Learners’ Self Checking and Its Effectiveness in Conceptual Data Modeling Exercises

Seiichi KOMIYA††††, Members, Yuki HIRAI††††, Nonmember, and Keiichi KANEKO††, Member

SUMMARY Conceptual data modeling is an important activity in database design. However, it is difficult for novice learners to master its skills. In the conceptual data modeling, learners are required to detect and correct errors of their artifacts by themselves because modeling tools do not assist these activities. We call such activities self checking, which is also an important process. However, the previous research did not focus on it and/or the data collection of self checks. The data collection of self checks is difficult because self checking is an internal activity and self checks are not usually expressed. Therefore, we developed a method to help learners express their self checks by reflecting on their artifact making processes. In addition, we developed a system, KIfU 3.0, which implements this method. We conducted an evaluation experiment and showed the effectiveness of the methodology. From the experimental results, we found out that (1) the novice learners conduct self checks during their conceptual data modeling tasks; (2) it is difficult for them to detect errors in their artifacts; (3) they cannot necessarily correct the errors even if they could identify them; and (4) there is no relationship between the numbers of self checks by the learners and the quality of their artifacts.

key words: Conceptual data modeling, self checking, artifact-making process, UML

1. Introduction

Conceptual data modeling (CDM) [1], [2] is an important activity in database design [3], [4] in which we detect concepts and construct a model of them with relations among them, where the concepts are things and events that are handled in software [2]. Many computer science departments in universities teach CDM in classes of database design [5]–[7]. It is important for learners not only to understand the methodology of CDM but also to acquire practical skills regarding the methodology. However, CDM is a difficult task for novice learners [8], [9]. Teachers typically conduct exercises to cultivate and evaluate these skills, which focus on topics close to real problems. In each exercise, the teacher tries to understand each learner’s achievement as well as the tendency of the learners as a group (e.g. common mistakes made by many learners) by checking the learners’ artifacts. However, only checking the artifacts submitted by learners is not enough for the teacher to grasp the learners’ processes of making artifacts and their thinking behind these processes. Without understanding the processes and thoughts going through the learners’ minds, it is difficult for the teacher to evaluate the degree to which the learners can utilize the learned methods in practice.

In exercises of software modeling including CDM, it is important for the learners to acquire the skill to find out errors in their artifacts by themselves. In programming exercises, learners can recognize existence of errors from error messages or unexpected execution results when they compile their source codes or execute the compiled codes. However, there is no such support in software modeling environments. Therefore, it is important for learners to repeat self checking of their artifacts actively to find out errors, and correct them. Self-checking is an important practical skill for learners.

In this paper, we define a self check (SC) as a three-step activity: ‘Start of SC’, ‘Search for Errors’, and ‘Correction of Errors’. We show a success model of self checking in Fig. 1. In the exercise, a learner starts to make an artifact. The learner conducts SCs of his/her artifact one or more times. The learner searches for errors in his/her artifact after he/she starts an SC. If the learner discovers errors, he/she modifies the artifact to correct them. Then the learner comes back to make his/her artifact again. The exercise ends with the learner submitting the finished artifact.

CDM consists of two steps: (1) Making concepts and (2) Making relations. In general, learners make conceptual data models in the order of Step (1) and Step (2). Of course, learners often notice errors in their models in Step (2) and return to Step (1) to correct them. Therefore, timings that learners conduct SCs can be summarized as shown in Table 1. To make high-quality models, it is necessary to conduct SCs at ‘Before submission’. Returning to the previous step takes more effort to correct errors. To prevent it, it is necessary to detect errors at the earlier timings. For this purpose, it is also important to conduct SCs properly in the making process of artifacts. However, it is unclear when the learners actually conduct SCs.

In general, it is difficult for novice learners to find...
out errors in their artifacts by themselves. Tanaka et al. [10], [11] reported that novice learners have a tendency to submit their artifacts even though they include errors. This is caused by unsuccessful SCs. Therefore, it is important for teachers to know at which of the three steps of SCs (Fig. 1) the learners stumble. If we can clarify the steps that are difficult for the learners, teachers will be able to provide the instructions specific to the steps. However, it is difficult for the teachers to grasp the situations of learners’ SCs because SCs are learners’ mental activities, and the learners do not usually express their SCs to the teachers. Hence, we propose a method to collect learners’ SCs by supporting the learners to express SCs and that can record them in association with their artifact making processes.

