**Abstract**

In recent years, software development process is required to support for diversification and multi-platform by the shorter delivery time. In order to develop a system to efficiently respond to such requests, software development using component has spread. When defining the structure and behavior of software components, UML diagrams are often used. UML is a standardized modeling language developed by OMG. Especially in upstream development, UML sequence diagrams are frequently used to understand and verify the behavior of components.

However, the specification of sequence diagram is complicated and flexible. Also, in many cases, the detailed behavior has not been determined in early development stage. For example, developers can design login process as "return a success message if the user name and the password are correct, return a failure message if the user name and the password are invalid". Here, if the way how passwords are stored and they are checked has not been designed yet, it cannot be described deterministically whether it should return a success message or a failure message. For these designs, it is important to verify designs leaving the nondeterminism like "return a success message or a failure message nondeterministically". Using conventional methods, it was difficult to handle those nondeterminism. Therefore, it was difficult to verify sequence diagrams automatically. It has relied on manual review to find mistakes such as inconsistencies and insufficient refinements between sequence diagrams. If such mistakes are found in a late development stage, it may take a lot of time and cost to correct them.

In this paper, we define a subset of sequence diagrams with formal semantics and propose a method to verify consistency of the sequence diagrams. With this method, developers can clarify the specifications by using formal description and find bugs by using automatic verification.

In order to verify sequence diagrams, we propose a synthesis method of a formal expression called CSP (Communicating Sequential Processes) from sequence diagrams. This synthesis method consists of following steps: At first, an order of sending and receiving is extracted from each sequence diagram for each component and it is formally expressed as a CSP process. Next, two or more CSP processes extracted from a number of sequence diagrams for the component is combined to a CSP process which represents the whole behavior of the component. We define new CSP operators for synthesizing sequence diagrams. Finally, using expansion rules of the new CSP operators, the CSP process is converted into a standard CSP process without the new operators. The standard CSP process can be verified using existing CSP tools such as PAT and FDR2. In addition, counter examples found in the model checking can be translated back to sequence diagrams for supporting to correct their inconsistency.

Compared with the related works, the main advantage of this work is that nondeterminism can be considered. It means that our approach can handle abstract sequence diagrams. Sequence diagrams are often abstract in early development stage. Our approach can be applied to such diagrams.

We implemented a tool named SDVerifier, based on the proposed method, which can be used for verifying sequence diagrams. We also conducted experiments with real world case studies.

**Keywords**

Sequence Diagram, Process Algebra, CSP, Process Synthesis, Verification