OOExpert: An Agent Based System for Identifying and Refining Objects from Software Requirements Based on Object Based Formal Specification

Romi Satria Wahono

Indonesian Institute of Science (LIPI)
Department of Information and Computer Sciences, Saitama University

Keywords: object model creation process, object identification, object-oriented analysis and design

Abstract. This paper examines the issues associated with the methodology for object identification and refinement, and also the use of multi-agent system approach for collaborative object-oriented analysis and design. We propose an agent based system called *OOExpert* for solving problems on object model creation process by identifying and refining objects from software requirements based on object based formal specification.

Introduction

There are numerous object-oriented analysis and design methods being advocated at the present time, all fairly similar but with significant differences in approach and notation. However, the challenges of object-oriented analysis and design are, to identify the objects, attributes, associations between the identified objects, and refine objects and organize classes by using inheritance to share common structure [6]. Researchers and software designers have come to a conclusion that object identification and refinement process are an ill-defined task [5] [21] [22] [23], because of the difficulty of heuristics and there is no unified methodology for object-oriented analysis and design.

Although there are many projects focusing on Computer Aided Software Engineering (CASE) tools for object-oriented analysis and design, there are only a few focusing on the formalization and implementation of the methodology for object model creation process. This paper presents a methodology for object identification and refinement, and also the use of agent-based approach for identifying and refining objects from software requirements based on object based formal specification [15] [16] [22].

Requirement Acquisition and Specification Based on Object Orientation

The primary goal of the requirements document is to be a reference for the software designers, facilitating improved software design through detection of incompleteness, inconsistency and ambiguity. Most of the faults found during testing and operation result from poor understanding or misinterpretation of requirements. Until now, there are only a few effective methods and tools to guarantee a complete, consistent, and unambiguous requirement model [17]. Recent advances in software technology such as the development of the Unified Modeling Language (UML) for object-oriented design have not reduced the need for better requirement acquisition and specification.

In the traditional approach to software analysis, system analyst interviews end-users to capture requirement. We propose an approach where end-users take an active role in the analysis by specifying requirements using Object-Based Formal Specification (OBFS). We use OBFS to guide end-users in describing their problem. OBFS is composed of Description Statements (DS), Collaborative Statements (CS), Attributive Statements (AS), Behavioral Statements (BS), and Inheritance Statements (IS). This approach also takes advantage of end-users' domain knowledge.

Object-Based Formal Specification

Description Statements (DS)

Description statements are used to guide for writing an overview of the system that we want to build. *Description statements* contain four kinds of elements: *Requirements ID, Requirements Name, Language, and Description*. The description statements should state what is to be done and not how it is to be done. It should be a statement of needs, not a proposal for a solution.

$$DS = \{reqID, reqName, Language, Description\} \ldots (2)$$

Collaborative Statements (CS)

Collaborative statements (CS) are used to identify objects, and associations between objects. The first step in the object model creation process is to identify relevant objects and their association from the application domain. Objects include physical entities and all objects must make sense in the application domain. All objects are explicit in the collaborative statements. Objects correspond to nouns that are identified from collaborative statements. *CS* consists of a set of forms with contains Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (E) natural language that is based on *CS syntax rules*.

$$CS = \{(S_1, V_1, O_1)_{cs}, (S_2, V_2, O_2)_{cs}, (S_3, V_3, O_3)_{cs}, \ldots\} \text{ and } \forall CS \in E \quad \ldots (3)$$

S_{cs} and O_{cs} will be identified as a tentative object (OBJ_t), and V_{cs} will be identified as a tentative association (ASS_t) in terms of object-oriented paradigm.

$$\forall CS \in E \ [S_{cs} \Rightarrow OBJ_t] \text{ and } \forall CS \in E \ [O_{cs} \Rightarrow OBJ_t] \quad (4)$$

$$\forall CS \in E \ [V_{cs} \Rightarrow ASS_t] \quad (5)$$

The *CS syntax rules* are listed as follows. Predicates are extracted from synonym data dictionary (thesaurus) [14].

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

Attributive Statements (AS)

Attributive statements (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. Attributive statement must contain properties of each object identified at the previous step. *AS* consists of a set of forms with contains Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (*E*) natural language that is based on *AS syntax rules*.

$$AS = \{(S_1, V_1, O_1)_{as}, (S_2, V_2, O_2)_{as}, (S_3, V_3, O_3)_{as}, \ldots\} \text{ and } \forall AS \in E \quad \ldots (6)$$

O_{as} will be identified as a tentative attribute (ATT_t) in the term of object-oriented paradigm. And S_{as} is identified and refined objects (*OBJ*) from tentative object (OBJ_t), as the final result of object identification's process.

$$\forall AS \in E\ [O_{as} \Rightarrow ATT_t]\ \cdots (7)$$

$$\forall AS \in E\ [S_{as} = OBJ]\ \cdots (8)$$

The *AS syntax rules* are listed as follows.

