Improvement of RETUSS to Ensure Traceability between Sequence Diagram in UML and Java Source Code in Real Time

Kaoru Arima*, Tetsuro Katayama*, Yoshihiro Kita†,
Hisaaki Yamaba*, Kentaro Aburada*, Naonobu Okazaki*
**Department of Computer Science and Systems Engineering, Faculty of Engineering, University of Miyazaki,*
1-1 Gakuen-kibanadai nishi, Miyazaki, 889-2192, Japan
†Department of Information Security, Faculty of Information Systems, Siebold Campus, University of Nagasaki,
1-1-1 Manabino, Nagayo-cho, Nishi-Sonogi-gun, Nagasaki, 851-2195, Japan
arima@earth.cs.miyazaki-u.ac.jp, kat@cs.miyazaki-u.ac.jp, kita@sun.ac.jp,
yamaba@cs.miyazaki-u.ac.jp, aburada@cs.miyazaki-u.ac.jp, oka@cs.miyazaki-u.ac.jp

Abstract

Ensuring traceability of software deliverables is one of the methods to ensure software quality. RETUSS (Real-time Ensure Traceability between UML and Source-code System) is a tool that saves labor and time, and eliminates mistakes by human handling in ensuring traceability between UML and source codes. However, RETUSS is not useful due to its limited scope of application. This paper improves the usefulness of RETUSS by extending the scope of application of sequence diagram in UML and Java source code.

Keywords: software quality, traceability, UML, sequence diagram, Java

1. Introduction

The importance of software in society is increasing, and system failures and software bugs cause significant economic and social impact. Therefore, ensuring the quality of systems and software has become more important. Ensuring traceability of software deliverables is one of the methods to ensure software quality.[1] It can specify the scope of the impact due to the modification in the requirements and remove the gap between the documents and the source codes. However, it has the following two problems.

- Taking much labor and time to modify similarly other related deliverables in modifying a part of deliverables
- Having a risk that you cannot ensure traceability because of causing mistakes to ensure traceability by human handling

In order to solve them, our laboratory developed RETUSS (Real-time Ensure Traceability between UML and Source-code System).[2,3] RETUSS ensures traceability between UML (Unified Modeling Language)[4] and source codes by transforming them to each other in real time. Therefore, RETUSS can save labor and time, and eliminate mistakes by human handling in ensuring traceability between UML and source codes. RETUSS has the following functions.

- Description of class diagram
- Description of sequence diagram
- Description of Java source codes
- Description of C++ source codes
- Bidirectional transformation between class diagram and Java source codes
- Bidirectional transformation between class diagram and C++ source codes
- Bidirectional transformation between sequence diagram and Java source codes

However, RETUSS is not useful in ensuring traceability between sequence diagram and Java source codes due to its limited scope of application.

This paper improves the usefulness of RETUSS by extending the scope of application of sequence diagram and Java source codes.

2. The extended RETUSS

This paper extends the following two functions of the existing RETUSS.

- Description of sequence diagram
- Bidirectional transformation between sequence diagram and Java source codes

Fig. 1 shows the interface of the extended RETUSS. The extended RETUSS has the main window and the source code window. The main window is a window for describing UML. The source code window is a window for writing source codes.

Fig. 2 shows the structure of the extended RETUSS. The extended RETUSS consists of five parts: display part, correspondence part, description part, transformation part, and storage part. To extend the function of the description of sequence diagram, we mainly extend the display part, correspondence part, and description part. To extend the function of the bidirectional transformation between sequence diagram and Java source codes, we mainly extend the description part, transformation part, and storage part.

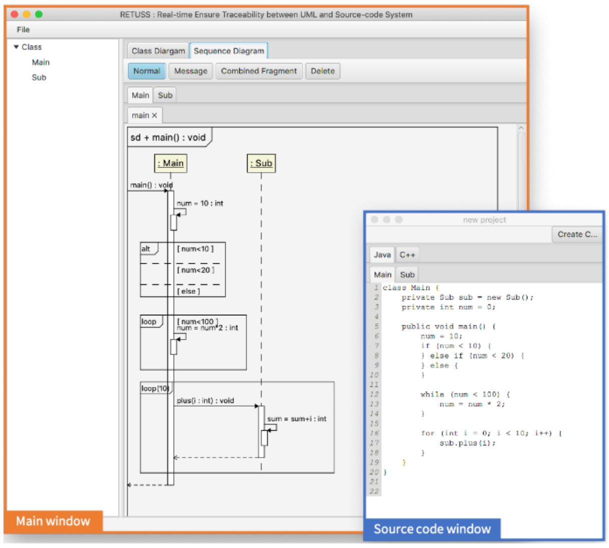
2.1. *Extending the function of the description of sequence diagram*

We add the following three functions to the function of the description of the sequence diagram.

- Adding a message
- Adding a combined fragment
- Deleting elements

By adding these functions, a user can edit the sequence diagram directly on RETUSS.

