

Usage Result of Problem Resolution Information Sharing System for a Software Engineering Course

Atsuo Hazeyama Kazuyuki Shimada and Yusuke Kobayashi

Department of Information Science, Tokyo Gakugei University
4-1-1 Nukuikita-machi, Koganei-shi, Tokyo, 184-8501 Japan
hazeyama@u-gakugei.ac.jp

Abstract. Software development is highly knowledge-intensive and collaborative work. Problem resolution processes are performed iteratively during software development. The authors have proposed a problem resolution process model that was based on reflection and collaboration for a software engineering project course. They have also developed a support system based on the process model and applied it to an actual university course for two years. The results from the stored log data and the contents from two years' usage showed that similar trend on the number of registered problem resolution information, ratio of classification by phase and by contents was seen in both years (problem resolution information on coding with respect to phase was registered most, and most information was that on technical with respect to contents). We observed knowledge transfer by the teaching staff in the discussion space and in the bulletin board system of group.

1. Introduction

Software development is knowledge intensive and collaborative work. Developers face many problems during software development and solve them by interactions with various resources [10]. There are no developers who possess all required knowledge. They gather necessary information while they progress development. They may ask others who possess expertise [10]. Ye described importance of knowledge collaboration in software development and developed a software engineering environment that enabled to register and retrieve sample programs, browse past discussions for the programs, and inquiry professionals [10].

Ishida et al. pointed out some issues on information and know-how sharing in industrial software organizations as follows: although software developers and/or managers have desire to know-how sharing, it does not function well [6].

Problem resolution information sharing systems have been developed for some areas [1, 2]. Answer Garden was developed to use in a help desk. It provided a branching network of diagnostic questions that helped users find answers. If the answer is not present, the system automatically sends the question to the appropriate expert and the answer is returned to the user and it is inserted into the branching network [1]. Although Answer Garden contributed to efficient problem resolution, it did not seem to focus on a learning aspect such as reflection.

Reflection is known as a process that is rooted knowledge acquired through problem resolution processes. Hatamura advocated “Shippaigaku” that learned from failures in order not to iterate similar ones [4]. “Shippaigaku” specified six attributes, i.e., event, background, progress, cause, disposition, and lessons learned, in order to describe a failure and transfer it to others.

We have been conducting a group-based software engineering project course. The goal of this course is for students to acquire knowledge and skills that are necessary for software development through their experience of collaborative software development by group. We proposed a problem resolution process model by collaboration and reflection, which aimed to solve problems that occurred during software development and to be rooted knowledge acquired through problem resolution [5]. We also developed a system based on the process model to share the problem resolution information (description of problem, the resolution process, solution, and lessons) students encountered. This system is different from an issue tracking system like Bugzilla, Dhruv [9, 2] from the viewpoint of the target scope. Issue tracking systems manage bug information and information on the disposition in the testing phase. On the other hand, we manage problem resolution information students encounter in all phases of software life cycle. We reported some results gained from its application to an actual university course as follows: 1) around 90% of the registered problem resolution information dealt with the implementation phase, 2) 95% of the registered problem resolution information dealt with the technical information and only 5% dealt with process information [5].

We would like to ascertain whether this is a transient phenomenon or not. We also ascertain state of reuse of shared knowledge.

The organization of this paper is as follows: section 2 gives an overview of our problem resolution process. Section 3 presents a support system. Section 4 reports application of the system to an actual university course and some results. Finally we conclude this paper.

2. A Process Model for Collaborative Problem Resolution

The problem resolution process we propose is consisted of the following four steps as shown Figure 1.

(1) Identification of a problem

When a learner or a group faces a problem, (s)he identifies the event and its background. Problems may come from results of inspection and/or testing, troubles in programming, and/or troubles in development environments.

(2) Information gathering

A learner or a learning group collects information which is necessary for problem resolution and considers causes of the problem. As the information source for problem resolution, we assume the supporting system we provide in this paper and/or external resources such as Web pages. We also assume to collect information by means of communications with peers in the course and/or the teaching staff.

(3) Disposition of the problem

The learner deals with the problem to remove the cause, which was considered based on the information collected in the step (2).

(4) Reflection

When the problem was solved, the learner reviews the problem resolution process and describes the lessons learned by the problem resolution process.

