Chen, W. et al. (Eds.) (2016). Proceedings of the 24th International Conference on Computers in Education. India: Asia-Pacific Society for Computers in Education

Networked Tutoring Support System for a Programming Class based on Reusable Tutoring Content and Semi-automatic Program Assessment

Yasuhiro NOGUCHI [a*], Ryota OSAWA [b], Koichi YAMASHITA [c], Satoru KOGURE [a], Tatsuhiro KONISHI [a] & Yukihiro ITOH [d]
[a]*Faculty of Informatics, Shizuoka University, Japan*
[b]*Graduate School of Informatics, Shizuoka University, Japan*
[d]*Faculty of Business Administration, Tokoha University, Japan*
[d]*Shizuoka University, Japan*
*noguchi@inf.shizuoka.ac.jp

Abstract: In programming classes, teachers and teaching assistants (TAs) cooperate to support students' programming exercises. A teacher is required to support many students and sometimes hand them over to another teacher or assistant. However, it is difficult to check students' source code without interrupting the students, to share tutoring information among teachers and TAs, and to reuse tutoring documents created for one student. In this paper, we propose a tutoring support system based on reusable tutoring content and semi-automatic assessment of students' code and report on a trial conducted in university programming classes that found the system could spontaneously promote tutoring content sharing and reuse.

Keywords: Class support system, novice programming education, information sharing, decision support

1. Introduction

Recent novice programming classes are typically one-to-many teaching situations. First, a teacher teaches programming theories to many students in the teaching section. Next, in the exercise section, the teacher and teaching assistants (TAs) cooperate to support the students' programming development (tutoring). In general, each student writes his own program code at his own computer display. Thus, the major role of the teacher and TAs (hereinafter, "teachers") is to circulate among the students to answer their questions, check the work of students having trouble, and to give them learning support. However, there are some problems in tutoring efficiently in exercise sections.

A) When teachers check a student's work, they can see only the section of code shown on the display, and if they need, they must interrupt the student and ask to see the rest of the code.
B) When a student asks the teacher about code she has just written, the question must be answered after rapidly reading the code and finding bugs. However, it is difficult for not only TAs but also experienced teachers to answer quickly.
C) Teachers cannot remember every detail of their discussions with students. Thus, later directions and discussions lack the earlier context.
D) It is hard for teachers to find the opportunity to share information during class, and scheduling regular meetings to share information reduces the time for tutoring. Therefore, when teachers handed students over, directions and discussions lack the earlier content or lose time to ask them to the students. In general, they cannot answer them correctly because they required help by their misunderstanding.

E) Teachers often create tutoring documents with hand-drawn figures when answering students. When another student asks a similar question, it is difficult to reuse hand-written tutoring documents, so similar figures must be made a new.

We argued that an educational support system that reduces these problems would enhance teachers' tutoring. Other classroom support systems focused on programming exercise classes have been developed. Harada et al. (2013) proposed a system to support question-answering in computer exercise classes. The system shows the seats of questioners and shares responses to common questions understandable without the details of each student's exercise situation. Kato and Ishikawa (2012) created a system to support learning portfolios for faculty development activities. This system creates a learning portfolio with the learning situation of each student and the class, adequacy reports for examination design, and records of instructions and discussions in classes. This system is designed to accumulate class/student information by hand to refine the class after it ends. While some aspects of these systems might be effective for teachers circulating among students, they do not fully address the particular problems. Kakimori and Yasutome (2012) proposed a web monitoring system that visualizes students' learning conditions by showing the problem each student is working on and the progress in graph and/or list format. TAs may be notified via the seating chart of students having trouble by estimating their progress and past achievements, and this system supports semi-automatic assessment of students' code based on the presence of specific statements in their code and test results using test cases defined by the teachers. Hachisu and Yoshida (2014) designed a system that generates error correction questions for programming exercise classes from correct programs and error patterns. This system can check whether students' answers are correct.

As mentioned above, some classroom supporting systems operate by visualizing students' exercise situations and/or sharing assessment information for students' modified source codes, but they are not focused on the needs of teachers circulating among students in exercise classes, especially as regards teachers' tutoring activities. In this paper, we describe the architecture of our class support system and report observations from using the system in three novice programming classes. Our system focuses on these activities and supports information-sharing among students and teachers by evaluating the students' source code and/or reusing tutoring contents.

2. Networked Tutoring Support Systems

2.1. Concept

We designed a decision support system with the following two functions that are considered necessary to address the five problems above.

(α) Automatic Code Collection and Assessment
When a student builds his program codes, his IDE should automatically transfer them and the results of the build process. This addresses problem A) so that the transfer can proceed implicitly in accordance with a student's general activities. Moreover, the transferred program codes should be automatically assessed by a semi-automatic program code assessment system, which addresses problem B) by showing teachers the error statements returned by the system. Totally, these functions enable teachers to prepare their support without interrupting students' coding activities.

