ISSN 0918-7685

PROCEEDINGS
OF
THE 9TH SCIENTIFIC MEETING
TEMU ILMIAH 2000
TI-IX PPI 2000

HAMAMATSU, September 2, 2000

Organized by

PPI-JEPANG

Membuka dunia untuk Indonesia

Indonesian Students Association in Japan

Persatuan Pelajar Indonesia di Jepang

PROCEEDINGS OF THE 9TH SCIENTIFIC MEETING
TEMU ILMIAH TI-IX PPI 2000

ISSN 0918-7685
pp. 283-286

Methodological Support for Object Identification from Formal Requirement Specification

Romi Satria Wahono*
Graduate School of Sciences and Engineering, Saitama University
Indonesia Institute of Sciences (LIPI)

Abstract: The challenges of object-oriented design are to identify the objects and classes needed to implement the software, and to define the behaviors and the attributes of the objects from requirement specification. These are very complicated challenges because of there dependence on heuristic. Formal requirement specification have the additional advantage over informal requirement specification because they are amenable to machine analysis and manipulation. The greatest benefit of applying a formal requirement specification is that system designers gain a deeper understanding of the specified system, because they have forced to be more abstract and precise about desired properties. Another important application of formal specification is that they can be used as a base to reason about the behavior of the desired system. Formal requirement specification will be first important step for solving the difficulties and ill-defined tasks in the object model creation process, including identification of objects, attributes, behaviors and organization of objects with inheritance. In this paper, we propose methodological support for object identification from formal requirement specification.

Keywords: object-orientation, object identification, formal requirement specification

1. INTRODUCTION

Requirement acquisition is considered one of the most important activities in software development. Most faults found during testing and operation result from poor understanding or misinterpretation of requirements. In spite of progress in analysis techniques, CASE tools support, prototyping, and early verification and validation technique, software development still suffers from poor requirements acquisition.

In the traditional approach to software analysis, system analyst interview end users to capture requirements. We propose an approach where end users take an active role in analysis by specifying requirements using structured formal requirement specification. We use a structured formal requirement specification to guide end users in describing their problem. This approach will be first important step for solving the difficulties and ill-defined tasks in the object model creation process, including identification of objects, relationships, attributes, behaviors and organization of objects with inheritance. This approach also takes advantage of end users' domain knowledge.

In this paper we present a structured formal requirement specification and methodological support for object model creation process, especially object identification from structured formal requirement specification.

* **Romi Satria Wahono**, Jinde 221-1 Syato Kouyama 102, Urawa, Saitama, Japan (338-0812). Tel./Fax: 048-856-1147
Email: romi@cit.ics.saitama-u.ac.jp
URL: http://www.cit.ics.saitama-u.ac.jp/~romi

2. REQUIREMENT SPECIFICATION

A *requirement* is a desired relationship among phenomena of the environment of a system, to be brought about by the software system that will be constructed and installed in the environment.

A *specification* describes system behavior sufficient to achieve the requirement. A specification is a restricted kind of requirement. All the environment phenomena mentioned in a specification are shared with the system. The phenomena constrained by the specification are controlled by the system, and the specified constraints can be determined without reference to the future. Specifications are derived from requirements by reasoning about the environment, using properties that hold independently of the behavior of the system [Jackson et al., 1995].

In other words, we can say that the difference between requirements and specification is that requirements refer to the entire system to be realized, whereas a specification refers only to the part of the system to be implemented by software.

Jackson and Zave [Jackson et al., 1995] consider specifications as special kind of requirements. A requirement is a specification if all actions constrained by the requirement are controlled by the software system, and all information it relies on is shared with the software system and refers only to the past, not the future. Requirements (and thus specifications) do not talk about the state of the software system. In contrast to this view, we consider a specification to be a model of the software system to be built in order to satisfy the requirements.

The software requirements specifications process consists of three steps:

1. *Requirements capture and analysis*
2. *Requirements definition and specification*
3. *Requirements validation*

The origin of most software system is the need of a client /user who desires a new software system. The final output of this process is a requirements document, which defines the system to be developed [Jalote, 1997].

3. FORMAL METHOD and FORMAL SPECIFICATION

Formal methods used in developing software systems provide frameworks for specifying, developing, and verifying systems in a systematic manner rather than ad hoc manner. Formal methods are used to reveal ambiguity, incompleteness, and inconsistency in a software system. System designer use formal methods to specify desired behavioral and structural properties [Ralston et al., 1993].Formal methods can be used in all phases of software's development and present an opportunity to develop new techniques to improve software production. One tangible product to applying a formal method is a formal specification.

Formal requirement specifications have the additional advantage over informal requirement specifications because they are amenable to machine analysis and manipulation. The greatest benefit of applying a formal requirement specification is that system designers gain a deeper understanding of the specified system, because they have forced to be more abstract and precise about desired properties. Another important application of formal requirement specification is that they can be used as a base to reason about the behavior of system's components.

The usefulness of formal requirement specification is more and more accepted by researcher and practical software engineers. But formal requirement specification techniques still suffer from two drawbacks.

First, research spends more effort to develop new languages than provide methodological guidance for using existing ones. Often, users of formal techniques are left alone with a formalism for which no explicit methodology has been developed.

