An Approach to Class Diagram Design

Chebanyuk Elena
Software Engineering Department, National Aviation University, Ave Komarova 1, Kyiv, Ukraine

Keywords: Model-Driven Architecture (MDA), Transformation Rules, Behavioural Software Model, Collaboration Diagram, Set Theory Tool.

Abstract: An approach to class diagram design is suggested in this paper. The approach is based on the analysis of behavioral software models represented by collaboration diagrams. An analytical form of collaboration diagrams representation is suggested. Rules that define relations between class diagram constituents by analyzing the analytical representation of collaboration diagrams are presented. An approach to defining the relations between class diagram constituents that is based on these rules is suggested. An example of class diagram design by analyzing collaboration diagrams according to the suggested approach is presented.

1 INTRODUCTION

Using models in software development processes increases productivity of various development activities, such as domain analysis, automated code generation, designing domain specific languages, representation of a software system with necessary details, testing, requirement analysis, software documentation, code reuse and other tasks. It is a background for development of special technics and approaches for software models transformation.

Often software models are represented as UML diagrams. Most of the models that are used in software development process can be divided into static and dynamic (behavioral).

Classification of software models according to principle of division them into static and dynamic was proposed in paper (Gupta, 2012).

Today Agile methodology is widely used to create software projects (Gandomani, 2013). Main feature of Agile is possibility to change software requirements, algorithms and other artifacts after every development iteration. It result to changing of behavioral software models.

That is why the task to design languages, technics, rules and other tools for transformation of behavioral (dynamic) software models into static ones is vital problem of Model-Driven Architecture (MDA).

The paper is organized by the following way. Section 2 represents the rewiev of papers that solve tasks that are devoded to transformation of models in MDA area. The tasks of recearch is formulated in Section 3. An analytical representation of collaboration diagrams is described in Section 4. Section 5 represents rules of defining relations between class diagram constituents, that are based on analytical representation of collaboration diagrams. Section 6 represents stages of realization of the proposed approach. Section 7 contains an exmple of class diagram designing. Section 8 describes an application of the proposed approach in software development process. Section 9 containes conclusions.

2 RELATED PAPERS

Actuality of the task to design transformation approaches and technics is a background of appearing series of papers that are devoted to different aspects of MDA, namely creating of analytical tools, generating new artefacts from behavioural software models (Gupta, 2012), estimation of code reuse effectiveness, tools for an analytical description of software static models (Chebanyuk, 2013) and other aspects that are based on software models represented in a form of UML diagrams (Acretoaie, 2013), (Whittle, 2009).

Paper (Gupta, 2012) represents an approach of generating test cases based on use case models that are refined by state diagrams. But an operation (Gupta, 2012) of representation UML activity diagram as state table and writing it into in to some file is rather consuming when activity diagram is large. Also mechanical errors are possible when test cases are generated.

Recently, several approaches adopting model transformation technics to software development processes have been proposed in paper (Whittle, 2009). These approaches use the concrete syntax of the source and target models to define transformation rules, and thus propose a change to the overall model transformation mechanism.

Paper (Acretoaie, 2013) is devoted to definition and implementation of a model transformation language that is focused on usability. But transformation templates of this language that are proposed in paper (Acretoaie, 2013) relate only to class diagram. Usage of an information from behavioural models allows clarifying patterns, designing new templates, and increasing an effectiveness of models refactoring procedure.

3 TASK

Task: to propose an approach to class diagram designing using collaboration diagrams. Collaboration diagrams must meet to the following requirements: completeness, information content, accuracy and not contradictory. For this purpose we have to do the following:

- to propose an analytical representation of collaboration diagrams allowing to trace a process of objects creation;
- to formulate rules that define relations between class diagram constituents by analyzing the analytical representation of collaboration diagrams.

4 AN ANALYTICAL FORM OF COLLABORATION DIAGRAMS REPRESENTATION

4.1 Denotations for Representations of Messages and Objects

A set of collaboration diagram objects is denoted as Δ.

A set of collaboration diagram messages is denoted as E.

Consider any collaboration diagram object $\delta \in \Delta$.

Consider a set of messages that are linked directly with the object $\delta \in \Delta$ as E^{δ} ($\mathrm{E}^{\delta} \subset \mathrm{E}$). In graphical notation such messages are represented by arrows that are directed to this object.

The note: an upper index in denotation (for instance E^{δ}) shows that component in question belongs to definite whole (for instance E^{δ} -is a set of messages that are connected with the object $\delta \in \Delta$). A lower index in denotation defines a number of an element in a set.