(þ) Supporting the Accumulation and Reuse of Tutoring Documents
Tutoring documents should be accumulated by the system. Reuse of tutoring documents follows three patterns. In Pattern 1, past tutoring documents for the student are reused. This helps the teacher tutoring the student grasp the student's context and addresses problem C). In Pattern 2, teachers reuse tutoring documents by sharing them to better support a student's learning, which addresses problem D). In Pattern 3, tutoring documents created for one student serve as the basis for new documents for another student. This prevents unnecessary duplication of work with each student, which addresses problem E). These patterns naturally occur in combination. Totally, teachers can share earlier context for each student by sharing documents and reuse them for another student.

2.2. *Use case*

This system is typically used as follows. Teachers use a wide display tablet computer connected to the system, and as they circulate among the students, their codes are entered into the system, semi-automatically assessed, and transferred to each tablet computer. Tutoring documents and learning support notes created in each tutoring session are synchronized between the system and all tablet computers. Notifications of updating information for each student are broadcasted to all tablet computers. The teachers know which students are going forward and/or pausing their development; they can focus on these students to find their trouble and to prepare to support.

1) While circulating among students, teachers find students having trouble personally and through the system's assessments of their code.
2) Once a teacher observes a student having trouble or is asked for help, she opens the student information page to confirm the note for the student. If needed, she may check the student's program code in more detail without interrupting the student's coding activities.
3) She addresses the problem with figures and/or documents that she or other teachers created earlier. She reuses the figures and/or documents to support the student. If needed, she creates new documents based on the earlier figures and/or documents. The new documents are automatically shared to other teachers.
4) She should note directions for the student.

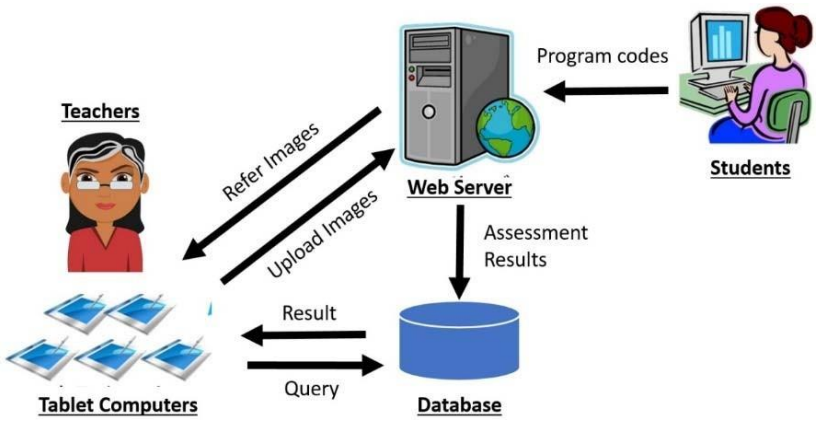

Figure 1: Architecture of the System

2.3. *Architecture*

The system is designed as a client-server system. Figure 1 shows the components of the system, which consists of a web server, database, tablet computers for each teacher, and a compiler with an automatic transfer function built into students' IDEs. In this system, the compiler automatically transferred program codes developed by each student and their build results to the web server. The system semi-automatically assesses them and returns the result to the database (Suzuki et al. 2007; Kogure et al. 2015). The teacher's tablet computer is synchronized with the database. When a teacher creates tutoring documents and/or learning support notes, these are registered in the database and update notifications are sent to each tablet computer.

This system was required some preparations. Teachers creates seating charts for their classes. By using an interface of the system, the teachers should arrange students on seats of the classroom. To use semi-automatically assessment function, the teachers should upload correct source codes with tags. This semi-automatically assessment function can accept extra tags for supporting variety of implementations. The teachers creates an example of correct source code; they add tags to chunks of statements that are permutable each other; they add tags to a chunk of statements that has other implementations. It enables the function to assess different variations on students' source codes that are correct. As for students, they should prepare their programming environment with the system's compiler that includes an automatic transfer function.

2.4. Interface

In this system, teachers can access the student information page in a table computer. While teachers circulate among students, they can check each student's status via "Seating Chart" of the classroom and/or "Search" by student's ID/name. The teacher can check the system's assessment of the student's source code and learning support notes for the student written in earlier tutoring session. The student's program codes highlighting where bugs might exist help teachers rapidly reference problems in a student's source code. The learning support notes enable teachers make their tutoring plan for the student based on other teachers' tutoring history for the student.

