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

Romi S. Wahono[†], Behrouz H. Far[††], Jingde Cheng[†]

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.

| | |
|---|---|
| **ReqID** | #001 |
| **ReqName** | A System for Supporting the Program Committee Chair (PC) of an International Conference |
| **Language** | English |
| **Description** | 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.. |

### 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 $S_{cs}$-$V_{cs}$-$O_{cs}$ notation for describing S-V-O natural language, which is based on *CS syntax rules*. The collaboration between $S_{cs}$ and $O_{cs}$ must be described in the *CS*.

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

$$\langle \textbf{ActionSentence(AcS)} \rangle ::= S_{cs} \langle AcSPredicate \rangle O_{cs}$$
$$\langle \textbf{AcSPredicate} \rangle ::= drive | work\ for | maintain | manage | own | execute | serve | use$$
$$\langle \textbf{LocationSentence(LcS)} \rangle ::= S_{cs} \langle LcSPredicate \rangle O_{cs}$$
$$\langle \textbf{LcSPredicate} \rangle ::= next\ to | goto$$
$$\langle \textbf{CommunicationSentence(CmS)} \rangle ::= S_{cs} \langle CmSPredicate \rangle O_{cs}$$
$$\langle \textbf{CmSPredicate} \rangle ::= talk\ to | communicate\ with | refer\ to$$

An example of CS for *IntConfProgChair* system is as follows.

| |
|---|
| PC *Create* Schedule, CFP, ProgramCommitteeList.<br>PC *Distribute* CFP.<br>PC *Arrange* KeynoteSpeakers, ProgramCommittee, Moderator.<br>Submitter *Send* Paper, CoverPaper.<br>PC *get* Paper. PC *give* PaperNumber. PC *SendConfirmationEmailTo* Submitter.<br>PC *SendPaperTo* ProgramCommittee.<br>ProgramCommitee *Select* Paper. ProgramCommittee *Review* Paper.<br>PC *Create* ReviewerQualification. PC *SendResultTo* Submitter.<br>PC *Create* CFPt. PC *Distribute* CFPt. |

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

$$\forall CS \in E\ [S_{cs} \Rightarrow OBJ_t]\ and\ \forall CS \in E\ [O_{cs} \Rightarrow OBJ_t] \wedge \ (1)$$
$$\forall CS \in E\ [V_{cs} \Rightarrow ASS_t] \wedge \ (2)$$
$$\neg (OBJ_t)_{red} \Rightarrow OBJ\ and\ \neg (ASS_t)_{red} \Rightarrow ASS \wedge \ (3)$$

*$X_t$=tentative X, $Y_{red}$=redundant Y, OBJ=object, ASS=association*

[†] Dept. of Information and Computer Sciences, Saitama University
[††] 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 $S_{as}$-$V_{as}$-$O_{as}$ as well as the English natural language based on *AS syntax rules* (*E*).

The *AS syntax rules* are listed as follows.

$$\langle \textbf{\textit{OwnershipSentence(OwS)}} \rangle ::= S_{as} \langle \textit{OwSPredicate} \rangle O_{as}$$
$$\langle \textbf{\textit{OwSPredicate}} \rangle ::= \textit{has (properties)|consits of|contain of}$$

An example of *AS* for *IntConfProgChair* system is as follows.

| |
|---|
| Schedule *consist of* DeadlinePaper, DeadlineCameraReady, ProgramCommitteDate, ReviewResultNotification, ProceedingsCreationDate.<br>ProgramCommitteList *consist of* Name, Address, Affiliation, Phone, Fax, Email, ResearchField.<br>CoverPaper *consist of* Title, Authors, ContactPerson, Address, Affiliation, Phone, Fax, Email, Abstract, Keywords.<br>ReviewerQualification *consits of* NotMyPaper, NotSameOrganization, NotSameCountry, NotKnowOtherReviewerPaper. |

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

$$\forall AS \in E\ [O_{as} \Rightarrow ATT_t] \textit{ and } \forall AS \in E\ [S_{as} \Rightarrow OBJ] \Lambda \ (4)$$
$$\neg (ATT_t)_{\text{red}} \Rightarrow ATT \Lambda \ (5)$$

$X_t$=*tentative X*, $Y_{red}$=*redundant Y, 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 $S_{bs}$-$V_{bs}$-$O_{bs}$ as well as the English natural language based on *BS syntax rules* (*E*).

