Chapter 7
Software Design Based on Using Ontologies and Algorithm Algebra

ABSTRACT

This chapter proposes an approach to the automated development of programs based on the use of ontological facilities and algebra-algorithmic toolkit for design and synthesis of programs (IDS). The program design ontology, developed using Protégé system and represented in OWL format, includes concepts from various subject domains (sorting, meteorological forecasting, and other) intended for description of main program objects: data, functions, and relations between them. IDS toolkit generates the initial (skeleton) algorithm scheme based on its ontological description extracted from OWL file. The generated scheme is the basis of further design of the algorithm and synthesis of a program in a target programming language. The approach is illustrated by examples of developing parallel sorting, meteorological forecasting, and N-body simulation programs.

INTRODUCTION

This chapter describes the approach to development of parallel programs using ontologies and algebra-algorithmic facilities. Ontology is a philosophical term that refers to the study of being, becoming, existence and reality and

DOI: 10.4018/978-1-5225-9384-3.ch007

was introduced to computer science through the field of artificial intelligence (Strmečki, Magdalenić, & Kermek, 2016).

In (Happel & Seedorf, 2006), main approaches for using ontologies in software engineering are listed:

- ontology-driven development: the use of ontologies at development time for describing the problem domain;
- ontology-enabled development: applying ontologies at development time to support developers in their tasks;
- ontology-based architectures: the use of ontologies as primary runtime artifacts;
- ontology-enabled architectures: applying ontologies as support to runtime software.

In (Calero, Ruiz, & Piattini, 2006), a broader classification of ontologies based on their subject of conceptualization is proposed:

- knowledge representation ontologies that are used to formalize knowledge under a specific paradigm;
- generic ontologies which represent reusable common-sense knowledge;
- high-level ontologies describing general concepts and notions;
- domain ontologies that offer vocabulary for concepts in a particular domain;
- task ontologies which describe the vocabulary related to a generic activity;
- domain task ontologies that are reusable only in a particular domain;
- method ontologies applicable to a reasoning process designed to perform a particular task;
- application ontologies that are dependent on the application and often specialize the vocabulary of a domain or task ontology.

Based on the moment when they are utilized, ontologies can be used during the development or in runtime. The former approach is called ontology-driven development, in which, for example, ontology's semantic content can be converted into a system component.

In (Gašević, Kaviani, & Milanović, 2009), the use of ontologies in software engineering throughout software lifecycle phases is researched. In the analysis phase, an ontology is commonly used for requirement engineering. In the design phase, ontologies are used as software models, business vocabularies