private workspaces or by making new structures in the shared workspace. Entities and attributes must be named with words in the Glossary of the selected problem to make students aware of the necessity of a common data dictionary and to make it easier for the coach to compare their solutions. The system doesn't attempt to do natural language understanding. Instead, students can type what they want, but if a word used is not in the glossary they are prompted to check the glossary and pick a synonym from there. After each change to the workspace, the changed object is highlighted in yellow. Then students are required to express their opinions using the **OK/NOT/?** buttons before making subsequent use of the shared workspace.

The initial problem solving helps to ensure individual participation and provides differences between students' solutions that form the basis for discussion. Students' initial solutions also provide the coach with useful information to identify learning opportunities. Previous approaches to support collaboration, such as the ones presented by Greer et al (1998) and Hoppe (1995), use a similar approach, assessing individual student models or profiles to inform group learning situations. Based on this information, they are able to suggest partners for a team or students who can help with a specific problem.

COLER's implementation is based on an open architecture for intelligent collaborative learning systems designed by Suthers & Jones (1997) and originally used for the implementation of the Belvedere software for collaborative critical inquiry (Suthers, *et al.* 2001). COLER's architecture is shown in Figure 4. The light objects indicate modules that are Belvedere extensions. The dark objects are new modules.

The COLER architecture includes Java 1.1 applets that are in charge of the different functions of the system, such as chat, floor control, voting, private and shared ER modeling. It uses Belvedere's diagrammatic classes and components for networking. The COLER database was constructed by extending the Belvedere database, using the Mini SQL 2.0.1 database management system from Hughes Technologies[1]. COLER's applets as well as the personal coach communicate with an mSQL database server via a JDBC "Object Request Broker." This broker also informs the Connection Manager of user changes.

The Connection Manager is a process on the server that keeps track of the clients using any diagram. It informs other clients via their Listener sockets of the changes to their diagram for "what you see is what I see" updating of the shared workspace, chat window, and opinion and team panels. Each student's client contains a personal coach. This personal coach is a Java thread that monitors participation, identifies and evaluates differences between diagrams and encourages students to discuss them.

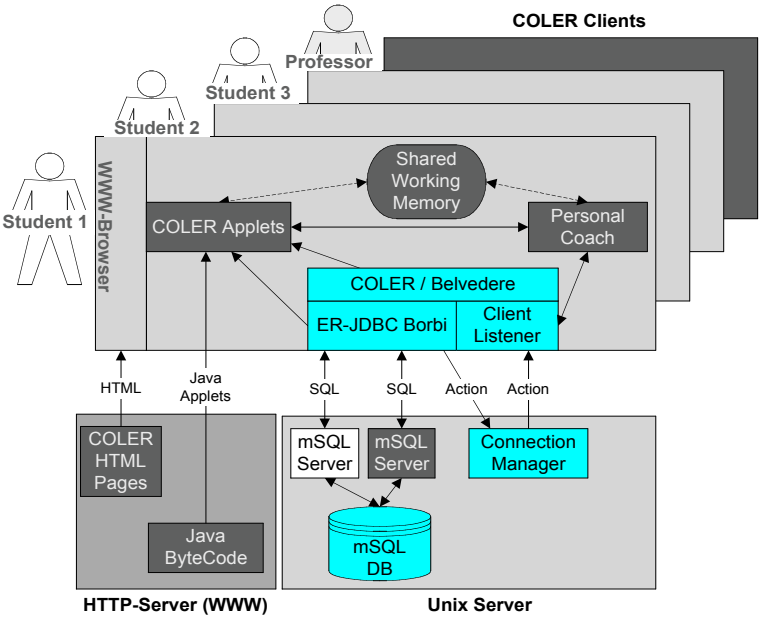

Figure 4: COLER Architecture

[1] http://www.Hughes.com.au

COLER'S COACH

According to Collins, Brown & Holum (1991), *coaching* is a technique in which the instructor observes students and provides hints, help and feedback while they try to complete a task. Since students often miss learning opportunities and get stuck on a certain level of proficiency, a coach can make students aware of further possibilities, provide unobtrusive assistance and create potential learning experiences that will improve individual's development. Burton & Brown (1982) developed the first computer coach by implementing an "issues and examples" paradigm in the WEST system, in which students learn by playing the children's board game called "How the West Was Won." In this paradigm a Differential Modeling technique is applied to identify differences between student and expert models and to find aspects of the domain of pedagogical interest to the system to tutor. The pedagogical agents defined by Gordon & Hall (1998) in an Entity-Relationship Modeling Virtual Learning Environment (ERM-VLE) are another example of domain-specific coaching. In this system learners receive immediate feedback about the correct steps they should take in their search for a solution.

COLER's coach is a pedagogical agent to facilitate collaboration. It does not tutor Entity Relationship modeling, but encourages students to discuss and participate during collaborative problem solving. The coach is implemented as a personal assistant for each student. It has the capability to perform specific tasks autonomously.

Motivations for the Coach

The coach's goal is to promote group-learning interactions and maintain balanced participation. The design of the coach was based on socio-cognitive and cognitive dissonance theories. According to the *socio-cognitive conflict theory* (Doise & Mugny, 1984) students learn from disagreements when they identify and resolve conflicts in their viewpoints, present alternatives, and request and give explanations. *Cognitive dissonance theory* (Festinger, 1957) states that the existence of disagreement among members of a group produces cognitive dissonance in the individual, who experiences pressure to reduce this dissonance, leading the individual to a process of social communication and revision of her position. The value of the disagreement depends less on the correctness of the opposing position than on the attention, thought processes and learning activities it induces.

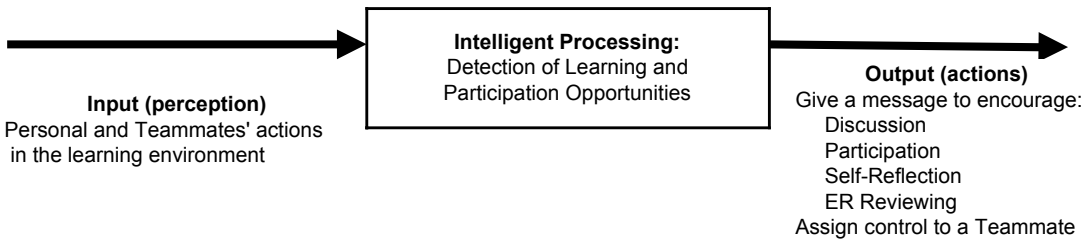

Figure 5: COLER Personal Coach as a black box

The coach helps to prevent missed opportunities for collaborative learning (Baker & Bielaczyc, 1995) by monitoring students' participation and recognizing differences between students' individual and group solutions. When relevant opportunities for learning are found, the coach tries to guide students to practice collaborative skills, providing advice such as encouraging students to participate and to discuss their differences. A general description of the coach is given in Figure 5, where the coach is represented as a black box.

Coach's Architecture

The current version of the personal coach is implemented as a local component of the COLER application. The coach relies completely on information that is local to the application. There is no direct communication between different students' personal coaching agents, although group

parameters are accessible through the local copy of the shared working environment. Local information is sufficient for each personal coach to detect differences between the group and the individual solution, monitor participation and provide personal guidance. There is no central guidance in the version reported here: each agent acts independently.

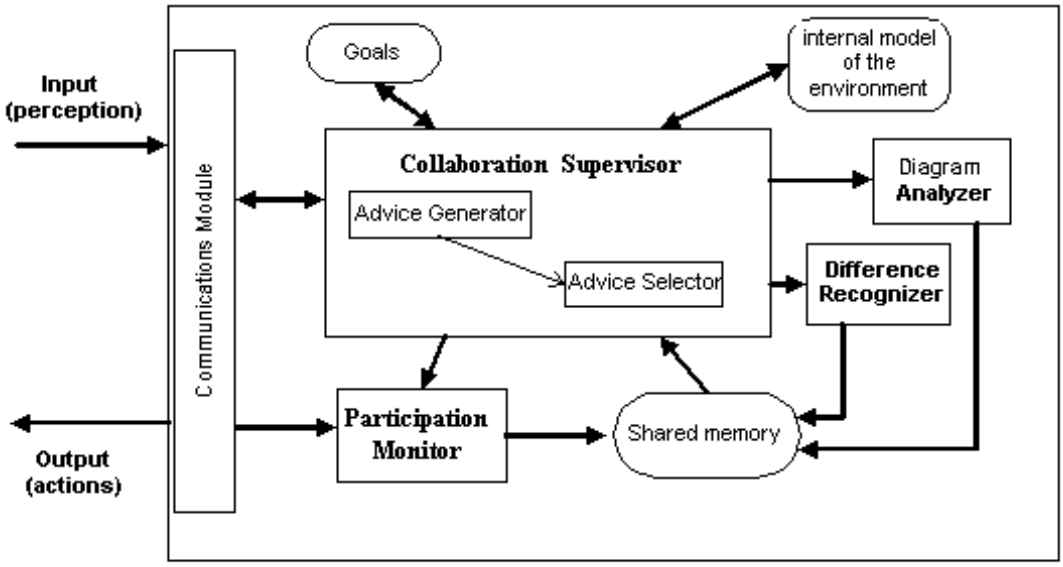

Figure 6. Coach Architecture

The coach involves different modules that cooperate in the solution of the main problem: when and what advice to give (Figure 6). The *Diagram Analyzer* is a simple module that identifies participation opportunities based on the detection of problems in the quality of the ER group diagram. It recognizes some common errors in the group ER diagram based on their structural and categorical characteristics. Detection of these problems was included to see how they could be used to encourage students' participation. The *Difference Recognizer* detects opportunities for students to collaborate by finding significant differences between individual and group ER diagrams. The Difference Recognizer can either find differences specifically related to the currently added object, or find all "extra work" that the student can contribute to the group. The *Participation Monitor* attends to the activity in the group diagram. If nobody has worked in the group diagram for a period of time, it reports this event. It also monitors whether each student is participating too much or too little. The Diagram Analyzer, Difference Recognizer and Participation Monitor communicate their results to the *Collaboration Supervisor* via a shared memory. The Collaboration Supervisor maintains an internal model of the environment, which includes the current group and individual diagrams, "the coached student" and team members' levels of participation, advice types, advice patterns and advice history, current received and given feedback, session phase and session start time. The Collaboration Supervisor operates in two phases: *Advice Generation* and *Advice Selection*. The Advice Generator computes the set of appropriate advice for a given situation using an AND/OR decision tree, while the Advice Selector chooses the most appropriate advice from this set based on control strategies. These processes are described in detail below. The *Communications Module* is in charge of getting real time information from the environment and communicating a coach's advice when it is required. This module enables the reactive capability of the coach.

