Collaborative Learning Environment to Improve Novice Programmer with Convincing Opinions

DINH THI DONG PHUONG
Computer Science Dept
Ritsumeikan university
JAPAN
phuong@de.is.ritsumei.ac.jp

HIROMITSU SHIMAKAWA
Computer Science Dept
Ritsumeikan university
JAPAN
simakawa@is.ritsumei.ac.jp

Abstract: At the present, many universities do not have enough teaching staffs for programming training. The teacher cannot give suggestions in a timely fashion to all of the students who need guidance for their programming problems. The training condition reduces motivation to learn programming of the students. We propose a model to promote collaborative learning among the students. In the model, the students are combined into small groups to practice programming. Although every group member has to do programming independently, they are encouraged to share programming problems with other group members to find out solution. The voting mechanism is applied to each group. If a group member gives an opinion which is evaluated convincing one, the member is rewarded by others with COOP points. For effective collaboration of students in each group, we combine students based on two features of each student. One is programming ability, which are scores of the source codes. The other is contribution to group, which are COOP points accumulated during the collaborative learning. Using the model, we developed a collaborative learning environment, Col-E. The Col-E has been applied to actual classes in Ritsumeikan university, Japan. The experiment results show that Col-E has good possibilities to improve any kinds of students.

Key–Words: Collaborative learning, convincing opinion, student combination, chat-based system, client-server architecture, Col-E

1 Introduction

Programming practicing is indispensable tasks for students to understand comprehensively what they learn from lectures. Practicing is to improve programming experience and skill. The more practice, the deeper and further understanding and experience can be achieved. Moreover, it brings foundation to understand better subsequent lectures and practices. Therefore, whenever novice programmers get stuck in problems, they should be helped with suggestion and guidance from teaching staffs so that they can progress their programming as the training expectation.

However, at the present, in many programming training places, programming training condition is not good enough. Especially there are not enough teaching staffs. There are usually 40 or more students learning programming in one class room but there is only one teacher or one teaching assistant - TA. The shortage of teaching staffs causes programming practice in computer rooms is not effective for both students and the teacher. The students frequently cope with problems but are not responded to by the instructor in a timely fashion. If they are not able to solve problem and have no idea to solve it, they reach an impasse. The teacher has to answer many questions from many students though these questions often include similar ones repeatedly asked or easy ones for other students to answer.

As stated above, whenever a student is faced with problems and cannot solve them, the student has to wait for guidance from the teacher. However, in many cases, students hesitate to consult with their classmates on the troubles. The reason may be they are not sure whether the person with whom they intend to consult can offer good opinions for them or not. They are afraid to interrupt others who are concentrating on doing their works.

More seriously, these actual states cause students to become bored with studying programming. They expect to achieve more programming experience. They expect to make a program successfully. But they are not satisfied because they cannot receive guidance and advice from their instructor when they need. As a result, they do not want to come to the programming practice class.

The teacher spends much time to answer similar questions time by time from different students. It also takes teacher much time to reply big number

of unimportant questions to students who are unpracticed or have holes of necessary programming matters. It would prevent the teacher from instructing students the instruction of whom is indispensable. It reduces time to encourage students to achieve creative solutions. These therefore make the teacher to be stuck in the situation where the supervision of the students is not effective and boring.

For the problems, we should utilize the power of classmates, instead of making the teacher repeat to answer the same simple questions to many students. Therefore, it is necessary to promote collaborative learning among students. Our method is combining the students into small groups to encourage students to search a solution for themselves. The collaborative work among a small number of students enables them to find a solution in a timely manner, even thought the teacher is not available at that time. Grouping the students is based on each student programming ability and contribution to group. The former is evaluated by source code. The latter is measured by COOP points. COOP points are the number of opinions from the student to be considered convincing enough to solve certain problems of other members of the group.

CoL-E is a collaborative learning environment using the above method. It is a chat-based system with the server and client architecture. The servers take in charge of logging the two features of the students to combine them. Client provides graphical user interface facilitating their collaborative learning activities.

2 Novice programmer training

2.1 Problems

Problems in novice programmer education can be classified as two main categories. One category is the mastering programming knowledge. Programming knowledge includes understanding theory, practicing and applying it. The shortage of these matters is covered from background, general to specific[1][2][3][4].

