DOMAIN ENGINEERING APPROACH OF SOFTWARE REQUIREMENT ANALYSIS

O.V. Chebanyuk, O.V. Palahin, K.K. Markov

Requirement analysis is one of the important processes in software development lifecycle management. In Agile approach requirements software models are the basic of generating other software development artifacts. Improving requirements approaches and techniques allows avoiding mistakes in other software development artifacts. Domain engineering fundamentals is the basic for “template oriented” approaches of software development artifacts designing. Reusing domain models and knowledge allows adding details in vertical “model to model” transformation operations, refine generated software development artifacts, organize systematic software reuse and perform many other activities. Paper proposes an approach of requirement analysis based on UML Use Case diagrams transformations into communication ones and the next refinements of them by means of information from domain models. The advantages of the proposed approach is the next: proposed transformation method involves “many to many” transformation in order to save the semantic of initial model. Domain knowledge are used to complete communication diagram by means of adding details after transformation to them. In order to perform Use case to communication transformation graph representation of software models is chosen.

Key words: Domain Engineering, Domain Analysis, Requirement Analysis, Software Model Transformation, UML diagram.

Introduction

In practice, domain engineering finds practical implementation in Software Product Line approach. There are software engineering standards with recommendations to organize lifecycle processes in AGILE approach (ISO 12207, ISO 15288, ISO 19770-1, ISO 29119-2, ISO 20000-4). General recommendations of software development lifecycle process organization are complicated by specific operations aimed to organize an effective reuse of different software development artifacts. As software models are central development artifacts in AGILE approach operations of their reuse will allow to avoid designing and other mistakes. In order to organize effective software artifacts reuse scheme it is necessary to answer on two research questions (RQ):

(RQ1) What should be reused? Other words: how to select proper domain knowledge for reuse?
(RQ2) How to merge domain model with software development artifacts?
Effective solving of these questions propose performing the next activities:
- forming request of searching in domain area through domain artifacts in repository;
- organizing search procedure and defining matching criterion;
- merging domain knowledge with software development artifacts.

Related papers and practical research

Involving Domain engineering into software artifacts reuse started in the end of the previous century. Reuse researches performed in two directions. Research laboratories of big companies accumulated practical achievements in this area. Scientific research directed to development of an analytical approaches.

As a result of research laboratories practices analysis shows that the next factors slowed the process of software artifacts reuse:

Successful search of software development artifacts in Motorola practices was limited because it was some difference rules using meta-information while preparing information about software artifacts and its further reuse during search.

IBM focused on architectural solutions reuse. As a procedure of architectural solutions adoption for future projects is quite complicated, architectural solutions may contain errors or rigid design characteristics.

Hewlett-Packard developer teams make some free procedure of adoption software development life cycle process including extra processes for preparing high – quality software development artifacts ready for the further reuse.

Table 1 Summarizing results of research laboratories IBM, Motorola and Hewlett-Packard companies.

<table>
<thead>
<tr>
<th>Application engineering requirements for effective reuse of software development artifacts</th>
<th>Motorola</th>
<th>IBM</th>
<th>Hewlett Packard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal apparatus of software artifacts reuse</td>
<td>+</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>Formal apparatus of software artifacts semantic similarity</td>
<td>–</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>Providing maturity level of software development lifecycle processes</td>
<td>–</td>
<td>–</td>
<td>+</td>
</tr>
</tbody>
</table>

Analysis of scientific papers devoted to domain engineering development pointed that there is a list of factors that slow the development of software artifacts reuse in domain engineering:

1. Absence of the common concept and complex approach of software artifacts reuse information that is based on gathering information while domain analysis and its further reuse in application engineering processes [1–4].

2. Existing approaches of software artifacts reuse estimation do not contain formal apparatus of choosing the best software development artifact from the possible ones. [5–8].

3. Complex of tasks needed to be solved for software artifact reuse usually performed by means of different software development tools that use different formats of data representation. Inaccurate data transition between formats can be a cause of their partially lost or appearing some not expected elements [9–12].

4. Absence of formal methods allowing synchronizing domain models structure when initial information about domain analysis is changed (text, audio, video, web-site etc.) [13–16].

5. Difficulty to collaborate results of software models processing in text and graphical representation [17–19].

6. Absence of formal approaches of preparation and reuse meta-information about software development artifacts [20–22].

Proposed approach

Proposed approach is grounded on collaboration of knowledge about problem domain that were accumulated in domain analysis procedure [23] and improvement of requirement analysis procedures. The aim of improvement requirement analysis procedure is to spread information about Use Case Diagram and design communication diagram that satisfy the requirements and store the semantics of requirement specification.