Consider a set of collaboration diagram objects that are connected directly with the object $\delta \in \Delta$ by messages from the set E^{δ} as Δ^{δ} ($\Delta^{\delta} \subset \Delta$). A number of elements in the sets E^{δ} and Δ^{δ} is denoted as N^{δ}. (Each object $\delta \in \Delta^{\delta}$ is linked with considering object by means of one message from the set E^{δ}.)

4.2 Input Stream of a Collaboration Diagram Object

Input stream of an object $\delta \in \Delta$ defines one of the variant of this object creation.

In graphical notation the input stream of the object $\delta \in \Delta$ corresponds to fragment of collaboration diagram, that consist from three components namely: this object, a message $\varepsilon^{\delta} \in \mathrm{E}^{\delta}$ and an object $\delta^{\delta} \in \Delta^{\delta}$ that is linked with the object $\delta \in \Delta$ by message $\varepsilon^{\delta} \in \mathrm{E}^{\delta}$. Denote the number of input streams of the object $\delta \in \Delta$ as N^{δ}.

A set of input streams of an object $\delta \in \Delta$ is represented as a subset of the Cartesian product of the following sets: objects Δ^{δ} and messages E^{δ} and is denoted as follows:

$$\begin{cases} E^{\delta} = \{e_1^{\delta}, e_2^{\delta}, \ldots, e_{N^{\delta}}^{\delta}\} \\ \Delta^{\delta} = \{\delta_1^{\delta}, \delta_2^{\delta}, \ldots, \delta_{N^{\delta}}^{\delta}\} \\ E^{\delta} x \Delta^{\delta} \supset \{< e_1^{\delta}, \delta_1^{\delta} >, < e_2^{\delta}, \delta_2^{\delta} >, \ldots, < e_{N^{\delta}}^{\delta}, \delta_{N^{\delta}}^{\delta} > \end{cases} \qquad (1)$$

, where $i = 1,.., N^{\delta}$ .0

A border object of a collaboration diagram is an object $\delta \in \Delta$ which corresponds to the condition $\mathrm{E}^{\delta} = \varnothing$.

4.3 Method of Forming Complete Input Streams for an Object

A complete input stream for the object $\delta \in \Delta$ is a tuple. A tuple is formed by concatenating of input streams of different objects from the set Δ Complete input stream for the object $\delta \in \Delta$ is denoted as φ^{δ} and defined as follows:

$$\varphi^{\delta} = \underset{\substack{i=1 \\ \delta^{\phi^{\delta}}, \varepsilon^{\phi^{\delta}} \in E^{\phi^{\delta}} x \Delta^{\phi^{\delta}}}}{\overset{N^{\delta}}{\bullet}} \{< \delta_i^{\phi^{\delta}}, \varepsilon_i^{\phi^{\delta}} >\} \tag{2}$$

where $E^{\phi^{\delta}} \times \Delta^{\phi^{\delta}}$ - is a tuple of input streams which form complete input stream for the object $\delta \in \Delta$. $N^{\phi^{\delta}}$ - is a number of elements in the tuple φ^{δ}.

The first component of the tuple φ^{δ} is an input stream of the object $\delta \in \Delta$. The last component is an input stream of a border object.

Introduce the method of forming complete input stream for an object $\delta \in \Delta$.

1. A tuple of input streams $E^{\delta} \times \Delta^{\delta}$ is formed using (1). This tuple consists from N^{δ} elements.

2. N^{δ} tuples ϕ_i^{δ} , $i = 1,\dots,N^{\delta}$ are formed. A component of a tuple ϕ_i^{δ} is one element from the tuple $E^{\delta} x \Delta^{\delta}$. These tuples are denoted as follows:

$$\phi_i^{\delta} = \{< \varepsilon_i^{\delta}, \delta_i^{\delta} >\} \tag{3}$$

Denote object $\delta_i^{\delta} \in \Delta^{\delta}$ as $\delta^*, \delta^* = \delta_i^{\delta}$.

3. Consider δ^*.

3.1 If δ^* is a border object then the tuple ϕ_i^{δ} is added to a set containing all complete input streams of the object $\delta \in \Delta$. Denote this set as $\Phi^{\delta} = \{\phi_1^{\delta}, \phi_2^{\delta}, \dots, \phi_{N^{\Phi^{\delta}}}^{\delta}\}$, where $N^{\Phi^{\delta}}$ - is the number of elements in this set.