Background problems include those related to from understanding how computer works to using the tools, understanding the task (programming exercises) and getting started. Poor understanding of naming principles and directory hierarchy causes students to fail the resources or to lose files. Since students can not understand the requirements of the exercise, they are unable to figure out solution. Even if students understand the requirement and be able to have an image of possible solution, they do not how to get started.

General problems are those that students have a general design view but are getting basic structural details wrong. These result from problems with basic algorithms and data structures. Besides, naming causes problems. They understand what the function of the thing is, but cannot name it adequately. It is also serious barrier not to be Being not familiar with programming processes such as analysis, design, coding, testing and maintenance.

Specific problems are those that are associated with programming languages and particular programming matters such as arrays, loops, expressions and strategies to apply them. For instance, with C programming language, many students mistake the result of an expression evaluation which is equivalent to 0 or 1, with the values of true or false of Pascal programming language.

The other category is derivative matters of the first category. Learners are eager to succeed in their programming, but they do not how to proceed when they encounter problems. They need teacher guidance to overcome the problems. In actual conditions, however, they have to waste time waiting guidance from the teacher. The situation reduces their motivation to study programming.

Whenever the programming training condition is not qualified enough, these problems continues. Shortage and weakness of programming teaching staffs are the main reasons. However, the problems cannot be solved in a short time.

2.2 Comparison with existing works

2.2.1 MEDD

The multistrategy error detection and discovery system - MEDD is a bug classification system[6]. Students can retrieve the bug libraries to solve their difficulties. However, there are considerable matters. The input of the system must be a program. This criterion is so difficult for the beginners. Problems of novice are so wide range that they are hard to be classified. Retrieval from the bug library also takes much time. Understanding the suggestion from the bug libraries is hard task for beginners.

2.2.2 Pair programming

Pair programming, a software development technique in which two programmers work together at one keyboard, is another considerable solution for this training condition.

From the study on problems encountered by novice pair programmers[3][5], a notable result is presented. The number of problems occur for the case of pairs much less than for the case of solo.

The experiment on 40 senior Computer Science students at the University of Utah by Laurie Ann

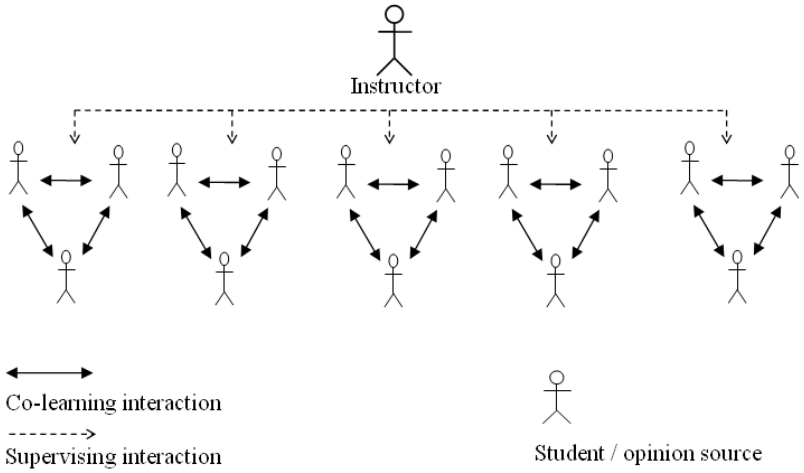

Figure 1: Col-E theme

Williams[7] also shows that paired students feel enjoyable and more confident than solos. The quality of source codes made by paired students are better than those made by solos.

However, if we let novice programmers pair when they practice programming, it would reduce the real practicing time of each member. Whenever a problem occurs, the one who has solution will seize the opportunity to solve it. The other even would not have chance to understand the problems and have no opportunity to practice it. This makes the student gradually become not confident enough to program solo after pairing.

Furthermore, if we apply pair programming to novice students, we cannot clarify exactly who makes the programs because we have no means to manage all the pairs. To make the matter worse, if we make the score of exercises important, students good at programming would finish most parts of the work, leaving ones poor in programming idle[8].

Besides of these, we do not know how to pair two students so that both of the two members can achieve their collaborative learning the most. We might combine a less experienced programmer with a more experienced one with the hope that the former will learn from the latter and can achieve the best result. As the real phenomena stated above, it is difficult to achieve this goal.