Each one of COLER's reasoning modules contributes distinct expertise, and is implemented as a Java thread. Coach-applet communication, as well as inter-applet communication, is implemented in two different forms: (1) using static variables stored in the shared working memory and (2) direct communication between applet instances. The types of messages they use in the communication are presented in Figure 7.

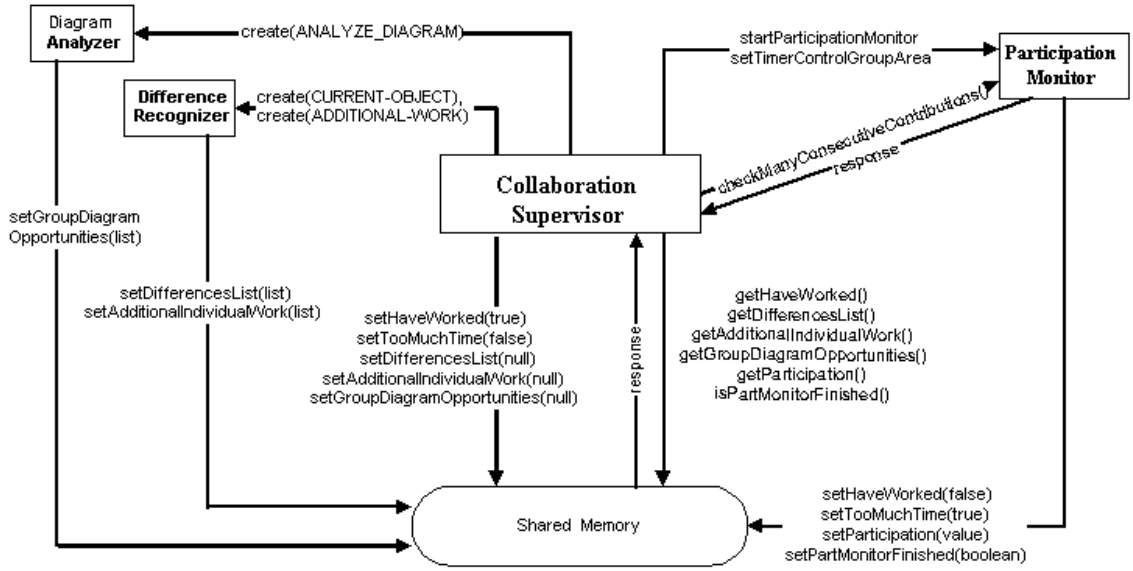

Figure 7. Communication between COLER coach Modules

The reasoning modules have different temporal extents and communicate in different ways. The Collaboration Supervisor is active during the complete group session. It starts the Participation Monitor when an "InitGroupSession" message is received from the Team/FloorControl Applet, and asks it to initialize some variables, such as the coached student, team members, individual and group diagrams, and timer for group work. The Participation Monitor thread stays alive until the session ends, so it is able to respond to specific requests from the Collaboration Supervisor at a given time, such as checking the coached student's consecutive contributions or setting the timer for the control of the group area. In contrast, the Difference Recognizer and the Diagram Analyzer are created only when the Collaboration Supervisor wants to detect specific learning opportunities for discussion or participation. When these modules finish their work, they store their results in a shared memory and are disposed. The Collaboration Supervisor can use this information later. The Participation Monitor also communicates its findings to the Collaboration Supervisor via the shared memory. Findings include whether students have been working on the group diagram (HaveWorked), whether a student has had the pencil too long (TooMuchTime) and students' participation status (various parameters to be discussed below). The shared memory enables the Collaboration Supervisor to check for the occurrence of a specific situation at a given time and to reason about it when such a situation occurs.

COLER works by detecting learning opportunities and deciding whether and how to respond to these opportunities by coaching collaboration. COLER recognizes learning opportunities by (1) evaluating a number of syntactic dissimilarities between individual and group ER diagrams and by (2) tracking participation in the group workspace. COLER coaches collaboration by (3) generating a set of advice and (4) selecting the advice to give based on control strategies. The following four sections describe these different types of knowledge and their use in the coach's reasoning.

RECOGNIZING DIFFERENCES BETWEEN SOLUTIONS

Difference detection, one of the basic points of this work, is carried out by the Difference Recognizer module (see Figures 6 and 7). The detection of conflicts to monitor the state of collaborative interaction has also been considered by Tedesco and Self (2000). Their focus was on detecting meta-cognitive conflicts in structured dialogue-oriented systems. Our focus is on detecting differences in structured representations, such as Entity Relationship (ER) diagrams.

COLER mainly recognizes differences by identifying a number of syntactic dissimilarities between individual and group ER diagrams. Only significant differences are detected, that is, ones that are semantically interesting. In order to have a reasonable basis of comparison between diagrams, a glossary of nouns was defined according to the problem being solved. This kind of knowledge is described below.

Knowledge: Significant Differences and Glossary

The Difference Recognizer uses a set of "Significant differences" that were defined based on four different information sources: (1) domain expert suggestions obtained by personal meetings with our colleague, Dr. Icaza of ITESM; (2) the first author's previous experience in teaching data modeling; (3) analysis of 13 students' solutions for a specific ER problem provided by the domain expert; and (4) common errors in ER modeling reported in the literature (Batra & Antony, 1994; Shanks, 1996). The significant differences considered in this research are presented in Table 1. A weight was assigned to each one of these differences depending on its impact. This weight is used to decide when to give advice.

Table 1: Significant Differences in ER Data Modeling

Difference Type	**Weight**
Missing entity	0.7
Extra entity	0.7
Missing relationship	0.7
Extra relationship	0.7
Missing key attribute	0.6
Difference in cardinality	0.5
Difference in existence	0.5
Attribute in different place	0.45
Missing attribute in relationship	0.4
Extra attribute in relationship	0.4
Missing attribute in entity	0.3
Extra attribute in entity	0.3
Difference in key	0.35
Difference in number of relationships	0.3
Difference in entity type (strong/weak)	0.2
Difference in existence of relationship's name	0.2
Difference in relationship's name	0.1
Attribute not identified as key	0.1

A glossary of terms was defined to enable difference detection. The glossary includes the names of entities, attributes and instances mentioned in the database problem, as well as some names that might correspond to students' mistakes. The glossary database contains the singular form of the nouns used in the problem scenario, their plural form, a synonym, and abbreviation. The singular form of the name is used to match entities and attributes. Relationships' names are not included in the glossary because students might use a large number of different names to indicate the same action, so they are not useful to match objects. Relationships instead are matched using dynamically generated internal names constructed from the associated entities (e.g. Employee+Project). A problem's glossary should be created for each ER problem based on the significant nouns mentioned in the problem description.

Reasoning: Diagram Matching

The Difference Recognizer undertakes subgraph matching between the private and group ER diagrams for the purpose of identifying differences. This matching is made tractable by using

the names in the glossary. The Difference Recognizer consists of three main methods: Compute Entity Differences, Compute Relationship Differences and Find Additional Individual Work[2]. Matching can either find differences specifically related to the currently added object (e.g. missing entity, extra attribute), or find all "extra work" that the student can contribute to the group. For each discrepancy detected, several information items are registered, such as the object involved, the type of difference, the user to whom this difference is against and additional information needed for particular types of differences when it is required. The Difference Recognizer module could be used for other purposes, e.g. to compare an expert diagram with individual and group solutions to detect misconceptions.

Example: Difference Recognition

We now introduce an example that will be used throughout the paper. Consider a three-student team, George, Frank and David, who are using COLER to construct an ER model to solve a research center database problem. George's ER diagram and the diagram being constructed by the group are shown in Figure 8 and Figure 9. Differences found between these two ER diagrams are shown in Table 2. This table indicates the node participating in the difference, the difference description and its weight. Besides the differences between these two existing diagrams, the extra objects George could add are also computed and presented in Table 3.

MONITORING PARTICIPATION

The Participation Monitor (see Figures 6 and 7) attends to the activity in the group diagram. It detects time-triggered events, such as inactivity in the group area or the coached student having control of the group area for a long time (pencil handling). Group diagram events, such as addition of an object to the group diagram, are also detected so it can monitor whether each student is participating too much or too little. The knowledge that this module uses is described below.

Knowledge: Strategic Parameters

Much of COLER's strategic reasoning is controlled by parameters that can be adjusted as needed to suit different problem domains or instructor's preferences. In this section we introduce 6 parameters pertaining to student participation and classified in three categories: Participation Balance (3 parameters), Progress on Task (1 parameter) and Waiting for Feedback (2 parameters). Examples will be provided after the parameters have been described.

Participation Balance

Three parameters control the desired balance in participation (activity in the workspace): MaximumStandardDeviation, MaximumConsecutiveContributions, and MinimumListenAdvice. COLER uses these parameters to monitor group's dynamics concerning the participation balance. The *MaximumStandardDeviation (MSD)* is used to determine the desired level of participation of each student compared with his/her teammates. If the value of this parameter is high, e.g. MSD > 1.4, the coach will encourage students to participate only when there is a large difference in their participation level. On the other hand, if this value is too small (MSD < 0.5), the coach will interrupt students almost after every action they do. *MaximumConsecutiveContributions* indicates the maximum number of consecutive contributions that the coached student can do before COLER suggests that he/she let others participate. *MinimumListenAdvice* indicates the minimum number of "listen" advice (e.g. LO: Listen to Others, LP: Let others Participate) that the coach should use to encourage the coached student let others participate before the coach takes the control of the group area from him/her.

