Journal of Robotics, Networking and Artificial Life, Vol. 2, No. 2 (September 2015), 135-139

Proposal of a Modification Method of a Source Code to Correspond with a Modified Model in MDA

Tetsuro Katayama[*], Yuuki Kikkawa[*], Yoshihiro Kita[†],
Hisaaki Yamaba[*], Kentaro Aburada[‡] and Naonobu Okazaki[*]
[*]University of Miyazaki, 1-1 Gakuen-kibanadai nishi, Miyazaki, 889-2192 Japan
[†]Kanagawa Institute of Technology, 1030 Shimo-ogino, Kanagawa, 243-0292 Japan
[‡]Oita National College of Technology, 1666 Maki, Oita, 870-0152 Japan
E-mail: kat@cs.miyazaki-u.ac.jp, kikkawa@earth.cs.miyazaki-u.ac.jp, y.kita@ccy.kanagawa-it.ac.jp, yamaba@cs.miyazaki-u.ac.jp, aburada@oita-ct.ac.jp, oka@cs.miyazaki-u.ac.jp

Abstract

This paper proposes a modification method of a source code to correspond with a modified model in MDA. The proposed method generates, translates, and modifies EAD (Extended Activity Diagram). Also, it generates a source code from the activity diagram. We use a simple ATM example to confirm availability of the method. The method can reduce time and effort to keep consistency between models and a source code after requirement specification is modified.

Keywords: MDA (Model Driven Architecture), Extended Activity Diagram, Activity diagram, Detail specification.

1. Introduction

MDA (Model Driven Architecture) is a concept of software development.[1] MDA defines five models: business model, requirement model, platform independent model (PIM), platform specific model (PSM), physics model.[2] Each Model has different abstraction level. A developer defines models and generates a less abstract model by software development in MDA. Here, MDA Tool is used to generate a less abstract model. A developer uses UML (Unified Modeling Language) [3] for modeling PIM and PSM.

Before generation of less abstract model, a developer must create generation rule of high abstract model. A method to support the creation of a generation rule is researched. [4]

One of the MDA's problems is how to keep consistency between the original model and an edited model which is generated from the original. A modification method of PIM to keep consistency with PSM is researched.[5]

Also, there is no consistency way if a developer edits the original model. A developer can keep consistency if MDA Tool generates models from the edited models again. Here, Some MDA Tool can generate a complete models from models including detail specification. A framework that generates the executable source code from a class diagram and a state machines diagram is researched.[6] However, MDA Tool cannot generate complete models from abstract models because these models do not have detail specification of a system. The developer must modify generated models to fit the modified original models or generate a new model from the modified models with MDA Tool and then add the detail specification to the new model by hand again.

The purpose of this study is to improve the efficiency of software development using MDA. This paper proposes a modification method of a source code to correspond with a modified model in MDA.

2. Proposal Method

As shown in Fig. 1, the proposed method has four functions: generate a source code from the activity diagram, generate EAD, modify EAD to correspond with the modified activity diagram, and generate a source code from the modified EAD. The proposed method consists of six steps as below.

(i) generate a source code from the activity diagram
(ii) add detail specification to the generated source code
(iii) generate an EAD
(iv) modify the activity diagram
(v) modify the EAD
(vi) generate a source code from the modified EAD

2.1. *Generate a source code from the activity diagram*

The proposed method generates a source code from the activity diagram. The steps to generate source code are shown as below.

(i) Acquire the function name
The method generates a skeleton of source code. The function name is the activity name that is described in the activity diagram. Here, the type and the parameter of the function is void.
(ii) Select the initial node
(iii) Implement the function
The method executes the process as below depending on the type of the selected node.

- Call activity node
  Write the name of the call activity node to the source code.
- Decision node
  Write an if-statement to the source code. Condition of if-statement is guard condition of this node.
- Activity final node
  Finish the generation of the source code.
- Other than the above
  Do nothing.

(iv) Select another node
The method reselects the node connected by the outgoing edge of the selected node and go to (iii).

