

---

## METHOD OF BEHAVIORAL SOFTWARE MODELS SYNCHRONIZATION

Elena Chebanyuk

**Abstract:** A method of behavioral software models synchronization is represented in this paper. Implementing this method behavioral software models, which are changed after communication with customer, are synchronized with other software models that are represented as UML diagrams. Method of behavioral software artifacts synchronization makes the Model-Driven Development (MDD) approach more effective. For synchronization of different behavioral software models, transformation approach in the area of Model-Driven Architecture (MDA) is proposed. Synchronization operation is executed using analytical representation of initial and resulting models. Initial behavioral software model is represented by UML Use Case Diagram. Resulting behavioral software model is represented as UML Collaboration Diagram. Analytical representation of UML Use Case diagram allows considering data flows. For this representation set-theory tool operations are used. As a Collaboration Diagram usually contains more information in comparison with Use Case one, method defines two types of Use Case diagram fragments. From the beginning Use Case diagram fragments that can be transformed directly to resulting diagram constituents are considered. Then the rest of Use Case diagram fragments are processed to represents rules of placement Collaboration Diagram messages. These rules help to designate data flows, represented in Collaboration Diagram, more accuracy.

Method, proposed in this article, can be used both separately and be a part of more complex transformation technics, methods and frameworks solving different tasks in MDA sphere. Also the example of proposed method realization for solving task "designing of Vector hodograph of density lying function" (VHDLF) is represented. A process of designing Collaboration Diagram, considering Use Case diagram and ontology knowledge analysis is represented. The constituents of Collaboration Diagram, which are designed using different sources, namely Use Case diagram, ontology knowledge and requirements specification are defined.

**Keywords:** Software model transformation, Use Case diagram, Set-theory tool, behavioral software model synchronization, Vector hodograph of density laying function.

**ACM Classification Keywords:** D.2.2. Design Tools and Techniques, D.2.11. Software Architectures

---

### Introduction

Today software development process according to agile methodology becomes more widespread. One of the peculiarities of Agile is possibility to change software requirements in every development iteration.

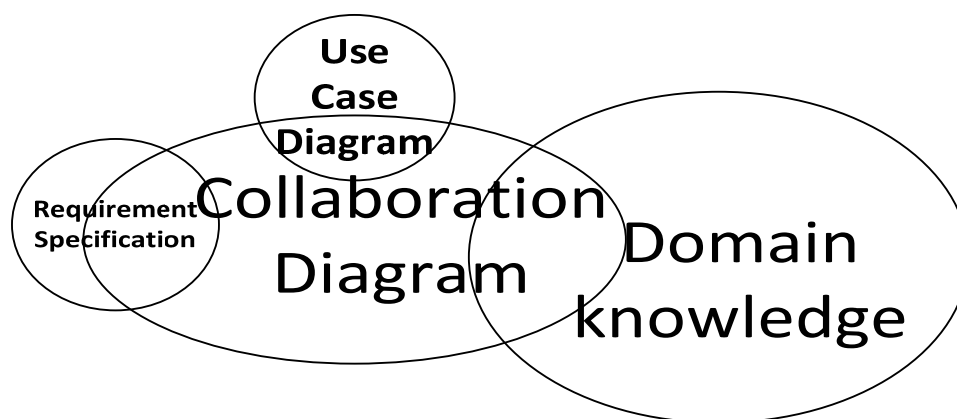
When requirements are changed often it's necessary to design methods for quick synchronization of software models that are renewed after communication with a customer and other software artifacts. Among software artifacts that are renewed after communication with customer are UML Use Case diagrams or their varieties (user stories).

Central software models in MDD approach are Collaboration Diagrams. These diagrams can both represent processes and define objects that are used for executing these processes. Collaboration Diagrams also are

sources for refinement of algorithms, generation of test cases, analysis of objects, editing of processes and designing of static software models (Class and Packages diagrams).

Use Case and Collaboration Diagrams are examples of behavioral software models [Gupta, 2012]. They are also called Computation Independent Models (CIM) in MDA approach. The purpose of CIM models in MDA approach is to represent processes and algorithms of solving software tasks and an order of objects collaboration [Marín, 2013].

It is very important after every requirement changing to obtain actual Collaboration Diagram. As Collaboration Diagram contains more information in comparison with Use Case diagram, it is necessary to use additional sources for executing synchronization operation of these diagrams. Other sources of information for designing Collaboration Diagrams are domain knowledge, requirement specification and other behavioral software models that are represented as UML diagrams (Figure 1). Keeping this condition one can obtain a Collaboration Diagram satisfying Model Driven Engineering (MDE) approach requirements.



**Figure 1.** Information sources for designing of Collaboration Diagram

Application of approaches, allowing behavioral software models transformation helps to raise effectiveness of solving the next tasks:

- Design model transformation tools, methods and technics in MDA sphere;
- Develop techniques of software models processing and analyzing;
- Synchronize software artifacts.

An analytical representation of information about behavioral software models can be used for successful solving of the next tasks:

- Maintaining history of artifacts changing;
- Designing tools and frameworks for checking whether software model corresponds to MDE requirements;
- Checking, merging, reusing and executing other operations of software models processing;
- Designing new and extending existing notation of formats for saving information about software models (for example XML);
- Requirements elicitation process and other activities.

---

That is why the task to design a method for synchronization of Collaboration Diagrams with artifacts that are changed after communication with customer using an analytical representation of input and resulting software diagram is actual.

---

### Related works

---

Necessity for software artifacts synchronization is a cause of appearing series of papers that are devoted to this question.

Authors [Tombe, 2014] proposed using UML Use Case diagrams for maintaining requirements specification to capture scenario requirements as per the software maintenance tasks to be performed. Then Use Case diagrams are translated into Use Case model, from which the analysis model is derived, and then the models of the subsystem are designed from the analysis model to map into the existing architectural design of the ready system.

Paper [Daud, 2014] presents review of requirement engineering tools. Using this tools one can execute tracing requirements activities for software development process.