[2] These methods were implemented by Vincent Trifot and Justin Peltier as part of a student project.

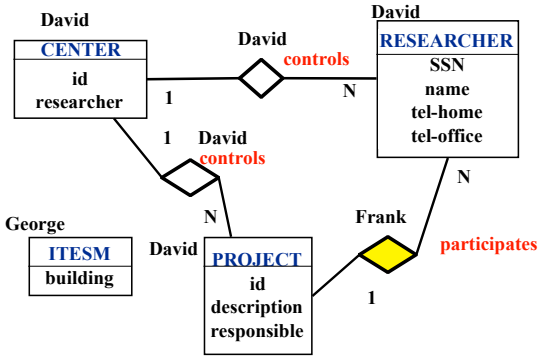

Figure 8: Group Diagram

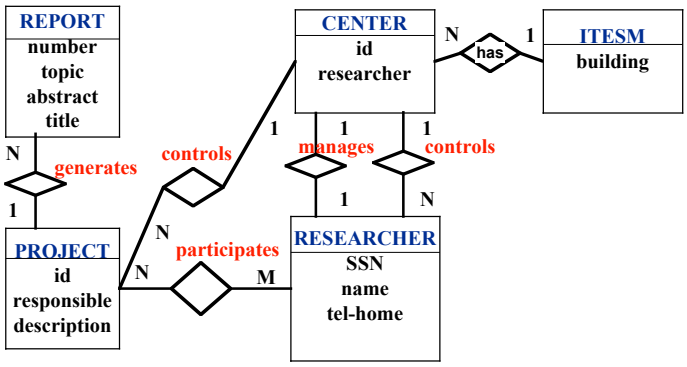

Figure 9: George's Diagram

Table 2: Differences between example ER diagrams

Node	Difference	Weight
RESEARCHER	Missing attribute in entity: tel-office	0.7
CENTER+PROJECT	Difference in relationship's name: controls VS controls	0.1
CENTER+RESEARCHER	Difference in number of relationships	0.3
PROJECT+RESEARCHER	Difference in cardinality: N:M VS 1:N Difference in relationship's name: participate VS participates	0.5 0.1

Table 3: George's Additional Work

Node	Difference	Weight
REPORT	Extra Entity	0.7
PROJECT + REPORT (generates)	Extra Relationship	0.7
CENTER + ITESM (has)	Extra Relationship	0.7
CENTER + RESEARCHER (manages)	Extra Relationship	0.7

Progress on Task

TimeoutNoAction is a parameter that was defined to encourage students to devote adequate time to the task of constructing the shared solution. *TimeOutNoAction* refers to the maximum period of inactivity in the group diagram that COLER waits before suggesting that the coached student take an action in the group workspace. Every time an action is performed in the group diagram (e.g. add, delete, change object), a timeout is set to verify that students are not just chatting for a long time, but are also working on the construction of the group diagram. This parameter should be defined in accordance with the total time assigned to the group session. If TimeOutNoAction is too small, the coach will constantly pressure students to work in the group area, with almost no time to discuss anything. If this value is too large, the student might not realize how the time is going and spend a lot of time chatting without any alert message from COLER.

Waiting for Feedback

Two parameters, TimeoutTeammateAction and TimeoutMyStudentAction, were defined to enable COLER decide whether to say or not something regarding to the use of the opinion buttons. A "Give Feedback" suggestion is considered when a teammate has performed an action in the shared area (add, delete, update), the *TimeoutTeammateAction* time has passed and the coached student has not pressed any opinion button ("OK," "not OK," or "unsure"). When the *TimoutMyStudentAction* time has passed and the coached student has not received any feedback from his/her teammates after his/her contribution, an "Ask For Feedback" suggestion is considered.

Reasoning: Monitoring Participation

The Participation Monitor tracks the coached student's number of specific contributions (SC_i), incrementing the value each time the student adds something to the shared area. In this version of the coach, only the *add object* action to the group diagram is counted as a contribution. Future versions of the Participation Monitor could consider updating and deleting actions as contributions, by assigning them different weights. To evaluate participation, a Standard Deviation (SD) is computed based on students' contributions. If the standard deviation exceeds the threshold of the MaximumStandardDeviation (MSD), explained previously, the monitor assumes there is a problem in participation and individual students are checked:

$$ProblemInParticipation(PIP) = \begin{cases} true & SD > MSD \\ false & otherwise \end{cases}$$

The participation status of each student is computed by the following equation. If the difference between a given student's contributions and the mean exceeds the MaximumStandardDeviation, it is assumed that the student is part of the problem, as follows:

$$ParticipationStatusStudent_i = \begin{cases} TooMuch: & PIP \rightarrow (SC_i - Mean > MSD) \\ NotEnough: & PIP \rightarrow (Mean - SCi > MSD) \\ Acceptable: & otherwise \end{cases}$$

where SC_i = *Number of contributions of Student i in the group diagram*

Time triggered events, in contrast, are not related to a specific event, but they are detected by a processing cycle. In this cycle, the Participation Monitor first initializes the *lastActionTime* and *controlTakenTime* variables to null, to indicate students have not started working in the group diagram and have not taken the pencil. Then it initializes a Contribution table to register each team member's contribution to the group diagram. In each cycle, after waiting some time,

if nobody has worked in the group diagram for the time specified in TimeoutNoAction, the monitor computes the participation status of the coached student and indicates this event by changing the value of the variable *HaveWorked* in the shared memory. A similar process is followed to change the variable *TooMuchTime*, to indicate a student has had the pencil for a long time.

Example: Monitoring Participation

Following the example presented previously, the following parameter values associated with the Participation Monitor were defined:

- MaximumStandardDeviation: 1.2
- MaximumConsecutiveContributions: 3
- MinimumListenAdvice: 3
- TimeoutNoAction: 240,000 milliseconds = 4 minutes
- TimeoutTeammateAction: 60,000 miliseconds = l minute
- TimeoutMyStudentAction: 60,000 miliseconds = 1 minute

The contributions to the group diagram presented in the above example were as follows: David 5, Frank 1, and George 1. Based on these values, George's coach computes the participation's mean and standard deviation:

- Mean = 2.333
- StandardDeviation = 1.8856

Since the StandardDeviation, 1.8856, is greater than the MaximumStandardDeviation (1.2), then, it is concluded that a problem in participation exists (PIP = true). Therefore, George's agent needs to determine whether George, its coached student, is part of the problem. George's agent computes ParticipationStatusStudent for George using the formula given above. His participation status is computed as "NotEnough" since these conditions are satisfied, as shown below:

$ParticipationStatusStudent_{George}$ = "NotEnough," since
PIP ∧ (Mean - SCGeorge > MaximumStandardDeviation)
= True ∧ (2.333 - 1 > 1.2) = True ∧ (1.333 > 1.2) = True

GENERATING ADVICE

The Advice Generation phase of the Collaboration Supervisor Module (see Figure 6) uses event-driven inference to detect learning opportunities for discussion and participation and to generate advice appropriate for those opportunities. An event is evaluated using an AND/OR decision tree. The generation of the advice requires knowledge that is described below, before detailing this event-driven reasoning.

Knowledge: Advice Categories and Strategic Knowledge

The knowledge for advice generation includes the advice itself along with strategic knowledge used to determine when different categories of advice are relevant. This strategic knowledge, such as session phases and discussion encouragement intensity, is an integral part of the advice generation process. A detailed description of the advice categories and types, as well as of the strategic knowledge follows.

Table 4: Coach Advice Types

Advice Category	*Advice Type Abbreviation*	*Advice Type Description*
Discussion	ED	Express Disagreement
	AE	Ask for Explanation
	AJ	Ask for Justification
	GE	Give Explanation
	GJ	Give Justification
	EU	Express Uncertainty
	AA	Analyzing Alternatives
	RA	Reflect with teammates about...
Participation	GC	General Contribution
	SC	Specific Contribution
	CT	Continue working on Task
	GP	Explain, in general, the importance of participation
	LO	Listen to Others
	LP	Let Others Participate
	IP	Invite others to Participate
	LM	Listen to Others, Mandatory
	LC	Ask a teammate to let you contribute
Feedback	AF	Ask for Feedback
	GF	Give Feedback
Self-Reflection	CD	Check Own Discrepancies
ER Modeling	ER	Entity-Relationship Modeling: Connect a disconnected Entity, draw a relationship, add an entity or attribute, define a key.
	RW	Review Work Completeness
Welcome	IW	Individual Welcome
	GW	Group Welcome
Goodbye	IG	Individual Goodbye
	GG	Group Goodbye

Advice Categories and Types

COLER's advice is expressed as suggestions or questions that try to encourage students to discuss and participate. They are not imperative, so students should feel free to follow the advice or discard it when they believe it to be inappropriate. Advice types and categories were defined based on the collaborative learning literature and "Wizard of Oz" studies in which the human expert coached through the chat interface: see Constantino-González (2000).

The present version of COLER (Table 4) includes seven advice *categories*. The first two categories, *Discussion* (in chat) and *Participation* (in the group workspace), are the main categories related to coaching collaboration. *Feedback* messages are related to student's pressing of COLER opinion buttons. The *ER Modeling* category includes suggestions related to some common errors in the domain. The *Self-Reflection* category consists of suggestions to think about a problem or situation. COLER also uses *welcoming* and *goodbye* messages.

Types of advice were defined and classified within these categories (Table 4). For each advice type, several advice *templates* were defined using different wording to provide linguistic variety. The templates can be contextualized by binding variables from the current situation, including the student's name ($MyStudentName), the object's type ($ObjectType), the object's name ($ObjectName), and the problem type. An example template (translated from the Spanish) follows:

$MyStudentName, $ObjectName $ObjectType proposed in the diagram is different from what you've got. If you do not agree with this, you should express and justify your viewpoint.