(2) To clarify the difficult steps of SCs for learners: ‘Start of SC’, ‘Search for Errors’, ‘Correction of Errors’.

In this paper, we define RQ1 for Goal (1), and RQ2 to RQ5 for Goal (2) as follows:

RQ1: Is our method effective for collecting learners’ SCs?
RQ2: What proportion of learners can start SCs in their artifact making processes?
RQ3: When do learners start SCs in their artifact making processes?
RQ4: At what percentage of SCs can learners detect errors?
RQ5: At what percentage of SCs can learners correct the errors they found?

RQ2 and RQ3 are related to ‘Start of SC’ in Goal (2). We clarify whether ‘Start of SC’ is difficult or not for learners by answering RQ2. Then, we clarify the timings of learners’ SCs by answering RQ3. We also clarify whether ‘Search for Errors’ is difficult or not for learners by answering RQ4. Furthermore, we also clarify whether ‘Correction of Errors’ is difficult or not for learners by answering RQ5.

3. Related Work

In this section, we describe related work on augmentation of SCs and collection of thoughts by developers during software modeling and/or requirement analysis.

Suraweera and Mitrovic developed KERMIT, which is a learning environment of Entity Relationship (ER) modeling [12]. KERMIT provides feedback for each learner in ER modeling through an agent. The feedback includes suggestions of actions that learners should take next, or indications of errors in their artifacts. The feedback also includes recommendations for the learners to start SCs with a check list. However, we cannot grasp whether learners conducted SCs or not through KERMIT. Moreover, we cannot elicit the learners’ thinking behind SCs.

Batra and Davis clarified similarities and differences between experts and novices engaged in a CDM task [1]. They collected thoughts of experts and novices by think-aloud protocol analysis. However, it is difficult to apply this method to many learners without thought-expression training, which makes it difficult to apply this method to actual exercises.

Oyama et al. developed a CAPIS-model method that helps software designers express thoughts and decisions in software design tasks [13]. Oyama et al. conducted an experiment to show that experts could express thought processes by the CAPIS model. However, the CAPIS model requires the software designers to describe many items in detail, which takes much time and effort. Therefore, this method is difficult for novices, who are the focus of our research.

![Fig. 1](image-url) A success model of self checking.

<table>
<thead>
<tr>
<th>Table 1 Timings of SCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing</td>
</tr>
<tr>
<td>Making concepts</td>
</tr>
<tr>
<td>Between making concepts and making relations</td>
</tr>
<tr>
<td>Making relations</td>
</tr>
<tr>
<td>Before returning</td>
</tr>
<tr>
<td>Before submission</td>
</tr>
</tbody>
</table>
Nakahama et al. developed a method to collect learners’ thoughts during the processes of generating requirement specifications in software development exercises [14]. They reported that a reflection on the changes in the specifications is effective for the learners to recall the thoughts they had while coming up with specifications. Although the target of their study is different from ours, their method to collect learners’ thoughts is applicable to CDM exercises in which we are interested.

Tanaka et al. developed a system that supports learners to express their thoughts during CDM exercises by reflecting on their artifact making processes [15]. They defined an “edit event”, and regarded the artifact making process as a sequence of edit events. An edit event is an aggregation of data, and includes the type (Create, Update, Delete) and the target of the corresponding edit operation on the artifact, the time stamp when the edit operation occurred, and so on. Their system collects artifact making processes as sequences of edit events, and provides learners a function for reflecting on their artifact making processes by replaying them. According to them, reflection on the artifact making processes is effective for learners to recall what they thought during these processes. In the study presented here, we extend their system KIfU 2.0 to collect SCs.

4. A Method to Collect Learner’s SCs

In this section, we describe a method to collect learner’s SCs during a CDM exercise with the artifact making process. Concretely, the method collects data that can replay the timings of the SCs conducted by the learner in the artifact making process, the contents of the SCs, and the states of the artifact at these times.

