A Collaborative Learning Support System for Software Engineering Education

Longming ZHANG* and Atsuo HAZEYAMA**
* Graduate School of Education, Tokyo Gakugei University, Japan
E-mail:m103306m@st.u-gakugei.ac.jp
** Department of Information Science, Tokyo Gakugei University, Japan
E-mail: hazeyama@u-gakugei.ac.jp

Abstract

Recently, the study of the field of Computer Supported Collaborative Learning (CSCL) is actively conducted along with the development of the Internet. This study focuses on collaborative learning theories and developed a system which could perform collaborative learning between learners or between learners and teachers online for software engineering education. The system provides functions Cognitive Apprenticeship suggests. The learners can also do interactions such as discussions and/or question and answer between learners by a question and answering feature of the system that Distributed Cognition suggests. According to the analysis of the results that applied the system to the course of a university, we found some patterns of collaborative learning that collaborative learning theories suggest.

1. Introduction

Research and development of Computer Supported Collaborative Learning (CSCL) has been actively conducted along with advancement of the Internet [5]. Social constructivism is a theoretical basis of CSCL [5]. Social constructivism emphasizes on existence of other learners in the process of knowledge acquisition. It supposes that knowledge and/or cognitive functions of learners are acquired by internalization by learners through interactions between learners and external worlds or others [5]. Learning theories such as Cognitive Apprenticeship (CA), Legitimate Peripheral Participation (LPP), Distributed Cognition (DC), and so on are those based on social constructivism.

Software engineering education aims at learners acquiring various types of techniques and/or skills to develop software. They include not only those on programming but also those on requirement definition, design, and so on. One goal of software engineering education is for learners to develop software by using techniques and/or skills that learned in lecture. For that goal, it is necessary for learners to tackle on authentic problems [7]. The instructors usually give tasks to the learners. One approach that the instructors support for the learners to accomplish the tasks is to follow the method of Cognitive Apprenticeship. However, it is difficult for the instructors to undertake the guidance by the Cognitive Apprenticeship style in a classroom many learners study. On the other hand, LPP refers to the existence of learners who acquire knowledge through learning in the community and become full participation [6]. Distributed Cognition also supposes existence of other learners. We suppose that the guidance is done by not only the instructors but also the learners in the classroom. We expect to facilitate learning with each other.

This paper proposes a learning system that supports the learning process based on Cognitive Apprenticeship and the interactions among the stakeholders of the learning based on Distributed Cognition that considered characteristics of software engineering education.

The rest of this paper is organized as follows: section 2 describes an overview of some collaborative learning theories. Section 3 introduces some related work. Section 4 proposes our learning support system. We applied the support system to an actual university course. Section 5 shows a scenario of the experiment, results and discussions. Finally we conclude this paper in section 6.

2. Collaborative learning theories

Cognitive Apprenticeship is a theory of critical-observing the processes by which an expert thinks and practicing these skills under the guidance of the expert [1]. Cognitive Apprenticeship is consisted of six methods, *modeling* (involving an expert's performing a task so that the students can observe and build a conceptual model of the processes that are required to accomplish it), *coaching* (consisting of observing students while they carry out a task and offering hints, scaffolding, feedback, modeling, reminders, and new tasks aimed at bringing their performance closer to expert performance), *scaffolding* (referring to the supports the teacher provides to help the student carry out the task), *reflection*

CollabTech 2012 , August 27-29, 2012, Hokkaido, Japan.

(involving enabling students to compare their own problem-solving processes with those of an expert, another student, and ultimately, an internal cognitive model of expertise), *articulation* (involving any method of getting students to articulate their knowledge, reasoning, or problem-solving processes), *exploration* (involving pushing students into a mode of problem solving on their own).

Legitimate Peripheral Participation (LPP) describes how newcomers become experienced members and eventually old timers of a community of practice or a collaborative project [6].

Distributed Cognition captures aspects that learners exchange their knowledge one another and conduct their tasks by utilizing external resources under the recognition that no learners exist who have all expertise in a learning community [3].

3. Related work

This section describes related work regarding software development support based on the learning theories.