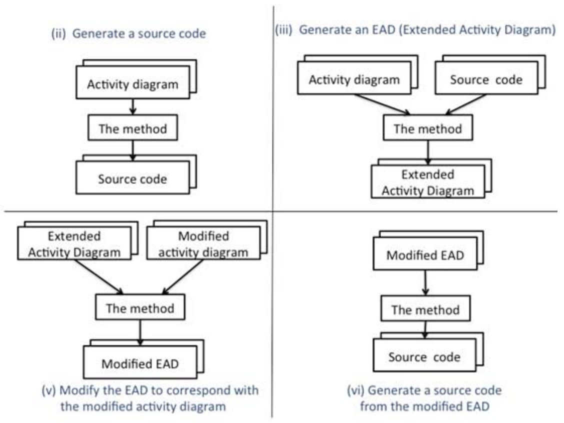

(ii) Generate a source code
Activity diagram
The method
Source code
(iii) Generate an EAD (Extended Activity Diagram)
Activity diagram
Source code
The method
Extended Activity Diagram
Extended Activity Diagram
Modified activity diagram
The method
Modified EAD
(v) Modify the EAD to correspond with the modified activity diagram
Modified EAD
The method
Source code
(vi) Generate a source code from the modified EAD

Fig. 1. Functions of the proposed method.

2.2. *Add detail specification to the generated source code*

A developer adds detail specification to the generated source code.

2.3. *Generate an EAD*

The method generates an EAD from an activity diagram and a source code added detail specification. At the first step, the method selects the first line of the source code added detail specification. Thereafter, we call the selected source code "LOS (line of selected)". The method executes the process shown as below.

(i) Extract a source code from activity diagram by the steps shown in section 2.1.
(ii) Select the next line of LOS.
(iii) If (i) and (ii) are not same character string, the method executes the process shown as below.
  (a) Write LOS to the activity diagram.
  (b) Encircle lines written in (a) as a node.
  (c) Select the original node of the extracted source code in (i).
  (d) Connect the incoming edge for the selected node to the node extracted in (b).
  (e) Make an edge connected with the selected node and the node extracted in (a).
  (f) Go to (ii).
(iv) If encircled nodes are connected each of them, the method collects them as one node.
(v) Go to (i).

2.4. *Modify the activity diagram*

At the second step, a developer modifies the activity diagram to fit the changed requirement specification.

2.5. *Modify the EAD*

The method modifies the EAD to correspond with the modified activity diagram. The method executes the process shown as below.

(i) Select initial nodes of the activity diagram.
- Call the selected node “AD selected node”.

(ii) Select initial nodes of the EAD.
- Call the selected node “EAD selected node”.

(iii) Execute the process shown as below depending on the case.
- EAD selected node is encircled node.
  - (a) Change EAD selected node to the next node of the current EAD selected node.
- AD selected node and EAD selected node have the same name.
  - (a) If the incoming edge of the AD selected node does not include guard condition, the method write guard condition to the edge of the EAD selected node.
  - (b) Change EAD selected node to the next node of the current EAD selected node.
  - (c) Change AD selected node to the next node of current AD selected node.
- AD selected node and EAD selected node do not have the same name.
  - (a) Write the AD selected node to the EAD
  - (b) Make an edge to the previous node of the EAD selected node and the node written in (a).
  - (c) Make an edge to connect the EAD selected node and the node written in (a).
  - (d) Change EAD selected node to the next node of (a).
  - (e) Change AD selected node to the next node of current AD selected node.

(iv) Go to (iii).

2.6. *Generate a source code from the modified EAD*

The method generates a new source code from the modified EAD. It has information about detail specification and is applied the changed requirement specification. Therefore, a new source code generated

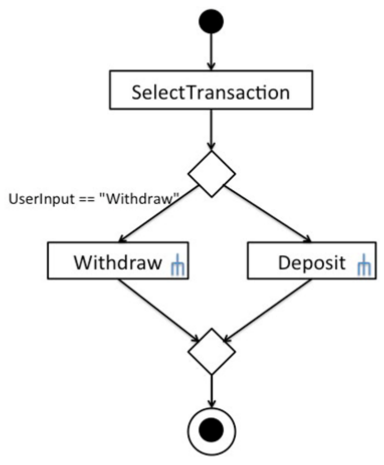
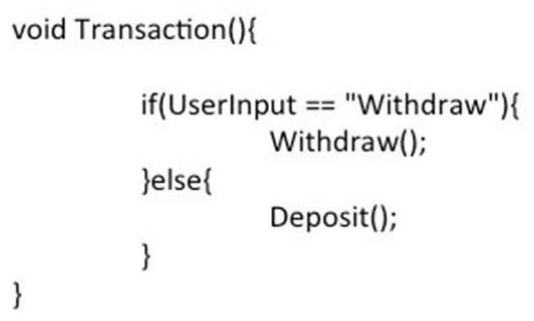