Collaborative Session Phases

The group session was divided into several phases, according to the progress in the construction of the group diagram (number of objects) and the elapsed time (Figure 10). Depending on the session phase, advice patterns with different semantics are defined, so it is possible to give more appropriate suggestions according to the current phase. During the collaborative session, students have a time limit to work and learn together and to generate a group solution. Eight phases were defined for the group session: "Init," "Waiting," "Ready," "Started," "Middle," "Verification," "FinishingTime" and "End." (These and other constant values will be quoted in the discussion to distinguish them from parameters and variables.)

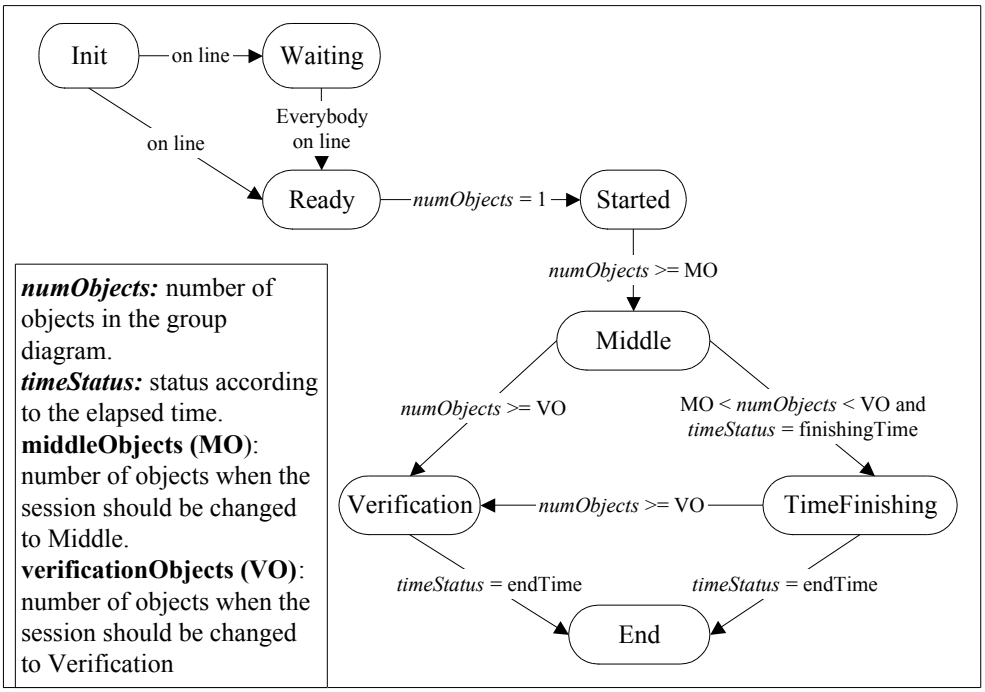

Figure 10: Group Session Phases

The collaborative session begins with the "Init" phase. When the student logs into the collaborative session the phase is either changed to "Ready" if everybody is prepared, or "Waiting" otherwise. The phase is changed from "Waiting" to "Ready" when everybody is on line. Once in the "Ready" phase, the phase is set to "Start" after the first object is added to the group diagram. How the sessions evolve from "Middle," "Verification," and "FinishingTime" to "End" depends on five parameters that should be provided by the professor: two concerned with time and three related to the contents of the group diagram.

Time-related parameters include the time in seconds the collaborative session will last (TimeLimit) and the time in seconds for the reviewing period (TimeToReview), when the session is close to finish. Using these parameters, it is possible to compute the timeStatus, which, as illustrated in Figure 10, is required to compute the session phase:

$$timeStatus = \begin{cases} WorkingTime: & PastTime \leq TimeToReview \\ FinishingTime: & TimeToReview < pastTime < TimeLimit \\ EndTime: & PastTime \geq TimeLimit \end{cases}$$

where

pastTime = CurrentTime − InitTime

InitTime = Session starting time

CurrentTime = time when the formula is computed

Group diagram parameters include the expected number of objects of the problem group solution (TotalObjects) and diagram construction milestones (MiddlePercentage and VerificationPercentage). These latter indicate progress percentage in diagram construction. MiddlePercentage (MP) indicates the percentage of TotalObjects when the session should be changed to "Middle." VerificationPercentage (VP) indicates the percentage of TotalObjects when the session phase should be changed to "Verification." Default values for these two parameters are 30% and 90% respectively. These percentages are used to compute the middleObjects and verificationObjects variables, which indicate the number of objects when the session phase should be changed to "Middle" and "Verification" respectively.

Collaborative Session Phase Example

Continuing the description of the previous example the timeStatus and session phase are computed using the values of the session phase parameters:

- TotalObjects (expected number of objects in the problem solution): 10
- TimeLimit : 90 minutes
- TimeToReview: 80 minutes
- MiddlePercentage: 30%
- VerificationPercentage: 90%
- pastTime: 50 minutes
- numObjects (total number of objects in the current group diagram): 7.

First, the timeStatus is determined as follows:

- Since pastTime < TimeToReview, timeStatus = "WorkingTime"

Next, the middleObjects (MO) and the verificationObjects (VO) are determined as follows:

- middleObjects = round (MiddlePercentage * TotalObjects) = round(30%*10) = 3
- verificationObjects = round (VerificationPercentage * TotalObjects) = round(90%*10) = 9

In this specific example, the session state changes to "Middle" when the group diagram has 3 objects. It changes to "Verification" when the group diagram contains 9 objects.

Considering that the value for timeStatus is "WorkingTime" and that the group diagram is not finished (middleObjects < numObjects < verificationObjects), then the value computed for session is "Middle". This value is used to select the set of relevant advice for generation, as described in the next section.

Discussion Encouragement Intensity

Advice is not necessarily given every time a difference is found, since not all differences have the same importance. COLER uses three parameters to define the Discussion Encouragement Intensity, i.e., the extent to which the coach encourages students to discuss their differences: ThresholdImportantDifference (TID), ThresholdHighTotalWeight (THTW) and ThresholdMediumTotalWeight (TMTW). They are considered in the decision of when to interrupt students for discussion. The first one, ThresholdImportantDifference, is used when a single difference exists. It indicates the value when a single difference is important. Its value

ranges from 0 to 1. The last two, ThresholdHighTotalWeight and Threshold MediumTotalWeight, are applied for multiple differences. *ThresholdHighTotalWeight* indicates the sum of several differences that could be considered as high. *ThresholdMediumTotalWeight* indicates the sum of several differences that could be considered as of medium importance. These values should be greater than zero. The larger theses values, the less often discussion is encouraged. The values defined for these parameters should be in concordance with the weights defined for each type of difference in the model analyzed. The importance of multiple differences (totalWeightImportance) can be computed using the following equation:

$$totalWeightImportance = \begin{cases} Significant: & TW \geq THTW \\ Medium: & TMTW < TW < THTW \\ Low: & TW \leq TMTW \end{cases}$$

where

$$TW = \sum_{i=1}^{n} Difference[i].weight$$

$$n = Number\ of\ differences\ found\ at\ a\ given\ time$$

This value is used by the Advice Generator phase of the Collaboration Supervisor to decide whether it is appropriate to give discussion advice.

Discussion Encouragement Intensity Example

To compute Discussion Encouragement Intensity in the previous example, the following values were defined:

- TID (Threshold Important Difference): 0.6
- THTW (Threshold High Total Weight): 0.6
- TMTW (Threshold Medium Total Weight): 0.3

When the current object (Project+Researcher:participates) was added by Frank, the following differences are computed by the Difference Recognizer:

Number	Differences	Weight
1	Difference in cardinality: N:M VS 1:N	0.5
2	Difference in relationship's name: participate VS participates	0.1

Then TW (totalWeight) was computed as follows:

TW = difference[1].weight + difference[2].weight = 0.5 + 0.1 = 0.6

Therefore, since TW >= THTW, it is concluded that

totalWeightImportance = “Significant”.

This value is used by the Advice Generator module of the coach, as explained in the following section.

Reasoning: Event-Driven Application of AND/OR Decision Trees

The Advice Generation phase of the Collaboration Supervisor module uses event-driven reasoning. Three main types of events are attended: (a) time-triggered events, such as inactivity in the group diagram, (b) group and individual diagram events, for example, the addition, change or removal of an object, and (c) voting events, such as the receiving and giving of feedback. Given an event, the Collaboration Supervisor identifies the event type, analyzes the

situation and then decides what kinds of advice to give. The reasoning for each event uses an AND/OR decision tree defined for this event, as illustrated in Figure 11. A more detailed description of how these events are managed is given below.

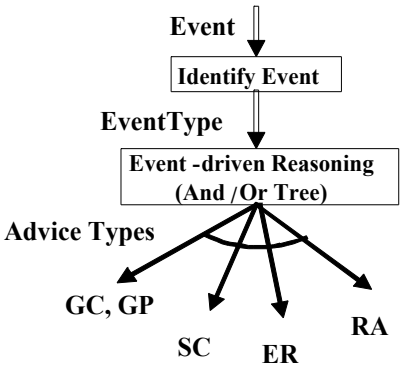

Figure 11. Advice Generation

Time-Triggered Events

Time-triggered events are followed by the processing cycle shown in Figure 12. At the beginning of the session, the Collaboration Supervisor initializes the coach's internal model by setting values for initial data such as individual and group diagrams, category preferences, advice types, advice patterns, session phase and session start time. In each cycle, reviewPhaseSession determines the current phase. If this phase corresponds to a group-learning phase (e.g. "Ready," "Started," "Middle," "Verification," "FinishingTime"), the Collaboration Supervisor checks students' activity in the group area by accessing the corresponding variable in the shared memory set by the Participation Monitor. If there has been inactivity in the group diagram for a long time, then the Collaboration Supervisor, using the AND/OR tree for this event, decides what advice to give. This cycle stops when the time available for the group session ends, that is when the session phase is End.

```
Initialize-State();
repeat
sleep(t);
reviewPhaseSession();
if (isGroupLearningTime()) {
        checkHaveWorked();
} else if (phase == END_SESSION))
         finishSession();
end repeat
```

Figure 12. Collaboration Supervisor's Processing Cycle for Time-triggered events

Diagram Events

Diagram events are generated when the coached student works on his/her individual diagram or when he/she or a teammate works on the group diagram. When an object is added to the group diagram, the coach updates its internal state by setting a timer for feedback and updating several elements such as the corresponding diagram, the current object being analyzed and the student executor of the action. Then, the AND/OR decision tree for this event is followed. When an object is deleted or updated, the current version of the coach does not do any reasoning. It only updates the corresponding diagram and the current node being analyzed. The same occurs when these actions are made in the individual diagram.