Tholander and Karlgren constructed a learning environment for object-oriented modeling learning that provides three major features based on "Cognitive Apprenticeship" and "Situated Learning" [7]:

- Function that presents video which recorded problem solving process by expert modelers to learners.
- Function that provides the pattern library for learners to support understanding of abstract concepts
- Function that facilitates reflection and way of thinking of meta-cognition by agent

Their study focuses on supporting of activities of individual learners. On the other hand, it doesn't support interactions between learners and the instructors such as coaching in "Cognitive Apprenticeship." Our study focuses on supporting of not only individuals' learning but also interactions between learners and the instructors and among learners, resulting in knowledge construction of learners.

Ye captured software development from the viewpoint of "Distributed Cognition" and developed a programming support environment called STEP_IN that supported social aspects of problem resolution in programming [8]. STEP_IN provides some features as follows: storing and sharing of sample programs, presenting of discussion history for each sample program, and inquiring experts.

4. Collaborative learning support system for software engineering education

This section describes our learning support system. It is composed of functions for the teaching staff and those for learners. In the following, we show functions for supporting learners.

(1) Viewing of materials: we provide lecture notes, in addition, artifacts created in our Project-Based Learning (PBL) course (we call the artifacts model) and the review comments for the artifacts [2]. The artifacts created by groups in the PBL course were reviewed by the teaching staff and revised. Therefore, they reach definitive level of quality. Figure 1 shows a process how an artifact assigned to a learners group is completed through inspection and revision. This is an aspect of authenticity in software development. We think this function corresponds to "modeling" support in Cognitive Apprenticeship.

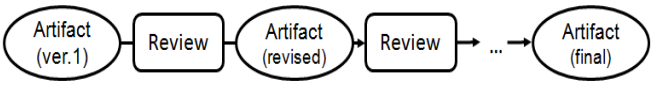

Figure 1. Artifact revision process

(2) Viewing of tasks: this is a function to view tasks the instructor gave. This is a first step in "scaffolding" in Cognitive Apprenticeship.

(3) Communication support: Cognitive Apprenticeship and Distributed Cognition refer to the interaction process (communication) among learners. A communication support function is a core function for collaborative learning.

(4) Reflection and articulation: as this system is a learning support system, we embed functions for reflection and articulation into the system. Comments can be given for lecture notes and the models. It is important for a learner to express his/her own situations, opinions and/or what he/she did for the inquiries when he or she asks them. When advices and/or supports from other learners result in problem resolution for the learner, to reflect the process is important activities to adhere the knowledge.

We implemented a prototype learning support system as a web application using JSP/Servlet technologies.

5. Experiment

5.1. Scenario

We applied the system to our actual course in the 2011 academic year. This course gives lectures on software engineering, such as Unified Modeling Language (UML) and web application development using Java and database (DB) technologies. We applied this system to the learners who took the course during 13 June 2011 through 18 July 2011. The instructor presented a statement for the task to the learners and asked to submit requirement specification, use case diagram including use case descriptions, class diagram, DB design and source code for the task statement. Requirement specification, use case diagram including use case descriptions, class diagram, DB design

for another task were provided as the model in the system. A sample source code was provided as one of the lecture notes.

We collected the usage log of the system and the result of the questionnaire that includes items to investigate usefulness of each function. However, as the response rate of the questionnaire was low (around 23 percent), we evaluated the system based on the log data in this paper.

5.2. Evaluation results of each function of the learning support system

Table 1 shows the usage results of the system.

- Learners logged into the system eight times in average
- Learners browsed samples four times in average
- The number of samples was seven while the number of viewing of them was 170, therefore, 24 times per sample in average were viewed.
- Only one question was submitted and no answers submitted. On the other hand, the number of viewing of question and answer was thirty-six. In this experiment, as few questions and answers were exchanged, usefulness of this function was not confirmed.