3.2 If δ^* is not a border object then expression (3) is complimented by concatenating of the following components: expression (3) and an input stream of the object δ^*. The concatenation is made according to the next rules:

3.2.1 If $N^{\delta^*} = 1$ then expression (3) is complimented by an input stream of the object δ^* using concatenation operation. The expression (3) will look like:

$$(\phi_i^{\delta})^* = \{< \phi_i^{\delta} > \bullet < \varepsilon^{\delta^*}, \delta^{\delta^*} >\} \tag{4}$$

where $\delta^{\delta^*} \in \Delta^{\delta^*}$, $\varepsilon^{\delta^*} \in \mathrm{E}^{\delta^*}$.

3.2.2 If $N^{\delta^*} > 1$ then N^{δ^*} new tuples are formed by the following way:

$$\phi_{j+N^{\delta}}^{\delta} = \{< \phi_i^{\delta^*} > \bullet < \varepsilon_j^{\delta^*}, \delta_j^{\delta^*} >\} \tag{5}$$

where $j = 1,\dots,N^{\delta^*}, \delta_j^{\delta^*} \in \Delta^{\delta^*}$, $\varepsilon_j^{\delta^*} \in \mathrm{E}^{\delta^*}$,

An object of the last input stream in the tuple (4) or (5) is considered as δ^*.

4. Actions that are described in point 3 are repeated. Doing this tuple (4) or (5) is complimented by an input stream of the object δ^* , namely the last component of the considered tuple.

5 RULES OF DEFINING RELATIONS BETWEEN CLASS DIAGRAM CONSTITUENTS

Consider a collaboration diagram that meets the following requirements: completeness, information content, accuracy and not contradictory Denote this collaboration diagram as Complete Diagram. Consider an collaboration diagram object $\delta \in \Delta$. This object has an attribute d. This attribute matches with the name of another object in the same collaboration diagram. Denote a class C of a class diagram named – "name" as $C(name)$. Consider class diagram containing classes with names $C(\delta)$ and $C(d)$. In order to define relations between these classes we have to do the following:

1 A set Φ^{δ} of the complete input streams for the object $\delta \in \Delta$ is formed.

2. The tuple $E^d x \Delta^d$ of input streams of the object $d \in \Delta$ is formed using (1).

3. Common components of tuples from all the tuples $\varphi^{\delta} \in \Phi^{\delta}$ and the tuple $E^d x \Delta^d$ are defined. In order to do this a following set is formed:

$$\mathrm{H}^{\delta} = E^d x \Delta^d \bigcap_{\substack{i=1 \\ \phi_i^{\delta} \in \Phi^{\delta}}}^{N^{\Phi^{\delta}}} \phi_i^{\delta} \tag{6}$$

3.1 If $N^d \le N^{\Phi^{\delta}}$ and $E^d x \Delta^d = H^{\delta}$ then classes $C(\delta)$ and $C(d)$ are linked by a composition relation. Explain this:

a) Presence of the attribute d in the object $\delta \in \Delta$ proves the fact that at least one operation of the object $\delta \in \Delta$ is executed using object $d \in \Delta$ or some of its properties.

b) Expression $E^d x \Delta^d = H^\delta$ and condition that collaboration diagram is complete shows that relation between classes $C(\delta)$ and $C(d)$ corresponds to whole-part relation.

c) Expression $N^{\Phi^\delta} - N^d \le 0$ and condition that collaboration diagram is complete show that the lifetime of the object $d \in \Delta$ depends upon the lifetime of the object $\delta \in \Delta$.

According to definition [Byrne, 2013] such a relation corresponds to composition relation between classes.

3.2. If $N^d > N^{\Phi^\delta}$ and $E^d x \Delta^d = H^\delta$ then classes $C(\delta)$ and $C(d)$ are linked by an aggregation relation. Explain this:

a) The first and the second points of this explanation match with the previous explanation (p. 3.1.).

b) Expression $N^{\Phi^\delta} - N^d > 0$ and condition that collaboration diagram is complete show that the object $d \in \Delta$ participates in operations that are executed by other collaboration diagram objects and the lifetime of the object $d \in \Delta$ does not depend upon the lifetime of the object $\delta \in \Delta$.

According to definition [Byrne, 2013] such a relation corresponds to aggregation relation between classes.

3.3. If $E^d x \Delta^d \ne H^\delta$ then classes $C(\delta)$ and $C(d)$ are linked by an association relation.

Explain this: expression $E^d x \Delta^d \ne H^\delta$ proves the fact that to create the object $\delta \in \Delta$ it is not necessary to involve the object $d \in \Delta$ and these objects are not in a whole-part relation. From the other hand presence of the attribute d in the object $\delta \in \Delta$ proves the fact that to execute at least one operation it is necessary to involve the object $d \in \Delta$. According to definition [Byrne, 2013] such a relation corresponds to association relation between classes.

