

Towards The Use of Intelligent Agents in Collaborative Object-Oriented Analysis and Design

Romi Satria Wahono^{*1}Behrouz Homayoun Far^{*2}^{*1} ^{*2} Department of Information and Computer Sciences, Saitama University

Software design often requires collaborative working between members of a software design project team. In many cases, the members are geographically distributed making the need for effective information and communication technologies acute. Implementing distributed artificial intelligence in intelligent agents is an alternative approach to achieve tasks on distributed computer systems. This paper examines the issues associated with the use of intelligent agents within the software analysis and design, especially we concern to solve the problems identified in object model creation process for object-oriented analysis and design. This is intended to serve as a useful decision support system for designers, and should allow faster, better, and more economic, collaborative object-oriented analysis and design.

1. Introduction

Object oriented analysis and design has now become a major approach in the design of software systems. The state of object-oriented analysis and design is evolving rapidly. 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 and their attributes needed to implement the software, describe the associations between the identified objects, define the behavior of the objects by describing the function implementations of each object, and refine objects and organize classes by using inheritance to share common structure [Holland et al., 1996].

Researchers and software designers have come to a conclusion that identifying objects including associations, attributes and behaviors is an ill-defined task [Booch, 1991], regarding of the difficulties of heuristic [Holland et al., 1996] [Kato, 1998] and there is no unified methodology for object-oriented software analysis and design. This is mainly due to lack of formalism for object-oriented software analysis and design.

This paper examines the issues associated with the use of intelligent agents within the software analysis and design, especially we concern to solve the problems identified in object model creation process for object-oriented analysis and design. This is intended to serve as a useful decision support system for designers, and should allow faster, better, and more economic, collaborative object-oriented analysis and design.

We are developing a multi-agent system that aims to help designers while designing object-oriented software by automating the difficulties in the object model creation process. We formulate design rules and cases for solving the above problems, and store them in the knowledge bases. This system was named *OOExpert* [Romi et al., March 1999] [Romi et al., June 1999].

2. Overview of the Object Model Creation Process

As shown in Figure 1, object-oriented analysis and design begins with a problem statement (requirement) generated by users and possibly customer. The requirement may be incomplete, informal, and identification processes make it more precise and expose ambiguities and inconsistencies. The real-world system described by the requirement must be understood and identified, and its essential features abstracted into a model.

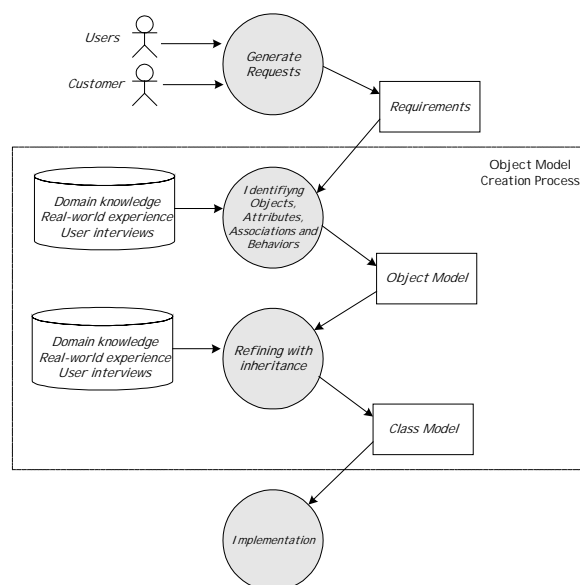


Figure 1: Overview of the Object Model Creation Process

Identifying objects, attributes, associations and behaviors of the object are important steps in constructing an object model. Then, the next step is to organize classes by using inheritance to share common structure. Inheritance can be added in two ways [Rumbaugh et al., 1991]: by generalizing common aspects of existing classes into a superclass (*bottom up* or *generalization* approach), or by refining existing classes into specialized subclasses (*top down* or *specialization* approach). The object identification and refinement process are together called *object model creation process*.

3. Models for Object Model Creation Process

3.1 Object Based Formal Specification

Figure 2 shows a first step towards solving the object model creation process. We propose an approach where end users take an active role in 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)*.