3 Collaborative learning with convincing opinions

3.1 Collaborative learning

The project of the Wisconsin Center for Education Research studies the collaborative learning intensivety. Their web page[9] says "Collaborative learning or co-learning is an educational approach to teaching and learning that involves groups of students working together to solve a problem, complete a task, or create a product. Collaborative learning is based on the idea that learning is a naturally social act in which the participants talk among themselves. It is through the talk that learning occurs."(Gerlach, 1994) Co-learning method has been used somehow and brought many interesting advantages[10][11][12].

For novice programmers to study programming effectively, practice programming on their own must be strengthened. In addition to that, the collaborative learning and the pair programming methods should be applied in the fashion of taking their advantages and diminishing negative effects. By approaching this way, in practice time, every student is demanded to practice programming by itself on the assignment given by a teacher. When a student cope with a problem, it itself would try to search for clues from others. The student itself would practice these suggestions and make up its decisions, make the understanding into experience and vice versa. We call this training method as collaborative learning or co-co-learning.

3.2 The overview of the model

The model overview is described in figure 1. The students of the class are grouped. Each group member is called an opinion source. Inside a group, opinion sources can give opinions on programming problems bravely. For effective co-learning of each group, the following factors are taken into account. Fist, the number of each group should be logical. It should be 3 because of the balance of many matters. If there are more than 3 students in a group, one member would be interrupted too much, while it has to focus on its own programming. The group member would not have a sense of responsibility to others, either. From the view point of the receiver, more than two different opinions are puzzling. Opinions from the other two members are enough to help the receiver. In case these opinions are not convincing, they can consult the teacher. Second, the combination of students for each group would be appropriate. The grouping algorithm based on contribution to group and programming ability of the students presented in section 3.4 aims at this purpose. Third, a proper communication means must be stepped up among group members. In case all group members are engaging in their programming, the student with a problem can broadcast it without hesitation. The broadcasting is supported with the system described in section 4.

3.3 Convincing opinions

Figure 2 illustrates the co-learning mechanism in each group. Whenever a group member gets in problems, it can broadcast the problems to all remaining members using the co-learning supporting system. When the others of the group receives signals of requesting suggestion, it would try to offer suggestion back to the requesting one. The requesting one itself would examine and practice the guidance to solve its problems. If the receiver considers an opinion convincing, it gives a COOP(COnvincing OPinion) point to the offerer. Giving COOP points is similar to commending and rewarding mechanism. If a person contributes a good opinion, the person is rewarded with a COOP point.

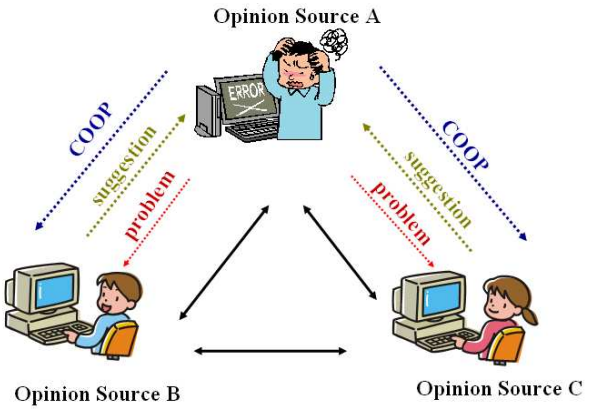

Figure 2: Co-learning activities in one group

Interaction of COOP points is a mental encouragement. From actual experience, it is obvious that in case a student has to cope with so many problems to reach to a final good result, the student wants to express the problems to others with expectation of good advice. When one is praised, appreciated, rewarded because it has done something good, or helpful to others, a big encouragement is brought back to the author. This is a strong motivation that inclines students to try more and be willing more to do their work.

Because motivation of students is increased, the discussion between them is promoted. It brings effective co-learning.

To encourage students offering opinions among them, we should consider COOP points of a student as a part of its achievement.

3.4 Grouping method

Several grouping criteria are studied to improve the learning among students[16]. We group student based on its programming ability and its contribution to the group.

Suppose the students have to practice programming many times during a semester. After each co-learning session, every student submits its source code to the server. A teacher grades these source codes for students. Each student will have:

1. A score of its source code, and
2. COOP points which are accumulated when it practices programming.

The two features of every student, as shown in figure 3, are basic to specify type for a student. The example in figure 4 adopts 4 types: type I for strong programming ability and contribution, type II for strong ability but poor contribution, type III for poor ability but strong contribution, and type IV for poor ability and contribution. The type of a student at a specific time will be identified based on the average values of COOPs points and scores over all its past practicing sessions. We assume the types are ranked as I, II, III, and IV in the descending order.