Authors proposed to divide the requirement engineering tools available in the market into two main categories: the commercial Requirements Engineering (RE) tools and the RE research tools. The requirement engineering tools for the comparison analysis comprises of three commercial requirement engineering tools, namely the RAQuest, QPack Tool and Enterprise Architect and four requirement engineering research tools which are the EA-Miner Tool, LRS Requirements, WikiReq system and Nocuous Ambiguity Identification (NAI). Main fourth requirement engineering activities were defined by authors, namely requirements elicitation and analysis, requirements validation and requirements management. Both the commercial and research requirement engineering tools support just a part of the requirement activities and focus on a partial solution for a particular requirements management activity.

Also involving both analytical representation of behavioral software model and transformation approaches into software development process [Chebanyuk, 2014] simplifies operations of static and behavioral software models synchronization, model transformation operations, improves model checking process, requirement validation and verification operations.

Research, presented in paper [Goknul, 2014], and has been devoted to relating requirements and design artifacts from source code. This paper presents an approach for generation and validation of traces between requirements and architecture. The approach directed for improving the currently observed practices by providing a degree of automation that allows faster trace generation and improves the precision of traces by validating them.

First, by using architectural verification, traces that otherwise would be missed in case of manual assignment and informal reasoning are discovered. Second, by using trace validation, we may reveal traces that are false positive traces.

An analytical background of approach that helps to trace requirements includes a representation of processes as subsets of Cartesian products of different sets. Other operations from the set theory tool give the schematic view of the relations between requirements and architecture. The definition of the requirements trace types formalizes the intuition that a part of software architecture is an implementation of a set of requirements. Approach proposes some iteration. Number of links may be increased after every requirements elicitation operations.

It is necessary to notice [Diskin, 2014], that the task of software artifacts synchronization is considering in MDD approach as vital. MDE approach poses several challenges for transformation tools, e.g. support of bidirectionality, instrumentality, informational symmetry, and ultimately concurrent updates. Having taxonomy of synchronization behaviors, with a clear semantics for each taxonomic unit, could help to manage these problems. Authors of paper presented a taxonomic space of model synchronization types and provided it with formal semantics. They considered computational and form a taxonomic plane classifying pairs of mutually inverse transformation operations. Also they proposed to classify relationships of organizational dominance between the models to be kept in synchronization. This allows infer the requirements for model transformations stools and theories to be applied to the problem. This knowledge can be useful both for MDE tools users and MDE tools builders for specifying MDE tools capabilities and behavior.

### Collaboration Diagram designing using both Use Case diagrams and other knowledge sources

#### Analytical representation of Use Case diagram

Use Case diagram consists from subsystems. A set of all subsystems is denoted as follows:  $\Omega$ . Consider a Use Case diagram subsystem  $\omega \in \Omega$  and its constituents, namely: precedents, actors, precedents with marks <<include>> precedents with marks <<extends>> and comments.

Introduce the following notation:

A set of actors in subsystem  $\omega \in \Omega$  is denoted as follows  $A^\omega$ .

A set of precedents in subsystem  $\omega \in \Omega$  is denoted as follows  $P^\omega$ .

A set of precedents in subsystem with <<include>> mark is denoted as follows  $P^\omega(\text{include})$ .

A set of precedents in subsystem with <<extends>> mark is denoted as follows  $P^\omega(\text{extends})$ .

A set of comments in subsystem is denoted as follows  $K^\omega$ .

A set of conditions transition between subsystem elements in subsystem  $\omega \in \Omega$  is denoted as follows  $Y^\omega$ .

A set of associations between elements in subsystem  $\omega \in \Omega$  is denoted as follows  $T^\omega$ .

Define  $\omega \in \Omega$  as a subset of Cartesian product of the following sets:  $A^\omega, P^\omega, P^\omega(\text{include}), P^\omega(\text{extends}), K^\omega, Y^\omega, T^\omega$ .

$$\left\{ \begin{array}{l} \omega \subseteq P^\omega \times K^\omega \times A^\omega \times P^\omega(\text{include}) \times P^\omega(\text{extends}) \times Y^\omega \times T^\omega \\ P^\omega \times K^\omega \times A^\omega \times P^\omega(\text{include}) \times P^\omega(\text{extends}) = \{ \langle p_1 p_2 \rangle, \langle a_1 p_1 \rangle, \langle a_1 k_1 p_1 \rangle, \langle p_1 k_1 p_2 \rangle, \\ \langle p_1 v_1 p_2 \rangle, \langle p_2 \tau_1 p_4 \rangle, \langle p_1(\text{include}) p_3 \rangle, \langle p_1(\text{extends}) p_4 \rangle, \langle p_5 p_6 \bullet p_5 p_7 \rangle, \\ \langle \text{sign} p_2 \text{sign} p_4 \rangle \} \end{array} \right. \quad (1)$$

where  $p_i \in P^\omega, i = 1, \dots, 7, k_1 \in K^\omega, a_1 \in A^\omega, p_3(\text{include}) \in P^\omega(\text{include}),$

$p_3(\text{extends}) \in P^\omega(\text{extends}), v_1 \in Y^\omega, \tau_1 \in T^\omega.$

The expression  $P^\omega \times K^\omega \times A^\omega \times P^\omega(\text{include}) \times P^\omega(\text{extends})$  describes an analytical representation of all possible combinations of constituents for subsystem  $\omega \in \Omega$ .

Preparing an analytical representation using expression (1) it is necessary to consider that the order of constituents in "<" and ">" brackets corresponds to the order of Use Case diagram elements.

Matching of constituents when a Use Case diagram is transformed to Collaboration one is denoted as follows (Table. 1):

**Table 1.** Matching of Use Case diagram constituents to Collaboration Diagram constituents

Use Case diagram constituent	Collaboration diagram constituent	Reasoning of matching
Actor	Object	Both actors and objects make actions
Precedent	Message	Both precedent and messages serve for representing actions
Comment	Comment	The same meaning

When a Use Case diagram is transformed to Collaboration Diagram the next **rule of precedent arrangement in Use Case diagram** is used: the order of precedent placement should repeat the sequence of processes that should be realized in software. If the Use Case diagram precedent has less number the process, matching to this precedent should be executed before process or part of the process represented by precedent with bigger number.