We assume here that the data of an SC consist of the following six elements E1 to E6, though not all of these elements may be collected. These are called the elements of SC (ESCs).

- **E1 (Timing):** The timing when the learner started the SC.
- **E2 (Artifact):** The snapshot of the artifact when the learner started the SC.
- **E3 (Checked items):** The items checked by the learner in the SC.
- **E4 (Thoughts):** What the learner was thinking in the SC.
- **E5 (Error detection):** The boolean value that indicates whether the learner detected an error in the SC.
- **E6 (Error content):** The content of the error(s) that the learner detected in the SC.

There is a problem regarding the collection of ESCs. If we ask a learner to express his/her ESCs during the artifact making process, his/her cognitive load increases. On the other hand, if we ask him/her after the process, it may be difficult for him/her to recall all of his/her ESCs. Therefore, we apply an enhancement of the method proposed by Tanaka et al. [15] to collect ESCs. This method was designed to support the learner to express his/her thoughts during the artifact making by reflecting on the process. We have modified this method so that we can collect ESCs without imposing a significant cognitive load on the learner. Our method asks the learner to express the **Timings** of SCs while he/she was making the artifact. In the method by Tanaka et al., the learner expresses his/her thoughts during the artifact making process as free descriptions. However, it is difficult for the learner to express his/her ESCs as free descriptions because ESCs have many items to describe. Therefore, we propose an improved method in which the learners can describe each item of ESCs. Our method consists of two steps of data collection during and after the artifact making process.

In the first step, we automatically collect the learner’s edit events in a manner similar to Tanaka et al. [15]. The learner expresses **Timings** of SCs when he/she starts SCs in the artifact making process through the system. If the learner recognizes the **Checked items** of an SC clearly when he/she expresses the **Timings** of the SC, he/she can express them at the same time. We reduce the learner’s cognitive load in expressing his/her SCs by providing the system that enables him/her to express the ESCs of SCs by easy operations. To collect authentic data of SCs, we do not impose any load on the learner after he/she starts SCs.

In the second step, the learner recalls and expresses his/her ESCs (**Checked items, Thoughts, Error detection**, and **Error contents**) by replaying and reflecting the artifact making process. In addition, if the learner recalls those SCs that were not expressed in the artifact making processes during the reflection, he/she can also express them.

By using this method, we can collect learners’ SCs associated with their artifact making processes. Data of an artifact making process is a series of edit events. Each edit event includes the kind of the edited target (Class, Attribute, or Relation). Therefore, from the edit events before and after an SC, we can guess the timing (Table 1) in which the SC was conducted. For example, if a learner mainly edited classes and/or attributes before an SC and edited relations after the SC, we can guess that the SC was conducted at “Between making concepts and making relations”.

5. Development of KIfU 3.0

This section describes the development of KIfU 3.0, which implements our method.

5.1 Functional Requirements

Functional requirements to implement our method are as follows:

1. Data collection of artifact/CDM-making processes,
2. Replay of artifact/CDM-making processes,
3. Recording of ESCs.

Requirement (1) is necessary for the data collection during the artifact making process. Requirement (2) is necessary for the data collection after the artifact making process. Requirement (3) is necessary for both of the data col-
lection during and after the artifact making process.

Requirements (1) and (2) have already been implemented by the system KIfU 2.0 developed by Tanaka et al. [15]. Here, we implement the function of recording of ESCs by extending KIfU 2.0.

5.2 Implementation of the Function for Recording of Reflection Contents

This section describes the implementation of Requirement (3). We show an overview of the artifact making window of KIfU 3.0 in Fig. 2. The area indicated by (2) is a canvas to draw a UML class diagram. The three buttons indicated by (3) correspond to ‘save’, ‘submit’, and ‘undo’ operations. When a learner wants to save the diagram, he/she can use the save button. If the learner clicks the save button, KIfU 3.0 records an edit event ‘Save’ and the state of the diagram. When a learner wants to cancel the previous edit, he/she can use the undo button. If the learner clicks the undo button, KIfU 3.0 records an edit event ‘Undo’, and restores the diagram to the previous state. At this time, no edit events are deleted. In addition, there is no change to the SC data. The learner can cancel his/her successive edits by clicking the undo button repeatedly.