Second, formal requirement specification techniques are not well integrated with the analysis phase of software engineering. Often, formal requirement specifications begin with very short description of the system to be implemented, and detail is added during the development of the formal requirement specification. Such a procedure does not

adequately take into account the need to thoroughly analyze the system to be implemented and the environment in which it will operate before a detailed requirement specification is developed.

4. METHODOLOGY FOR IDENTIFYING OBJECT FROM STRUCTURED FORMAL REQUIREMENT SPECIFICATION

Figure 1 shows our strategy to solve the object model creation process, including object, behavior, association identification and also classes' organization by using inheritance. We propose an approach where end users take an active role in analysis by specifying requirements using structured formal requirement specification. We use a structured formal requirement specification to guide end users in describing their problem. This approach will be first important step for solving the difficulties and ill-defined tasks in the object model creation process, including identification of objects, relationships, attributes, behaviors and organization of objects with inheritance.

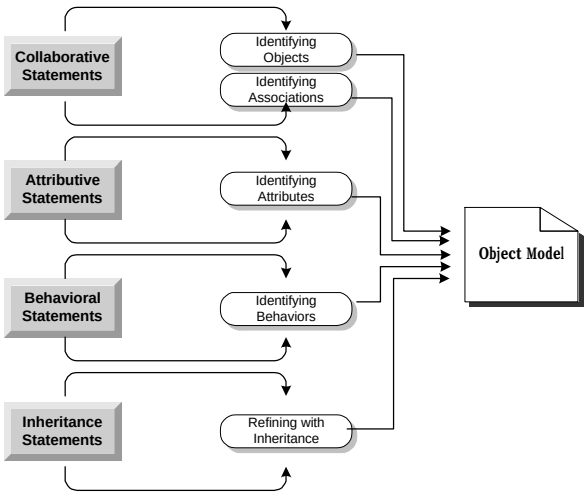

Figure 1: Object Model Creation Process by Using Structured Formal Requirement Specification

4.1. Collaborative Statements

Collaborative statements are used to identify objects, and association between objects. The first step in object model creation process is to identify relevant object and its 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 identified from collaborative statements.

Any dependency between two ore more objects in the collaborative statements is an object association. A reference from one object to another is also an association. Associations show dependencies between objects at the same level of abstraction as the objects themselves. Associations can be implemented in various ways, but such implementation decisions should kept out of the analysis model to preserve design freedom. Associations often correspond to verbs or verb phrases. These include physical location (next to, part of, contained in), directed actions (drives), communication (talks to), ownership (has, part of), or satisfaction of some condition (works for, married to, manages).

The algorithm for identifying objects and association from collaborative statements is shown in Figure2.

Figure 2: Object and Association Identification from Collaborative Statements

4.2. Attributive Statements

Attributive statements are used to identify object 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.

4.3. Behavioral Statements

Behavioral statements are used to identify object behaviors. Behavior is how an object acts and reacts, in terms of its state changes and message

passing [Booch et al., 1991]. Behavioral statement must contain behaviors of each object identified at the previous step.

4.4. Inheritance Statements

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 is a relationship between classes where one class is the parent (base/super/ancestor/etc.) class of another. Inheritance statement provides sentences that have *is-a-kind-of* relationship. For example, mountain bikes, racing bikes, and tandems are all different kinds of (is-a-kind-of) bicycles.

5. CONCLUSION

We proposed an approach where end users take an active role in analysis by specifying requirements using structured formal requirement specification. We use a structured formal requirement specification to guide end users in describing their problem. In this paper we presented methodological support for object model creation process, especially object identification from structured formal requirement specification.

REFERENCES

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

[Booch et al., 1999] Grady Booch, James Rumbaugh, and Ivar Jacobson, "The Unified Modeling Language User Guide," *Addison-Wesley*, 1999.

[Iglewski et al., 1997] Michal Iglewski and Tomasz Muldner, "Comparison of Formal Specification Methods and Object-Oriented Paradigms", *Journal of Network and Computer Applications*, Vol. 20, No. 4, 1997, Academic Press.

[Jackson et al., 1995] Michael Jackson and Pamela Zave, "Deriving Specifications from Requirements: an Example", *Proceedings of the 17th International Conference on Software Engineering*, Seattle, WA USA, April 1995.

[Jalote, 1997] P. Jalote, "An Integrated Approach to Software Engineering", *Springler-Verlag New york Inc*., 1997.

[Ralston et al., 1993] Anthony Ralston, Edwin D. Reilly, "Encyclopedia of Computer Science", *Van Nonstrand Reinhold, IEEE Press*, 1993.

[Romi et al., 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.

BIOGRAPHY of AUTHOR

Romi Satria Wahono, Was born in Madiun-Indonesia on October 2nd 1974, Received B.Eng. in Information and Computer Sciences in 1999, from Saitama University. He is currently a researcher at the Indonesian Institute of Sciences (LIPI), and a M.Eng. candidate at the Department of Information and Computer Sciences, Saitama University. The research fields of his interests are Distributed Artificial Intelligence, Multi Agent Systems, Reasoning System, Software Engineering, and Object-Orientation. He is a member of the Association for Computing Machinery (ACM), The Institute of Electrical and Electronics Engineers (IEEE) Computer Society, The Institute of Electronics, Information and Communication Engineers (IEICE), Japanese Society for Artificial Intelligence (JSAI), and Indonesian Society on Electrical, Electronics, Communication and Information (IECI).