A Methodology for Identifying and Refining Objects from the Software Requirements Based on Object-Based Formal Specification

Romi S. Wahono¹, Behrouz H. Far¹¹, Jingde Cheng⁷

This paper presents a methodology for object identification and refinement from the software requirements, which is based on object-based formal specification (OBFS). This methodology provides the mean of understanding the object-oriented paradigm easily, and supports us with identifying and refining the objects. As a case study, we have implemented a system for supporting the program committee chair of an international conference.

1. Introduction

The challenges of object-oriented analysis and design can be summarized as: identifying objects, its attributes and behaviors, defining associations between objects from the software requirements, and refining objects and organize classes by using inheritance [2]. The above mentioned object identification and refinement process is an ill-defined task [1] [5]. This is mainly due to the complexity of heuristics, and lack of unified methodology for this process.

We propose a methodology for identifying and refining objects from the software requirements based on object-based formal specification (OBFS) [4]. OBFS is a pure object-oriented requirement model, which is based on the object-oriented paradigm and its features. We described a system for supporting the program committee chair (PC) of an international conference (IntConfProgChair), based on this methodology.

2. A Methodology for Identifying and Refining Objects by Using OBFS

Requirement acquisition and specification are considered as one of the most important activities in software development. We propose an approach where end-users take an active role in the analysis by specifying requirements using OBFS. We use OBFS to guide end-users in describing their problem based on object-oriented paradigm. OBFS is composed of Description Statements (DS), Collaborative Statements (CS), Attributive Statements (AS), Behavioral Statements (BS), and Inheritance Statements (IS). OBFS use English natural language based on the constraint syntax rules.

2.1. Description Statements (DS)

DS is used to guide the writing of an overview of the system that one wants to build. DS is composed from four elements: Requirement ID, Requirement Name, Language, and Description. DS should specify what is to be done, but not how it is to be done. It should be a statement of needs, not a proposal for a solution.

An example of DS for IntConfProgChair system is as follows.

<table>
<thead>
<tr>
<th>ReqID</th>
<th>#001</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReqName</td>
<td>A System For Supporting The Program Committee Chair (PC) of an International Conference</td>
</tr>
<tr>
<td>Language</td>
<td>English</td>
</tr>
<tr>
<td>Description</td>
<td>This system supports the program committee chair of an international conference, not only in the management of the paper submission, but also in the creation of a letter for submitter, etc.</td>
</tr>
</tbody>
</table>

2.2. Collaborative Statements (CS)

CS is used to identify objects, and associations between the objects. CS consists of a set of forms and contains Subject-Verb-Object (S-V-O) as well as the English natural language based on CS syntax rules (E). We use $\text{S}_\alpha\cdot\text{V}_\alpha\cdot\text{O}_\alpha$ notation for describing S-V-O natural language, which is based on CS syntax rules. The collaboration between $\text{S}_\alpha$ and $\text{O}_\alpha$ must be described in the CS.

The CS syntax rules are listed as follows. Predicates are extracted from synonym data dictionary (thesaurus) [3].

\[
\text{(ActionSentence(AcS)) := } \text{S}_\alpha(\text{AcSPredicate})\text{O}_\alpha \\
\text{(AcSPredicate) := drive|work for|maintain|manage|own|execute|serve|use} \\
\text{(LocationSentence(LcS)) := } \text{S}_\alpha(\text{LcSPredicate})\text{O}_\alpha \\
\text{(LcSPredicate) := next|go} \\
\text{(CommunicationSentence(CmS)) := } \text{S}_\alpha(\text{CmSPredicate})\text{O}_\alpha \\
\text{(CmSPredicate) := talk to|communicate with|refer to} \\
\]

An example of CS for IntConfProgChair system is as follows.

PC Create Schedule, CFP, ProgramCommitteeList. 
PC Distribute CFP. 
PC Arrange KeynoteSpeakers, ProgramCommittee, Moderator. 
Submitter Send Paper, CoverPaper. 
PC get Paper. PC give PaperNumber. PC SendConfirmationEmailToSubmitter. PC SendPaperToProgramCommittee. 
ProgramCommittee Select Paper, ProgramCommittee Review Paper. 
PC Create ReviewerQualification. PC SendResultToSubmitter. 
PC Create CFPs. PC Distribute CFP.

The objects and its associations can be identified by using the following formulas.

\[
\forall CS \in E \ [ \text{S}_\alpha \Rightarrow \text{OBJ} ] \text{ and } \forall CS \in E \ [ \text{O}_\alpha \Rightarrow \text{OBJ} ] \Lambda (1) \\
\forall CS \in E \ [ \text{V}_\alpha \Rightarrow \text{ASS} ] \Lambda (2) \\
(\neg(\text{OBJ}.)_{\text{nat}} \Rightarrow \text{OBJ} ) \text{ and } (\neg(\text{ASS}.)_{\text{nat}} \Rightarrow \text{ASS} ) \Lambda (3) \\
X=\text{tentative } X, Y=\text{redundant } Y, \text{OBJ=object, ASS=association} 
\]

¹ Dept. of Information and Computer Sciences, Saitama University
¹¹ Dept. of Electrical and Computer Engineering, University of Calgary
2.3. Attributive Statements (AS)

AS are used to identify the attributes of objects. Attributes are properties of individual objects. Attributes usually correspond to nouns followed by possessive phrases, and sometimes are characterized by adjectives or adverbs. AS must contain properties of each object identified at the previous step. AS consists of a set of forms and contains $S_w$, $V_w$, $O_w$ as well as the English natural language based on AS syntax rules (E).