We ask learners for describing items that correspond to the steps. We adopt “Shippaigaku” as a framework for describing the problem resolution process as shown in section 1. Shippaigaku is a discipline, which aims at learning from failures and prevents similar failures by sharing them [4]. In the learning process, we regard reflection as activities of describing all the attributes specified by “Shippaigaku” for the corresponding problem. The problem resolution information that will be stored through the abovementioned learning process is available for other learners. At this time we think learning effectiveness will not be obtained if a learner only refers to the information. Therefore we provide a function that referrers describe their lessons based on the information they referred. These lessons lead to feedback information for the submitters of the problem resolution information.

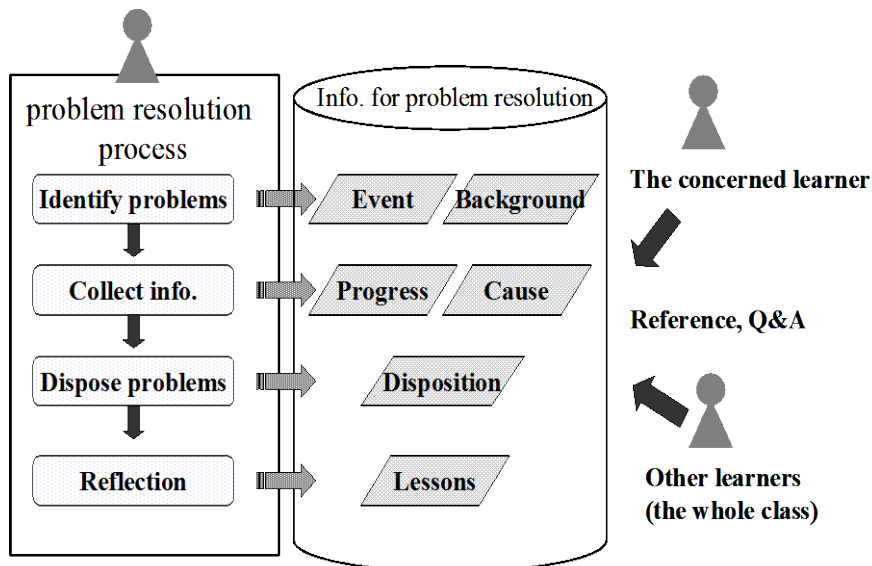


Fig. 1 Problem Resolution Process Model

3. Support System

We implemented a support system based on the abovementioned problem resolution process model. The system was implemented as a web application and a sub-system of “Shin-Gi-Tai” (Mind-Skill-Force) that was a group-based collaborative software development environment we developed [7]. “Shin-Gi-Tai” provides the following functions: document management, BBS (Bulletin Board System)-based

communication support, project planning and progress reporting, and issue tracking. The system we developed in this study has the following major functions. The information that is stored by the functions is available to all groups in “Shin-Gi-Tai”.

- * Registration of the problem resolution information: a user registers the problem resolution information. By registering it according to the input form that corresponds to the attributes defined by “Shippaigaku”, the context information is attached and reflection will be made by the user.
- * Browsing of the problem resolution information: users browse the registered problem resolution information. They can evaluate the information and/or give comments for it. Figure 2 shows an example screen shot of the problem resolution information.
- * Discussion space for the problem resolution: this function creates a space for discussions specific to an encountered problem. A user who would like to ask a question registers his/her question. In particular, as the context information is important for the encountered problem to be answered, description on the goal and on what (s)he has tried is mandatory. Then users enable to take three types of actions, i.e., browsing of questions, submission of messages, and browsing of the messages. From this function, a user can register the problem resolution information if the problem was resolved. The system associates the problem resolution information with the discussions.