First, we discuss the learner’s expression of the Timings of SCs during artifact making process. A learner expresses the Timing of his/her SC by clicking the SC button indicated by (1). If the learner clicks the SC button, KIfU 3.0 records the event as a ‘Check’ event, and the check list indicated by (4) will appear. Then, the learner checks the Checked items of the SC if it is possible for him/her to specify them. If the learner clicks the submit button placed at the bottom of the list, the SC will be recorded into the system. The question “Why did you SC at the timing?” in Fig. 2 (4) is not referred in this study.

Next, we discuss the learner’s expression of ESCs by reflecting on the artifact making process. We show an overview of the reflection window of KIfU 3.0 in Fig. 3. The buttons indicated by (1) represent the functions for the teachers, and they are invisible to the learners. The area indicated by (3) shows a list of edit events including the ‘Check’ events. The learner can replay the artifact making process by selecting an event or scrolling the list. The artifact corresponding to the selected edit event is shown in the canvas indicated by (5). The area indicated by (4) shows a list of SCs. The buttons indicated by (2) are to control replaying edit events and to add/delete SCs to/from the SC list.

When the learner selects an SC by clicking the recorded SC in the SC list, the SC editor indicated by (1)
will appear as shown in Fig. 4. The question “Why did you SC at the timing?” in Fig. 4 (1) is not referred in this study. The learner inputs the ESCs of the SC through the editor.

The editor includes the forms to input the following items:

(i) Check list of Checked items,
(ii) Free description area of Thoughts,
(iii) Radio button of Error detection,
(iv) Free description area of Error contents.

The learner can add an SC for an edit event by selecting the edit event and by using the SC button. Then, the learner can express the SC that he/she failed to express its Timing in the artifact making process.

6. Evaluation Experiment

In this section, we describe an evaluation experiment of our method by using KIfU 3.0. The purpose of the experiment is data collection to answer the RQs.

6.1 Participants

The participants of this experiment were 32 second-year undergraduate students of Nippon Institute of Technology who were attending a course ‘Database Design’. At the beginning of the class, they were all novices of CDM. They all had learned basics of database design for two months. Concretely, they had already learned a method to make conceptual data models from requirement specifications by detecting concepts, their attributes, and relations among the concepts. In addition, they had also learned how to detect errors from conceptual data models. According to the result of this experiment, no participant achieved the expected answer perfectly. From this, we consider that there was not any expert-level participant.

6.2 Choices of Checked items to Express ESCs

The participants had learned database design based on a textbook [16], which proposes the following check list to verify conceptual data models:

C1: Is there any attribute that is not basic data?
C2: Is there any attribute that is not entity specific?
C3: Is there any attribute that is not an attribute but a value?
C4: Is there any attribute of an entity that must be an attribute of a relation?
C5: Is there any multiplicity that does not follow the business rules?
C6: Is there any relation that must be modeled as an entity?

‘Basic data’ in C1 means that the data cannot be derived from other data. An ‘entity specific’ attribute means that the attribute must belong to an entity, and must not belong to other entities. In this experiment, we used them as Checked items.

6.3 Experimental Procedure

The experiment was conducted in a class of database design, and took about 100 minutes. The procedure was as follows:

(1) Tutorial of KIfU 3.0 (20 min)
We distributed the operation manual of KIfU 3.0 to the participants and explained the functions of the system to them. Then, we asked each participant to practice the system operations by making a small tutorial model.

(2) Exercise (40 min)
The topic of the exercise was “an order management system for online shopping” where the expected answer consists of 5 entities, 5 relations, and 19 attributes. We distributed the specification of the system, and asked each participant to do an exercise in which he/she makes a conceptual data model using KIfU 3.0. Also, we asked him/her to express Timings (and Checked items, if possible) of his/her SCs every time he/she conducts an SC during the exercise.

(3) Reflection (30 min)
We asked each participant to recall and express the ESCs by reflecting on his/her artifact making process. The participant replayed his/her artifact making process by using KIfU 3.0.