Fig. 2. The activity diagram.



```
void Transaction(){

        if(UserInput == "Withdraw"){
                Withdraw();
        }else{
                Deposit();
        }
}
```

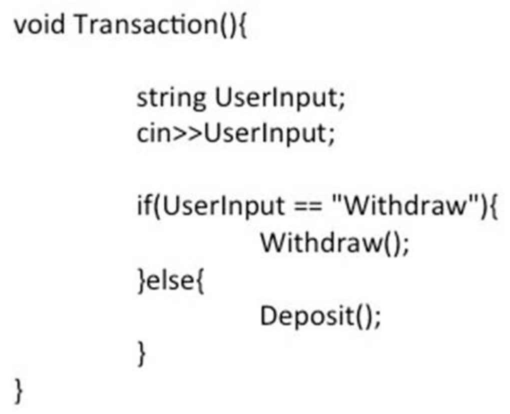
Fig. 3. The source code generated from activity diagram.

from the modified EAD corresponds with the changed requirement specification and has information about detail specification.

Here, the method can treat only if-statement. The way to treat other statements is a future issue. In addition, the method can treat with addition only to a generated source code and an activity diagram, but it cannot treat with deletion or revision. Correspondence to them is a future issue.

3. Application Example

We use a simple ATM as an example to confirm availability of the method. This ATM system executes a withdrawal process or a depositing process depending on a user input. Fig. 2 shows the activity diagram that expresses processing flow of the ATM system.



```
void Transaction(){

        string UserInput;
        cin>>UserInput;

        if(UserInput == "Withdraw"){
                Withdraw();
        }else{
                Deposit();
        }
}
```

Fig. 4. The source code added detail specification.

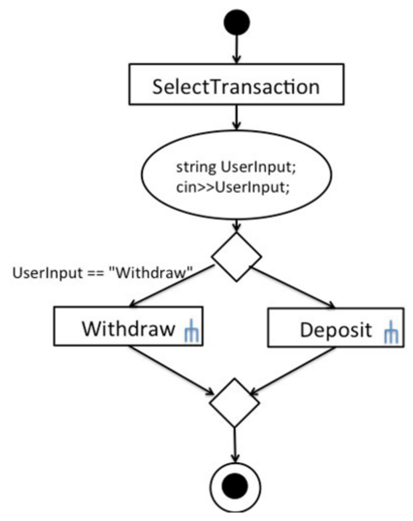


Fig. 5.   The Extended Activity Diagram.

The method generates a source code from the activity diagram. Fig. 3 shows a generated source code.

The developer adds the detail specification to the generated source code in order to execute it. The source code added the detail specification shown in Fig. 4.

Suppose a case that the requirement of specification is changed to add a process of balance checking after adding the detail specification.

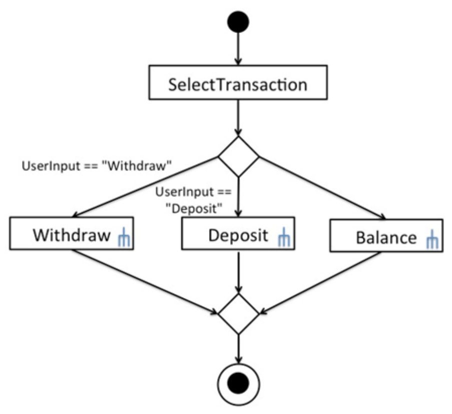


Fig. 6. The modified activity diagram.