KNOWHOW MENU: [list](#) [registration](#)

Browsing of the problem resolution information

event	Eclipse build path error?	dev. env.	tomcat: 5 mysql: 5 java: 1.4 eclipse: 3.2
phase	Coding&unit testing	usecase	?
back ground	Eclipse did not compile automatically after my program enabled to connect a database.	cause	Build path error occurred because of an unnecessary mysql connector
disposition	I searched for solution for my problem on the web, but did not find solution. Then I submitted a question to the discussion space and resolved my problem.	lessons Error messages tell me a hint of problem resolution, and the discussion space contributed to early resolution to my problem. Thanks a lot!
Web sites attached file	None	keyword	Eclipse, build
	None	link2QA	Link

 edit comment

2人中**2**人が Two out of two are assessed useful.
 このノウハウは役に立つと登録しています

Fig. 2 Screen shot of browsing the problem resolution information

Nakakoji et al. categorized communication in software development into coordination communication and expertise communication [8]. This function corresponds to expertise communication. Coordination communication is done in bulletin board system (BBS) of each group.

4. Evaluation

We show the results from two years' application of the system.

4.1 Overview of the course

We applied the system to an actual software engineering project course at our university in the 2007 and 2008 academic year. In this course, four or five students form a group. Each group selects one task from the two given by the instructor and is expected to complete their development via requirement analysis, design, implementation, and testing from scratch. Each group is required the system is implemented as a Web application with the Java technologies. During the project, the teaching assistants and the instructor (hereafter we call them the teaching staff) conduct inspection for requirement specification and several design documents. Faults detected during the inspection are required to be revised (the follow-up step was conducted). Acceptance testing is performed by the teaching staff for the system that is uploaded to a server machine and the development group conducts system testing for the same system. Faults are kept track by an issue tracking tool we developed.

Our department offers one and half years' programming courses by C language, one semester's course "automaton and language theory," and one semester's course "introduction of software engineering" just before the project course as related to the project course. "Introduction of software engineering" gives lectures on some software life cycle models, concept of object-orientation, modeling by UML (Unified Modeling Language), Web application development with Java technologies (including exercises). However, as all knowledge necessary for the assigned task can't be taught in the course, the students are required to investigate and/or exchange information to resolve their encountered problems in the project course.

In the 2007 academic year, the number of student developers was twenty-two and five groups were organized. The development was during 31 October 2007 through 22 January 2008. All groups finished their requirement analysis and design phases till mid December 2007. Integration of source codes written by their members was conducted around 10 January 2008. System testing by developers and acceptance testing by the teaching staff were done in the following two weeks.

In the 2008 academic year, the number of student developers was twenty-six and six groups were organized. The development was during 30 October 2008 through 21 January 2009. All groups finished their requirement analysis and design phases till mid December 2008. Integration of source codes written by their members was conducted around 10 January 2009. System testing by developers and acceptance testing by the teaching staff were done in the following two weeks. Table 1 summarizes a profile of the two years' projects.

Table 1 Summary of two years' projects.

Items	Year	
	2007	2008
No. of students	22	26
No. of groups	5	6
Duration of project	31 Oct. 2007 – 22 Jan. 2008	30 Oct. 2008 – 21 Jan. 2009

All groups finished their development in both years. At the start of the project, we asked all the students for registering at least one problem resolution information they encountered in the project. To register duplicated information with others is allowed if the information was acquired in the course of a student's problem resolution process.

4.2 Results

We show the results from the following viewpoints: classification by phase, classification by contents (technical versus process), and situation of knowledge collaboration.

4.2.1 Classification of the problem resolution information by phase

39 problem resolution information were registered in 2007 and 42 in 2008 (a student registered 1.8 in average in 2007, and 1.6 in average in 2008). Figure 3 shows the ratio of the registered problem resolution information by phase in the 2007 and 2008 course. We classified the phase into requirement analysis and specification, design, coding (including information on a development environment), testing, project management (PM), and others. As this figure shows, both courses present similar trend by this classification. That is, information on coding was registered quite most and then information on design was registered. No information on requirement analysis and PM was registered.