(4) Questionnaire (10 min)
We asked each participant to answer a questionnaire consisting of the following questions:

Q1: What percentage of SCs could you express during the exercise by using the SC button?
Q2: Did you encounter any problems with the task of making the artifact while expressing Timings of your SCs during the exercise?
Q3: Was the reflection on the artifact making process useful to recall Thoughts in each SC?
Q4: Was the reflection on the artifact making process useful to recall Error detection and Error content?

7. Experimental Results and Analysis

7.1 Results of Questionnaire

The results of Q1, Q2, Q3, and Q4 are shown in Tables 2, 3, 4, and 5, respectively.

7.2 Results of Data Collection of SCs

We calculate the number of participants who expressed one or more SCs as an indicator to answer RQ2.
The 32 participants expressed 122 SCs in total. All of the participants expressed at least one SC. Among them, 30 participants expressed multiple SCs. The maximum, minimum, and average numbers of SCs expressed by a participant are 10, 1, and 3.8, respectively.

### 7.3 Analysis of ‘Start of SC’

We investigate the tendency of Timings when the participants started SCs.

The average number of edit events (Create, Edit, and Remove) between two successive SCs is 11.7. Based on this value, we decide to classify each SC based on 10 edit events just before and 10 edit events just after the SC, respectively. We define six classes to whose targets are relations in the 10 edit events just before and just after the SC, respectively. Similarly, for an SC, whose targets are entities in the 10 edit events just before and just after the SC, we define four classes to whose targets are entities in the 10 edit events just before and just after the SC, respectively. We define six classes to whose targets are relations in the 10 edit events just before and just after the SC, respectively. Similarly, for an SC, whose targets are entities in the 10 edit events just before and just after the SC, we define four classes to whose targets are entities in the 10 edit events just before and just after the SC, respectively. We define six classes to whose targets are relations in the 10 edit events just before and just after the SC, respectively. Similarly, for an SC, whose targets are entities in the 10 edit events just before and just after the SC, we define four classes to whose targets are entities in the 10 edit events just before and just after the SC, respectively. We define six classes to whose targets are relations in the 10 edit events just before and just after the SC, respectively. Similarly, for an SC, whose targets are entities in the 10 edit events just before and just after the SC, we define four classes to whose targets are entities in the 10 edit events just before and just after the SC, respectively. We define six classes to whose targets are relations in the 10 edit events just before and just after the SC, respectively. Similarly, for an SC, whose targets are entities in the 10 edit events just before and just after the SC, we define four classes to whose targets are entities in the 10 edit events just before and just after the SC, respectively.

### 7.4 Analysis of ‘Search for Errors’

We compute the proportion of SCs by which the participants detected errors. In the first step, we examine Checked Items and Thoughts of all SCs to select those SCs in which the intentions of the participants are clear. Then, among these SCs, we further select those SCs where each of the Artifacts actually contains at least one error. Let \( N \) (54 in our experiment) be the number of such SCs.

In the second step, among these \( N \) SCs, we select the SCs where each of the Artifacts includes at least one error corresponding to the intention of the participant. Let \( N_1 \) (34 in our experiment) be the number of such SCs. In the third step, among the \( N_1 \) SCs, using Error detection, Error content, and Thoughts, we choose the SCs by which the participants found at least one error corresponding to the intentions. Let \( n_1 \) (7 in our experiment) be the number of such SCs. Finally, we calculate the proportion of SCs by which the participants detected the errors that correspond to the intentions of the participants as the ratio \( n_1/N_1 = 0.21 \) (21% in our experiment).
In a similar manner, we also compute the proportion of SCs by which the participants detected the errors that do not correspond to their intentions. The result shows that the proportion is 0.

7.5 Analysis of ‘Correction of Errors’

We compute the number of errors that were detected and corrected by the participants. First, we count the number of SCs in which Error detection is true. In our experiment, 13 such SCs expressed by 9 participants were detected. However, we exclude two of them expressed by one participant because we cannot estimate the intentions from the ESCs. In addition, we also exclude two other SCs by another participant because he regarded the correct parts as errors. Next, for the remaining 9 SCs expressed by 7 participants, we calculate the ratio of the errors that were corrected in the artifact making process, 0.78 (= 7 of 9 SCs) (78%). At the same time, we calculate the ratio of the participants who corrected at least one error, 0.71 (= 5 of 7 participants) (71%).