6 APPROACH TO CLASS DIAGRAM DESIGNING

Introduce of the approach to class diagrams designing using rules of defining relations between its constituents. The source information for class diagram designing is an analytical representation of collaboration diagram.

Approach to Class Diagram designing

1. A collaboration diagram which represents all problem domain processes is designed. It should meet the following requirements: completeness, information content, accuracy and not contradictory. The correctness of a collaboration diagram is checked by a domain expert.

2. Names of class diagram classes are defined by means of matching with names of the objects in the collaboration diagram. It is denoted as follows:

$$C(\delta) = \delta \tag{7}$$

3. An analytical description of input streams for each $\delta \in \Delta$ are formed using (1). Input streams are not formed for border objects.

4. Defining of an association relations between all pairs of classes when one class is $C(\delta)$ and another one is a class $C(\delta^\delta)$, $\delta^\delta \in \Delta^\delta$. All collaboration diagram objects are considered.

5. Complete input streams for each object of the collaboration diagram are formed using (2).

6. Class diagram is complimented by composition aggregation and association relations. These relations are defined between class $C(\delta)$ and $C(\delta^{\text{I}})$ where δ^{I} matches with the name of an attribute that belongs to the object $\delta \in \Delta$.

The rules of defining relations between class diagram constituents are represented in the previous section.

7. Actions that are described in p.6 are made for every collaboration diagram object.

8. Verifying relations between classes. if between some classes association relation was set and after that aggregation or composition relation was defined then more strong relation remains. An association relation changes to composition or aggregation. Aggregation relation can be changed only to composition relation.

7 EXAMPLE OF DESIGNING A CLASS DIAGRAM ACCORDING TO THE SUGGESTED APPROACH

Consider an example to class diagram design using collaboration diagram

1. A collaboration diagram is designed (fig. 1).

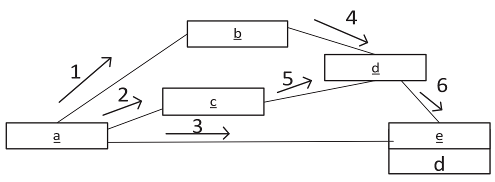

Figure 1: Collaboration diagram.

2. Names of the classes in the class diagram are defined. Thus classes are named by the following way: $C(a)$, $C(b)$, $C(c)$, $C(d)$, $C(e)$.

3. Form an analytical representation of input streams for each collaboration diagram object using (1).

$$\begin{cases} \Delta^b = \{a\} \\ E^b = \{1\} \\ E^b x \Delta^b \supset \{<1, a>\} \end{cases} \tag{8}$$

$$\begin{cases} \Delta^d = \{b, c\} \\ E^d = \{4,5\} \\ E^d x \Delta^d \supset \{<4, b>, <5, c>\} \end{cases} \tag{9}$$

$$\begin{cases} \Delta^e = \{a, d\} \\ E^e = \{3,6\} \\ E^e x \Delta^e \supset \{<6, d>, <3, a>\} \end{cases} \tag{10}$$

4. An association relations are set between the next pairs of classes: $C(a)$ and $C(b)$, $C(a)$ and $C(c)$, $C(b)$ and $C(d)$, $C(c)$ and $C(d)$, $C(d)$ and $C(e)$, $C(a)$ and $C(e)$.

5. Form the complete input stream for the object e using (2) and analyzing (8), (9) and (10).
5.1 $N^e = 2$, thus: $\phi_1^e = \{<6, d>\}, \phi_2^e = \{<3, a>\}$
5.2 $N^d = 2$, thus: $\phi_3^e = \{<\phi_1^e > \bullet <4, b>\}, \phi_4^e = \{<\phi_1^e > \bullet <5, c>\}$
5.3 $N^b = 1$, thus:

$$\phi_3^{e^*} = \{<\phi_3^e > \bullet <1, a>\} \tag{11}$$

The object a is the border object. The forming of complete input stream is stopped.
5.4 Other input streams of the object e are formed similarly:

5.4.1 Input stream ϕ_4^e

$$\phi_4^e = \{<\phi_1^e > \bullet <5, c> \bullet <2, a>\} \tag{12}$$

5.4.2 Input stream ϕ_5^e

$$\phi_5^e = \{<3a>\} \tag{13}$$

$\Phi^e = \{\phi_3^{e^*}, \phi_4^e, \phi_5^e\}$ thus $N^{\Phi^e} = 3$.