$\langle \textbf{OwnershipSentence(OwS)} \rangle ::= S_{as} \langle OwSPredicate \rangle O_{as}$

$\langle \textbf{OwSPredicate} \rangle ::=$ *has (properties)|consits of|contain of*

Behavioral Statements (BS)

Behavioral statements are 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 with contains Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (*E*) natural language that is based on *BS syntax rules*.

$$BS = \{(S_1, V_1, O_1)_{bs}, (S_2, V_2, O_2)_{bs}, (S_3, V_3, O_3)_{bs}, ...\} \text{ and } \forall BS \in E\ \ldots (9)$$

O_{as} will be identified as a tentative behavior (BEH_t) in the term of object-oriented paradigm. And S_{bs} is identified and refined objects (*OBJ*) from tentative object (OBJ_t), as the final result of object identification's process.

$$\forall BS \in E\ [O_{bs} \Rightarrow BEH_t]\ \cdots (10)$$

$$\forall BS \in E\ [S_{bs} = OBJ]\ \cdots (11)$$

The *BS syntax rules* are listed as follows.

$\langle \textbf{CapabilitySentence(CpS)} \rangle ::= S_{bs} \langle CpSPredicate \rangle O_{bs}$ |
$S_{bs} \langle CpSMinusPredicate \rangle O_{bs}$

$\langle \textbf{CpSPredicate} \rangle ::=$ *has (a capability to)|has (a capacity for)|*
can (capabilities)|able to (capabilities)

$\langle \textbf{CpSMinusPredicate} \rangle ::=$ *has not (a capability to)|has not*
(a capacity for)|can not (capabilities)|not able to (capabilities)

Inheritance Sentences (IS)

Inheritance statements are used to organize classes by using inheritance, to share common object attributes and behaviors. Inheritance provides a natural classification for kinds of objects and allows for the commonality of objects to be explicitly taken advantage of in modeling and constructing object systems. *Inheritance statements* provide sentences that describe *is-a-kind-of* relationship. *Inheritance statements* consists of a set of forms with contains Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (*E*) natural language that is based on *IS syntax rules*.

$$IS = \{(S_1, V_1, O_1)_{is}, (S_2, V_2, O_2)_{is}, (S_3, V_3, O_3)_{is}, ...\} \text{ and } \forall IS \in E\ \ldots (12)$$

O_{is} will be identified as a tentative superclass (SCL_t) in the term of object-oriented paradigm. And S_{is} is identified and refined objects (*OBJ*) from tentative object (OBJ_t), as the final result of object identification's process.

$$\forall IS \in E\ [O_{is} \Rightarrow SCL_t]\ \cdots (13)$$

$$\forall IS \in E\ [S_{is} = OBJ]\ \cdots (14)$$

The *IS syntax rules* are listed as follows.

$\langle \textbf{InheritanceSentenceA(IhSA)} \rangle ::= S_{is} \langle IhSAPredicate \rangle O_{is}$

$\langle \textbf{IhSAPredicate} \rangle ::=$ *is a kind of|is spesialization of*

$\langle \textbf{InheritanceSentence}\ B\textbf{(IhS}\ B\textbf{)} \rangle ::= O_{is} \langle IhSBPredicate \rangle S_{is}$

$\langle \textbf{IhSBPredicate} \rangle ::=$ *is generalization of*

Object Identification and Refinement Process

Figure 1 shows our strategy for the object identification process. We use *collaborative statements (CS)* from *OBFS* to guide end-users in describing their problem, especially for collaborative process in the system that end-users want to build. The first step in the object identification process is to extract *S* and *O* written in the *collaborative statements* to be tentative objects (OBJ_t) (4).

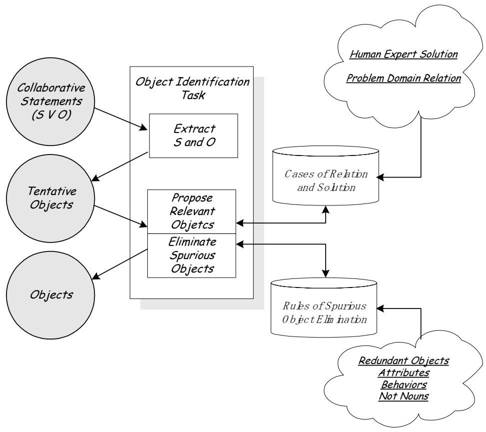

Figure 1: Object Ident ification Process

The next step is to eliminate spurious objects and propose relevant objects using Rule-Based Reasoning (RBR) and Case-Based Reasoning (CBR) paradigms. In RBR, the system will discard unnecessary and incorrect objects according to the following criteria: *redundant objects* (OBJ_{red}), *not noun objects* (OBJ_{non}), *attributes* (OBJ_{att}), *behaviors* (OBJ_{beh}), and *associations* (OBJ_{ass}).