8. Discussion

In this section, we discuss the results of the analysis and address the RQs.

8.1 Effectiveness of Our Method and Response to RQ1

In our approach, learners express the Timings of SCs in the artifact making process. From Table 2, about half of the participants respond that they could express over 50% of all actual SCs. From Table 3, the cognitive load of expressing Timings in SCs is not critical for the participants. Based on these results, we suggest that our approach is effective to collect exact Timings of learners’ SCs.

On the other hand, from Table 4 and Table 5, it is clear that the reflection on the artifact making process is effective for learners to recall ESCs.

To compare our method with the previous one by Tanaka et al. [15], we reinvestigated the data collected in the previous study. The 46 participants in the previous study belonged to the same university as the participants in our study and they were attending the same course ‘Database Design’ in the previous academic year of our study. Hence, we assume that they had the similar levels of CDM ability to the levels of the participants of this study. The previous participants tackled a CDM exercise by using KIFU 2.0 in 60 minutes. The expected answer has 5 entities, 6 relations, and 14 attributes, and the size of the diagram is similar to that of this study. Then, they expressed their thoughts in their artifact making processes as comments in 15 minutes.

Among the comments, we extracted the comments by which we could guess that the participants had conducted SCs. The extraction criterion was that the comment contains a description such as: “I checked the artifact”, “I noticed errors”, or “I corrected errors in the artifact”. As a result, we extracted 34 such comments in total by all 46 participants, and we regard that we could collect that number of SC data. Hence, the average number of the SC data by one participant collected in the previous study is 0.73. On the other hand, the average number of the SC data in this study is 3.8. We conducted a t-test to investigate whether there is a significant difference between the average numbers of SC data collected by the previous and our methods. As a result, we obtained a significant difference between the two methods (p-value = 1.3 × 10^-9 < 0.05).

From these results, we claim that our method is effective to collect data of the learners’ SCs.

8.2 Number of SCs and Response to RQ2

As mentioned in Sect. 1, to make a high-quality conceptual model, it is necessary for each learner to conduct SCs in the making process and before submitting the artifact. Hence, at least two SCs are necessary. From the result described in Sect. 7.2, we found that all of the participants conducted at least one SC. In addition, we found that more than 90% of the participants conducted multiple SCs. These results indicate that ‘Start of SC’ was not difficult for the participants.

8.3 Timings of SCs and Response to RQ3

Table 7 shows that the number of the SCs conducted at “Before submission” is the largest. That is, almost all of the participants conducted SCs at this timing. The response to RQ3 is that almost all learners conducted SCs at “Before submission”. The number of the SCs conducted at “Between making concepts and making relations” is the second largest. However, only about half of the participants conducted SCs at this timing. In addition, only 31% of participants conducted SCs at “Making concepts”. SCs at “Making concepts” and “Between making concepts and making relations” are important because they contribute not only to improve the quality of the model but also to prevent backtracking. From the results, we can say that there is more room for teachers to encourage learners to conduct SCs at these timings. Hence, teachers should guide learners to conduct SCs at these timings.

8.4 Participants’ Skill of ‘Search for Errors’ and Response to RQ4

As shown in Sect. 7.4, the ratio of the SCs by which the participants could detect errors corresponding to the intentions was 21% (= n_1/N_1). It shows that it was very hard for the participants to detect errors by themselves. Furthermore, if the errors do not correspond to the intentions, the result was 0% (= n_2/N_2). Based on these results, we can say that it is unlikely that novice learners can detect errors by their SCs even if they know Checked items to be checked in their SCs. Hence, we claim that error detection in the step of ‘Search for Errors’ shown in Fig. 1 is difficult for the novice learners.

The check list refers to the textbook [16], which describes check methods for each check item. The teacher had
explained the check methods to learners in the class before the experiment. It must be easy for the learners who had understood the check methods to recall them from the check list. Therefore, we consider that the check list did not affect the difficulty of ‘Search for Errors’.