Voting Events

Voting events include those actions related with the opinion panel. When a feedback is given or received, the applet in charge of feedback monitoring asks the coach to check this action. The Collaboration Supervisor then analyzes it and decides what advice to give if one is required.

AND/OR Decision Trees

An AND/OR decision tree is associated with each event type. This tree generates different advice types given an event type. Every branch of the tree represents a possible set of suggested advice. Several suggestions might be generated for any given event because several leaves may be reached at once via the "and" arc of the tree. Also, many of the leaves of each tree generate multiple advice, and trees for different events may be invoked at the same time. An advice selector will later choose the most appropriate advice from the ones generated.

The AND/OR decision tree describing the coach's reasoning for the "Node added in group diagram" event is shown in Figure 13. Other AND/OR decision trees for other events such as "Inactivity in group diagram" may be found in Constantino-González (2000). The tree shown has two main "or" branches, depending on who performed the action.

If a teammate added the object (left hand branch of Figure 13), processes to evaluate (1) participation, (2) feedback, and (3) discussion are executed (branches connected by an "and" arc):

(1) If the Participation Monitor indicates that the coached student has not participated, the coach generates advice shown in the leftmost sub-tree: General Contribution (GC), General Participation (GP), Reflection About an issue (RA) and Specific Contribution (SC). Specific Contribution suggestions are computed according to extra individual work done by the coached student and identified by the Difference Recognizer. ER Modeling suggestions correspond to syntax-based problems found in the group diagram by the Diagram Analyzer. They are generated according to a sub-tree, not shown, attached at the @ in the left branch of Figure 13.

(2) Feedback evaluation simply checks whether the student has selected one of the **OK**/**NOT**/**?** buttons.

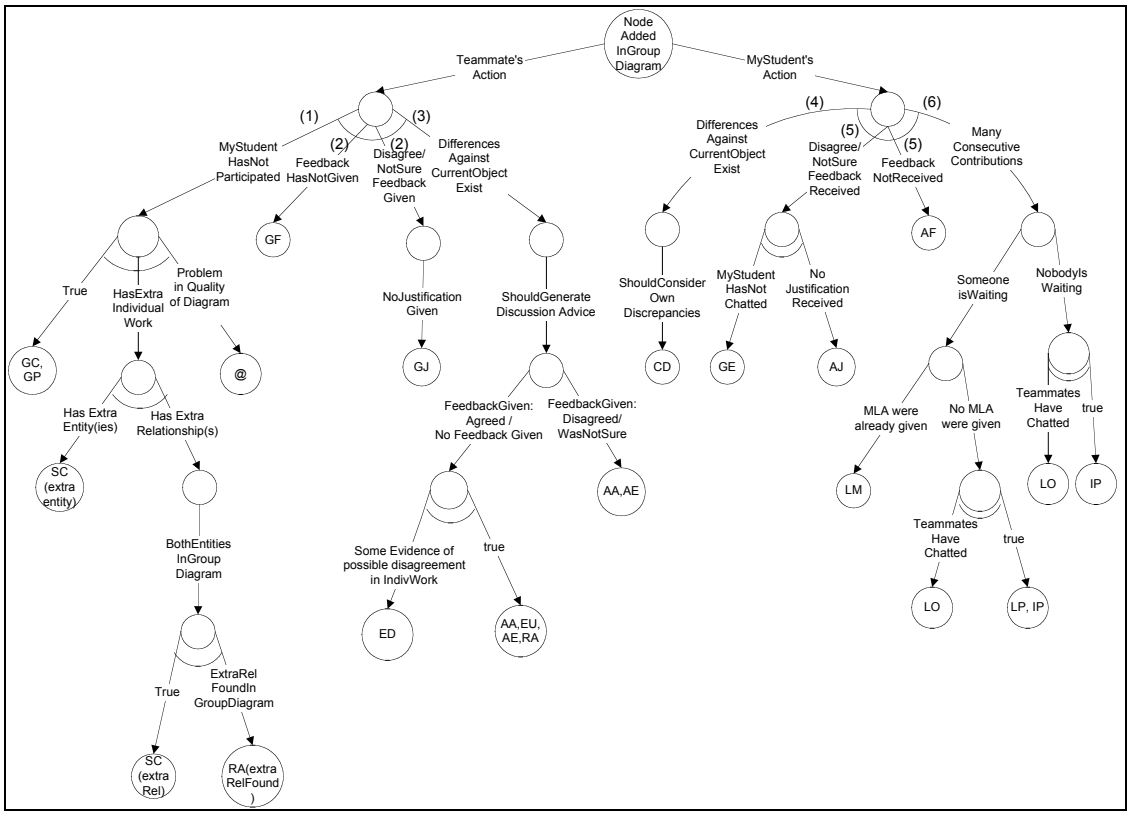

Figure 13. AND/OR Decision Tree for the "Node added in Group Diagram" event

(3) The discussion evaluation process takes into account the number and type of differences found by the Difference Recognizer and the student's participation computed by the Participation Monitor. Discussion suggestions are generated by the evaluation of the differences found, as shown in the large subtree in the center of the figure.

If the coached student was the one who added the object (right hand branch of Figure 13), processes to evaluate (4) discrepancy checking, (5) received feedback and (6) participation are executed:

(4) A suggestion to Check Discrepancies (CD) is generated based on an evaluation of the differences found.

(5) If opinions from others (OK, NOT, ?) have not been received, then Ask for Feedback (AF) is generated.

(6) If the Participation Monitor indicates that the coached student has done many consecutive contributions, it generates advice from the following types: Listen to Others (LO), Let Participate (LP), and Invite others to Participate (IP).

The diagram in Figure 13 also shows different situations in which chat information is considered to decide the kind of advice to give (e.g. LO, IP, GE). This information is specifically required in the process to decide whether discussion advice should be generated when differences against the current object exist (3). Both the coached student's chat messages and his/her teammates' messages are considered. Chat information includes who has chatted and the chat messages' length. See Constantino-González (2000) for further details of the generation process.

The AND/OR decision trees used by the Advice Generator were represented directly in Java code. A method was defined for each specific type of event such as adding a node, giving/receiving feedback or not working for a period.

Example: AND/OR Decision Tree

Following the description of the previous example, the coach has already given several messages that are saved in an advice history area: IndividualWelcome, GroupWelcome, GeneralParticipation and AnalyzeAlternatives. An event has just occurred: Frank, George's teammate, has just added the Project+Researcher relationship (participates) to the group diagram (shaded diamond, Figure 8). Even though George had some differences, he pressed the OK button to indicate his agreement.

The Coach's reasoning is based on the AND/OR decision tree presented in Figure 13. Since the action was performed by a teammate (Frank), the left hand side of the tree is analyzed: (1) George's participation status is computed based on all the students' contributions. As shown previously, it is concluded that there is a problem in participation and the coached student (George), who has one contribution, has not participated enough. Therefore, this branch of the tree is analyzed and the following advice types are generated:

- GeneralContribution
- GeneralParticipation
- SpecificContribution(Report)
- SpecificContribution(Center+ITESM:has)
- SpecificContribution(Center+Researcher:manages)
- EntityRelationship(ITESM disconnected)
- EntityRelationship(Center,key)

From the SpecificContribution types (SC), an algorithm selects the two relationships as the best ones. The SC(Center+ITESM:has) is randomly selected from the two relationships (e.g. Center+ITESM:has, Center+Researcher:manages). (2) Since George has pressed the OK button the feedback branches are not executed. (3) Differences are evaluated by comparing George's and the group's diagrams. As a result, as explained before, the TW (TotalWeight) of the

differences is 0.6. Since TW >=THTW, it is concluded that the coach should generate advice from the discussion advice category. Then, considering that the feedback given was OK and there was no extra evidence of disagreement, the advice types generated are AskForExplanation, AnalyzeAlternatives, ExpressUncertainty and ReflectAbout. Once this set of advice has been generated, one must be selected. This process is explained below.

SELECTING ADVICE

Advice Selection is the second phase of the Collaboration Supervisor Module (see Figure 6). Advice selection is sensitive to the relative importance of different forms of advice and to the context of the advice (problem solving and advice giving history). Six *control strategies* were specified to control selection and timing of advice: Preferences, Collaborative Session Phases, Discussion Encouragement Intensity, Participation Balance, Time on Task and Waiting for Feedback. Most of these strategies, except the first one, are employed to filter advice during the generation process and have already been described in previous sections as strategic knowledge. The control strategy that utilizes *Advice Preferences* in the selection process is described below.

Knowledge: Advice Preferences

Preferences are used to select the advice to give from among the candidate generated advice. A *preference* is a predicate that compares two proposed advice and chooses one as being preferable to the other in importance. Three kinds of preferences were defined: New Advice (don't repeat advice type during the session), Many Instances (prefer advice of a type that applies more than once) and Category preferences (e.g., for Discussion, Participation, or Feedback advice). When the session starts default values are assigned to these preferences, such as New (1), Many (2), and Category (3). The default order for category preferences is: Feedback (F), Discussion (D), Participation (P), Self-Reflection (SR) and ER Modeling (ER). This indicates that at the beginning suggestions to discuss are preferable to suggestions to participate. However, the preferences in use can change during the group session according to the group's performance.