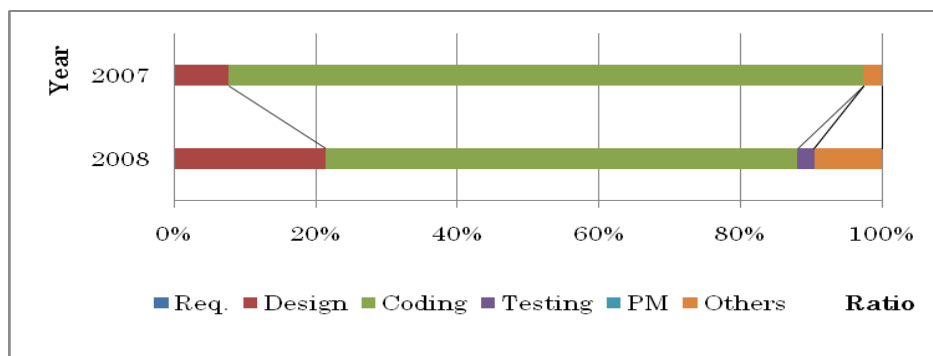


Fig. 3 Classification of the registered problem resolution information by phase

4.2.2 Classification of the problem resolution information by content

We classified the problem resolution information into categories proposed by Ishida et al. [6]. We classified the content into technical information and process information. Figure 4 shows the result. As this figure shows, more than 90% were technical information. Figure 2 shows an example of the registered problem resolution information.

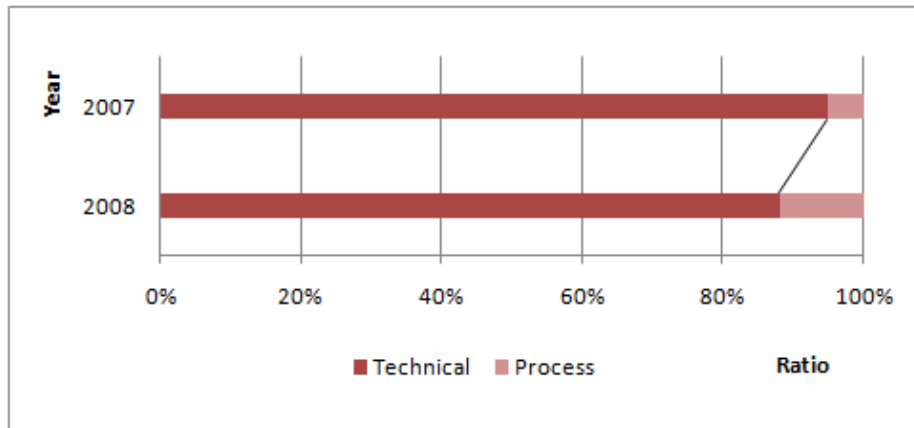


Fig. 4 Classification based on the contents of the registered problem resolution information.

4.2.3 Result on knowledge collaboration

(1) Knowledge collaboration in the discussion space

In the discussion space, eight questions were submitted in 2007, and one was submitted in 2008. They all led to resolution. Six questions were resolved by only one response. One question was followed by a counter question by a teaching assistant (TA) because the contents of the question were too short to answer it. In this case, the student who asked this question described detailed explanation, in addition, seemed to find a root cause for the problem as the result of his/her investigation. The TA who asked a counter question gave advice with respect to a solution to remove the cause. The rest of questions was “Current date-time can’t be inserted into a database” and “Build problem.” These two questions were resolved by exchanging messages between the questioner and responder(s) several times.

Question “Current date-time can’t be inserted into a database” was resolved by the following process: a teaching assistant asked the questioner for presenting details on the current situation and what (s)he wanted, and gave advice based on the responses from the questioner and the investigation results from internet search by the teaching assistant. Finally the problem was resolved in five turns.

Question “build problem” was also resolved by exchanging messages between a questioner and responder(s). In this case, first of all, a student (S1) asked the following question “*I created a file using Eclipse and updated it. After updating, I executed a program. However, the update was not reflected.*” A member of another group (P1) predicted a root cause. However, (s)he did not suggest a concrete solution. Then a teaching assistant (P2) asked the questioner for several inquiries and gave

several advices, but the advices did not resolve the problem. Furthermore another teaching assistant (P3) who is a master course student gave advice to check a log file. By contents of the log file presented by the questioner, P2 found out a solution and gave a concrete advice to solve the problem. We show the message sequence as follows. This case worked escalation mechanism well.