$$\forall OBJ \in E\ [\neg OBJ_{red} \wedge \neg OBJ_{att} \wedge \neg OBJ_{beh} \wedge \neg OBJ_{non} \Rightarrow OBJ] \quad \dots (15)$$

Other identification and refinement processes are similar, although use different rule for their processes. The summary of object identification and refinement processes are shown in Figure 2.

Object Model Creation Process	Pre-Input (OBFS)	Extract (S V O)	Input	Rules for Reasoning					Output
				Rules for Elimination					
Object Identification	Collaborative Statements	S and O	Tentative Object	Redundant Object	Not Noun	Attribute	Behavior	Association	Object
Association Identification	Collaborative Statements	V	Tentative Association	Redundant Association	Not Verb	Behavior	Object	Attribute	Association
Attribute Identification	Attributive Statements	O	Tentative Attribute	Redundant Attribute	Not Noun	Object	Association	Behavior	Attribute
Behavior	Behavioral Statements	O	Tentative Behavior	Redundant Behavior	Not Verb	Association	Attribute	Object	Behavior
Object Refinement with Inheritance	Inheritance Statements	S and O	Object Hierarchy						Class Hierarchy
			Identified Object from Object Identification Process	Rules for Similarity Searching		Rules for Superclass Naming			Class Hierarchy
				Attribute	Behavior	Similar Object's Name	Given Name from User		

Figure 2. Summary of the Proposed Approach for Object Model Creation Process

OOExpert Design and Implementation

In our approach, object model creation process is viewed as a society of software agents that interact and negotiate with each other. We have devised six types of agents (*OOExpert Agents*):

requirement acquisition agent, object identification agent, attribute identification agent, association identification agent, behavior identification agent, and object refinement agent (Figure 3, 4, 5).

The responsibility of each agent is as follows. Firstly, the *requirements acquisition agent* manages the task concerning the requirements *acquisition from* object-based formal specification (OBFS). The *object identification agent* manages the task concerning the identification of objects. The *attribute identification agent* manages the task concerning the identification of object attributes. The *association identification agent* manages the task concerning the identification of associations between the identified objects. The *behavior identification agent* manages the task concerning the identification of object behaviors. And finally, the *object refinement agent* manages the task concerning to refine objects and organize classes by using inheritance to share common structure.

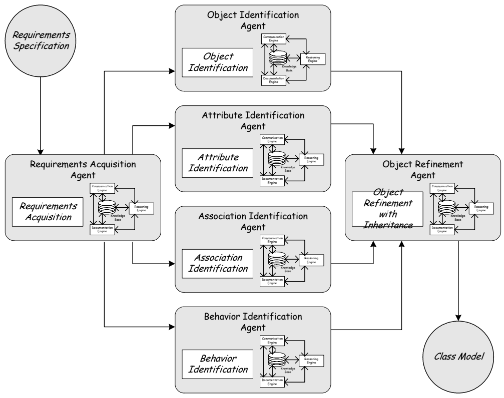

Figure 3: Intelligent Agent Architecturefor Object Model

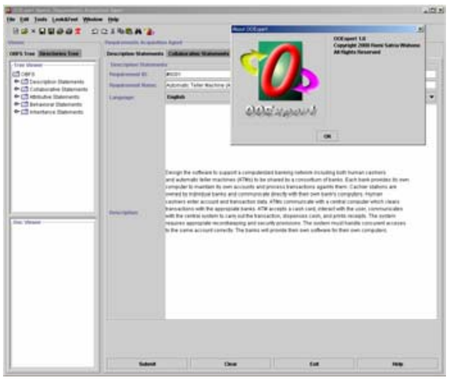

Figure 4: Requirements Acquisition Agent

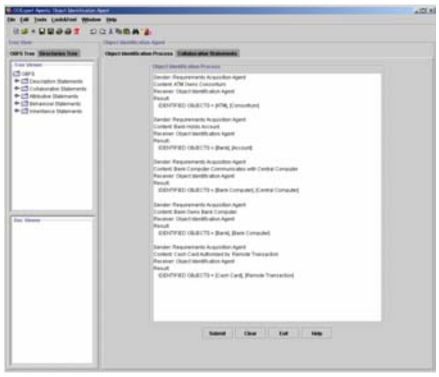

Figure 5: Object Identification Agent

Conclusions

This paper presented the methodology for object identification and refinement, and also the use of multi-agent system approach for collaborative object-oriented analysis and design. We propose an agent based system called *OOExpert* for solving problems on object model creation process by identifying and refining objects from software requirements based on object based formal specification.