For several preliminary sessions, students are grouped randomly. Let e be the number of the preliminary sessions. After session i ends, the following

Table 1: List of groups of students

Student	Q	E	D	H	L	F	I	C	M	B	A	G	K	O	P
Type	I	IV	III	II	I	I	I	II	II	III	III	IV	IV	II	III
Group No	3	2	4	1	3	4	5	1	1	4	5	2	3	2	5

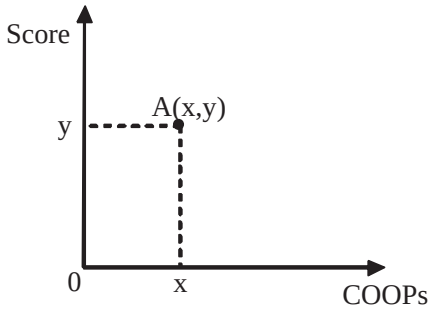

Figure 3: Two features of one student

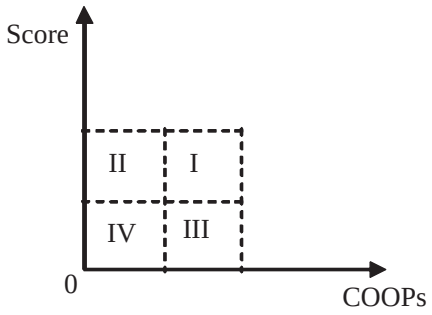

Figure 4: 4 types of students

procedure is used to determine new student groups for session $(i+1)$, where $i > e$.

1. Figure out the type of each student over all the past practice sessions.
2. Evaluate whether a combination of students is good or not. If the types of session i of all the group members are greater or equivalent to the types figured out in the previous step, the combination is regarded good. This good combination of student types is counted up in the statistic table.
3. Group students based on their types and the well combinations from the statistic table.

Table 2: Statistics of effective combinations

Combination of types			Good times	Combination of types			Good times
I	I	I	0	II	II	II	6
III	III	III	1	IV	IV	IV	0
I	I	II	4	I	I	III	3
I	I	IV	4	II	II	I	1
II	II	III	5	II	II	IV	2
III	III	I	4	III	III	II	2
III	III	IV	1	IV	IV	I	2
IV	IV	II	5	IV	IV	III	3
I	II	III	2	I	II	IV	5
I	III	IV	4	II	III	IV	1
I	I		0	I	II		0
I	III		0	I	IV		1
II	II		2	II	III		1
II	IV		0	III	III		0
III	IV		0	IV	IV		0

The first 2 rows in table 1 shows an example of the random list of the students with their current types. Table 2 shows statistic data of well combinations of students accumulated from the beginning session. From the table 2, the combination of students typed II, II and II is the most effective. Therefore, students H, C and M will be in the same group in session $(i+1)$. The next is the combination of students typed II, II, III, the combination of student typed IV, IV, II, and the combination of students typed I, II, IV . It brings students E, G and Q to the same group, and so on. The lowest row in table 1 shows the groups of students for session $(i+1)$.

4 Co-learning environment with convincing opinions, Col-E

The co-learning environment with convincing opinions is a chat-based system[14] using client-server architecture. The system aims at two main functions. One is logging students COOP points and scores of