Before transforming a Use Case diagram to Collaboration one, all the Use Case Diagram precedents should be numbered according to this rule. Also messages in resulting Collaboration Diagram are numbered according to this rule too.

#### **Rules of placement messages for designing Collaboration Diagram from the Use Case one**

When a Use Case Diagram precedent matches to a Collaboration Diagram message (Table 1) three cases can be considered:

- One precedent corresponds to one message;
- ONE precedent corresponds to several messages;
- Several precedents correspond to one message.

Also a precedent in Use Case diagram does not define all messages in Collaboration Diagram because a Collaboration Diagram contains more information in comparison with Use Case diagram (Figure1). Transforming Use Case diagram precedents to Collaboration Diagram messages we define rules for placement of Collaboration Diagram messages sequence. In order to do this notation of meta-language for description of problem domain processes is used [Chebanyuk, 2013].

According to this notation a sequence from  $n$  operations that are executed for solving some task is denoted as follows:

$$H = \{p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_n\} \quad (2)$$

where  $p_i$  - is an operation from the sequence H. The sign  $\rightarrow$  shows that sequence of operations is important.

Consider a fragment of a Use Case diagram that consist from two precedents  $p_1$  and  $p_n$  connected by a link (Figure 2a). Define such a fragment as  $P^{1n}$ .

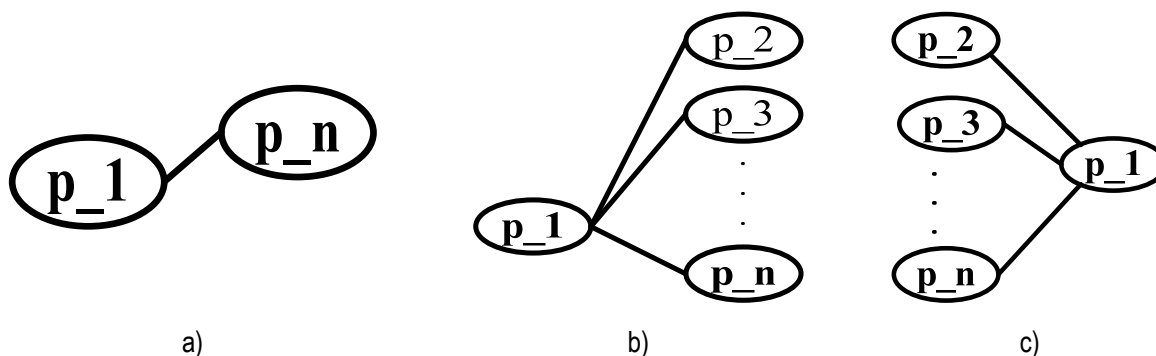


Figure 2. Use Case diagram fragments

Analytical representation of fragment  $P^{1n}$  (Figure 2a) corresponds to the next expression:  
 $P^{1n} = \langle p_1 p_n \rangle$  where  $p_1, p_n \in P^{\omega}$ .

As Use Case Diagram precedents correspond to Collaboration Diagram messages (Table 1), an analytical representation of Use Case Diagram fragment  $P^{1n}$  (Figure 2a) is denoted as follows:

$$H = \{p_1 \rightarrow P^l \rightarrow p_n\} \quad (3)$$

where  $P^l$  - is a set of operations that can be executed between operations  $p_1$  and  $p_n$ . Existence of the set  $P^l$  is explained by the fact that Collaboration Diagram contains more information in comparison with Use Case Diagram.

**The first rule of Collaboration Diagram messages placement** is formulated as follows: when fragment  $P^{1n}$  (Figure 2a) of Use Case diagram is transformed to Collaboration one, message  $P_1$  should be placed before message  $p_n$ . In some cases the set  $P^l$  can be empty.

Formulate the second rule of Collaboration Diagram messages placement. Consider precedents that are located as it is represented in Figure 2b. From precedent  $p_1$ ,  $n$  brunches are started. Every brunch links the precedent  $P_1$  with one of the precedents  $p_{1+i}$ ,  $i = 2, \dots, n$ . Denote this fragment as  $P^{1(23\dots n)}$ . An analytical form for representation such a fragment is denoted as follows:

$$P^{1(23\dots n)} = \langle p_1 p_2 \bullet p_1 p_3 \dots \bullet p_1 p_n \rangle = \langle p_1 \bullet (p_2 \bullet p_3 \bullet \dots \bullet p_n) \rangle \quad (4)$$

In expression (4) the order of multipliers matches with the order of precedents arrangement (see Figure 2b). In other word from the beginning operation  $p_1$  is executed then one operation from the set  $\{p_2 p_3 \dots p_n\}$  is executed.

Branching of the precedents shows that every pair of precedents  $p_1$  and  $p_i, p_i = 2, \dots, n$  can be represented using expression (3). But it is necessary to take into account the fact, that the choice of transition variant is determined by some condition. This condition is defined by peculiarities of concrete task. A set of all possible conditions providing transition from the precedent  $p_1$  to precedent  $p_i, p_i = 2, \dots, n$  is denoted as follows:

$T^{p_1} \subseteq T^\omega$ . Using (3) and  $\tau_k \in T^{p_1}$  where  $T^{p_1} \subseteq T^\omega$  an analytical representation of the Use Case Diagram fragment  $P^{1(23\dots n)}$  (Figure 2b) is formed:

$$\left\{ \begin{array}{l} H_1 = \tau_i \{p_1 \rightarrow P^1 \rightarrow p_2\} \\ H_2 = \tau_j \{p_1 \rightarrow P^2 \rightarrow p_3\} \\ \cdot \\ \cdot \\ H_n = \tau_k \{p_1 \rightarrow P^n \rightarrow p_n\} \end{array} \right. \quad (5)$$

Every row of the expression (5) shows that if some condition from the set  $T^{p_1} \subseteq T^\omega$  is "true", then the sequence of operations  $H_i, i=1, \dots, n$  is executed.