Table 1. Quantitative results of the experiment

Usage results by learners	Duration	4 weeks
	No. user registration	44
	No. login	364
	No. viewing samples	170
	No. viewing for the detailed info.	221
	No. submissions for task	193
	No. re-submissions for task	79
	No. total learners who registered acknowledgement	77
	No. total learners who acknowledgements were registered	186
	No. questions	1
	No. answers for the questions	0
	No. viewing of questions and answers	36
Usage results by teaching staff	No. samples registered	7
	No. tasks registered	5

5.3. Analysis of the learning process in a face-to-face manner

This sub-section analyses the interactions among the learners by using the log data. In this experiment, the asynchronous communication feature was few used (only one question was submitted). We also collected the data of the acknowledgement function (log data of face-to-face learning) as one of the log data. The system provided a function to register “acknowledgements” [4] that a learner thanked those who supported the learner. As table 1 showed, the number of submissions for task was 193, and the number of total learners who acknowledgements were registered was 186. This means around one acknowledgement was registered per a submission for task. As mentioned earlier, since asynchronous communications by the function of the system were few, almost all communications were done in a face-to-face manner.

From the data stored by the acknowledgement function, we analyzed what kinds of interactions were done. The tasks assigned to the students in this experiment were five tasks: creation of requirement specification, use-case diagram, class diagram, DB design document and source code written in Java language. Figure 2 to 6 show a social graph for each task based on the acknowledgement data respectively. The graphs were depicted by using JUNG2.0. In the graph a node represents a participant in the subject. An arc represents who acknowledges whom. A source node represents a learner who asked an inquiry. A destination node represents a respondent who answered the inquiry.