Reasoning: Advice Selection

The advice selection process involves two steps. First, select an advice pattern from the advice types generated by each one of the leaves. Second, select an advice from the resultant advice set.

The selection from the advice types generated by each leaf is implemented as follows. First, for each advice type indicated by the AND/OR tree's leaf, a list of advice instances is generated according to the session phase and the advice templates defined for this advice type. If the leaf includes several types of advice, the one that has been used just previously is eliminated, implementing a preference to avoid giving the same kind of advice twice. If the leaf contains only one advice type and this is the same as the last advice type given, this advice type is selected only if a no-advice limit has been reached. This limit is defined by the parameter MaximumNoAdvice, which represents the number of times that an advice could have been given but was not because its type was the same as the type of the last advice given. After this verification is done, an advice instance is randomly chosen from the list for each leaf node and then contextualized by binding variables from the situation (e.g., $MyStudentName, $ObjectType, and $ObjectName). Random selection from the list provides advice variety.

The second step, selection from the advice generated by all of the leaves, starts with a revision of the category preferences, which may change depending on the group's performance. In the current version, only discussion and workspace participation preferences are interchanged. If the group seems to need more participation advice, this category of advice is promoted. Otherwise, discussion is encouraged. Finally, a preference-based sort algorithm (Suthers, 1993) is run if needed to choose between multiple advice instances. This sort splits the

advice into more preferred and less preferred based on the first preference, and does this recursively on each partition with the remaining preferences, concatenating the results to yield a sorted list. From the sorted list, the coach gives the more preferred advice. The others are stored in a list to be given on demand.

Example: Application of Preferences

Recall that instances of advice types AnalyzeAlternatives, AskForExplanations, EntityRelationship(Center, key), EntityRelationship(ITESM disconnected), ExpressUncertainty, GeneralContribution, GeneralParticipation, ReflectAbout, and SpecificContribution(Center+ITESM:has) were generated in our example. As described previously, advice selection starts by eliminating instances of the type of the last advice given (e.g. AnalyzeAlternatives in this case), and then randomly selecting one advice type from each leaf. Therefore, the result is as follows: ER: EntityRelationship(Center, key), GP: GeneralParticipation, RA: ReflectAbout, and SC: SpecificContribution(Center+ITESM:has). Examples of the advice patterns are shown below. Discussion advices (e.g. RA) are generated in a general context, without mentioning the kind of difference detected.

> GP: George, participation is a learning opportunity. I suggest that you leverage it. Come on, participate! :)
>
> SC: George, you could share your work with your teammates by adding CENTER+ITESM relationship to the diagram.
>
> ER: George, you could define the key of the CENTER entity.
>
> RA: George, PROJECT+RESEARCHER relationship has been just added by Frank. What do you think about it? Is it correct? I suggest that you discuss it with your teammates.

Before this list is sorted according to the preferences, the ordering of category preferences are evaluated to account for the current group dynamics. Since there is a problem in participation, this kind of advice is preferred. Therefore, the Participation and Discussion preferences are interchanged resulting in the following order: Feedback, Participation, Discussion, SelfReflection and EntityRelationship.

Finally, the preferences-based sort is applied considering COLER preferences (New, Many and Category). According to the "New" preference, the advice is partitioned into (SC, ER, RA) and (GP) since GP is in the advice history. Then, the "Many" preference is applied without producing any difference. Finally, these sublists are partitioned based on the ordered set of "Category" preferences, according to the order: Feedback, Participation, Discussion, SelfReflection, and EntityRelationship. As a result, concatenating all the sublists, the following order is produced: SC, RA, ER, GP. The coach then gives the SC advice:

> SC: George, you could share your work with your teammates by adding CENTER+ITESM relationship to the diagram

The rest of advice patterns generated are stored for future use, being available for advice on demand. They would be given in the following order:

> RA: George, PROJECT+RESEARCHER relationship has been just added by Frank. What do you think about it? Is it correct? I suggest that you discuss it with your teammates.
>
> ER: George, you could define the key of the CENTER entity.
>
> GP: George, participation is a learning opportunity. I suggest that you leverage it. Come on, participate! :)

EVALUATION

The evaluation reported here assesses the quality of advice generation and selection algorithms, and the contributions of knowledge sources in the generation of reasonable advice. Future publications will report detailed evaluation of the relationship between group functioning and COLER's advice.

Summary of Method and Procedure

This laboratory evaluation of COLER involved participants who had taken or were taking a database course. Our domain expert, a computer science professor, was also present in two sessions. A pilot session was run to test COLER's usability and functionality. Then, five sessions were conducted to generate data and scenarios for the different types of evaluations. In each of these sessions, three students were presented with a simple database design problem. They first solved the problem individually, and then convened to construct a group solution. Students and coach activities were recorded in a log file. The pilot study and the two sessions in which the Expert was present were used for preliminary evaluation, detecting some problems in COLER's user interface and coach algorithms. The last three sessions, in which the expert was not present, were used to evaluate COLER's algorithms and the quality of its advice, as described below.

Documents for Expert Evaluation

For each student of each of the last three sessions, two documents were generated for the Expert's Evaluation: the Environment Document and the Advice Document. These documents describe the chronological sequence of events of the collaborative session in reference to a specific student, and the context of each event (current state of the environment). The *Environment document* provided the expert with the same information that the computer coach had during the collaborative session (e.g. group diagram, event type, voting response, contributors of chat messages). After each event, a space was left for the Expert to indicate the advice he would give, if any. The *Advice document* was used to evaluate COLER's algorithms and advice acceptability. For each event, the expert was asked to rank the suitability of all of COLER's advice types, and to indicate a cutoff of which advice was "Worth saying," "So-So" and "Not Worth Saying," all without knowing which COLER actually considered. Then a new page showed and solicited comments on the advice types the coach actually generated, and on the advice type selected. Subsequently, each student's individual diagram and the chat transcript of the collaborative session were printed and given to the human expert to evaluate whether his advice would change if he could see more than the coach did. The advice generation algorithm was evaluated by comparing COLER's generated advice to those generated by the expert, as well as through the expert's ranking of all advice available to COLER. The advice selection algorithm was evaluated by comparing COLER's ranking to the expert's ranking.

Results

A number of advice instances were generated from each advice category: 34 were participation, 23 Discussion, 6 Self-Reflection/Discrepancy and 9 Feedback advice. Participation and Discussion advice were given the most. Although Participation advice was the more used, Discussion advice was also important since the participation advice "continue task" is usually given to all group members at a similar time while discussion advice are usually given in different situations.

Coverage of COLER's Knowledge Sources

The *overall knowledge available to COLER* was evaluated by comparing expert and COLER advice for each situation, with 67% of the advice given by the expert not given by the coach

(Figure 14). Thus, as expected the expert has a greater repertoire of advice, although COLER's limited knowledge sources produced the same advice as the expert in 33% of the situations. Figure 15 depicts the distribution of the missing advice: 69% would require new advice types and new branches in the AND/OR decision tree (New), 21% involved situations already considered in the AND/OR tree but requiring that new advice types be attached to them (Considered), and 10% involved advice that COLER could give with minor adjustments to parameters (Parametric).

According to the results, some existing advice types need to be extended to mention a specific context, such as suggesting that students reflect on a specific difference or inviting someone in particular to participate. The findings also suggested situations in which a new "Self-Reflection" advice type could be given.

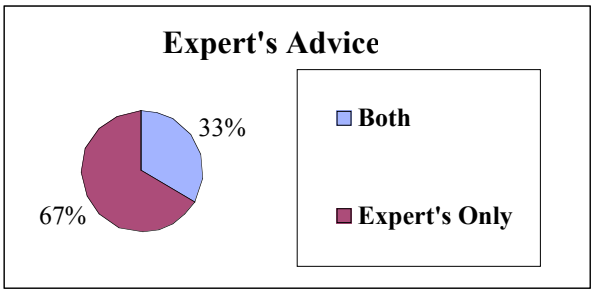

Figure 14. Expert's Advice, Overall Results

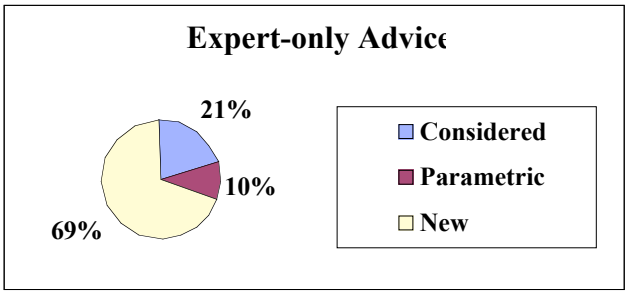

Figure 15. Expert-only Advice

A new category of advice, "Social Interaction," could be included to establish a closer relationship between COLER coach and the student. This category could include different advice types such as thanking the student for listening to advice, and otherwise commenting on student actions. However, these kinds of messages are not closely related to the main objective of the coach. Rather the coach is trying to assist students by helping them to visualize differences and encouraging them to leverage learning opportunities provided by the online group.

Quality of Generated Advice

Advice Generation was evaluated by using the expert's classification of the advice available to COLER into "Worth saying," "So-So" and "Not Worth Saying." Results showed that 73% of the advice generated by COLER was worth saying, 7% was "so-so" and 20% was not worth saying (Figure 16). Some reasons for "Not Worth Saying" advice are change in conditions making the advice obsolete and failure to match entities due to spelling errors and unidentified

synonyms. Reviewing the conditions for each specific advice type before giving the advice could solve the obsolescence problem. Spelling errors could be managed by diminishing the importance of differences in relationship's names for generating "Check Discrepancy" advice, or by using a distance metric between the spellings. Also, some advice, such as "Analyze Alternatives" and ER related advice were given in different situations than the Expert, so they should be reviewed with the Expert in order to modify the corresponding AND/OR decision tree. Results also indicated the need to define a new parameter that specifies the time the coach should wait before suggesting that the coached student give an explanation when a NOT or ? (not sure) feedback has been received and the coached student has not given any response.