From domain analysis artifacts, controlled vocabulary is used. Requirement analysis of artifacts consists of requirement specification and Use Case Diagram.

Proposed approach is based on performing transformation from Use Case to Communication Diagram, transforming whole structure of Use Case. Graph representation of UML diagram is chosen. Initial information for transformation is prepared composing all graph paths from textual representation (XMI) of UML diagram. The concept of “text to model” transformation is proposed in paper [24].
Data flow of the proposed approach is represented in the figure 1.

Figure 1. Data flow of the proposed approach

In order to solve this task propose the **next denotations:**
Graph representation of Use Case Diagram, that consider data streams

\[
SM_{use-case} = \{ \text{path}_1, \text{path}_2, \ldots, \text{path}_n \}, n = |SM_{use-case} | \\
\text{path} = (\text{esg}_1, \text{esg}_2, \ldots, \text{esg}_n) \\
\text{esg} = (\text{ob}_1, \text{link}, \text{ob}_2) \\
\text{ob}_1, \text{ob}_2 \in \{ \text{p}, \text{a}, \text{c} \} \\
\text{link} \in \{ \text{[include]}, \text{[extends]}, \text{[inh]} \}
\]

where \( SM_{use-case} \) – denotation of whole Use Case Diagram,
path\_i – path in the Use Case Diagram representing one data stream (path in graph).

esg – elementary sub-graph, describing two directly linked objects \(ob_1\) and \(ob_2\) by means of link.

Objects (ob) in notation of Use Case Diagram can be the next type \(a\) – actors; \(p\) – precedents; \(c\) – comments.

Links in Use Case diagram can be the next types – \(l\) (include) – include, \(l\) (extends) – extends, \(l\) (inh) – inheritance.

\[
SM_{\text{com}} = \{path_1, path_2, ..., path_n\}, n \Rightarrow SM_{\text{com}}
\]

\[
path = (esg_1, esg_2, ..., esg_p),
\]

\[
esg = (ob_1, m, ob_2)
\]

\(ob_1, ob_2 \in \{a, c, ob\}\)

where \(SM_{\text{com}}\) – Communication Diagram,

\(ob_1, ob_2\) – Communication Diagram objects,

\(m\) – Communication Diagram message.

Denote transformation operation from Use Case Diagram to Communication one as: \(SM_{\text{use\_case}} \triangleright \triangleright SM_{\text{com}}\),

where \(\triangleright \triangleright\) is a set of transformations rules, which are applied when Use Case diagram is transformed into communication one.

A set of domain entities in controlled vocabulary (\(ConVoc\))

\[
ConVoc = \{c_1, c_2, ..., c_n\}, n \Rightarrow \{ConVoc\}.
\]

A set of Use Case diagram precedents is:

\[
P_{\text{use\_case}} = \{p_1, p_2, ..., p_k\}
\]

\(p = (w_1, w_2, ..., w_l), p \in P_{\text{use\_case}}\).

Let define the transformation rules using proposed denotations.

In order to perform transformation from Use case to Communication diagrams, Transformation rules represented in the paper [25] are used. Grounding on these rules, it is proposed rule for transforming whole Use Case diagram into communication one.

Rules for obtaining skeleton of communication diagram

\[
PATH_{\text{use\_case}} \triangleright \triangleright PATH_{\text{com}}
\]

\[
path_{\text{use\_case}} \triangleright \triangleright path_{\text{com}}
\]

\[
esg_{\text{use\_case}} \triangleright \triangleright esg_{\text{com}}
\]

\(TRANS = \{trans_1, trans_2\}\)

\(trans_1 : (a, l, p) \rightarrow (a, m, obj)\)

\(trans_2 : (p_1, l, p_2) \rightarrow (obj, m, obj)\),

where \(path_{\text{use\_case}}\) – path in Use Case Diagram,

\(path_{\text{com}}\) – path in Communication Diagram,

\(esg_{\text{use\_case}}\) – elementary sub-graph in Use Case Diagram,

\(esg_{\text{com}}\) – elementary sub-graph in Communication Diagram,

\(obj\) – Communication Diagram object.

After performing such a transformation, the next task is to give a name for obtained objects. Denote named objects as \(obj(name)\). The rule of naming object is written in the following way:

\[
ConVoc \cap p = \{w = c \mid w \in p, c \in Convoc\}
\]

\[
ConVoc \cap p \neq \emptyset \rightarrow obj(name) = ConVoc \cap p .
\]
The last transformation task is to optimize Communication Diagram structure by means of applying “self-message” rule. Self-message is the message that is outcomes and incomes to the same communication diagram object.

\[
if \ obj_i = obj_j in (obj_i, m, obj_j) \\
then (obj_i, m, obj_j) \rightarrow (obj, m(self), obj)
\]

Graphically such communication diagram fragment (figure 2,a) is changed to the next (figure 2,b).