Questioner (S1) at 21 December 2007: *I created a file using Eclipse and updated it. After updating, I executed a program. However, the update was not reflected.*

P1 at 08:37, 21 December 2007: *I think build is not done.*

Questioner (S1) at 12:46, 21 December 2007: *Maybe I think so. I check "automatic build" now. I checked "build all" and tried to build my program, but it did not work. Where should I modify?*

P2 at 14:31, 21 December 2007: *Do you mention your update is not reflected into the program? Cache may cause your problem. Are there errors in the "work" folder?*

Questioner (S1) at 14:43, 21 December 2007: *Where should I examine for that matter?*

P2 at 15:16, 21 December 2007: *There will be "work" folder under your eclipse project. Compiled JSP files are stored there. Cache will also be stored there. Troubles happen if the folder has errors or old cache remain. I have one question: do you have any concrete error messages? Isn't your update of files merely recognized by Tomcat?*

Questioner (S1) at 15:35, 21 December 2007: *Maybe I think so. I wrote "System.out.println();" in a servlet program, but it was not executed. I have deleted the contents of the "work" folder. Was that incorrect?*

P2 at 15:54, 21 December 2007: *No. If you find "X" mark on the "work" folder, please delete the contents of the "work" folder. In what situations did this problem emerge? That is, did you encounter this problem suddenly while the program ran well till yesterday? Or has not the problem resolved so far? Is the version of Java compiler 1.4? You can check the version and level of Java compiler by menu "window" -> "setting" -> "Java" -> "compiler" -> "compiler level" in eclipse.*

P3 at 16:34 21 December 2007: *If you can't resolve your problem by checking the abovementioned advice, please check the following: is a message "project can't be built until the build path is resolved" shown in a list of "problem" from "presentation of view" of "window" menu?*

Questioner (S1) at 18:04, 21 December 2007: *Dear P2, I suddenly encountered this problem. Yesterday, the system ran well. I checked the version number and level of the compiler. The level was 1.4.*

Dear P3,

I found following two error messages: "The project will not be built until error with respect to the build path is resolved" and "a library necessary for project 'C:\eclipse3.2.1\workspace\KOKKO\WEB-INF\lib\mysql connector-java-3.0.9-stable-bin.jar' is not found." I do not know how to read these error messages. Thank you for your help.

P2 at 18:15, 21 December 2007: *That will be a true cause!! Please append a path of the jar file of the mysql connector to the build path of your project. You can paste the jar file in the WEB-INF folder and then append the build path by an operation of clicking right button on the project.*

Questioner (S1) at 19:13, 21 December 2007: *Two mysql connectors existed in my project. When I deleted one of them, the problem has resolved!! As I could not resolve this problem by myself, thank you very much for your support.*

(2) TA and instructor as knowledge broker

In spite of problem resolution information being registered in the system, some students failed to find it out that helped to resolve their problem. The teaching staff played a role of knowledge broker to inform them of the applicable information. The following is examples:

Example 1

Student S2: *I do not know how to implement a function of sending a mail, although I investigated the way.*

Instructor: *The problem resolution information registered at 20:26, 22 January 2008 may help you. Please refer to the information. I clearly remember that the student who registered this information spent many efforts and finally resolved the problem.*

Example 2

Student S3: *After compiling a system, I designated a URL in my web browser. However the 404 error happened.*

A teaching assistant: *You can find solution by accessing the problem resolution information registered at 09:34, 1st November 2007.*

(3) State of reuse of the problem resolution information

From the feedback for the problem resolution information, we found at least seven problem resolution information was reused to solve problems of some students (how to backup and restore a database, control from a sub-page to a main page, automatic mail distribution, and association class). However, it is difficult to ascertain true state of reuse of the information, because users have to visit the page and evaluate it after their problem has fixed (they may not come back the page after fixing their problem).

4.3 Discussions

4.3.1 On problem resolution information

Similar trend on the number of registered problem resolution information, ratio of classification by phase and by contents was seen in both years. Problem resolution information on coding with respect to phase was registered most. In this course, students are assigned to different use cases. Thus technologies required differ respectively. A problem of a student must basically be resolved by him/herself. Therefore I think problems seem to be overt. On the other hand, design is group work in this course. This course also exposes design inspection. Groups are required to revise artifacts according to the inspection comments. It may be difficult to abstract from inspection comments to know-how. Much of the problem resolution information in the design phase was that on how to use tools (UML editor) (in 2007, two out of

three problem resolution information and in 2008, two out of six were that on how to use tools).