To add these three functions, we extend the display part, the correspondence part, and the description part. The extended display part displays three buttons in the main window: Message button, Combined Fragment button, and Delete button. The extended correspondence part has three new processes: displaying the dialog to add a message, displaying the dialog to add a combined fragment, and displaying the dialog to delete elements. The three processes are called from the event handlers of the three buttons displayed by the extended display part. The extended description part has four new processes: adding a message, adding a combined fragment, deleting a message, and deleting a combined fragment. These four processes are called by correspondence part and they change the sequence diagram information in the storage part.



```
class Main {
    private Sub sub = new Sub();
    private int num = 0;

    public void main() {
        num = 10;
        if (num < 10) {
        } else if (num < 20) {
        } else {
        }

        while (num < 100) {
            num = num * 2;
        }

        for (int i = 0; i < 10; i++) {
            sub.plus(i);
        }
    }
}
```

Fig. 1. The interface of the extended RETUSS

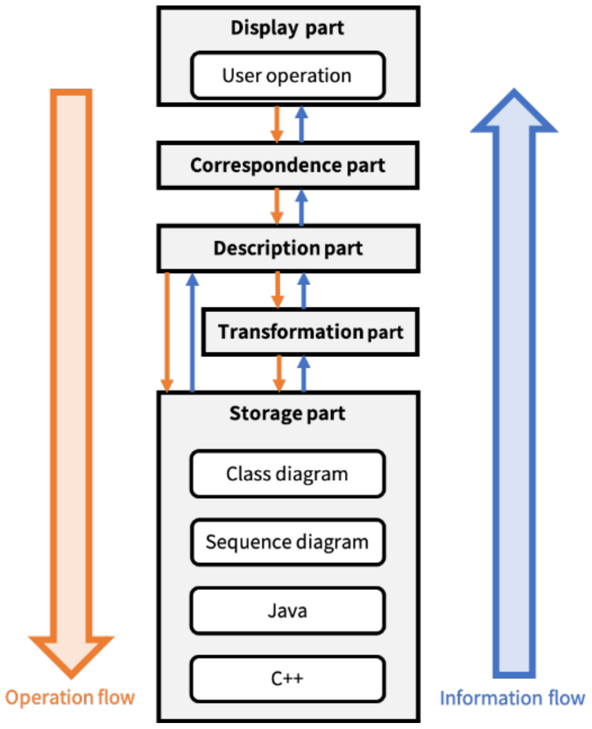
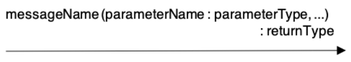
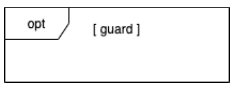
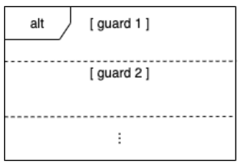
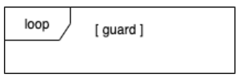
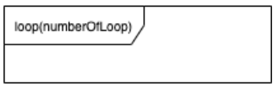


Fig. 2. The structure of the extended RETUSS

2.2. *Extending the function of the bidirectional transformation between sequence diagram and Java source codes*

We define four new transformation rules to extend the scope of the bidirectional transformation between sequence diagram and Java source codes. Table 1 shows

Table 1. The four new transformation rules for the bidirectional transformation between sequence diagram and Java source codes.

Name in sequence diagram	**Notation in sequence diagram**	**Name in Java**	**Syntax in Java**
Message	 messageName (parameterName : parameterType, ...) : returnType	Method invocation expression	methodName(parameter, …);
		Method declaration	accessModifier returnType methodName(parameterType parameterName, …) { … }
Combined fragment opt		if-then statement	if (expression1) { … }
Combined fragment alt		if-then-else statement	if (expression1) { … } else if (expression2) { … } …
Combined fragment loop		while statement	while (expression) { … }
		for statement	for(int i=X; i<Y; i++) { … }
			for(int i=X; i<=Y; i++) { … }
			for(int i=X; i>Y; i--) { … }
			for(int i=X; i>=Y; i--) { … }

the four new transformation rules for the bidirectional transformation between sequence diagram and Java source codes. By the transformation rules in Table 1, the extended RETUSS can transform the following sequence diagram elements into Java source code syntaxes.

- Message of operation invocation with parameters
- Combined fragment opt
- Combined fragment alt
- Combined fragment loop

In addition, the extended RETUSS can transform the following Java source code syntaxes into sequence diagram elements.

- Method invocation expression with parameters
- Method declaration
- if-then statement
- if-then-else statement
- while statement
- for statement

Here, the extended RETUSS does not support nested structure of the above syntaxes.