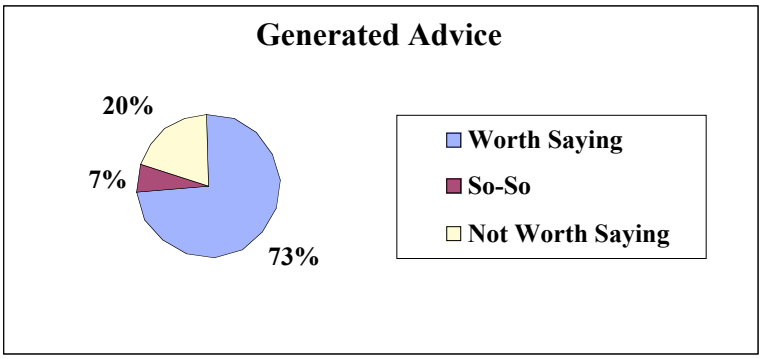

Figure 16. Generated Advice

Quality of Advice Ranking

The *Advice Selection* module was evaluated by analyzing events in which (1) several candidate advice with different rankings exist, and (2) some advice was suggested by the Expert. To evaluate the selection algorithm independently of the generation algorithm, a ranking of the candidate advice generated by COLER for each situation was computed based on the Expert ranking of all advice with respect to that situation. The disparity between COLER's and the Expert's ranking of this generated subset was measured using the Euclidian distance between the individual ranks:

$$d_{CE} = \sqrt{\sum_{i=1}^{n} (x_{Ci} - x_{Ei})^2}$$

The COLER-Expert Euclidian distance d_{CE} is the square root of the sum of squared differences across a set of advice types, where n = the number of advice types generated by COLER, x_{Ci} is the value of COLER's rank for the i^{th} advice type and x_{Ei} is the value of the Expert's rank (of the generated subset) for the same advice type. The COLER-Expert Euclidian distance is not a measure of COLER's advice quality, but a measure of the precision of the ranking assigned by the Selection Algorithm to COLER's advice types. Selection among several advice types was needed only a few times in this study. There were 2.33 average selections per session, each selecting between an average of 4.78 generated advice items. The Euclidian distance obtained was 0.9, i.e., less than one disagreement in ranking per selection event. This seems adequate although leaving room for minor improvements.

The limited number of selections per session is partially due to the fact that the advice generation algorithm randomly selected an instance from the advice instances generated by each leaf of the AND/OR tree. In retrospect we believe that a better design would allow the advice selection preferences to consider all of the advice instances generated by the leaves, and apply

random selection only when the set of most preferred advice instances contains more than one element.

Role of Knowledge Sources

The contribution of each knowledge source in the generation of reasonable collaboration advice was evaluated by "ablating" it analytically, i.e., identifying the advice that relied on the knowledge source and hence would be lost if the knowledge source were removed. We focused on the advice that the expert ranked as "reasonable." This analysis used the Environment and Advice documents to identify the situations in which COLER gave advice and the rank the expert assigned to this advice, and the AND/OR trees to identify the type of knowledge used in each situation.

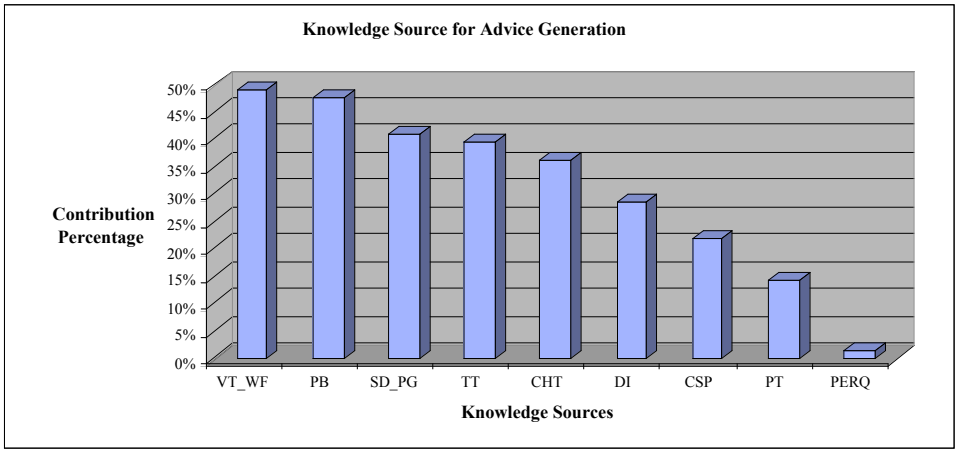

Figure 17. Knowledge Contribution for Generation of Collaborative Advice

Figure 17 shows the contribution of knowledge sources to generation of advice judged by the expert to be "reasonable" as follows: VT_WF: Voting Tracking and Waiting for Feedback timeout (49%), PB: Participation Balance (48%), SD_PG: Significant Differences and Problem Glossary (41%), TT: Time on Task (40%), CHT: Chat Tracking (37%), DI: Discussion Encouragement Intensity (29%), CSP: Category and Sort Preferences (22%), PT: Pencil Tracking (14%) and PERQ: Problems with ER diagram Quality (2%). Some knowledge sources were used to generate different categories of advice (hence the percentages reported above sum to greater than 100) while others were more marginal and only were used in a specific advice category. For instance, knowledge of significant differences and glossary is used to produce Discussion, Participation and Self-Reflection advice, while knowledge of participation balance, time on task, pencil tracking and problems with ER diagram quality are used to generate only Participation advice. Similarly, the generation of reasonable advice (e.g. Discussion and Participation) in this study required the conjunction of several types of knowledge (e.g. Significant Differences and Problem Glossary, Participation Balance) and confirmed the hypothesis that knowledge on problem solving activity could be used to generate reasonable collaboration advice. The knowledge about problems in quality of ER diagrams was used very little in this study since the coach's primary goal is promote discussion and participation instead of teaching ER modeling.

CONCLUSIONS AND FUTURE WORK

This work is part of a research agenda that seeks to characterize the knowledge needed to facilitate collaborative learning processes. The present focus has been on determining how much leverage can be obtained by a basic ability to detect semantically interesting differences

between representations of two problem solutions, together with simple tracking of individuals' levels of participation (e.g. contributions in the shared Workspace) and feedback given (e.g. opinion buttons). The study showed that reasonable collaboration advice can be generated without the need for expert solutions or discourse understanding, although the addition of these knowledge sources would improve the quality and range of advice generated and selected by the system (at the cost of additional knowledge engineering and system complexity). Specifically, 73% of the advice given was considered to be reasonable by the expert, and 33% of COLER's advice was the same as that which the expert would give. (It would be interesting to compare this to the level of agreement between two human coaches.) These results indicate that COLER is a viable advisor, albeit different in style from our expert. Response time should also be considered: our expert pointed out that he came up with his advice after careful and time-consuming study of the Environment and Advice documents while COLER generated advice based on the same information in real time.

Although COLER is presently limited to Entity Relationship modeling, its long-term goal is to support other domains in which problem solutions can be expressed in structured representations that can be compared to identify significant differences, such as finite automata modeling, object oriented modeling and systems dynamics. The following process would be required to apply the coach's design to other areas: (1) identifying how diagram objects can be compared (e.g. lexical units in finite automata), (2) identifying the significant differences in the domain (e.g. missing link, extra link, missing a node, extra node), (3) modifying the Difference Recognizer accordingly, (4) adjusting the differences' weights according to their relevance, (5) reviewing participation and discussion parameter settings and (6) adjusting minor portions of the Advice Generator. Several of the coach's modules can be reused. Reusable modules include the Participation Monitor, the Advice Selector and a fraction of the Advice Generator, such as the AND/OR decision trees for the "Have not worked" event, and a section of the "Node added to the group diagram" event. However, some of COLER's reasoning modules should be developed for each domain, such as the Difference Recognizer, the Diagram Analyzer and a small fraction of the Advice Generator.

A secondary contribution of this work is to illustrate several useful evaluation methodologies, including (1) evaluation of overall knowledge by comparing freely chosen expert advice to advice generated by the system; (2) evaluation of advice generation and selection by comparing generated and selected advice to an expert ranking of all advice available to the system; (3) evaluation of advice selection by Euclidean distance between expert and coach rankings; and (4) evaluation of the contribution of each knowledge source by analytical ablation. Other evaluations reported elsewhere (Constantino-González & Suthers, 2001) or underway include student opinions, whether the advice was timed appropriately for the group's collaborative activity, and whether students take the advice given. Overall, a mixture of empirical and analytic methodologies is advocated to fully understand the design of complex systems.

One limitation of this research is that only one expert evaluated the system. Therefore, we plan to conduct more experiments in which several experts can evaluate the system, so a more precise value of the coach's advice accuracy and adequacy can be obtained.

In this work, a personal assistant in each client viewed a given student's private workspace and the shared workspace in order to help prevent missed opportunities for collaborative learning. Future work may investigate a single *global* coach endowed with the ability to inspect all students' private workspaces as well as the shared workspace. Such a coach would be able to identify conflicts between solutions in private workspaces and encourage the students to share the relevant part of their solutions, thereby *creating* conflict opportunities for collaborative learning. Little additional knowledge engineering would be required.

This work demonstrates the possibility of coordinating distributed agents via their shared situation without explicit negotiation. Personal agents can then provide complementary advices. For instance, while a personal coach says "<student>, why don't you invite someone else to contribute?" to the student who has participated too much, another personal coach would say "<student>, contribute to the construction of the group diagram" to the one who has not

participated. It would be interesting to investigate how much coordination is possible by watching the same knowledge sources without explicit negotiation.