6. Compliment the class diagram by composition aggregation and association relations.

Consider the object e containing the attribute d. Define relations between classes $C(e)$ and $C(d)$.
6.1 The set of the input streams for the object d is formed (10).

6.2 The set H^e is formed using (6). Thus, consider the expression $E^d x \Delta^d$ and (11), (12) and (13). Thus: $H^e = \{4b, 5c\}$.

As $N^d < N^{\Phi^e}$ and $\Phi^d = H^e$, then between classes $C(e)$ and $C(d)$ aggregation relation is set.

According to the p.8 the association relation between classes $C(e)$ and $C(d)$ is changed to the aggregation one.

Class diagram that was designed according to this approach is represented in fig. 2.

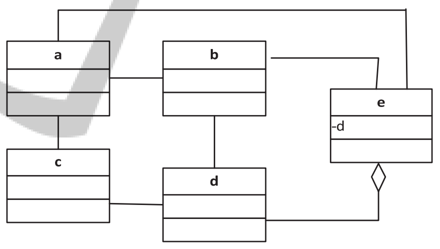

Figure 2: Class diagram, designed according to the proposed approach.

8 APPLICATION OF THE PROPOSED APPROACH

The schema of development iterations organization according to the proposed approach is represented in fig.3.
Changing aspects in software development iterations, namely algorithms, software functional requirements, business needs and processes correct behavioral software models in every iteration. It initiates correction of class diagrams (adding, replacement or removement some components). Due to possibility of providing automatic transformation the proposed approach allows stakeholders to concentrate on software behavioral aspects, instead verification, checking and making correction to class diagram.

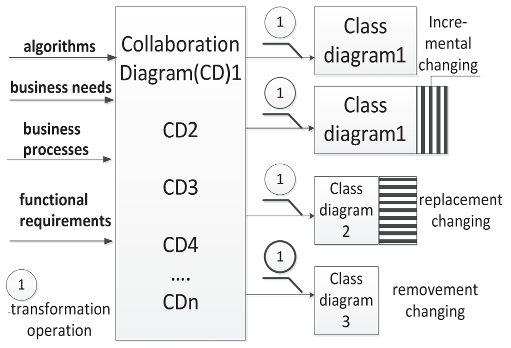

Figure 3: the schema of iteration organization according to the proposed approach.

9 CONCLUSIONS

The survey of papers on software model transformation and investigation of other MDA problems shows that researchers usually present transformation technics for software models of the same type (for example, from static into static or from dynamic into dynamic).

But the task of designing approaches to transformation of dynamic software models into static ones is a vital task of MDA.

The use of the approach to class diagram design and of the rules of defining relations between its constituents that are suggested in this paper will allow:

- designing new and specifying the existing source software models for refactoring in paper (Acretoaie, 2013);
- generating test cases in the approach suggested in paper (Gupta, 2012), at the same time using information contained in several dynamic software models based on analytical representations of dynamic software models;
- extending the syntax suggested in paper (Whittle, 2009) for representing source software models.

Using the approach for transformation of behavioral software models into static ones will allow efficiency improvement of solving problems concerning further processing of changed software static models.

The rules of defining relations between class diagram constituents suggested in this paper can be used in methods and techniques employed for solving software model transformation problems. The use of such methods and techniques is helpful in solving many vital problems of MDA.

REFERENCES

Gupta, S., Singla, J., 2012. A component-based approach for test case generation. *International Journal of Information Technology 5.2*,Pages 239-243, 2012.

Gandomani, T. J., Zulzalil, H., Ghani, A. A. A., & Sultan, A. B. M., 2013. *Towards Comprehensive and Disciplined Change Management Strategy in Agile Transformation Process*. http://www.maxwellsci.com/print/rjaset/v6-2345-2351.pdf.

Chebanyuk, E., Algebra describing software static models. *International journal "Information technologies & knowledge" 7.1*, Pages 83-93, 2013.

Acretoaie, V., Delivering the Next Generation of Model Transformation Languages and Tools. *Europian conference of object oriented programming, Pages 2-10*, 2013.

Whittle, J., Jayaraman, P., Elkhodary, A., Moreira, A., Ara_ujo, J., MATA: A United Approach for Composing UML Aspect Models Based on Graph Transformation. In: *Transactions on Aspect-Oriented Software Development VI - Special Issue on Aspects and Model-Driven Engineering, Pages* 191-237, Springer, 2009.

Byrne, B., M., and Yasser S., Q., *The use of uml class diagrams to teach database modelling and database design.* Friday 5th July 2013 University of Sunderland, 2013.