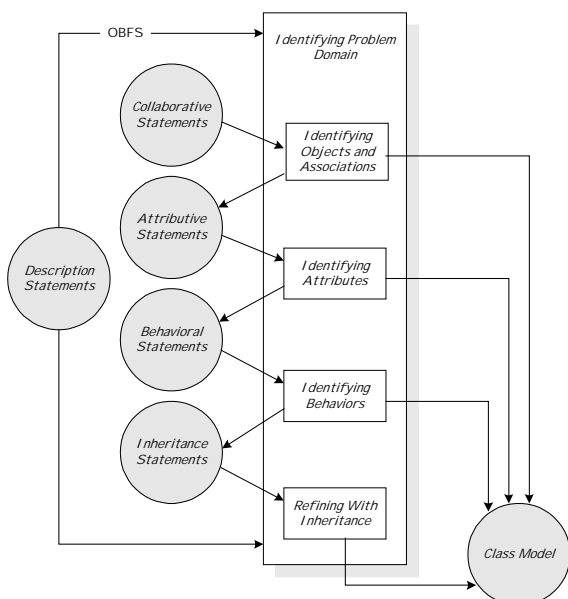


Figure 2: Object Model Creation Process by Using OBFS for Specifying Requirements

(1) 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: *Requirement ID*, *Requirement Name*, *Language*, and *Description*. The description statements 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.

(2) Collaborative Statements (CS)

Collaborative statements are used to identify objects, and association between objects. The first step in 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, and objects are corresponding to nouns that are identified from collaborative statements. Collaborative statement consists of Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (*E*) natural language.

(3) Attributive Statements (AS)

Attributive statements are used to identify object's attributes. 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. Attributive statement consists of Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (*E*) natural language.

(4) Behavioral Statements (BS)

Behavioral statements are used to identify object's behaviors. Behavior is how an object acts and reacts, in terms of its state changes and message passing. Behavioral statement must contain behaviors of each object identified at the previous step. Behavioral Statement consists of Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (*E*) natural language.

(5) Inheritance Statements (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 statement provides sentences that have is-a-kind-of relationship. Inheritance statement consists of Subject (*S*), Verb (*V*), and Object (*O*) as well as the English (*E*) natural language.

3.2 Object Identification Process

Figure 3 shows our strategy for solving the object identification process.

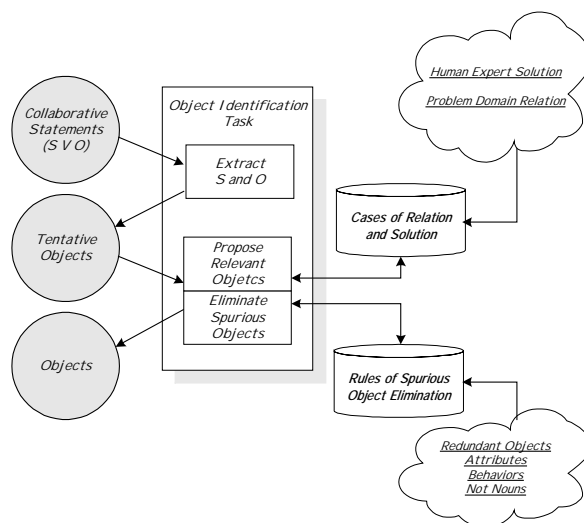


Figure 3: Object Identification Process

We use collaborative statements from OBFS to guide end users in describing their problem. The first step in object identification process is to extract *S* and *O* written in the collaborative statements to be tentative objects (*OBJ_t*).

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

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

$$\neg OBJ_{red} \wedge \neg OBJ_{att} \wedge \neg OBJ_{beh} \wedge \neg OBJ_{non} \Rightarrow OBJ$$

Furthermore, two kinds of case-base indexed in our CBR are: *Human Expert Solution (HES)* and *Problem Domain Relation (PDR)*. The final result of the object identification process is a relevant object (*OBJ*).

3.3 Association Identification Process

Figure 4 shows our strategy for solving the association identification process.

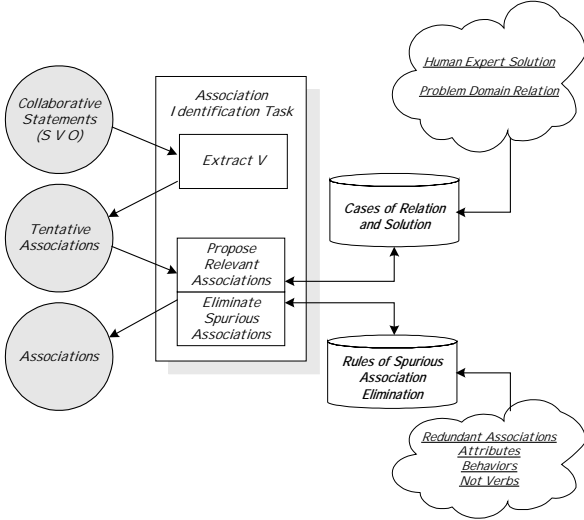


Figure 4: Association Identification Process

We use collaborative statements from OBFS to guide end users in describing their problem. The first step in association identification process is to extract *V* written in the collaborative statements to be tentative associations (*ASS_t*).

$$\forall CS \in E [V_{cs} \Rightarrow ASS_t]$$

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

$$\neg ASS_{red} \wedge \neg ASS_{att} \wedge \neg ASS_{beh} \wedge \neg ASS_{nov} \Rightarrow ASS$$

The final result of the association identification process is a relevant association (*ASS*).

