# UML 다이어그램을 이용한 Reverse Engineering을 통한 요구사항의 도출에 관한 연구

# An Approach to Derive Requirements through Reverse Engineering using UML Diagram


**요 약**

Reverse engineering aims to derive information from a system to renovate the system or to understand the system. In this paper, we propose an approach composed of three steps, which are all been proposed by different authors, but has not been attempted to integrate each of them.


## 1. Introduction

In software engineering, people usually generate requirements specification before developing software systems. But in an actual development field, systems may be developed without or with minimal requirements specifications. In this case, when entering the maintenance phase in software development lifecycle, a lot of effort must be made to understand the system when a person who has not developed the system proceeds with the maintenance. Since we need requirements specification to understand the system and proceed the maintenance phase, we need to generate requirements specification properly.

Reverse engineering is a process to develop a set of specifications from an existing system [1]. By reverse engineering, we can derive various information from the system, including requirements specification, which can be used to renovate the system or to understand the system.

In this paper, we propose an approach to derive requirements by integrating three existing approaches. And we will show a case study to show that an approach proposed in this paper works properly, by applying forward engineering to generated requirements specification.

This paper is organized as follows: Section 2 introduces related work, which attempted to generate requirements specification through performing reverse engineering. Section 3 introduces three existing approaches, which have been integrated in this paper to generate requirements specification. Section 4 describes a case study to show that the integrated approach works properly. Finally, Section 5 concludes this paper.

## 2. Related Work

Usually, reverse engineering is used to construct documents about design specification, by using code or data. But in this study, we are trying to produce requirements specification through reverse engineering.

A study of Fahmi and Choi [2] tried to derive requirements through integration of forward engineering and reverse engineering. In their study, they made use of a fact that forward engineering and reverse engineering has a point of a contact. They make goal model and get non-functional requirements by performing reverse engineering. Then, they perform requirements engineering to generate new requirements and compare it with a reverse engineering output, which is an abstract model of an original source code.

In this paper, we do not perform forward engineering. Instead, we use UML diagrams, especially sequence diagrams and statecharts to perform reverse engineering and generate a requirements specification.

## 3. Proposed Approach

In this paper, we propose an approach to derive requirements from an existing system, which has no or least requirements specification document. This approach is composed of three steps, which are all been proposed by different authors, but has not been attempted to integrate each of them. Total step is shown in Figure 1, and each of the steps is as follows:

(1) Constructing a sequence diagram from a system by looking into the code;

(2) Synthesizing a statechart from sequence diagrams by using an approach proposed by Ziadi et al. [3];

(3) Generating a requirements specification from statecharts using an approach proposed by Glinz [4]

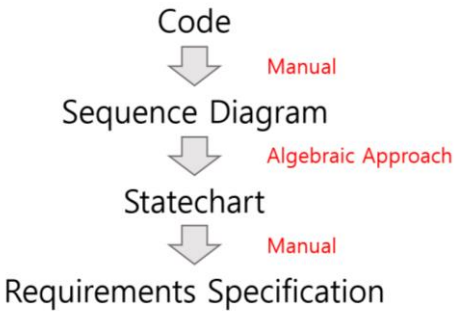


**Figure 1** Total Process

In step 1, we are going to look into the code and draw sequence diagram from the code. It will take a long time to convert code into sequence diagram, but still it is an essential step to perform.

In step 2, authors synthesize statechart from sequence diagrams using an algebraic approach. Since statechart is an intra-object description and sequence diagram is an inter-object view of a system [3], we cannot just convert sequence diagram into statechart. So They describe basic sequence diagram and flat statechart as a tuple and apply an algorithm to convert sequence diagram into statechart.

In step 3, author generates requirements specification from statechart using characteristics of a statechart. There are features of statechart, such as broadcasting events, hierarchical states and orthogonal states, single event assumption, synchronous event processing. The author considers such features when deriving functional requirements from statecharts.

## 4. Case Study

We applied a proposed approach to an example of coffee vending machine system, which is a small-scale system. We chose such an example to simply show that this approach can work properly. Since step 1 in the proposed approach takes long time, we needed to choose a small-scale example to show that this approach is efficient enough. There is a code for a coffee vending machine system and only a few functional and non-functional requirements. To get requirements specification, we applied this approach.

First, we draw sequence diagram from the given code. Figure 2 shows an example of one execution path, which is that actor choose a coffee menu and insert enough coin.

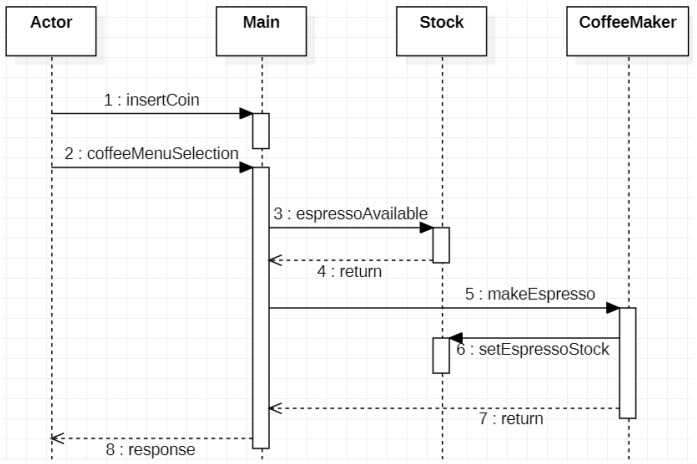


**Figure 2** Sequence diagram of making espresso

Then, we draw a statechart from sequence diagrams through algebraic approach proposed by Ziadi et al.. Finally, we generate requirements specification from statechart. Some examples of generated requirements specification are as follow:

1. Coffee vending machine should be able to turn off the power when manager inputs correct password and activate turn off menu.
2. When a user tries to get coffee menu without enough money, a machine should not provide a menu.
3. When a user tries to get coffee menu with enough money, a machine should provide a menu in 10 seconds.

## 5. Conclusion

In this paper, we proposed an approach to derive requirements specification from a code for a software system. We integrated three existing approaches to the

system, and these approaches are (1) constructing sequence diagram from a code; (2) synthesizing statechart from sequence diagrams; (3) generating requirements specification from statechart. With this integrated approach, we were able to generate requirements specification. But still, there are problems that this approach cannot be automated, and it depends much on engineer's capabilities. We are planning add formalism to this approach to automate some part of the approach.

## References


[1] M. G. Rekoff, "On reverse engineering," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 2, pp. 244-252, March-April 1985, doi: 10.1109/TSMC.1985.6313354.

[2] S. A. Fahmi and H. Choi, "Software Reverse Engineering to Requirements," 2007 International Conference on Convergence Information Technology (ICCIT 2007), 2007, pp. 2199-2204, doi: 10.1109/ICCIT.2007.228.

[3] T. Ziadi, L. Helouet and J. -M. Jezequel, "Revisiting statechart synthesis with an algebraic approach," Proceedings. 26th International Conference on Software Engineering, 2004, pp. 242-251, doi: 10.1109/ICSE.2004.1317446.

[4] M. Glinz, "Statecharts For Requirements Specification - As Simple As Possible, As Rich As Needed", Proceedings of the ICSE 2022 International Workshop on Scenarios and State Machines: Models, Algorithms and Tools, 2002.