The AS syntax rules are listed as follows.

\[
\begin{align*}
(OwSentence(OwS)) & \iff S_w (\text{OwSPredicate}) O_w \\
(OwSPredicate) & \iff \text{has (properties)} \text{consists of contain of}
\end{align*}
\]

An example of AS for IntConfProgChair system is as follows.

Schedule consist of DeadlinePaper, DeadlineCameraReady, ProgramCommitteeDate, ReviewResultNotification, ProceedingsCreationDate. ProgramCommitteeList consist of Name, Address, Affiliation, Phone, Fax, Email, ResearchField. CoverPaper consist of Title, Authors, ContactPerson, Address, Affiliation, Phone, Fax, Email, Abstract, Keywords. ReviewerQualification consist of NotMyPaper, NotSameOrganization, NotSameCountry, NotKnowOtherReviewerPaper.

The object attributes can be identified by using the following formulas.

\[
\forall AS \in E \left[ O_w \Rightarrow \text{ATT} \right] \text{ and } \forall AS \in E \left[ S_w \Rightarrow \text{OBJ} \right] (4)
\]

\[
\neg \text{ATT}_{\text{att}} \Rightarrow \text{ATT}_{\text{att}} (5)
\]

\[
X_{\text{att}} = \text{tentative} \ Y_{\text{att}} = \text{redundant} \ Y, \text{OBJ=object, ATT=attribute}
\]

2.4. Behavioral Statements (BS)

BS is used to identify object behaviors. Behavior is how an object acts and reacts, in terms of state changes and message passing. A behavioral statement must contain behaviors of each object identified at the previous step. BS consists of a set of forms and contains $S_w$, $V_w$, $O_w$ as well as the English natural language based on BS syntax rules (E).

The BS syntax rules are listed as follows.

\[
\begin{align*}
(\text{CapabilitySentence(CpS)}) & \iff S_w (\text{CpSPredicate}) O_w \\
S_w (\text{CpSMinusPredicate}) & \iff \text{has (a capability to)} \text{has (a capacity for)} \text{can (capabilities) able to (capabilities)} \\
(\text{CpSMinusPredicate}) & \iff \text{not has (a capability to)} \text{not has (a capacity for) not able (capabilities)}
\end{align*}
\]

An example of BS for IntConfProgChair system is as follows.

The object behaviors can be identified by using the following formulas.

\[
\forall BS \in E \left[ O_w \Rightarrow \text{BEH} \right] \text{ and } \forall BS \in E \left[ S_w \Rightarrow \text{OBJ} \right] (6)
\]

\[
\neg \text{BEH}_{\text{att}} \Rightarrow \text{BEH}_{\text{att}} (7)
\]

\[
X_{\text{att}} = \text{tentative} \ Y_{\text{att}} = \text{redundant} \ Y, \text{OBJ=object, BEH=behavior}
\]

2.5. Inheritance Statements (IS)

IS is used to organize classes by using inheritance, to share common object attributes and behaviors. IS provide sentences that describe is-a-kind-of relationship. IS consists of a set of forms and contains $S_w$, $V_w$, $O_w$ as well as the English natural language based on IS syntax rules (E).

The IS syntax rules are listed as follows.

\[
\begin{align*}
(\text{InheritanceSentenceA(IhSA)}) & \iff S_w (\text{IhSPredicate}) O_w \\
(\text{IhSPredicate}) & \iff \text{is a kind of specialization of}
\end{align*}
\]

An example of IS for IntConfProgChair system is as follows.

Reviewers is a kind of ProgramCommittee. Moderator is a kind of ProgramCommittee.

The object and its class hierarchy organization can be refined by using the following formulas.

\[
\text{IhSA} \iff \forall IS \in E \left[ O_w \Rightarrow SCL \right] \text{ and } \forall IS \in E \left[ S_w \Rightarrow OBJ \right] (8)
\]

\[
\text{IhSB} \iff \forall IS \in E \left[ S_w \Rightarrow SCL \right] \text{ and } \forall IS \in E \left[ O_w \Rightarrow OBJ \right] (9)
\]

\[
X_{\text{att}} = \text{tentative} \ Y_{\text{att}} = \text{redundant} \ Y, \text{OBJ=object, SCL=superclass}
\]

3. Concluding Remarks

It is argued that object identification and refinement are an ill-defined task. Although there are many projects focusing on the requirement engineering models for object-oriented analysis and design, there are only a few focusing on the formalization of object-oriented features and the methodology for identifying and refining objects. We presented OBFS and its roles to be a methodological support for the object identification and refinement.

References