3.4 Attribute Identification Process

Figure 5 shows our strategy for solving the attribute identification process. We use attributive statements from OBFS to guide end users in describing their problem. The first step in attribute identification process is to extract *O* written in the attributive statements to be tentative attribute (*ATT_t*).

$$\forall AS \in E [O_{as} \Rightarrow ATT_t]$$

The next step is to eliminate spurious attributes and propose relevant attributes using Rule-Based Reasoning (RBR) and Case-Based Reasoning (CBR) paradigms. In the RBR, the system will discard unnecessary and incorrect

attributes according to the following criteria: *redundant attributes (ATT_{red})*, *objects (ATT_{obj})*, and *behaviors (ATT_{beh})*.

$$\neg ATT_{red} \wedge \neg ATT_{obj} \wedge ATT_{beh} \Rightarrow ATT$$

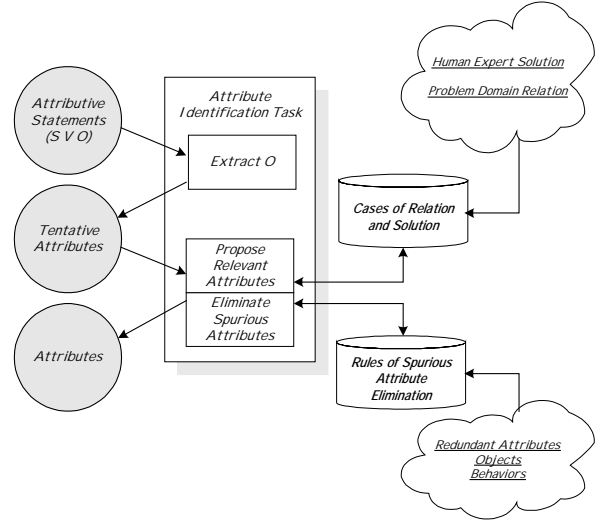


Figure 5: Attribute Identification Process

The final result of the attribute identification process is a relevant attribute (*ATT*).

3.5 Behavior Identification Process

Figure 6 shows our strategy for solving the behavior identification process.

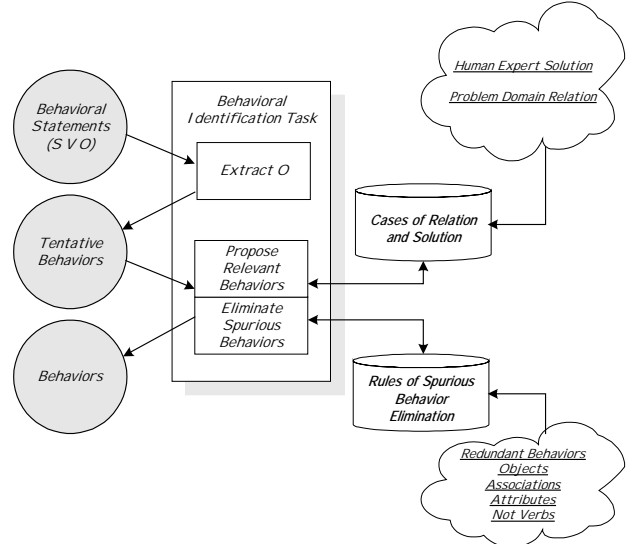


Figure 6: Behavior Identification Process

We use behavioral statements from OBFS to guide end users in describing their problem. The first step in behavior identification process is to extract *O* written in the behavioral statements to be tentative behavior (*BEH_t*).

$$\forall BS \in E [O_{bs} \Rightarrow BEH_t]$$

The next step is to eliminate spurious behaviors and propose relevant behaviors using Rule-Based Reasoning (RBR) and Case-Based Reasoning (CBR) paradigms. In the RBR, the

system will discard unnecessary and incorrect behaviors according to the following criteria: *redundant behaviors* (BEH_{red}), *objects* (BEH_{obj}), *associations* (BEH_{ass}), *attributes* (BEH_{att}), and *not verb behaviors* (BEH_{nov}).

$$\neg BEH_{red} \wedge \neg BEH_{obj} \wedge \neg BEH_{ass} \wedge \neg BEH_{att} \wedge \neg BEH_{nov} \Rightarrow BEH$$

The final result of the behavior identification process is a relevant behavior (BEH).