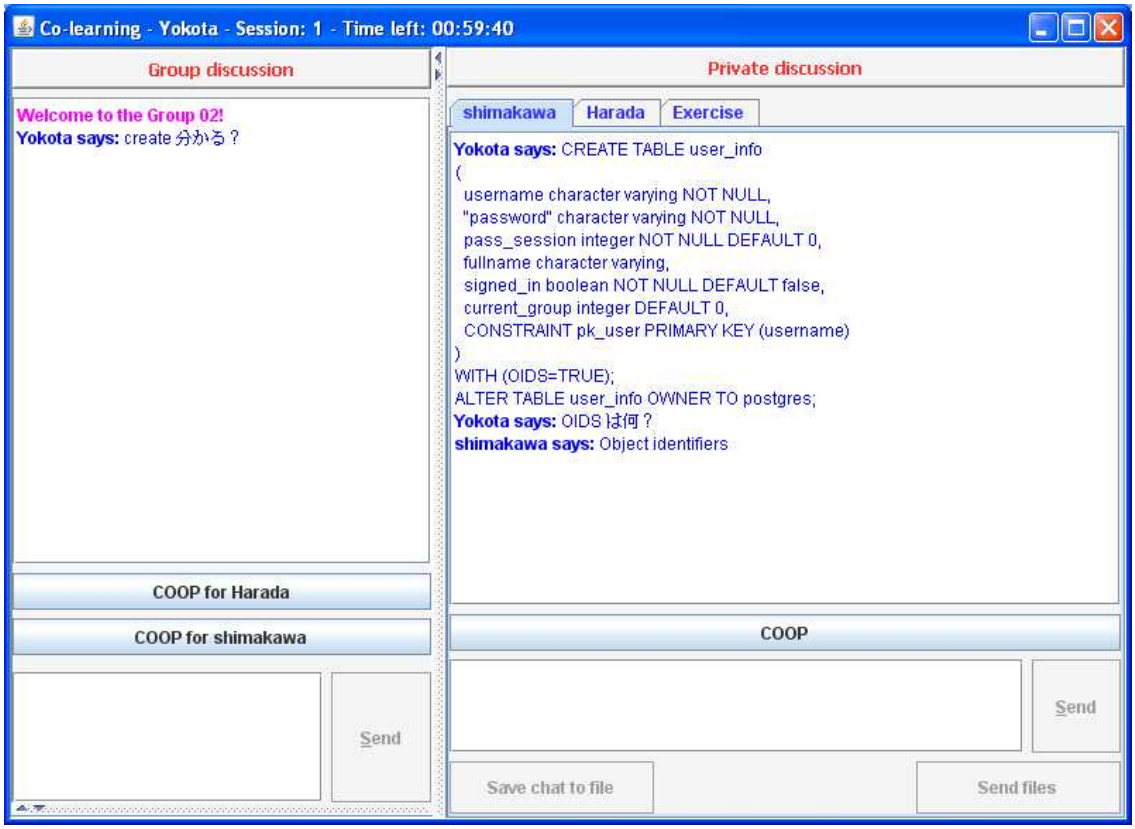

Figure 5: Discussion areas with COOP buttons

exercises, analyzing them to recommend good students combinations for next co-learning session. The other is facilitating communications among opinion sources of each group.

The server program undertakes the former function and provides some facilities for the teacher to manage the class. The client program provides graphical user interface (GUI) to support co-learning among students.

4.1 Client

The procedure to use the client program is described as:

1. Log in.
2. Get exercise requirements and work on them.
3. Submit the result source code to the server.

If a student logs in successfully, the co-learning GUI occurs. Student full name, session number, and time left for this session are displayed in the title of this window. The window has two main areas: left area for group discussion and right area for private discussion, *i.e.*, discussion with a specific person in the group as shown in figure 5. The exercise will be displayed in the tab *Exercise* of private discussion part.

When a message such as question, comment is sent with the left area for group discussion, the message is broadcasted to all of the group members. A student presses the button ”COOP” to give a COOP point to the group member who has offered the convincing opinion. The button can be pressed at any time.

In the right area for private discussion, there are 3 tabs windows. The names of the first and the second tabs will be the names of other sutdents who are in the same group. A student can choose a group member to whom it wants to chat[15]. Button ”COOP” is prepared to give COOP points to the chat partner.

Some students prefer group discussion, while others private discussion. The wider the area for the preferred discussion, the better utility is brought to students. The separator between the left area and the right area allows ”Click and Drag” to broaden the either area.

The button ”Submit” in Exercise Tab enables to submit the source code to the server. The teacher will grade these source codes to enroll the scores into the database.