References

[1] Celesta G. Ball and Rebecca L. Kim, An Object Oriented Analysis Of Air Traffic Control, WP-90W00542, the MITRE Corporation, McLean, Virginia, August 1991.

[2] F.P Brooks, No Silver Bullet, Essence and Accidents of Software Engineering, IEEE Computer, Vol. 20, No. 4, pp. 10-19, April 1987.
[3] Gerhard Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.
[4] Grady Booch, James Rumbaugh, and Ivar Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1999.
[5] Grady Booch, Object-Oriented Analysis and Design with Application, Benjamin/Cummings, 1991.
[6] Ian M. Holland and Karl J. Lieberherr, Object-Oriented Design, ACM Computing Surveys, Vol. 28, No. 1, pp. 273-275, March 1996.
[7] James F. Peters and Witold Pedrycz, Software Engineering An Engineering Approach, John Wiley & Sons, Inc., 2000.
[8] James Rumbaugh, Ivar Jacobson, and Grady Booch, The Unified Modeling Language Reference Manual, Addison-Wesley, 1999.
[9] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorenson, Object-Oriented Modeling and Design, Prentice Hall, 1991.
[10] J.M. Drake, W.W. Xie, and W.T. Tsai, Approach and Case Study of Requirement Analysis Where end-users Take an Active Role, in Proceedings of the 15th International Conference on Software Engineering, IEEE Computer Society Press, pp. 177-186, 1993.
[11] L.B. Becker, C.E. Pereira, O.P. Dias, I.M. Teixeira and J.P. Teixeira, MOSYS A Methodology for Automatic Object Identification from System Specification, Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, Newport Beach, California, March 15-17, 2000.
[12] Maritta Heisel and Jeanine Souquieres, Methodological Support for Requirements Elicitation and Formal Specification, Proceedings of the 9th International Workshop on Software Specification and Design, Ise-Shima (Isobe), Japan, April 16-18, 1998.
[13] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge, A Roadmap of Agent Research and Development, in Autonomous Agents and Multi-Agent Systems, pp. 7-38, Kluwer Academic Publishers, Boston, 1998.
[14] Robert L. Chapman, Roget's International Thesaurus, HarperCollins Publishers, 1992.
[15] Romi Satria Wahono and B.H. Far, Hybrid Reasoning Architecture for Solving Object Class Identification Problem in the OOExpert System, Proceedings of the 14th Annual Conference of Japanese Society for Artificial Intelligence, Tokyo, Japan, July 2000.
[16] Romi Satria Wahono and B.H. Far, OOExpert: Distributed Expert System for Automatic Object-Oriented Software Design, Proceedings of the 13th Annual Conference of Japanese Society for Artificial Intelligence, pp.456-457, Tokyo, Japan, June 1999.
[17] Ruqian Lu and Zhi Jin, Domain Modeling-Based Software Engineering, Kluwer Academic Publishers, 2000.
[18] Seiichi Komiya, Junzo Kato, Morio Nagata, Shuichiro Yamamoto, Motoshi Saeki, Atsushi Ohnishi, Hisayuki Horai, A Method for Implementing a System to Guide Interview-driven Software Requirements Elicitation, The 4th Joint Conference on Knowledge-Based Software Engineering (JCKBSE2000), Brno, Czech Republic, 2000.
[19] Software Engineering Standards Committee of the IEEE Computer Society, IEEE Guide for Developing System Requirements Specifications, IEEE Std 1233-1998, IEEE, New York, 1998.
[20] Software Engineering Standards Committee of the IEEE Computer Society, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, IEEE, New York, 1998.
[21] Ying Liang, Daune West, and Frank A. Stowell, An Approach to Object Identification, Selection and Specification in Object-Oriented Analysis, in Information Systems Journal, Vol. 8, No. 2, 1998, pp. 163-180, Blackwell Science Ltd., 1998.
[22] Romi S. Wahono and Behrouz H. Far, A Framework for Object Identification and Refinement Process in Object-Oriented Analysis and Design, Proceedings of the First IEEE International Conference on Cognitive Informatics (ICCI 2002), pp. 351-360, IEEE Computer Society Press, Canada, August 2002.
[23] Dong Liu, Kalaivani Subramaniam, Behrouz H. Far and Armin Eberlein, An Agent-based System for Class Elicitation and Modeling in Object Oriented Analysis and Design, 2nd ASERC Workshop on Software Architecture, pp. 18-19, Canada, February 2003.