The *BS syntax rules* are listed as follows.

$$\langle \textbf{\textit{CapabilitySentence(CpS)}} \rangle ::= S_{bs} \langle \textit{CpSPredicate} \rangle O_{bs} \mid S_{bs} \langle \textit{CpSMinusPredicate} \rangle O_{bs}$$
$$\langle \textbf{\textit{CpSPredicate}} \rangle ::= \textit{has (a capability to)|has (a capacity for)| can (capabilities)|able to (capabilities)}$$
$$\langle \textbf{\textit{CpSMinusPredicate}} \rangle ::= \textit{has not (a capability to)|has not (a capacity for)|can not (capabilities)|not able to (capabilities)}$$

An example of *BS* for *IntConfProgChair* system is as follows.

| |
|---|
| PC *has a capability* to CreateSchedule, CreateCFP, DistributeCFP, CreateCFPt, DistributeCFPt, CreateProgramCommitteList, ArrangeKeynoteSpeakers, ArrangeProgramCommittee, ArrangeModerator, GetPaper, GivePaperNumber, SendConfirmationEmailToSubmitter, SendPaperToProgramCommitte, CreatesReviewerQualification, SendResultToSubmitter.<br>ProgramCommitee *has a capability* to SelectPaper, ReviewPaper |

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

$$\forall BS \in E\ [O_{bs} \Rightarrow BEH_t] \textit{ and } \forall BS \in E\ [S_{bs} \Rightarrow OBJ] \Lambda \ (6)$$
$$\neg (BEH_t)_{\text{red}} \Rightarrow BEH \Lambda \ (7)$$

$X_t$=*tentative X*, $Y_{red}$=*redundant Y, 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 $S_{is}$-$V_{is}$-$O_{is}$ as well as the English natural language based on *IS syntax rules* (*E*).

The *IS syntax rules* are listed as follows.

$$\langle \textbf{\textit{InheritanceSentenceA(IhSA)}} \rangle ::= S_{is} \langle \textit{IhSAPredicate} \rangle O_{is}$$
$$\langle \textbf{\textit{IhSAPredicate}} \rangle ::= \textit{is a kind of|is spesialization of}$$
$$\langle \textbf{\textit{InheritanceSentence B(IhS B)}} \rangle ::= O_{is} \langle \textit{IhSBPredicate} \rangle S_{is}$$
$$\langle \textbf{\textit{IhSBPredicate}} \rangle ::= \textit{is generalization of}$$

An example of *IS* for *IntConfProgChair* system is as follows.

| |
|---|
| Reviewer *is a kind of* ProgramCommitte.<br>Moderator *is a kind of* ProgramCommitte. |

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

$$\textbf{\textit{IhSA}}\ \ \forall IS \in E\ [O_{is} \Rightarrow SCL_t] \textit{ and } \forall IS \in E\ [S_{is} \Rightarrow OBJ] \Lambda \ (8)$$
$$\textbf{\textit{IhSB}}\ \ \forall IS \in E\ [S_{is} \Rightarrow SCL_t] \textit{ and } \forall IS \in E\ [O_{is} \Rightarrow OBJ] \Lambda \ (9)$$
$$\neg (SCL_t)_{\text{red}} \Rightarrow SCL \Lambda \ (10)$$

$X_t$=*tentative X*, $Y_{red}$=*redundant Y, 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


[1] Grady Booch, *Object-Oriented Analysis and Design with Application*, Benjamin/Cummings, 1991.

[2] Ian M. Holland and Karl J. Lieberherr, *Object-Oriented Design*, ACM Comp. Surveys, Vol. 28, No. 1, pp. 273-275, 1996.

[3] Robert L. Chapman, *Roget's International Thesaurus*, HarperCollins Publishers, 1992.

[4] Romi S. Wahono and Behrouz H. Far, *Towards The Use of Intelligent Agents in Collaborative Object-Oriented Analysis and Design*, International Session of JSAI'01, Japan, 2001.

[5] Ying Liang, Daune West, and Frank A. Stowell, *An Approach to Object Identification, Selection and Specification in Object-Oriented Analysis*, Information Systems Journal, Vol. 8, No. 2, 1998, pp. 163-180, Blackwell Science Ltd., 1998.