3.6 Object Refinement with Inheritance

We use bottom-up (generalization) concepts as a basic approach to build a new model of object refinement process. As shown in Figure 7, object refinement with inheritance process begins by listing objects found in the previous object model creation process, and searching similar object names, attributes, and behaviors. If the similar objects are found, the object will be a sub class and a tentative superclass will be generated automatically.

The next process is to give the superclass a name. The superclass can be given from user, or automatically generates from similar object names. The result of this process is a class model with inheritance structure. Oppositely, if the similar object cannot be found, the object will be a class model (without inheritance structure) directly. The final result of the object refinement process is a class model, which is the combination of class model with inheritance and class model without inheritance.

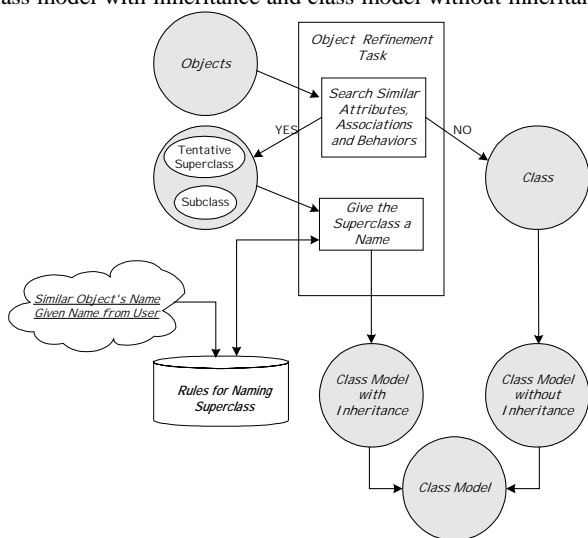


Figure 7: Object Refinement with Inheritance

4. System Architecture and Design

Figure 8 shows an overview of intelligent agents for object model creation process in object-oriented analysis and design. 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: *requirement acquisition agent*, *object identification agent*, *attribute identification agent*, *association identification agent*, *behavior identification agent* and *object refinement agent*.

Each agent is intelligent in its own field and may interact with its human counterpart or behave autonomously. Each agent has a local knowledge base and a reasoning engine. All agents have a

communication engine, which facilitate communication and navigation of each agent on the network environment.

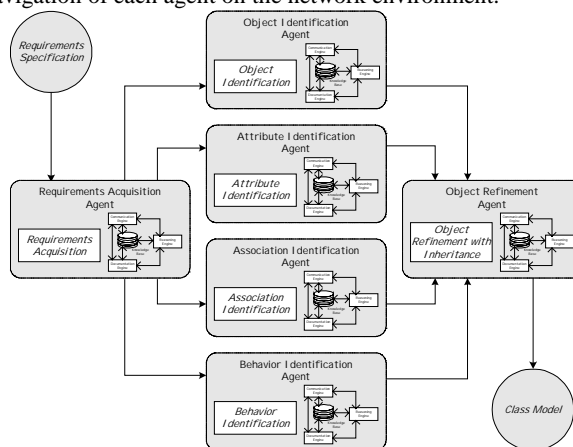


Figure 8: Intelligent Agents Architecture for Object Model Creation Process

5. Conclusion and Future Works

In this paper, we presented the outlines of an ongoing research project to build an intelligent agents system for supporting object-oriented analysis and design. The requirements acquisition agent, object identification agent, attribute identification agent, association identification agent, behavior identification agent and object refinement agent are already developed. The strategy for implementing and indexing two kinds of case-base: *Human Expert Solution (HES)* and *Problem Domain Relation (PDR)* used in *OOExpert* are under development.

References

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

[Holland et al., 1996] Ian M. Holland and Karl J. Lieberherr, "Object-Oriented Design", ACM Computing Surveys, Vol. 28, No. 1, March 1996.

[Kato, 1998] 加藤貞行, "オブジェクト識別についての一考察とその効果", 情報処理学会研究報告, Vol.90, No.100,1998.

[Romi et al., March 1999] Romi Satria Wahono, B.H. Far, "Distributed Expert System Architecture for Automatic Object-Oriented Software Design", Proceedings of the Third Workshop on Electro-Communication and Information, pp. 131-134, Japan, March 1999.

[Romi et al, June 1999] 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.

[Romi et al., July 2000] Romi Satria Wahono and Behrouz 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, pp.230-231, Tokyo, Japan, July, 2000.

[Rumbaugh et al., 1991] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorenson, "Object-Oriented Modeling and Design", Prentice Hall, 1991.