8.5 Participants’ Skill of ‘Correction of Errors’ and Response to RQ5

According to the result described in Sect. 7.5, the total number of the SCs that their values of Error detection are true is 9, and they were conducted by 7 participants. The number of the SCs that the detected errors were corrected is 7, and they were conducted by 5 participants. Hence, 0.78(= 7 of 9) (78%) is the answer of the RQ5.

Tanaka et al. [15] reported that KIFU 2.0 attains high usability with respect to the function of creating class diagrams. KIFU 3.0 is equipped with an equivalent function. Hence, we consider that KIFU 3.0 is also easy to use and has little adverse effect on the error correction activities by the participants. Then, the results indicate that the novice learners cannot necessarily correct the errors even if they could recognize them properly. Therefore, we claim that the step of ‘Correction of Errors’ is not always easy for the novice learners. This result suggests that instruction of the CDM should include some activities to improve learners’ skill of ‘Correction of Errors’.

8.6 Effects of Learners’ SCs

In this section, we describe the case of a successful SC and discuss the validity of the model of SCs shown in Fig. 1. Then, we study the relationship between the numbers of learners’ SCs and the quality of their artifacts. Furthermore, we discuss the effectiveness of the learners’ SCs.

8.6.1 An Example of Learner’s Successful SC

Let us describe a case of a successful SC by a participant where he could detect and correct an error. He conducted an SC when the state of his artifact was as shown in Fig. 5.

The ESCs of the SC are shown in Table 8. In the artifact shown in Fig. 5, ‘evaluationValue’, which means the evaluation value of the ‘Review’, is modeled as an attribute of ‘Review’. However, this is an error because the specification says that there are multiple ‘evaluationValue’s for one ‘Review’. The participant noticed this error by the SC and corrected it by removing the ‘evaluationValue’ from the attributes of ‘Review’ and by modeling the ‘EvaluationOfReview’ as another concept (class). Then he conducted another SC to confirm that this error is modified correctly.

Figure 6 shows the modified artifact, from which we can understand that the error was corrected. Table 9 shows the ESCs of the SC just after the error correction. It shows that the participant confirmed that the error was corrected. This case shows a successful flow of an SC in a practical context. The flow includes the steps of SCs described in Fig. 1. From this, we conclude that the model of SCs is appropriate to represent the flow of successful SCs.

8.6.2 Effects of SCs on Quality of Artifacts

In this section, we discuss the effects of learners’ SCs on the quality of their artifacts. The average number of the SCs by a participant is 3.8. Using this value as the dividing line, we divided the participants into two groups G1 and G2: the par-
Table 9: ESCs of SC after error correction.

<table>
<thead>
<tr>
<th>Items of ESC</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checked items</td>
<td>none</td>
</tr>
<tr>
<td>Thoughts</td>
<td>I verified the integration of the modified points by making relations in my image.</td>
</tr>
<tr>
<td>Error detection</td>
<td>No</td>
</tr>
<tr>
<td>Error content</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 10: Numbers of SCs and evaluation values of artifacts.

<table>
<thead>
<tr>
<th>G1</th>
<th># SCs</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>145</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>209</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>128</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G2</th>
<th># SCs</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>113</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>204</td>
</tr>
</tbody>
</table>

Participants who expressed more than 3.8 SCs were classified into the G1 group while the others were classified into the G2 group. Then, we statistically tested whether the average quality of the artifacts by the G1 group is significantly different from that by the G2 group. We blindly evaluated the quality of the participants’ artifacts by the G1 group is significantly different from that by the G2 group. We blindly evaluated the quality of the participants’ artifacts by comparing it with the correct answers in reference to Tanaka et al. [11], where the full score was 238.

We show the relationship between the numbers of SCs and the evaluation values of the artifacts in Table 10.

The null hypothesis $H_0$ and an alternative hypothesis $H_1$ for the statistical test are as follows:

$H_0$: There is no significant difference between the average evaluation values of the artifacts by G1 group and G2 group.

$H_1$: There is a significant difference between the average evaluation values of the artifacts by G1 group and G2 group.