When a teacher starts tutoring a student, she can create tutoring documents on the system for him. Figure 2 shows such a tutoring document. The system supports handwritten and keyboard input on the tablet computer. This system supports some drawing features: pen color, pen width, line type, and undo/redo function. If the teacher wrote on paper, only the final version of the tutoring document would remain, whereas the system retains not only the final version but each earlier stage of the document. In drawing her tutoring content, she saves the content by each stage, and these stages are saved as a sequence chained by links.

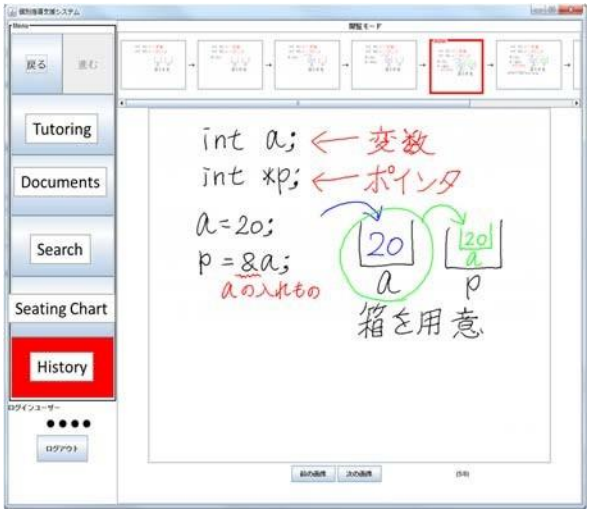

Figure 2. An Example of Reusing Content

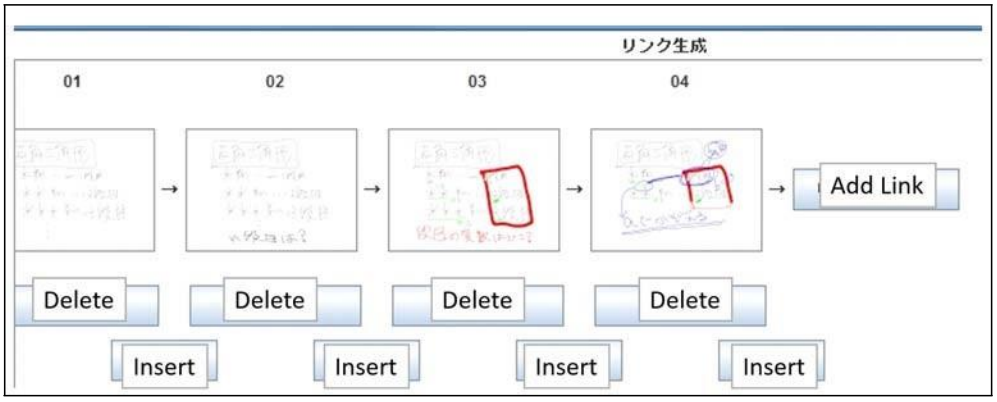

Figure 3. An Example of Creating Content with Reusing Existing Contents

The top of Figure 2 shows the stages of a tutoring document with each stage linked in sequence. Teachers can access each stage of the document by clicking its screenshot and/or the "Forward" or "Back" button. By using these buttons, the teachers can reproduce the picture-story show when the document was handwrote for the first student's question. This allows other teachers to use the document as well as the teacher who created it and to create new documents based on any stage of the document. Figure 3 shows that a new content is creating with reusing earlier contents in the system.

The teachers can pick up each stage or part of stages of documents to new content. They creates new contents by creating new stages and linking between earlier contents' stages and new stages. Even when a tutoring document was used to customize a new tutoring document, the system keeps the original, which allows the first teacher to recall the context of the document later.

3. Results and Discussion

3.1. Conditions of Use in Programming Class

Our system was used in several novice-programming classes in a university. These classes were open to 2nd-year university students studying business administration and used the C programming language. Table 1 shows the details of three classes that used our system. All three were managed by the same teacher, who managed Class 1 alone and Classes 2 and 3 each with a different TA. In these classes, students were strongly encouraged to ask the teachers anything. Our system was refined based on the feedback from Classes 1-3. We sought to determine following five points.

(1) The system does not obstruct the smooth operation of the class.
(2) The frequency that tutoring documents and/or direction notes are accumulated.
(3) The efficiency of the system in tutoring a student with the help of previous documents and directions.
(4) The efficiency of the system in information-sharing among teachers.
(5) The efficiency of the system when teachers reuse documents created for other students.

As Class 1 had only one teacher, it was excluded from (4). From practical use in the three classes, the following were obtained: (a) observations that these classes were operated with the system, (b) the records of all user operations in the interface of the system, (c) accumulated tutoring documents and direction notes, and (d) the opinions of the teachers in interviews.

