Chapter 16

USING ONTOLOGIES TO CREATE OBJECT MODEL FOR OBJECT-ORIENTED SOFTWARE ENGINEERING

Dencho N. Batanov¹ and Waralak Vongdoiwang²
¹School of Advanced Technologies, Asian Institute of Technology, Klong Luang, Pathumthani, 12120, Thailand; ²University of the Thai Chambers of Commerces, Din Daeng, Bangkok, 10400, Thailand

Abstract: In this paper we introduce and discuss our approach to creating an object model from a problem domain text description as a basic deliverable of the analysis phase in Object-Oriented Software Engineering using ontologies. For this purpose we first briefly compare object models with ontologies. The object model of a system consists of objects, identified from the text description and structural linkages corresponding to existing or established relationships. The ontologies provide metadata schemas, offering a controlled vocabulary of concepts. At the center of both object models and ontologies are objects within a given problem domain. The both concepts are based on reusability using intensively libraries. The major difference is that while the object model contains explicitly shown structural dependencies between objects in a system, including their properties, relationships and behavior, the ontologies are based on related terms (concepts) only. Because ontology is accepted as a formal, explicit specification of a shared conceptualization, we can naturally link ontologies with object models, which represent a system-oriented map of related objects. To become usable programming entities these objects should be described as Abstract Data Types (ADTs). This paper addresses ontologies as a basis of a complete methodology for object identification and their modeling as (converting to) ADTs, including procedures and available tools such as CORPORUM OntoExtract and VisualText, which can help the conversion process. This paper describes how the developers can implement this methodology on the base of an illustrative example.

Key words: Object Model; Ontologies; Software Engineering; Object-Oriented; Knowledge base
1. INTRODUCTION

Ontology is a specification of a representational vocabulary for a shared domain of discourse: definitions of classes, relations, functions, and other objects (Gruber, 1993) or, more generally, a specification of conceptualization (Gruber, 1994). The basic components of an ontology are concepts, relationships between concepts and attributes. Concepts, relationship types and attributes are abstracted from the objects and thus describe the schema (the ontology). On the other hand, the objects populate the concepts, values and relationships, instantiate the attributes of those objects and relationships among them respectively. Three types of relationships that may be used between classes or concepts in ontology are generalization, association, and aggregation. Ontology is well known as a structured description of declaration and abstract way to express the domain information of an application (Angele, Staab & Schurr, 2003). The concepts in an ontology are similar with objects in object oriented software engineering. To solve the problem of heterogeneity in developing software applications, there is a need for specific descriptions of all kinds of concepts, for example, classes (general things), the relationships that can exist among them, and their properties (or attributes) (Heflin, Volz, and Dale, 2002). Ontologies described syntactically on the basis of languages such as eXtensible Markup Language (XML), XML Schema, Resource Description Framework (RDF), and RDF Schema (RDFS) can be successfully used for this purpose.

Object models are different from other modeling techniques because they have merged the concept of variables and abstract data types into an abstract variable type: an object. Objects have identity, state, and behavior and object models are structural representation of a system of those objects [based on concepts of type, inheritance, association, and possibly class (ChiMu Corporation, 2003)]. In the artificial intelligence (AI) area, ontology has been focused on knowledge modeling. On the other hand, a lot of industry standards and powerful tools for object-oriented analysis, design, and implementation of complex software systems have been developed. And because of the closed connection between ontologies and object models, these maturing standards and tools can be used for ontology modeling (Cranefield & Purvis, 1999).

Object orientation is a commonly accepted paradigm in software engineering for the last few decades. The motto of object-oriented software development may be formulated in different ways, but its essence can be stated simply: “Identify and concentrate on objects in the problem domain description first. Think about the system function later.” At the initial analysis phase, however, identifying the right objects, which are vital for the system’s functionality, seems to be the most difficult task in the whole