![Diagram](image)

Figure 2. “Self-message” optimization rule: a – obtained communication diagram fragment; b – optimized communication diagram fragment

Describe the steps of the proposed approach of communication diagram designing that based on Use Case diagram (application engineering artifact) and controlled vocabulary (domain analysis artifact).

2. Design Use case diagrams from requirement specification.
3. Obtain a skeleton of communication diagram from the Use Case using proposed transformation rules

\[
SM_{use \_case}^{TRANS} \rightarrow SM_{com}
\]

4. Fill communication diagram skeleton by means of objects names using.
5. Entities from controlled vocabulary in Use Case diagram using (1).

Case study

Consider example of Use case diagram for visualizing data of accounting reports. Report settings are stored in profiles. Reports visualized in using graphics. Graphics are obtained considering time settings. Use Case Diagram is represented in the figure 3.

![Diagram](image)

Figure 3. Use case diagram of visualizing accounting reports

Analytical representation of this diagram is prepared using approach represented in [24]. A set of Path is containing from six elements. Some part of paths are duplicated. Analytical representation of Use case diagram contains the initial information for designing of communication diagram structure.
Expression (2) represents example of transformation Use case diagram path into communication one.

\[
\begin{align*}
\text{path}_{(\text{case})} &= (a_1, l_1, p_1), (p_1, l_2, p_3), (p_3, l(\text{include}), p_2) \\
\text{path}_{(\text{com})} &= (a_2, m_1, \text{obj}_1), (\text{obj}_1, m_2, \text{obj}_2), (\text{obj}_2, m_3, \text{obj}_3) . \\
\text{obj}_j &= \text{profile}, \text{obj}_2 = \text{profile}, \text{obj}_3 = \text{"to define"} \\
\end{align*}
\]

The note according to transformation rule names of different objects can be the same. It is pointed to the fact that the diagram needs the further optimization. Name of object “to define” points, that in order to define the name of communication diagram object the information from domain knowledge is used. After designing all paths of communication diagram, its skeleton is composed (figure 4).

Figure 4. Unoptimized “skeleton” of the Communication Diagram

After performing sequence of Communication Diagram refinement (implementing self-object messaging rule) obtain diagram that is represented in figure 5 and 6.
Conclusion

Known “model to model” transformation approaches do not use the whole structure of initial diagram. It may be cause of losing some information or performing additional efforts of domain analytics to organize the structure of resulting diagram. From the other hand, such approaches require additional time and efforts.

Proposed approach aimed to designing of Communication Diagram from Use Case one. It is grounded on usage of whole Use Case diagram structure while transformation operation is performed. Such a fact allows saving Use Case semantics after transformation. As proposed approach implements vertical transformation, resulting diagram complicated by information about problem domain from domain knowledge.

Further research

It is planned to design formal approach allowing reuse domain knowledge while designing different types of UML diagrams in Software Product Line.

References

Information about the authors:

Chebanyuk Olena Viktorivna,
PhD, associate professor of software engineering department,
PhD, associate professor.
Number of publications – approximately 75.
Publications in Ukrainian journals – 35.
Publications in foreign journals – 35.
PP Hirsh index=4, Scopus – 1.
https://orcid.org/0000-0002-9873-6010 (ORCID name Elena Chebanyuk),

Palahin Olexander Vasyliovych,
Doctor of Sciences, Academician of National Academy of Sciences of Ukraine,
Deputy director of Glushkov Institute of Cybernetics, head of department 205.
Publications in Ukrainian journals – 290.
Publications in foreign journals – 45.
http://orcid.org/0000-0003-3223-1391,

Markov Krassimir K.,
Professor Dr.
Number of publications: more than 135; 5 monographs.
PP Hirsh index – 11.
https://orcid.org/0000-0001-5041-1498 (ORCID name Krassimir Markov)

Authors’ place of work:

National Aviation University,
03058 ave. Lубомира Гузара 1,
Phone: 044-406-76-41,
E-mail: chebanyuk.elena@gmail.com (chebanyuk.elena@ithea.org)

V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine
40, Academician Glushkov Avenue, Kyiv, 03187, Ukraine

Institute of Information Theories and Applications, Sofia, 1000, P.O. Box 775, Bulgaria.
E-mail: markov@ithea.org