Applied Practice on the Guide Code Generation based on Model Driven Development for Developing Huge Systems on Distribute Environments

Sejun Jung
Software Engineering Laboratory
Hongik University
Sejong, Korea
bvcx79@hongik.ac.kr

So Young Moon
Software Engineering Laboratory
Hongik University
Sejong, Korea
whit2@hongik.ac.kr

R. Youngchul Kim
Software Engineering Laboratory
Hongik University
Sejong, Korea
bob@hongik.ac.kr

***Abstract*— Defining complex and multifunctional software requires a lot of requirements and design. In particular, the more complicated functions have, the more use case scenarios are created. Implementation and verification based on these requirements and design documents require a high understanding of design and requirements. Therefore, costs are incurred even before development begins. This research proposes a metamodel that integrates design documents. The structure of the metamodel makes it possible to trace related design elements. It also uses the metamodel to generate guide code automatically. The guide code helps developers understand the design and makes development more straightforward. This is expected to reduce the cost of development.**

Keywords—Metamodel, Automatic Code Generation, Usecase Specification, Unified Modeling Language, Class Diagram

I. INTRODUCTION

The scale and importance of smartphone applications, autonomous driving software, satellite software, and weapons software continue to grow. To define complex and many functions software, many requirements and design documents are required. In particular, software with many functions and complexities has design elements such as many systems, use cases, actors, and objects. This software increases the number and interaction of modules, and the size of the module itself increases. As a result, it becomes difficult to understand the module and implement it by reflecting on the design. This study proposes a metamodel that integrates high-level design and UML. Related design elements can be traced through the structure of the proposed metamodel. Moreover, this study automatically generates the guide code based on the class diagram and the use case specification stored in the metamodel. The guide code consists of class structure, comments, and functions. This guide code reduces the time required for module analysis and makes development more straightforward.

II. RELATED RESEARCH

It is difficult for software to trace the relationship to the outputs of each step that arise from the requirements. Tracking output typically identifies relationships through ID. A study creates a traceability matrix using a database to effectively visualize Id-based output tracking [1]. This study transforms the design into a metamodel, identifies it through id, and traces it through the structure. The metamodel is a model for defining a model. Metamodel uses Model to Model technology to treat similar but platform-dependent models as a single model. As a study using metamodel, some studies change the AST of different programming languages to the same metamodel ASTM [2] or change the smartphone UI model of other platforms to the same UI metamodel and automatically produce it [3]. In addition, there is a study to create test cases from natural language by defining a cause-effect model and a semantic sentence model as metamodel [4]. This study proposes a metamodel that integrates the high-level requirements and the UML model and suggests tracing design elements. The guide code is completed by generating detailed code and comments using the information of the high-level design model connected to the low-level design model.

III. USECASE SPECIFICATION MODELING

UML, requirements, and use case specification elements are defined as models to trace the outputs generated from the requirements. Figure 1 shows the core models and structures used to produce guide codes.

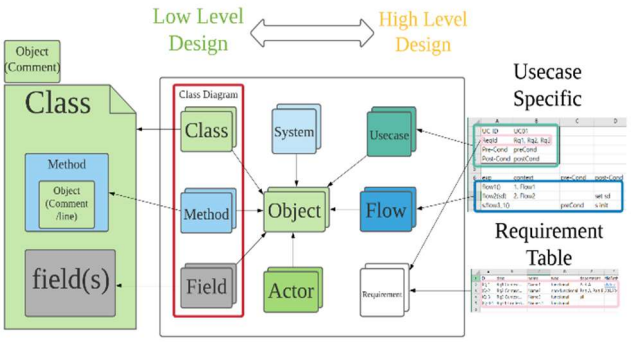

Fig. 1. Meta Model for Generate Guid Code

Design elements are viewed as objects without distinction between high-level and low-level designs. Furthermore, related design elements are identified through id and connected. All design elements can be recognized and connected as Objects. It is possible to trace design elements related to different design models derived from one design model. Finally, the design model of the class diagram is used to guide code production. Then, comments and requirements are created through the design information of the Object that is the basis of the class. The Method creates comments with the information of the underlying Object and makes the inside of the Method with the defined design information inside the Object.

IV. CASE STUDY

This paper is an example of applying the proposed technology to the monitoring system. This monitoring system has three system modules. First, there is SMS (Server Management System), where the administrator manages the server as a whole. Second, GUIS (Graphic User Interface System) shows the server status when SMS runs. Finally, the CS (Control System) is run from the GUIS. CS communicates and controls the input/output of server users and manages server data. Each system has various functions. Figure 2 is a use case diagram and a class diagram of the Server Manage System.