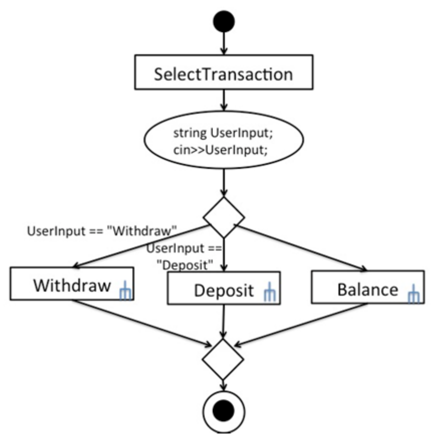


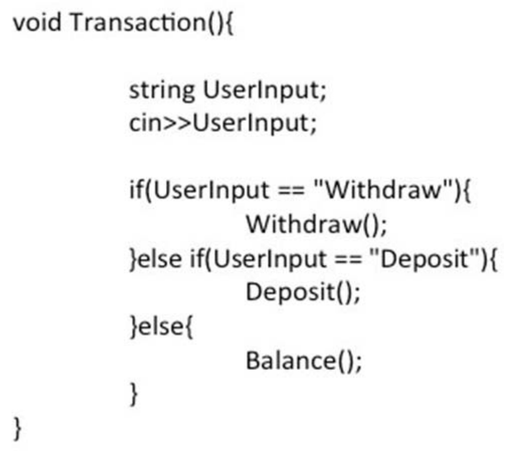
Fig. 7. The modified EAD.

The method generates the EAD from source code added the detail specification and the activity diagram shown in Fig. 2. Fig. 5 shows the generated EAD.

A developer adds the process of balance checking to the activity diagram as shown in Fig. 2. Fig. 6 shows the activity diagram added the process of balance checking.

The method modifies EAD to correspond with the modified activity diagram. Fig. 7 shows the modified EAD.

The method generates the source code from the modified EAD. Fig. 8 shows the generated source code. This source code has the detail specification.



```
void Transaction(){

            string UserInput;
            cin>>UserInput;

            if(UserInput == "Withdraw"){
                        Withdraw();
            }else if(UserInput == "Deposit"){
                        Deposit();
            }else{
                        Balance();
            }
}
```

Fig. 8. The source code generated from EAD.

4. Discussion

MDA Tool such as EA[7] (Enterprise Architecture) can generate a skeleton of a source code from a class diagram. In addition, EA can generate a source code from an activity diagram or a state machine diagram. However, The source code generated by EA does not have detail specification. It takes time and effort that the developer adds detail specification to the source code generated from the modified activity diagram.

The method can generate a source code including detail specification. The method can reduce time and effort to add detail specification to the source code generated from the modified activity diagram. Moreover, it can reduce time and effort to keep consistency between models and a source code after requirement specification is modified. Therefore, the method is useful for efficiency of software development.

5. Conclusion

This paper has proposed a modification method of a source code to correspond with a modified model in MDA. The method can generate the source code that corresponds with the modified activity diagram and has information about detail specification. We have confirmed that the method can generate a source code including the detail specification from the original activity diagram, the modified activity diagram, and the original source code. Therefore, the method is useful for efficiency of software development.

Future issues are as follows.

- Development of the tool implemented the method
- Improvement of the method to treat with deletion or revision to source code and activity diagram.
- Improvement of the method to treat with other statements except if-statement.

References

1. MDA (Model Driven Architecture), http://www.omg.org/mda (accessed February 16, 2015).
2. Wada H, Yasutake Y, MDA (Model Driven Architecture) and Actual Development Process (in Japanese), UNISYS TECHNOLOGY REVIEW, No.61 (2004), pp. 47-59.
3. UML (Unified Modeling Language), http://www.omg.org/spec/UML/2.4.1 (accessed February 15, 2015).
4. D. Lopes, S. Hammoudi, J. Bézivin, F. Jouault: Mapping Specification in MDA: From Theory to Practice, Interoperability of Enterprise Software and Applications, (Springer-Verlag London Ltd 2006), pp.253-264,
5. Ueno M, Omori M, An Approach to Keep Coherence between PIM and PSM of MDA (in Japanese), IPSJ SIG Technical Report, SE-156 (7), (2007), pp. 41-47.
6. A. Derezinska, Code Generation and Execution Framework for UML 2.0 Classes and State Machines, IMCSIT, (2008), pp. 517-524.
7. Enterprise Architect, http://www.sparxsystems.jp (accessed December 10, 2014).