**The second rule of Collaboration Diagram Messages placement** is formulated as follows: if in a Use Case diagram precedent  $p_1$  is linked with precedents  $p_i, p_i = 2, \dots, n$  pair wise then the message  $p_1$  should be located before any message  $p_i, p_i = 2, \dots, n$  in corresponded Collaboration Diagram. Also the set  $T^{p_1} \subseteq T^\omega$  of conditions should be formulated.

Reasoning by analogy formulates an analytical representation of Use Case Diagram fragment  $P^{(23n)1}$  (Figure 2c).

$$P^{(23n)1} = \langle p_2 p_1 \bullet p_3 p_1 \dots \bullet p_n p_1 \rangle = \langle (p_2 \bullet p_3 \bullet \dots \bullet p_n) \bullet p_1 \rangle \quad (6)$$

An analytical representation of the third rule of transformation a Use Case Diagram into Collaboration one is denoted as follows:

$$\left\{ \begin{array}{l} H_1 = \tau_i \{p_2 \rightarrow P^1 \rightarrow p_1\} \\ H_2 = \tau_j \{p_3 \rightarrow P^2 \rightarrow p_1\} \\ \cdot \\ \cdot \\ H_n = \tau_k \{p_n \rightarrow P^n \rightarrow p_1\} \end{array} \right. \quad (7)$$

Also a set  $T^{p_2 p_3 \dots p_n} \subseteq T^\omega$  of conditions, allowing passing from the precedent  $p_i, i = 2, \dots, n$  to precedent  $p_1$  is formed.

**The third rule of Collaboration Diagram Messages placement** is formulated as follows: if in a Use Case diagram precedents  $p_i, i = 2, \dots, n$  are linked with precedent  $p_1$  pairwise then in Collaboration Diagram message  $p_1$  should be located after all messages  $p_i, i = 2, \dots, n$ . Also the set of conditions  $T^{p_2 p_3 \dots p_n} \subseteq T^\omega$  should be formulated.

Consider cases of transforming Use Case Diagram fragment, looking as fragment in Figure 2a, but adding the condition that links between precedents  $p_1$  and  $p_n$  contain marks `<<include>>` or `<<extends>>` (Figure 3)



**Figure 3.** Use Case Diagram fragments that have different marks between precedents

Consider a Use Case Diagram fragment where two precedents are linked with `<<include>>` mark (Figure 3a).

An analytical form of representation such a fragment is:  $\langle p_1(\text{include})p_n \rangle$ .

Formulate an analytical representation of the fourth rule of transformation a Use Case Diagram into Collaboration one. Consider expression (3). According to Use Case diagram notation if link between two precedents contains `<<include>>` mark, actions  $p_1$  and  $p_n$  should executed consequently. That is why  $P^1 = \emptyset$ . As a result we obtain expression:

$$H = \{p_1 \rightarrow p_n\} \quad (8)$$

**The fourth rule of Collaboration Diagram Messages placement** is formulated as follows: if in a Use Case Diagram precedents  $p_1$  and  $p_n$  are linked with `<<include>>` mark then in corresponding Collaboration Diagram messages  $p_1$  and  $p_n$  are located sequentially.

Consider a Use Case Diagram fragment where two precedents are linked with `<<extends>>` mark (Figure 3b).

An analytical representation of such a fragment is denoted as follows:  $\langle p_1(\text{extends})p_n \rangle$ .

An analytical representation of the fifth rule of transformation a Use Case Diagram into Collaboration one is denoted as follows:

$$H = \begin{cases} \tau_1 \{p_2 \parallel p_1\} \\ \tau_2 \{p_1 \rightarrow p_2\} \\ \tau_3 \{p_1\} \\ \tau_4 \{p_2\} \end{cases} \quad (9)$$



The sources of formulating the set  $T^{p_1(\text{extends})p_n} \subseteq T^\omega$  of conditions are Use Case Diagram requirement specification document and domain knowledge, namely information about business processes.

$\tau_1 = \text{true}$  if messages  $p_2$  and  $p_1$  are executed in parallel.

$\tau_2 = \text{true}$  if message  $p_1$  is executed before message  $p_2$ .

$\tau_3 = \text{true}$  if existence of the message  $p_2$  in Collaboration Diagram is not obligatory.

$\tau_4 = \text{true}$  if existence of the message  $p_1$  in Collaboration Diagram is not obligatory.

If some conditions cannot be defined the variant  $H = \{p_1 \rightarrow p_2\}$  is chosen by default.

**The fifth rule of Collaboration Diagram Messages placement** is formulated as follows: if in a Use Case Diagram precedents  $p_1$  and  $p_2$  are linked with <<extends>> mark then in corresponding Collaboration Diagram messages  $p_1$  and  $p_2$  are located sequentially. The order of execution and necessity of presents one of these messages can be defined more exactly by means of conditions from the set  $T^{p_1(\text{extends})p_n} \subseteq T^\omega$ .

Consider a case when two Use Case Diagram precedents are linked by means one of some association type, namely: (1 1), (1 \*), (\* 1) or (\* \*).

An analytical form of representation such a fragment is:  $\langle p_1^{sign} p_2^{sign} \rangle$ , where sign specifies the association type, namely 1 or \*.

An analytical representation of the sixth rule of transformation a Use Case Diagram into Collaboration one is denoted as follows:

$$H = \{p_2 \xrightarrow{\text{sign} \quad \text{sign}} p_1\} \quad (10)$$

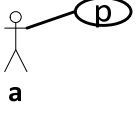
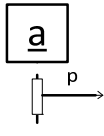
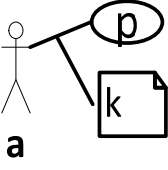
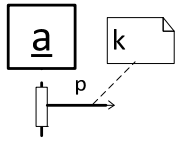
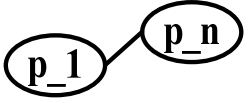
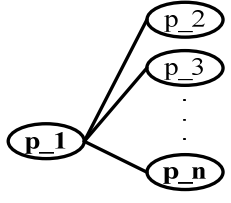
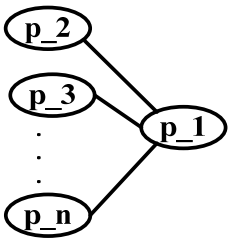
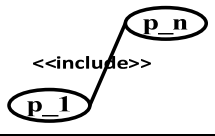
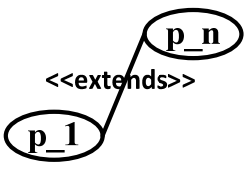
The first mark "sign" specifies multiplicity for the precedent  $p_1$  second  $p_1$  respectively.