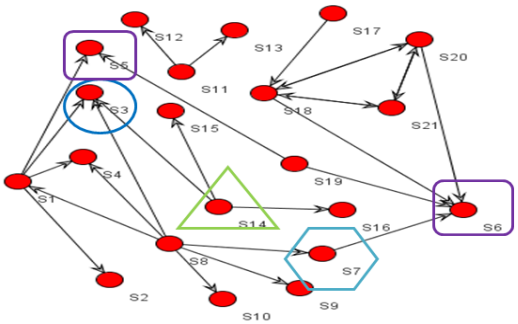

Figure 2. Social graph in the task of creation of requirement specification

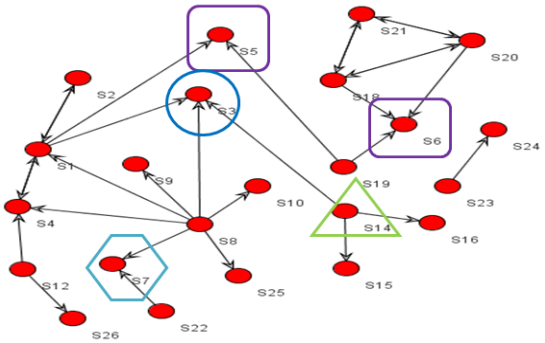

Figure 3. Social graph in the task of creation of use case diagram

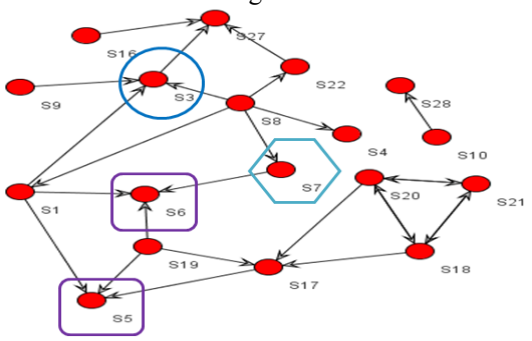

Figure 4. Social graph in the task of creation of class diagram

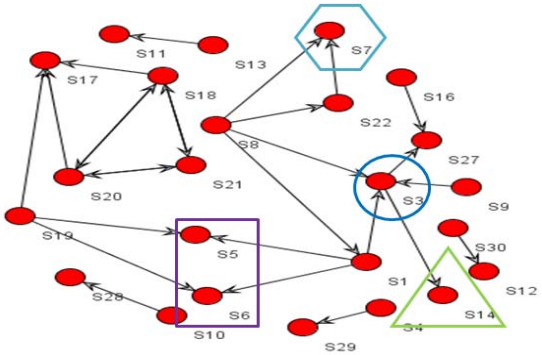

Figure 5. Social graph in the task of creation of database design

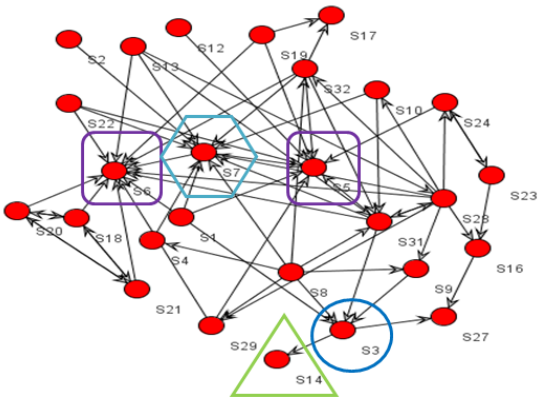

Figure 6. Social graph in the task of coding

We found several interaction patterns as follows:

(1) A case of transitions from the side that gets support to the side that gives support: Let's focus on learner S14. S14 only asked inquiries in the task of creating requirement specification and use case diagram but didn't get any acknowledgements from other learners (see Figure 2 and 3). However S14 got acknowledgements from other learners in the task of creation of DB design document and source code (see Figure 5 and 6). From this fact, we can consider one possibility that S14 required helps from other learners in the early phase because of his/her shortage of knowledge but acquired knowledge by getting helps and could give supports for other learners later. As another possibility, S14 might not be good at creating requirement specification and use case diagram, but S14 was good at implementation such as DB design and programming.

(2) Small learning community: Learners S18, S20 and S21 gave and took supports each other in all the tasks. They formed a small complete graph in this experiment.

(3) Giving supports in all the tasks: S5 and S6 gave supports to other learners in all the tasks. Especially, in the coding task, many learners acknowledged them. S7 and S3 got supports from other learners but they also gave supports for other learners in the early phases. They gave more supports in the coding task.

(4) Taking supports in all the tasks: On the other hand, S8 got supports from several learners in all the tasks.

(5) Reverse LPP: S13 gave support for a learner in the task of creation of requirement specification. However, in the later phases, the tasks of DB design and coding, (s)he got supports from other learners, especially in the coding. This shows a reverse LPP pattern. Probably it shows his/her strong and weak areas.

From the abovementioned considerations, we observed that learners S3, S7 and S14 did learning activities like LPP or Distributed Cognition. As S5 and S6 gave supports in all the phases and as time goes by, their supports increased. Therefore they became full participation in the learning community. Existence of them becomes important factors to success of the learning community.

6. Conclusions

This paper has proposed a learning support system for software engineering education based on the collaborative learning theories. We applied the system to an actual university course. While few communications were exchanged by the system, we could capture the learning process off-line by the function of registration of acknowledgements. We observed some learners behaved like LPP or Distributed Cognition, others behaved full participation in LPP.

References

[1] A. Collins, J. S. Brown, and A. Holum, Cognitive apprenticeship: Making thinking visible, American Educator, 1991.

[2] A. Hazeyama, A Case Study of Undergraduate Group-based Software Engineering Project Course for Real World Application, Proc. 1st International Symposium on Tangible Software Engineering Education (STANS2009), pp.39-44, 2009.

[3] J. Hollan, E. Hutchins and D. Kirsh, Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research, ACM Transactions on Computer Human Interaction, Vol.7, No.2, pp.174-196, 2000.

[4] S. Iio, A. Shiomi, Y. Matsuzawa and S. Sakai, Social network connected by acknowledgements in collaborative report writing, Proc. GN Workshop 2010, pp.69-74, IPSJ (In Japanese).

[5] A. Inaba and J. Toyoda, Underlying Learning Theories and Recent Researches on CSCL, JSiSE Magazine, Vol.16, No.3, pp.111-120, 1999 (In Japanese).

[6] J. Lave, and E. Wenger, Situated Learning - Legitimate Peripheral Participation, Cambridge University Press, 1991.

[7] J. Tholander and K. Karlgren, Support for Cognitive Apprenticeship in Object-Oriented Model Construction, Proc. CSCL2002, 2002.

[8] Y. Ye, Socio-Technical Support for Knowledge Collaboration in Software Development Tools, Proc. Workshop on Software Engineering and Usability Engineering, 2005.