Table **1**: Programming Classes Using the System.

	Name	Teacher and TAs	Class Length	No. Students	Topics
1	Programming I	1 Teacher	180 min	5	Arrays and double loop
2	Exercises for Professional Basis	1 Teacher and 1 TA	90 min	5	Arrays and double loop
3	Exercises for Professional Basis	1 Teacher and 1 TA	90 min	5	Arrays and double loop

3.2. Results and Discussions

On point (1), the teacher stated in the interview that the classes were not delayed by using the system compared with previous classes not using the system. The records of the users' operations showed that the users performed some operations indirectly until they became familiar with the system. On point (2), 62 tutoring documents and 6 learning support notes were accumulated in the three classes. The system could accumulate a certain number of tutoring documents. The number of learning support notes accumulated were none in Class 1, four in Class 2, and two in Class 3. On point (3), records of the users' operations show that a user used the same tutoring documents for the same student four times. In this case, documents from an earlier tutoring were used because the learning support note did not have a link to the documents. There is also an instance of tutoring documents being shared between a teacher and a TA, so that a different teacher could follow the student from the last topic. On point (4), accumulated information was shared among the teachers. In the interviews, the teachers confirmed that the learning support notes could be shared to show the level and/or current situation of the student. The records of the users' operations showed that shared learning support notes and/or tutoring documents were especially used when handing on a student to the other teacher during times of busy questioning. In the interviews, they stated that the tutoring documents worked for guessing what the student needed even if the learning support notes could not be found. Above all, students' information could be shared without interrupting the other teacher. On point (5), the records

showed tutoring documents created for one student were reused more than 10 times; according to the users, this increased efficiency by reducing the need to write duplicate figures.

These results are summarized as follows.

- The system helped teachers accumulate and reuse their tutoring documents.
- The teachers could use shared tutoring documents to guess the tutoring activities of the other teacher for a given student.
- Learning support notes were not accumulated as much as tutoring documents and thus could not be used as effectively in tutoring.

In another interview, the teachers stated that because the function for learning support notes was located in a different category of tutoring documents in the system interface, they did not write learning support notes after they wrote their tutoring documents, and they tried to guess what the other teacher told the student from the tutoring documents. On the other hand, they commented that shared tutoring content was effective evidence of the TAs' work. It enables the teacher to adjust the TAs' tutoring guidelines, while the TAs can spontaneously adjust their tutoring content following the teacher's policy, which allows them to mutually enhance the quality of their tutoring.

4. Conclusion

In this paper, we proposed a networked tutoring support system based on reusable tutoring content and semi-automatic code assessment. It was applied in several university classes, where it was observed that the teachers used the system spontaneously for sharing tutoring information, and the system promoted the accumulation and reuse of tutoring documents. Students could be tutored based on the records of the other teacher. However, even though the system included learning support notes to record notes for each student, teachers guessed at what was needed from the shared tutoring documents. In the future, we plan to redesign students' tutoring records and the roles of tutoring documents and learning support notes to improve the system interface. In addition, we think the system should allow students to share their information actively.

Acknowledgements

This study was supported by Japanese Grants-in-Aid for Young Scientists (B) 15K1610.

References

Harada, M., Yamaguchi, T., & Nagai, Y. (2013). Individual Student Support System for Teacher and TAs using Mobile Devices in Exercise Classes, *Proceedings of the 18th International Symposium on Artificial Life and Robotics 2013*. (pp. 126-129)

Kato, T., & Ishikawa T. (2012). Design and Evaluation of Support Functions of Course Management Systems for Assessing Learning Conditions in Programming Practicums, *Proceedings of the 12th International Conference on Advanced Learning Technologies*, (pp. 205-207).

Kakimori, T., & Yasutome, A. (2012). TA assignment system for TA activity, *Proceedings of the 37th Annual Conference of JSiSE*, (pp. 374-375). (in Japanese)

Hachisu, Y., & Yoshida, A. (2014). A Support System for Error Correction Questions in Programming Education, International Association for Development of the Information Society, Paper presented at the International Conference e-Learning.

Suzuki, H., Konishi, T., & Itoh, Y. (2007). A system assisting a teacher in evaluating learner's programs based on algorithm representations represented by using abstract data structures, *Transactions of Japanese Society for Information and Systems in Education*, *24*(3), 167-186. (in Japanese)

Kogure, S., Nakamura, R., Makino, K., Yamashita, K., Konishi, T., & Itoh, Y. (2015). Monitoring system for the effective instruction based on the semi-automatic evaluation of programs during programming classroom lectures, *Research and Practice in Technology Enhanced Learning (RPTEL)*, *9*(3).