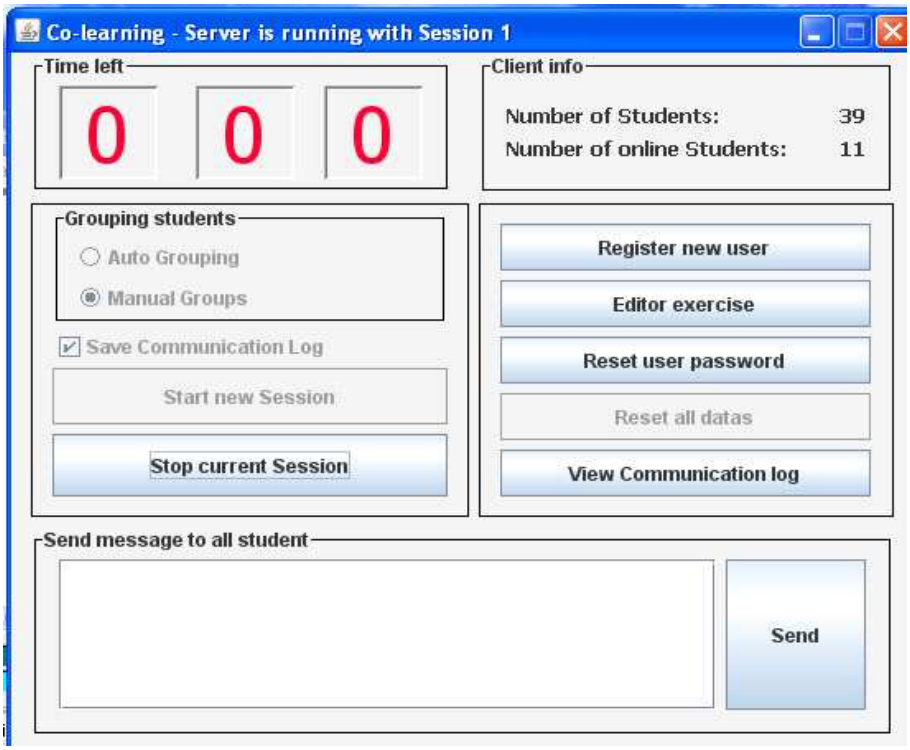

Figure 6: GUI of the Col-E server

4.2 Server

The functions of the server are arranged into two groups. One group is functions to manage an exercise session. This includes selecting the way to group students. Two ways to group students are provided here. Auto grouping is the grouping method done by the grouping algorithm described in section 3.4. Manual group is used for teacher to assign the group number of each student to the database. The other group consists of functions to manage student information such as co-learning usernames, passwords, exercise requirements, which are in the left side in figure 6. To send message to all students of the class, the teacher can utilize the textbox in the bottom part.

5 Experiment and results

5.1 Experiment condition

We applied our method into two actual classes of the second grade students in Ritsumeikan university. Each class has 37 students. They practices database programming with an ODBC library and C language. These students have learned C programming and database systems in previous semesters. Because of the strictness of the syllabus of the university, the experiment condition does not best match the evaluation of our method in the following reasons.

There are a teacher and 5 TAs for each class. Each group consisting 3 students is assigned with an exercise. Each group member takes in charge of a small part of the exercise. After 7 weeks, every student is required to submit the source code as well as presentation on its part. During programming, the students could use Col-E as a text communication means. COOP points are not considered an important achievement of the students, because their report documents have been declared as dominant criteria for the evaluation of their scores in the syllabus.

5.2 Results

Because students with problems can receive guidance from the teacher and TAs at any time, the learning condition described above is quite good for the students. We could not evaluate the real effect of our method due to the good learning condition. However, we still have several achievements.

1. Communication among group members are realized as following:

   At the beginning of the practice, when students discuss on who takes which part of the assignment, communication is done almost face-to-face. When students start to work on their own