**The sixth rule of Collaboration Diagram Messages placement** is formulated as follows: if in a Use Case Diagram precedents  $p_1$  and  $p_2$  are linked with multiplicity mark then sign=\* can denote a collection of objects, to which message is directed.

**Variants of Use Case Diagram fragments transformation to Collaboration Diagram fragments and rules**

Table 2 systemizes the variants of transformation Use Case Diagram fragments to Collaboration one (rows one and two). Also the Table 2 gives an analytical representation of the rules of Collaboration Diagram Messages placement (other rows). These rules are used when direct transformation doesn't contain all necessary information for obtaining Collaboration Diagram satisfying MDE requirements.

**Table 2** Matching Use Case Diagram fragments into Collaboration Diagram fragments and rules

Use Case diagram fragment	An analytical representation of this fragment	An analytical representation of rule defining Collaboration Diagram Messages placement	Collaboration diagram fragment
1	2	3	4
	$\langle ap \rangle$	-	
	$\langle akp \rangle$	-	
	$\langle p_1 p_2 \rangle$	$H = \{p_1 \rightarrow P^1 \rightarrow p_n\}$	-
	$\langle p_1 \bullet (p_2, p_3 \dots p_n) \rangle$	$\left\{ \begin{array}{l} H_1 = \tau_i \{p_1 \rightarrow P^1 \rightarrow p_1\} \\ H_2 = \tau_j \{p_1 \rightarrow P^2 \rightarrow p_2\} \\ \vdots \\ H_n = \tau_k \{p_1 \rightarrow P^n \rightarrow p_n\} \end{array} \right.$	-
	$\langle (p_2, p_3 \dots p_n) \bullet p_1 \rangle$	$\left\{ \begin{array}{l} H_1 = \tau_i \{p_2 \rightarrow P^1 \rightarrow p_1\} \\ H_2 = \tau_j \{p_3 \rightarrow P^2 \rightarrow p_1\} \\ \vdots \\ H_n = \tau_k \{p_n \rightarrow P^n \rightarrow p_1\} \end{array} \right.$	-
	$\langle (p_1(\text{include})p_2) \rangle$	$H = \{p_1 \rightarrow p_n\}$	-
	$\langle (p_1(\text{extends})p_2) \rangle$	$H = \left\{ \begin{array}{l} \tau_1 \{p_2 \parallel p_1\} \\ \tau_2 \{p_1 \rightarrow p_2\} \\ \tau_3 \{p_1\} \\ \tau_4 \{p_2\} \end{array} \right.$	-

The note: As comments' are transformed definitely they are not considered in the Table 2 (except row 2).

### Method of transformation a Use Case diagram into Collaboration one

1. An analytical representation of Use Case Diagram according to expression (1) is formulated.

Introduce the denotation for the representation of precedent brunches in Use Case Diagram. Example of brunches is represented in the Figure 2b. The precedent  $p_1$  is denoted as a root precedent. The sequence of linked precedents that are followed after the root precedent is denoted as follows:

$$p^1_{-p_i} (< \mu_1 >, < \mu_2 >, \dots < \mu_n >) \quad (11)$$

where  $p_i$  – can be root precedent for the new fragment or the last precedent in chain of precedents,  $< \mu_i >$  - is an element of chain number  $i$ , containing an analytical representation of a Use Case diagram fragment.  $n$  - number of fragments in the chain.

2. Define constituents in analytical representation of Use Case diagram that are transformed to Collaboration Diagram fragments directly (Table 2).

3. Define constituents in analytical representation of Use Case diagram that are used to formulate rules of Collaboration Diagram messages placement (Table 2).

4. Design a Collaboration Diagram using all data obtained in pervious points and other sources of information, namely requirements specification, domain knowledge, information about business processes.

5. Refine a Collaboration Diagram in order to define whether if satisfies to MDE requirements.

### Designing Collaboration Diagram for solving task "Plotting of vector hodograph of density laying function" for two details that have arbitrary shape of outer contour"

Consider a Use Case Diagram that was designed after communication with customer (Figure 4) for solving this task.

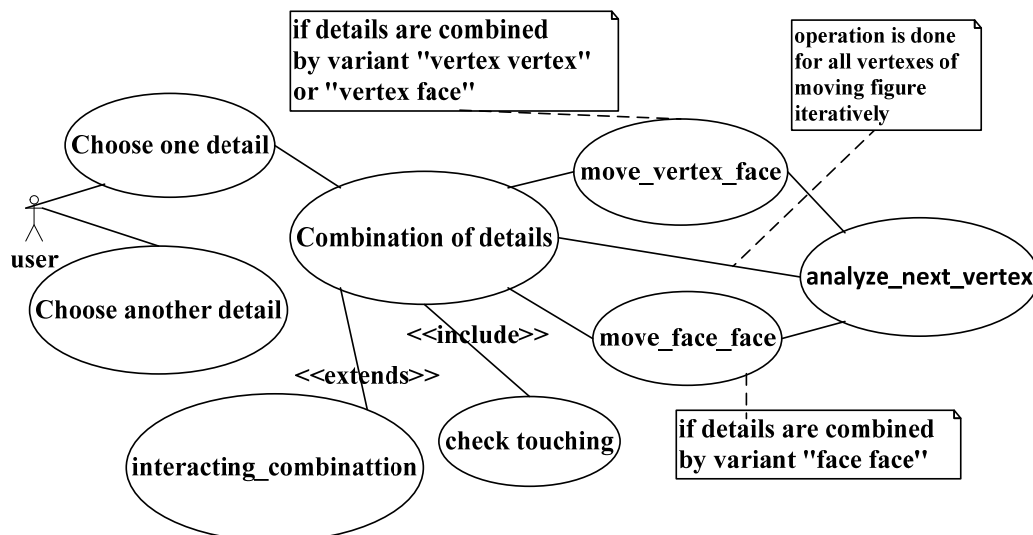


Figure 4. Use Case Diagram of task designing VHDLF defining of polygons mutual alignment on the plane

As for designing Collaboration Diagram, that meets MDE requirements, it is necessary to involve both Use Case diagram and other sources we represent *ontology knowledge* and *business processes description* of investigated problem domain.