To implement the transformation rules in Table 1, we extend the description part, transformation part, and storage part. The extended description part has a new process: transformation from Java source codes to Java information. The extended transformation part has a new process: transformation between sequence diagram information and Java information based on the transformation rules in Table 1. The extended storage part has three new classes in the sequence diagram information: interaction fragment class, combined fragment class, and interaction operand class. In addition, the extended storage part has three new classes in the Java information: If class, For class, and While class.

3. Application example

Fig. 3 shows the screenshot when the Java source codes are written in the extended RETUSS. It shows that the extended RETUSS can ensure traceability between sequence diagram and Java source code in writing if-then-else statement, while statement, and for statement of Java. In addition, we confirmed that the extended RETUSS can ensure traceability between sequence diagram and Java source codes in describing sequence diagram.

4. Evaluation

To evaluate the usefulness of the extended RETUSS, we experiment with four students of University of Miyazaki. The steps of the experiment are shown below.

(i) The experimenter prepares traceable sequence diagrams and Java source codes. We call these deliverables.
(ii) The experimenter instructs the participants to change the deliverables.
(iii) The participants change the deliverables as instructed

There are two types in the changes: sequence diagrams changes, Java source codes changes. There are two cases below for participants to change the deliverables.

- Case A: using the extended RETUSS.
- Case B: using EA (Enterprise Architect)[5] and a text editor.

Table 2 shows the times it took the participants to change in the two cases and the two changes. From Table 2, the time in case A was about 76.5% shorter than the time in case B, when the sequence diagrams changes was instructed. In addition, the time in case A was about 69.0% shorter than the time in case B, when the Java source code changes were instructed.

In summary, the extended RETUSS can save labor and time in ensuring traceability between sequence diagram and Java source codes. Therefore, the usefulness of RETUSS has improved by extending its scope of application while retaining the benefits of the existing RETUSS.

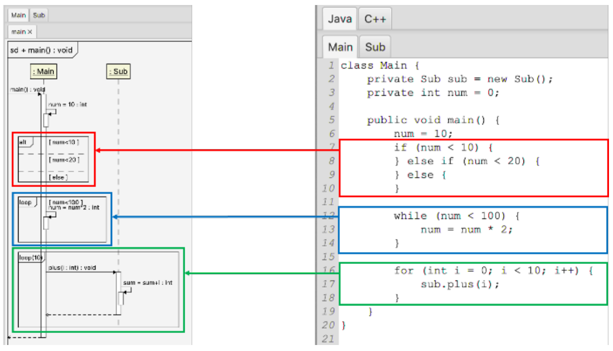
5. Conclusion

This paper has improved the usefulness of RETUSS by extending the scope of application of sequence diagram and Java source codes. The extended RETUSS allows you to edit sequence diagram directly on RETUSS, and also supports four new transformation rules for sequence diagram and Java source codes.

The experimental results showed that the extended RETUSS can save the time to ensure traceability between sequence diagram and Java source code by about 76.5% for sequence diagram changes, and about 69.0% for Java source code changes. Therefore, the usefulness of RETUSS has improved by extending its scope of application while retaining the benefits of the existing RETUSS.

The future works are as follows.

- Corresponding to other sequence diagram elements
- Corresponding to other Java source code syntaxes
- Corresponding to other UML diagrams
- Corresponding to other programming languages



```
class Main {
    private Sub sub = new Sub();
    private int num = 0;

    public void main() {
        num = 10;
        if (num < 10) {
        } else if (num < 20) {
        } else {
        }

        while (num < 100) {
            num = num * 2;
        }

        for (int i = 0; i < 10; i++) {
            sub.plus(i);
        }
    }
}
```

Fig. 3. The screenshot when the Java source codes are written in the extended RETUSS

Table 2. The times it took the participants to change (seconds)

Participants	Sequence diagrams changes		Java source codes changes	
	Case A	**Case B**	**Case A**	**Case B**
1	59	205	134	369
2	62	221	95	346
3	38	216	102	361
4	60	289	126	398
Average	54.75	232.75	114.25	368.50

References

1. SQuBOK Sakutei Bukai, *Guide to the Software Quality Body of Knowledge*, 2nd edn. Ohmsha, 2014 (in Japanese).
2. Tetsuro Katayama, Keisuke Mori, Yoshihiro Kita, Hisaaki Yamaba, Kentaro Aburada, Naonobu Okazaki: *RETUSS: Ensuring Traceability System between Class Diagram in UML and Java Source Code in Real Time*, Journal of Robotics, Networking and Artificial Life, Vol. 5(2), pp. 114–117, 2018.
3. GitHub, *RETUSS: Real-time Ensure Traceability between UML and Source-code System*, https://github.com/Morichan/Retuss (Accessed 2020-12-09)
4. The Object Management Group, *Welcome to UML Web Site!*, https://www.uml.org/ (Accessed 2020-12-09)
5. SPARX SYSTEMS, *UML modeling tools for Business, Software, Systems and Architecture*, https://www.sparxsystems.com/ (Accessed 2020-12-09)