JAIST Repository
https://dspace.jaist.ac.jp/

Title	モデル駆動開発における例外発生に伴う複雑度の抑制に関する研究
Author(s)	阿部，航
Citation	
Issue Date	2021-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/17151
Rights	
Description	Supervisor:鈴木 正人，先端科学技術研究科，修士（情報科学）

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

モデル駆動開発における例外発生に伴う複雑度の抑制に関する研究

1810008 阿部 航

In the process of software development, the source code becomes complicated every time specifications are added or changed. It becomes difficult to grasp the contents. As a system development method, UML (Unified Modeling Language) is a language that unifies the description when modeling a system in object-oriented design, and because it is visualized, it is easier to understand the contents than the source code. Model Driven Development (MDD) is a method that automatically generates a source code template from a UML model, and consists of a class diagram, sequence diagram, use case diagram, etc. as a model. MDD makes it easy to maintain model and code consistency in model development. However, UML itself may become complicated and maintainability may deteriorate. The sequence diagram itself, in which exceptions occur due to the addition and change of specifications and their processing appears, becomes complicated, and the maintainability deteriorates, which adversely affects the automatically generated source coat.

The purpose of this study is to propose a method to suppress the complexity of software that should consider numerous exception sequences for the addition of functions in MDD. The complexity of the sequence diagram is measured using several complexity metrics, and the factors that increase the complexity due to the addition of functions are clarified.

First a case study of system development is performed. Design the target system and upgrade it by adding functions multiple times. As the configuration becomes more complicated, the class diagram and sequence diagram become more complicated, which affects the decrease in maintainability. Therefore, each version evaluates this degree of complexity to identify the cause of the rapid increase in complexity. The system developed in the case study is a route search problem. There is a moving body and a mesh-like movable area, and the current position of the moving body can be obtained from the position detection device. The direction of movement is limited to four directions, north, south, east, and west, and the start point and end point of movement are given from the outside. In addition, the movement of the moving body to the movement start point is excluded. The functional requirement of version 1 is to move in the shortest path, and the constraint is not to go back. Class diagrams and sequence diagrams designed to meet this requirement were created, and the complexity of the sequence diagrams was measured using RFC(Response for a Class). As a result of measurement for each UC (Use Case), the complexity of all UC0, UC1 and UC2 was 1.

The functional requirement of version 2 is that the required time is set for each route and the sum of them is minimized. Also, as a constraint,

"required time" is added as information required for route calculation.The functional requirement of version 2 is that the required time is set for each route and the sum of them is minimized. Also, as a constraint, "required time" is added as information required for route calculation. As a result of RFC measurement for each UC, the complexity was unchanged for UC0 and UC1 and UC2 was 3. This value is three times that of V1.

The functional requirement of version 3 is that the required time changes depending on the time of day. In addition, "current time" is added as information required for route calculation. As a result of RFC measurement for each UC, the complexity was unchanged for UC0 and UC1 and UC2 was 5. This value is five times that of V1. In addition, when measuring with RFC considering the parameters, V3 was 11 for UC2, showing a significant increase.

When exception handling was added in version 4, the complexity increased sharply due to the increase in execution paths. Therefore, the reason for the rapid increase is that there are many messages with arguments in the execution path that accompanies the occurrence of an exception.

In order to suppress the increase in complexity, we propose a method of grouping the objects that appear in the sequence diagram. It was confirmed that this method can reduce the complexity.