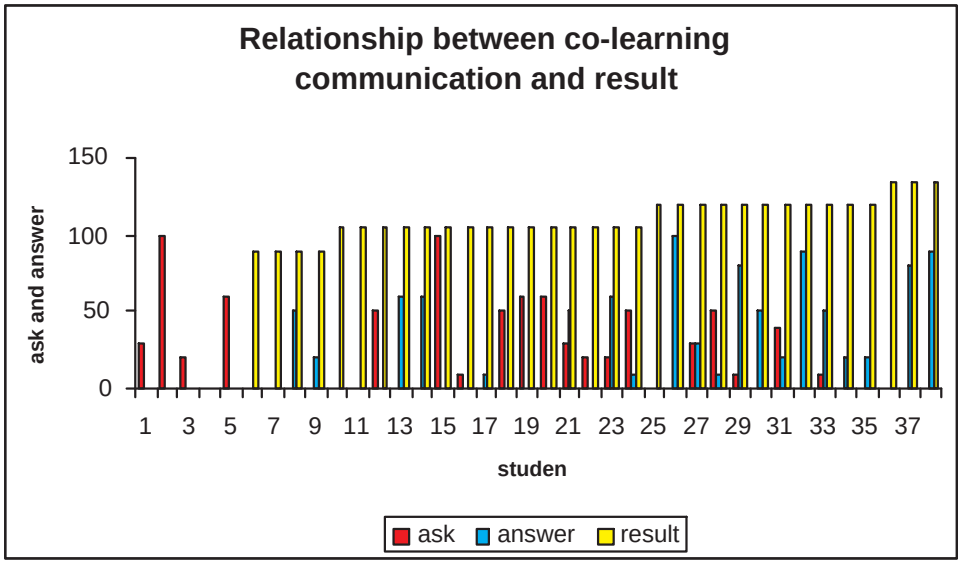

Figure 7: Relation of co-learning communication and result in Class 1

part, our system is a preferable means to exchange opinions with group members. This communication can be classified as:

Co-learning communication These are exchanges of opinions on problems, messages to ask friends ideas and guidance as well as the agreements of discussions. Some of them initiate a face-to-face discussion on certain problems. Examples are:

- How can I execute psql?
- Did I decide ID and name properly?
- I have checked my source code but I could not find any error. Can you come and help me?

Non co-learning communication Some communication messages are to ask or inform the work progress. Some students use Col-E to chat the things unrelated to the work. Following are examples:

- What are you doing now? - I am creating product table.
- I like this website.

2. Relation of student programming ability (score of the source code) and their co-learning communication:

   From the chat log of the students, we have analyzed co-learning and non co-leaning communications. From co-learning communication, we extract the asking, answering opinions. Figure 7 and 8 show the relevance of asking and answering with scores of source codes. The results shows that the co-learning activities occur at all sorts of student. Many students enjoy co-learning. It means the Col-E has good possibilities to improve capability and motivation of any kind of students.

3. The client GUI is as familiar as other chat GUI. Students have no difficulty to use it.

4. The system is implemented using Java. The installation is not complicated on Windows as well as Linux platform.

6 Discussion

With the proposed method, both students and teacher get more achievements from co-learning of students. The essence of this environment is to help students search by themselves for solutions, giving comments with each other on its problems to avoid wasting time. At the same time, it brings collaborations of students. Therefore, this CoL-E is effective for courses designed for students who start to learn programming such as programming courses including programming languages (C programming, Java programming) and data structure and algorithms. The grouping function aims at two main purposes. The first is to let students take chances to learn from many other different students so that one student can deepen its programming skill. The second is to find out the combination rules of students for effective co-learning if possible. At the time the rules are found out, we can fix these good combinations.

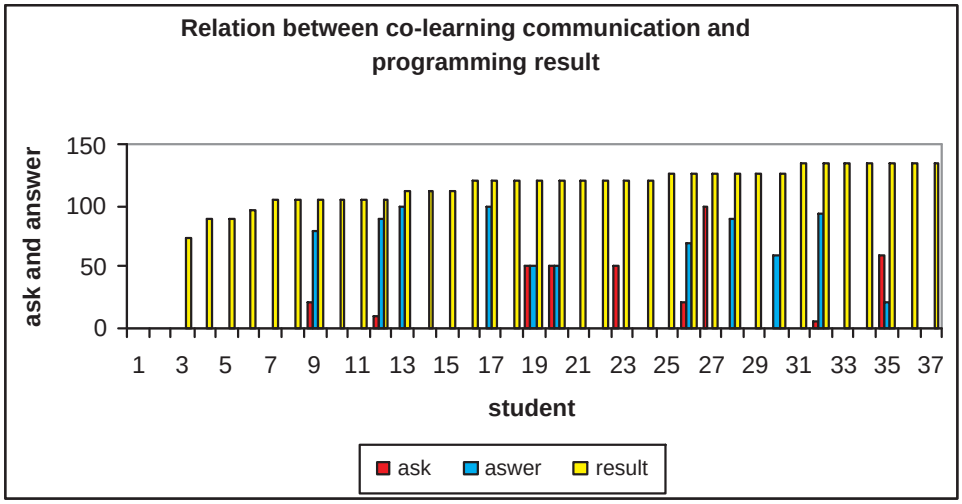

Figure 8: Relation of co-learning communication and result in Class 2

Depending on the the real purpose of the course, we can adjust the number of each group and the features of students. For example, for the courses which are emphasize on coding, the features of students may be ability of programming, experience of programming and contribution.