We conducted a $t$-test to validate $H_0$ and obtained the result that there is no significant difference between the average evaluation values of the artifacts generated by the G1 and the G2 groups ($p$-value = 0.69).

As we described in Sect. 8.4, it is difficult for novice learners to detect errors by SCs. Furthermore, as we described in Sect. 8.5, they cannot always correct errors even if they could detect them. For such learners, it does not help to improve the quality of artifacts when the teachers prompt them to conduct SCs. In the teaching of CDM, we propose that the teachers not only teach Checked items and/or the methods of SCs but also conduct practical exercises of error detection and/or correction.

9. Conclusion and Future Work

In this study, we focused on learners’ SCs in exercises of CDM. We developed a method and a system, KIfU 3.0, to collect the SCs. We conducted an application experiment of KIfU 3.0. The experimental results suggest that the method was effective to collect SCs. Furthermore, we analyzed the results and verified that (1) the novice learners conduct SCs during a CDM task (based on the result that all of the 32 participants conducted SCs); (2) it is difficult for the novice learners to detect errors in their artifacts (based on the result that the proportion of the SCs where the participants could detect the errors corresponding to the intentions was 21%); (3) they cannot necessarily correct the errors even if they recognize them (based on the result that the proportion of the SCs where the participants could correct the detected errors was 78%); and (4) there is no relationship between the numbers of self checks of the novice learners and the quality of their artifacts (based on a statistical test).

In the classes of CDM, we propose that the teachers should provide an opportunity to learners so that they can train SCs specifically related to error detection as well as error correction.

In our future research, we plan to design an exercise specifically to improve the learners’ SC skills. In addition, we will consider investigating differences between behaviors of experts and novices in the artifact making processes as well as their SCs. Also, we will increase the number of the Checked items because some descriptions in Thoughts in the experiment included the reasons of SCs that could not be classified into the Checked items.

Acknowledgments

We appreciate Dr. Bipin Indurkhya for proofreading this paper. This study was partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under Grant No. 17K01118.

References


---

Hiroaki Hashiura is an Assistant Professor at Nippon Institute of Technology, Japan. He received the degrees of B.E., M.MOT, and D.E. from Shibaura Institute of Technology in 2002, 2004, and 2008, respectively. His research interests include knowledge-based software engineering, software engineering education, project-based learning, and object-oriented software design. He is a member of IPSJ.

Atsuo Hazeyama received the degree of B.E. from Waseda University, Japan, the degree of M.E. from the University of Electro Communications, Japan, and the degree of D.E. from Shinshu University, Japan. He was with NEC from 1987 to 1999. He is now a Professor at Tokyo Gakugei University, Japan. His research interests are collaborative software development support, collaborative learning support, and knowledge management. He is a member of IEEE CS, ACM, IPSJ, JSAI, and JSiSE.

Seiichi Komiya received the degree of B.Sc. from Saitama University, Japan, in 1969 and the degree of D.E. from Shinshu University, Japan in 2000. He has been working for Hitachi Ltd. as a software engineer during 1970–2001, and has been on loan from Hitachi Ltd. to Information-technology Promotion Agency Japan during 1984–1999. He has studied the frameworks to construct various CASE tools, software specification/design process, CAI and Intelligent CAI there. He joined in Shibaura Institute of Technology as a Professor in 2001. Now, he is a Professor at National Institute of Informatics, Japan.

Yuki Hirai is a Lecturer at Shinshu University in Japan. His main research area is educational technologies. He received the B.L.A. and M.Ed. degrees from Tokyo Gakugei University in 2007 and 2009, respectively. He also received the Ph.D. degree from University of Tsukuba in 2012. He is a member of IPSJ and JSAI.

Keiichi Kaneko is a Professor at Tokyo University of Agriculture and Technology in Japan. His main research areas are functional programming, parallel and distributed computation, partial evaluation, fault-tolerant systems, and pedagogical systems. He received the B.E., M.E., and Ph.D. degrees from the University of Tokyo in 1985, 1987, and 1994, respectively. He is a member of ACM, IEEE CS, IPSJ, and JSiST.