*Terms from the vocabulary of problem domain ontology*

Pole of detail – any point inside of the detail.

Stationary detail – detail that does not move when VHGLF is defined.

Movable detail – detail that moves around stationary detail when VHGLF is defined.

Consider Figure 4. Triangle is movable detail, quadrangle is stationary ones.

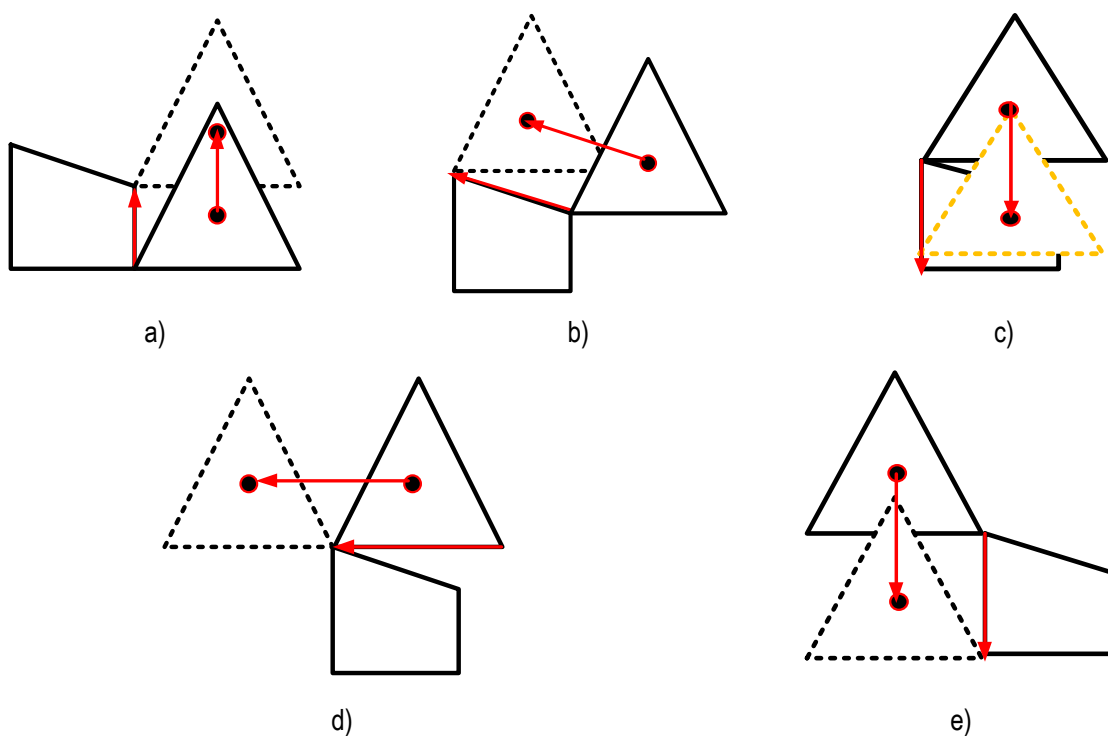
Offset vector – vector that defines shifting of movable detail for defining the next point of VHGLF.

*Peculiarities of business processes in considered problem domain*

Represent a short description of an algorithm allowing designing VHGLF and functional requirements to application that realizes described algorithm.

The Algorithm for designing VHGLF for two kinds of details which have an arbitrary form of outer contours:

1. Two details are packed densely (Figure 5a).
2. An offset vector for next position of movable detail is defined. The principles of an offset vectors defining are shown in the Figure 5. In some cases the offset vector is defined by face of stationary detail. (Figure 5a, 5b, 5e). When this kind of moving (Figure 5c) causes intersection of details the offset vector is defined by face of movable detail (Figure 5d).
3. Movable detail is shifted on the offset vector.
4. The coordinates of movable detail pole are added into array of VHGLF coordinates.
5. Check whether the current coordinate of movable detail pole matches to the first hodograph point. If no then go to the p.2 else the designing of VHGLF is finished.



**Figure 5.** VHGLF designing

Represent a short description of this Use Case Diagram (Figure 4).

Use Case Diagram represents main actions that are done when VHGLF is designed.

1. Two details, namely stationary and movable are packed in laying (Figure 5a). These actions are described by Use Case diagram precedents "choose one detail" and "choose another detail" on Figure 4.

The packing of details can be done in interactive mode (fragment of the Use Case Diagram <<extends>>, and the precedent "interactive mode") or in automated mode. Both modes require checking of condition whether details are touching in concrete point (fragment of the Use Case Diagram <<include>> and the precedent "Check touching" (Figure 4)).

The note: to design density laying for the same kind of detail one detail is chosen twice.

2. The first hodograph point is defined. It matches with point of details intersection.

3. The next hodograph points are defined. Doing these one of two operations can be done. One operation is movement of movable detail face by stationary detail face (Figure 5a, 5b, 5e). Another one when face of movable detail is moved by vertex of stationary detail (Figure 5d). These actions are represented in precedents "move vertex face" and "move face face" of the Use Case diagram.

4. Defining whether current coordinates of movable figure pole match with the first hodograph point. This action matches to precedent "analyze next vertex" of the Use Case diagram (Figure 3). If coordinates match the VHGLF designing is finished else go to the point 3.

#### **Example of transformation Use Case diagram into Collaboration one**

Analyzing Use Case diagram (Figure 4) consider that all its constituents are transformed to Collaboration Diagram constituents definitely.