The GUI of our system is another issue to be discussed. In case group members are working in one place, the face-to-face discussion is the most effective co-learning. However, the chat-based communication contributes to recoding discussion contents during the learning. When a student given guidance messages from others reviews them later, the record would bring deeper understanding to the student. In addition to that, the chat-based GUI can support the collaborative learning when students work on exercises in various places such as they work on homeworks.

We have adopted the programming ability and the COOP points as the criteria to make groups. There may be other parameters to be consideres as the criteria. We are planning to apply our method to many actual classes. We may find other parameters through the application.

7 Conclusion

We propose a collaborative learning environment to enhance capability and motivation of novice programmers. To promote the co-learning, convincing opinions are key elements. Moreover, combination of opinion sources and communication means are determining factors of the CoL-E.

Since attitudes toward learning as well as giving COOP points are much dependent on student characteristics, the effectiveness of this method depends on real programming conditions. To make this clear, our method needs to be applied to actual programming exercise classes.

References:

[1] Anthony Robins at al., *Learning and Teaching Programming: A Review and Discussion*, Computer Science Education, Vol.13, No.2, pp.137-172, 2003

[2] Sandy Garner, Patricia Haden, Anthony Robins, *My Program is Correct But It Doesn't Run: A Preliminary Investigation Of Novice Programmers' Problems*, Proceedings of the 7th Australasian Conference on Computing Education (ACE'05). pp.173-180, 2005

[3] Brian Hanks, *Problems Encountered by Novice Pair Programmers*, ACM Journal on Educational Resources in Computing, Vol.7, No.4, Article 2, Jan., 2008

[4] Richard E. Mayer, *Teaching and Learning Computer Programming*, Proceedings of Symposium on Research on Teaching and Learning Computer Programming, Washington D.C., 1987.

[5] Jun Soonjin, Kim Seungbum, Lee Wongyu, *Online Pair-Programming for Learning Programming of Novices*, WSEAS Transactions on Advances in Engineering Education, Issue 9, Volume 4, pp:187-192, 2007,

[6] Raymund C. Sison, Masayuki Numao, Masamichi Shimura, *Multistrategy Discovery*

and Detection of Novice Programmer Errors, Machine Learning, No.38, pp.157-180, Kluwer Academic Publishers, 2000

[7] Laurie Ann Williams, *THE COLLABORATIVE SOFTWARE PROCESS*, Department of Computer Science, the University of Utah, 2000.

[8] Juan A. Marin-Garcia, Jaime Lloret, *Improving Teamwork with University Engineering Students. The Effect of an Assessment Method to Prevent Shirking*, WSEAS Transactions on Advances in Engineering Education, Issue 1, Volume 5, pp.1-11, Jan., 2008,

[9] http://www.wcer.wisc.edu/archive/cl1/CL/default.asp

[10] Nira Hativa, *Teaching for Effective Learning in Higher Education*, Kluwer Academic Pub, 2000.

[11] Lynda Thomas, Mark Ratcliffe, John Woodbury, Emma Jarman, *Learning Styles and Performance in the Introductory Programming Sequence*, Proceedings of SIGCSE 2002, pp.33-37, 2002.

[12] Andrew Scott, Mike Watkins and Duncan McPhee, *Step Back from Coding - An Online Environment and Pedagogy for Novice Programmers*, Proceedings of the 11 Java in the Internet Curriculum Conference, The Hihger Education Acadamy, London Metropolitan University-UK, pp. 35-41, 2007

[13] Linda McIver, *The Effect of Programming Language on Error Rates of Novices*, 12th Workshop of the Psychology of Programming Interest Group, pp.181-192, 2000

[14] Andew Davison, *Killer Game Programming in Java*, Chapter 30, Network Chat, O'Reilly, 2005

[15] Thomas Huining Feng and Hans Vangheluwe, *Case Study:Consistency Problems in UML Model of a Chat Room*, Proceedings of International Conference on the Unified Modelling Language, Workshop on Consistency Problems in UML-based Software Development II, Oct. 2003

[16] Huikkola Miika, Silius Kirsi, and Pohjolainen Seppo, *Clustering and Achievement of Engineering Students Based on Their Attitudes, Orientations, Motivations and Intentions*, WSEAS Transactions on Advances in Engineering Education, Issue 5, Volume 5, pp.342-354, 2008