A greater investment in domain-specific knowledge would enable the coach to compare student solutions to expert solutions in order to (a) guide advice selection (e.g., encourage students to share solutions that are correct), and (b) comment directly on the correctness of solutions in the manner of traditional intelligent tutoring systems or Web-based adaptive systems, such as ELM-ART, an intelligent interactive educational system to support learning programming in LISP presented by Weber & Brusilovsky (2001) or the German tutor presented by Heift & Nicholson (2001). Other extensions that use models of discourse and natural language understanding to track the extent to which students are discussing and resolving differences in the chat medium can be envisioned.

ACKNOWLEDGMENTS

We thank our colleagues at the Learning Research and Development Center of the University of Pittsburgh for their support, in particular Alan Lesgold for hosting the first author as visiting scholar and employing the second as his research associate, and Sandy Katz, Arlene Weiner and Eva Toth for discussions about collaborative learning. We also thank Jose I. Icaza of ITESM, our domain expert, Moraima Campbell of ITESM for her support to conduct the formative studies, visiting students Vincent Trifot and Justin Peltier for implementing part of the personal coach, and all of the students who participated in the lab studies. The Center for Artificial Intelligence of ITESM provided research facilities. During this research, Ma. de los Angeles Constantino-González was funded by ITESM Campus Laguna and by CONACYT, and Daniel D. Suthers was funded by the Presidential Technology Initiative (while at LRDC) and by NSF's Learning and Intelligent Systems (while at the University of Hawai`i). José G. Escamilla de los Santos was affiliated with the Center for Artificial Intelligence of ITESM Campus Monterrey.

REFERENCES

Ayala, G. and Yano, Y. (1996*)*. Learner models for supporting awareness and collaboration in a CSCL environment. *Proceedings of the Third International Conference of Intelligent Tutoring Systems (ITS'96)* Montreal, Canada, 158-167.

Baker, M.J. and Bielaczyc, K. (1995). Missed Opportunities for Learning in Collaborative Problem-solving Interactions. In Greer, J. (Ed.) *Proceedings of the AI-ED 95–World Conference on Artificial Intelligence in Education,* Washington, DC, August 16-19. Association for the Advancement of Computing in Education, 210–217.

Baker, M.J. and Lund, K. (1996). Flexibly structuring the interaction in a CSCL environment. In P. Brna, A. Paiva & J. Self (Eds), *Proceedings of the European Conference on Artificial Intelligence in Education.* Lisbon, Portugal, Sept. 20 - Oct. 2, 401-407.

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

Batini, C., Ceri, S. and Navathe, S. B. (1992). *Conceptual Database Design: An Entity-Relationship Approach.* Benjamin/Cummings, Redwood City, California.

Batra, D. and Antony, S. R. (1994). Novice Errors in Conceptual Database Design*. European Journal of Information Systems*, 3(1), 57-69.

Bruce, T.A. (1992). *Designing Quality Databases with IDEF1X Information Models.* Dorset House, New York.

Burton, R. R. and Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems,* Academic Press, pp. 79-98.

Chen, P. (1976). The Entity-Relationship Model - Toward a Unified View of Data, *ACM Transactions on Database Systems,* 1(1), 9-36.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator*, *Winter*, 6-46.

Constantino-González, M. A. (2000). *A Computer Coach to Support Collaboration in a Web-based Synchronous Collaborative Learning Environment*. Unpublished dissertation, ITESM (Instituto Tecnológico y de Estudios Superiores de Monterrey), México.

Constantino-González, M. A., and Suthers, D. D. (2001). Coaching Collaboration by Comparing Solutions and Tracking Participation. In P. Dillenbourg, A. Eurelings, K. Hakkarainen (Eds.) *European Perspectives on Computer-Supported Collaborative Learning, Proc. First European Conference on CSCL*, Universiteit Maastricht, Maastrict, the Netherlands, March 22-24 2001, pp. 173-180.

Dillenbourg, P., Baker, M., Blaye, A. and O'Malley, C. (1996) The Evolution of Research on Collaborative Learning. In E. Spada & P. Reiman (Eds). Learning in Humans and Machine: Towards an interdisciplinary learning science, Oxford: Elsevier, 189-211.

Doise, W., and Mugny, G. (1984). The social development of the intellect. *International Series in Experimental Social Psychology*, 10, Pergamon Press.

Festinger, L. (1957). *A theory of cognitive dissonance.* Stanford University Press.

Gokhale, A. A. (1995). Collaborative Learning Enhances Critical Thinking, *Journal of Technology Education,* 7(1), 1995.

Gordon, A., and Hall, L. (1998). A collaborative learning environment for data modeling. *Proceedings of the 1998 Florida Artificial Intelligence Research Symposium*, Sanibael Island, FA:AAAI Press, 158-162.

Greer, J., McCalla, G., Collins, J., Kumar, V., Meagher, P., and Vassileva, J. (1998). Supporting Peer Help and Collaboration in Distributed Workplace Environments. *International Journal of Artificial Intelligence in Education*, 9, 159-177. [Available: http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_9/greer.html].

Heift, T. and Nicholson, D. (2001). Web Delivery of Adaptive and Interactive Language Tutoring. *International Journal of Artificial Intelligence in Education* (2001), 12, to appear.

Hoppe, U. (1995). The use of multiple student modeling to parameterize group learning. In Greer, J. (Ed.) *Proceedings of Artificial Intelligence in Education* (AIED'95), Washington, DC, August 16-19, 234-241.

Jermann, P., Soller, A., & Muehlenbrock, M. (2001). From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. In P. Dillenbourg, A. Eurelings, K. Hakkarainen (Eds.) *European Perspectives on Computer-Supported Collaborative Learning, Proc. First European Conference on CSCL*, Universiteit Maastricht, Maastrict, the Netherlands, March 22-24 2001, pp. 324-331

Johnson, D.W. and Johnson, R.T. (1994). *Learning Together and Alone*, Englewood Cliffs, NJ: Prentice Hall.

Johnson, D.W., Johnson, R. T. and Smith, K.A. (1991). Increasing College Faculty Instructional Productivity, *ASHE-ERIC Higher Education Report No. 4. School of Education and Human Development, George Washington University*.

Katz, S. and O'Donnell, G. (1999). The Cognitive Skill of Coaching Collaboration. In *Proceedings of the Computer Support for Collaborative Learning (CSCL) 1999 Conference*, C. Hoadley and J. Roschelle (Eds.) Dec. 12-15, Stanford University, Palo Alto, California. Mahwah, NJ: Lawrence Erlbaum Associates, 291-299. [Available at http://kn.cilt.org/cscl99/A36/A36.HTM].

McManus, M. M. and Aiken, R.M. (1995). Monitoring Computer Based Collaborative Problem Solving", *Journal of Artificial Intelligence in Education*, 6(4), 308-336.

Mühlenbrock, M., and Hoppe, U. (1999). Computer Supported Interaction Analysis of Group Problem Solving. In C. Hoadley & J. Roschelle (Eds.) *Proceedings of the Computer Support for Collaborative Learning (CSCL-99) Conference*, Stanford University, Palo Alto, Dec. 12-15, NJ: Lawrence Erlbaum Associates, 398-405.

Okamoto, T., Inaba, A. and Hasaba, Y. (1995). The Intelligent Learning Support System on the Distributed Cooperative Environment, In Greer, J. (Ed.) *Proceedings of Artificial Intelligence in Education AI-ED'95*, Washington, D.C., August 16-19, 588.

Paolucci, M., Suthers, D., and Weiner, A. (1996). Automated Advice-Giving Strategies for Scientific Inquiry. *Third International Conference on Intelligent Tutoring Systems* (ITS'96), Montreal,.June 12-14, 372-381.

Robertson, J., Good, J. and Pain, H. (1998). BetterBlether: A Computer Based Educational Communication Tool. In G. Ayala (Ed.) *Proceedings of Workshop "Current Trends and Applications of Artificial Intelligence in Education" at the 4th World Congress on Expert Systems,* Mexico City, Mexico, March 16, ITESM, *90-97.*

Shanks, G. (1996). Conceptual Data Modeling: An Empirical Study of Expert and Novice Data Modellers. In C.D. Keen, C. Urquhart, J. Lamp (Eds.), *Proceedings of the 7th Australasian Conference on Information Systems '96, Tasmania Australia, 11-13 December, Australian Computer Society INC, Tasmania, Vol 2, 641 – 652.*

Slavin, R. E. (1995). *Cooperative Learning*. Allyn and Bacon, 2nd. edition.

Suthers, D. (1993). Preferences for Model Selection in Explanation. *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93).* Chambery, France, August, 1208-1213.

Suthers, D.D. and Jones, D. (1997). An Architecture for Intelligent Collaborative Educational Systems. In B. D. Boulay and R. Mizoguchi (Eds.) *Artificial Intelligence in Education: Knowledge and Media in Learning Systems.* (Proceedings of AI-ED'97, 8th World Conference on Artificial Intelligence in Education, Kobe, Japan, August 18-22, Amsterdam: IOS, 55-62.

Suthers, D, Connelly, J., Lesgold, A., Paolucci, M., Toth, E., Toth, J., and Weiner, A. (2001). Representational and Advisory Guidance for Students Learning Scientific Inquiry. In K. D. Forbus and P. J. Feltovich (Eds.) *Smart Machines in Education: The Coming Revolution in Educational Technology.* Menlo Park: AAAI Press, 2001, pp. 7-35.

Tedesco, P. and Self, J. A. (2000). Using meta-cognitive conflicts in collaborative problem solving environment. *Proceedings of the 5th International Conference on Intelligent Tutoring Systems,* Montreal, Canada, 262-271.

Webb, N. and Palincsar, A. S. (1996). Group processes in the classroom. *Handbook of Educational Psychology*. D. Berlmer & R. Calfee (Eds.) Simon & Shuster Macmillan NY.

Weber, G. and Brusilovsky, P. (2001). ELM-ART: An Adaptive Versatile System for Web-based Instruction. *International Journal of Artificial Intelligence in Education* (2001), 12, to appear.