1. Formulate an analytical representation of the Use Case diagram.

Define elements of the set for subsystem  $\omega \in \Omega$ , namely  $A^\omega$ ,  $P^\omega$ ,  $P^\omega(\text{include})$ ,  $P^\omega(\text{extends})$ ,  $K^\omega$ ,  $Y^\omega$  and  $T^\omega$ .

$$A^\omega = \{a_0^w = \text{user}\},$$

$$P^\omega = \{p_0, \dots, p_7 \mid p_0 = \text{choose one det ail}, p_1 = \text{choose another det ail}, p_2 = \text{combination, of det ails}\} \\ p_3 = \text{interactive combination}, p_4 = \text{check touching}, p_5 = \text{move vertex face}, p_6 = \text{move face vertex} \\ p_7 = \text{analyze next vertex}\},$$

$$P^\omega(\text{include}) = \{p^w(\text{include})_0 = p_2(\text{include})p_3\},$$

$$P^\omega(\text{extends}) = \{p^w(\text{extends})_0 = p_2(\text{include})p_4\},$$

$$K^\omega = \{k^w_0, k^w_1, k^w_2\}.$$

Using (1) formulate an analytical representation of the Use Case diagram.

$$P^\omega \times K^\omega \times A^\omega \times P^\omega(\text{include}) \times P^\omega(\text{extends}) = \{ \langle a_0 p_0 \rangle, \langle a_0 p_1 \rangle, {}^{p_0} \langle p_0 p_2 \rangle, {}^{p_1} \langle p_1 p_2 \rangle, \langle p_2(\text{include}) p_3 \rangle, \langle p_2(\text{extends}) p_3 \rangle, \langle p_2 \bullet (p_5 \bullet p_6) \rangle, {}^{p_5} \langle p_5 p_7 \rangle, {}^{p_5} \langle p_7 p_5 \rangle, {}^{p_6} \langle p_6 p_7 \rangle, {}^{p_6} \langle p_7 p_5 \rangle, {}^{p_7} \langle p_7 p_5 \rangle \}$$

2. Combining p2 and p3 of the method of transforming Use Case diagram into Collaboration one form the table of Collaboration diagram fragments and rules of Collaboration diagram messages placement (Table. 3).

**Table 3.** Representation of Collaboration Diagram fragments and rules of message placement

Analytical representation of the Use Case Diagram fragments	Collaboration Diagram fragments and rules of messages placement
1	2
$\langle a_0 p_0 \rangle$ $\langle a_0 p_1 \rangle$	
${}^{p_0} \langle p_0 p_2 \rangle,$ ${}^{p_1} \langle p_1 p_2 \rangle,$ ${}^{p_5} \langle p_5 p_7 \rangle,$ ${}^{p_5} \langle p_7 p_5 \rangle$ ${}^{p_6} \langle p_6 p_7 \rangle,$ ${}^{p_6} \langle p_7 p_5 \rangle,$ ${}^{p_7} \langle p_7 p_5 \rangle$	$H_1 = \{p_0, P_1, p_2\}, P_1 = \emptyset$ $H_2 = \{p_1, P_2, p_2\}, P_2 = \emptyset$ $H_3 = \{p_5, P_3, p_7\}, P_3 = \emptyset$ $H_4 = \{p_7, P_4, p_5\}, P_4 = \emptyset$ $H_5 = \{p_6, P_5, p_7\}, P_5 = \emptyset$ $H_6 = \{p_7, P_6, p_5\}, P_6 = \emptyset$ $H_7 = \{p_7, P_7, p_5\}, P_7 = \emptyset$
$\langle p_2(\text{include}) p_3 \rangle$	$H_8 = \{p_2 \rightarrow p_3\}$
$\langle p_2(\text{extends}) p_3 \rangle$	$\tau_1 = \text{false}, \tau_2 = \text{false}, \tau_3 = \text{false}, \tau_4 = \text{true} \Rightarrow H_8 = \{\tau_4 p_3\}$ <i>interactive combination = true</i>
$\langle p_2 \bullet (p_5 \bullet p_6) \rangle$	$\left\{ \begin{array}{l} H_9 = \tau_5 \{p_2 \rightarrow P^1 \rightarrow p_1\}, P^1 = \emptyset, \tau_5 = \text{offset of moving det ail} \\ H_{10} = \tau_6 \{p_3 \rightarrow P^2 \rightarrow p_1\}, P^2 = \emptyset, \tau_6 = \text{offset of stay det ail} \end{array} \right.$

The resulting Collaboration Diagram after executing points 4 and 5 of the proposed method is represented on Figure 6.