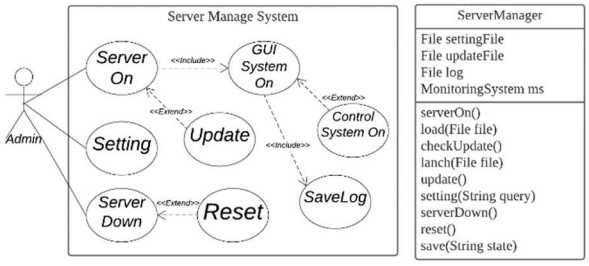

Fig. 2. Server Manage System – Admin, Use case Diagram

In the use case diagram in Figure 2, eight use cases result in 8 use case scenarios. The class diagram has the structure most similar to the source code. Although the class diagram has a structure similar to the source code, there is a problem that eight scenario documents need to be understood to implement the ServerManager class. Figure 3 is a part of the Java guide code of the ServerManager class generated by inputting the use case specification and UML design information using the method proposed in this study.

```
3   /**SM
4   RqSM-1:An administrator can open the server only through SM.
5   RqSM-1-1:CS can only be run through SM when the server is open.
6   RqSM-2:The SM program has a monitoring function.
7   RqSM-3:The server supports the update function.
8   RqSM-4:When the server is running, the parameters of the server are set through the setting file.
9   RqSM-5:Server reset is supported when the server is shut down.
10  RqSM-MS-1:The MS logs the status value of the monitoring system to SM log file.
11  */
12 public class ServerManager
13 {
14     File updateFile;
15     File settingFile;
16     Flie log;
17     MonitoringSystem ms;
18
19     /**UC01
20     Server On. If has patch, then Update.
21     */
22     public void serverOn()
23     {
24         //Load the start file. Start file has setting file path and update file path.       UC01-FL-1
25         load(file);
26
27         //Check there is update file.       UC01-FL-2
28         checkUpdate();
29
30         // Executes the server based on the setting value.      UC01-FL-3
31         launch(file);
32
33         // Run UC01-I1      UC01-FL-4
34         ms.runGui(log);
35
36     }
37
38     /**UC01-FL-2
39     Load the start file. Start file has setting file path and update file path.
40     @return Initialize Server Manager files.
41     */
42     public void loadFile(File file)
43     {
44     }
```

Fig. 3. Part of the ServerManager class guid code

Figure 3 is a part of the guide code of the automatically generated Server Manager class. Server Manager class is a class diagram designed based on Server Management System. Therefore, the ID and system requirements of the Server Management System are generated as JavaDocs from lines 3 to 11. Method serverOn in line 22 is defined in the class diagram based on Use Case UC01. The loadFile method in line 42 is a method of the class diagram depicted based on the event flow diagram of UC01. The relevant model information is imported, and the content description is generated with JavaDoc.

V. CONCLUSION

The method proposed in this study was applied to the monitoring system. Through this method, it was possible to see the design information through the structure of the source code and proceed with the implementation. It is expected that this will help understand complex systems and reduce the cost of documentation and development. Also, the inside of the guide code's method and comments are implemented based on the Use Case specification. Therefore, if the class structure can be

automatically generated through Use case diagrams and scenarios, it is possible to create guide codes with only high-level design. Therefore, to produce guide codes with only high-level design, we plan to conduct semantic analysis studies of use case specifications and improve models [4]. In addition, more detailed guide code generation is possible, and guide code generation can be expected from an unrefined specification file.

ACKNOWLEDGMENT

The research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R1I1A305040711) and the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2021R1I1A1A01044060) and the BK21 FOUR (Fostering Outstanding Universities for Research) funded by the Ministry of Education (MOE, Korea) (F21YY8102068).

REFERENCES

[1] Jin Hyub Lee, Kidu Kim, Bo Kyung Park, and R Young Chul Kim. "Example of Implementation of Requirements Tracking Based on the Traceability Matrix Mechanism." Proceedings of the Korean Society of Information Sciences 2018.6 (2018): 488-489.

[2] Hyun Seung Son,R Young Chul Kim. "xCodeParser based on Abstract Syntax Tree Metamodel (ASTM) for SW Visualization" INFORMATION -YAMAGUCHI- : 963-968.K. Elissa, “Title of paper if known,” unpublished.

[3] Hyunseung Son, Wooyeol Kim, and Youngcheol Kim. "A method of applying model transformation for the configuration of heterogeneous smartphone app development environments." Journal of the Information Science Society: The Practice of Computing and Letters 20.4 (2014): 238-242.

[4] Jang, Woo Sung, Se Jun Jung, and R. Kim. "Design of Sentence Semantic Model for Cause-Effect Graph Automatic Generation from Natural Language Oriented Informal Requirement Specifications." Annual Conference on Human and Language Technology. Human and Language Technology, 2020.