According to analysis of the registered problem resolution information from the viewpoint of contents, most information was that on technical. In 2007, two students who have less confidence on technical aspects registered the problem resolution information as follows: “*It is important to possess an attitude that asks how to investigate something, not that asks for the direct solution for the problem you face*”, “*You should not give up even though you can’t understand. Participate in your project in a positive manner, and your team members support you when you have troubles.*” They were on the process aspect how a problem should be resolved.

4.3.2 On knowledge broker

In this case study, we observed knowledge transfer by the teaching staff in the discussion space and in the BBS of a group.

Boden and Avram studied knowledge distribution between sites of geographically distributed software projects in small companies [3]. They concluded importance on oral communications by means of usage of Skype, business trips, bridging knowledge by developers who stay in another site.

On the other hand, our course is not geographically distributed, rather temporally distributed because students have their different schedules. Therefore synchronous communications are limited. People (developers and the teaching assistants) also change off very fast in the course. Oral communications are not suitable for this context because it is volatile. For these reasons, we adopted a method to describe, store, and share problem resolution information. Nakakoji et al. [8] proposed nine items as a design guideline for expertise communication support. They described that if documents or codes exist to obtain some information, they should be used as much as possible so that communications do not occur. Our system is satisfied with this item of the guideline.

5. Conclusion

This paper has analyzed state of the problem resolution information sharing in a software engineering project course from two years’ usage of the problem resolution information sharing system. Similar trend on the number of registered problem resolution information, ratio of classification by phase and by contents was seen in both years (problem resolution information on coding with respect to phase was registered most, and most information was that on technical with respect to contents). In this case study, we observed knowledge transfer by the teaching staff in the discussion space and in the BBS of a group.

We would like to ascertain the status of reuse of the problem resolution information as future work.

Acknowledgement

The authors would like to thank anonymous reviewers for their comments to improve this paper. They also would like to thank Mr. Hiroaki Yamada for his help to gather log data of the system usage.

References

1. M. S. Ackerman, and T. W. Malone, Answer Garden: a tool for growing organizational memory, Proceedings of the ACM SIGOIS and IEEE CS TC-OA Conference on Office Information Systems, pp.31-39, ACM Press, 1990.
2. A. Ankolekar, K. Sycara, J. Herbsleb, R. Kraut, and C. Welty, Supporting Online Problem-Solving Communities with the Semantic Web, Proceedings of the 15th World Wide Web Conference (WWW2006), pp.575-584, ACM Press, 2006.
3. A. Boden, and G. Avram, Bridging knowledge distribution - The role of knowledge brokers in distributed software development teams, Proceedings of the ICSE 2008 Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2009), ACM Press, 2009.
4. Y. Hatamura, Learning from Failures, Kodansha, 2000 (In Japanese).
5. A. Hazeyama, K. Shimada, and Y. Kobayashi, A Collaborative Problem Solving Support System for Group-based Software Engineering Project Course and Its Application, Proceedings of the Seventh International Conference on Creating, Connecting and Collaborating through Computing (C5 2009), pp.74-78, IEEE Computer Society Press, Kyoto, Japan, January 2009.
6. A. Ishida, et al., Research on the Problem in Information-sharing and Its Efficiency in Software Development, (In Japanese).
7. M. Miura, Y. Kobayashi, K. Shimada, K. Takahashi, S. Seiki, and A. Hazeyama, A Proposal of Integrating Personal and Community Support with Learning Environment for Group-based Software Engineering Course, Proceedings of the 2nd International Conference on Knowledge, Information and Creativity Support Systems (KICSS 2007), pp.144-151, 2007.
8. K. Nakakoji, Y. Ye, and Y. Yamamoto, Interaction Design for Supporting Knowledge Communication in Software Development, Annual Conference of the Japanese Society for Artificial Intelligence (JSAI 2009), Takamatsu, Japan, June, 2009 (In Japanese).
9. N. Serrano, and I. Ciordia, Bugzilla, ITracker, and other bug trackers, IEEE Software, Vol. 22, No.2, pp.11-13, 2005.
10. Y. Ye, Socio-Technical Support for Knowledge Collaboration in Software Development Tools, Proceedings of the Workshop on Integrating Software Engineering and Usability Engineering, pp.39-51, 2005.