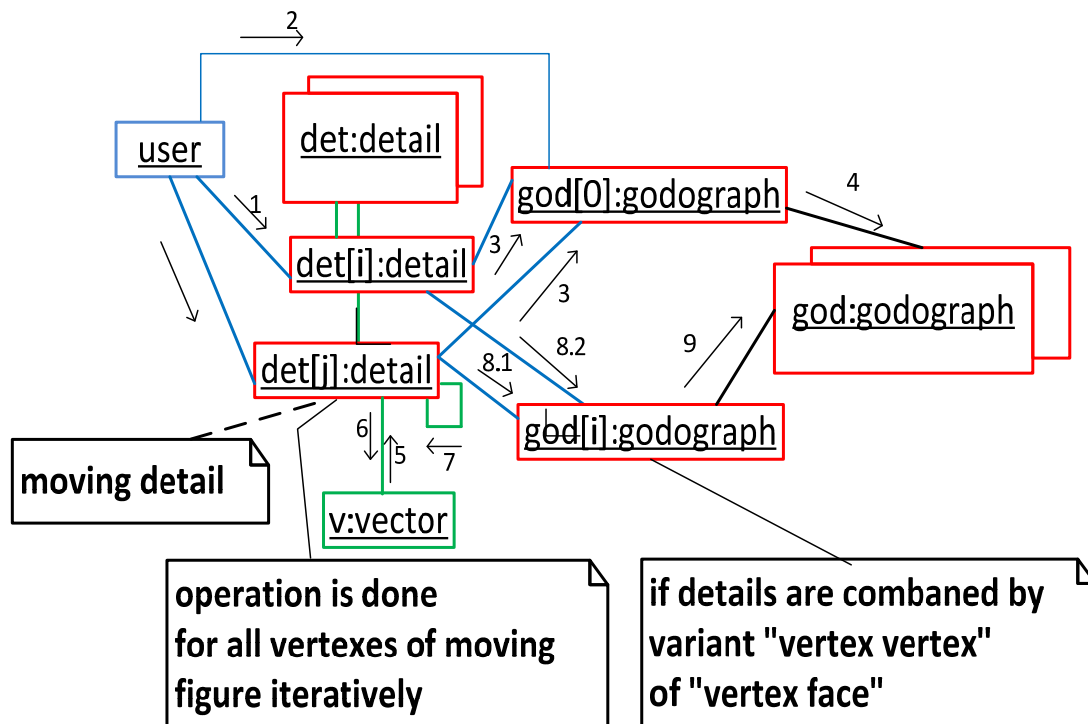


Figure 6. Collaboration Diagram of process designing of VHGLF

Represent an explanation of Collaboration Diagram messages

1. Message 1. Choose two details from an array of model's details.
2. Message 2. Define the first point of VHGLF in interactive mode.
3. Message 3. Define the first point of VHGLF in automated mode.
4. Message 4. Add the first point to the array of hodograph points.
5. Message 5, 6, 7. Define an offset vector analyzing positional relationship of the details (Figure 5).
6. Message 8.1, 8.2. Moving detail1 (detail2) define the next point of VHGLF by means of calculation coordinates of intersection point.
7. Message 9. Add a point to the array of hodograph points.

Objects and messages that are defined from the Use Case Diagram are marked by blue color.

Objects and messages that are defined from the ontology knowledge are marked by green color.

Objects and messages that are defined from the ontology vocabulary are marked by red color.

## Conclusion

---

The method of synchronization software artifacts that are changed after communication with customer is represented in this paper. Input behavioral software models are represented as Use Case diagram or its varieties and resulting model is Collaboration Diagram.

Involving this method into software development process let's to achieve the next advantages in MDD approach:

- Obtain behavioral software models that correspond to requirement specification;
- Allow further synchronization of software artifacts that are changed after Collaboration Diagram modifying [Chebanyuk, 2014].

An analytical apparatus for the representation of Use Case diagrams allows:

- Design tools for management software artifacts history;
- Generate new notation and formats for saving information about Use Case diagram;
- Design or modify techniques, tools and methods for merging, comparing, reusing, converting or changing software models.

Rules of Collaboration Diagram Messages placement can be used for:

- Checking whether behavioral software models meet MDE requirements, namely completeness of future software processes representation, validity and being non contradictory;
- Refinement other software artifacts, that describe software behavior;
- Save information about problem domain processes and interconnections between objects.

Using of the proposed approach for maintaining requirements specification to capture scenario requirements [Tombe, 2014] lets to improve tools for forming of an analytical models and designing of UML diagrams.

The review of environments of requirement engineering tools, presented in paper [Daud, 2014] shows that both the commercial and research RE tools support just a part of the requirement activities. Method of synchronization software models helps both designing new and improving existing methods for behavioral software models transformation and focus on a partial solution for a particular requirements management activity.

Involving analytical representation of behavioral software diagrams into requirement tracing activities one can simplify model conversion operations improve model checking process, requirement validation and verification operations [Goknul, 2014].

---

## Further exploration

---

Design an analytical apparatus and a framework for matching software requirements to architecture constituents. This framework will allow modifying architecture constituents after changing of requirements in automated mode. In order to achieve this goal it is necessary to do the following:

- Design an analytical apparatus for representation both static and dynamic models;
- Propose, check and verify mechanism for defining fully and partially matching elements of different software models types.



---

**Bibliography**

---

- [Chebanyuk, 2013] E. Chebanyuk Metalanguage for description problem domain processes. Міжнародна конференція "Сучасна інформатика. Проблеми, досягнення та перспективи розвитку", 11-13 вересня 2013 р. Київ. Ін-т програмних систем С. 63-64.
- [Chebanyuk, 2014] E. Chebanyuk. An Approach to Class Diagram Design. Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development. Pages 448-453.
- [Daud, 2014] N. Daud, M. Kamalrudin, S. Sidek, S. S. S. Ahmad. A Review of Requirements Engineering Tools for Requirements Validation Software Engineering Process Vol 1, No 1 (2014)
- [Diskin, 2014] Z. Diskin, A. Wider, H. Gholizadeh, K. Czarnecki. A Taxonomic Space for Increasingly Symmetric Model Synchronization. Generative Software Development Laboratory University of Waterloo University Avenue West, Waterloo, Ontario, Canada N2L 3G1, February 2014.
- [Goknul, 2014] A. Goknil, I. Kurtev, and K. Van Den Berg. "Generation and validation of traces between requirements and architecture based on formal trace semantics." Journal of Systems and Software 88 (2014): Pages 112-137.
- [Gupta, 2012] S. Gupta, J. Singla. A component-based approach for test case generation. International Journal of Information Technology 5.2, Pages 239-243, 2012.
- [Marín, 2013] B. Marín, J. Pereira, I G. Giachetti, F. Hermosilla, E. Serral. A General Framework for the Development of MDD Projects. Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development. Pages 257-260
- [Tombe, 2014] R. Tombe, Dr. S. Kimani, Dr. G. Okeyo. Method for software maintenance risk assessment at architecture level. Journal of international academic research for multidisciplinary ISSN: 2320-5083, Volume 2, Issue 1, February 2014.

---

**Authors' Information**

---



**Elena Chebanyuk** – lecturer in National aviation university, associate professor of software engineering department, Ukraine; e-mail: [chebanyuk.elena@gmail.com](mailto:chebanyuk.elena@gmail.com)

*Major Fields of Scientific Research: Model-Driven Architecture, Domain engineering, Code reuse.*