Chapter 1
Ontologies in Computer Science:
These New "Software Components" of Our Information Systems

Fabien L. Gandon
INRIA, France

ABSTRACT

Ironically the field of computer ontologies suffered a lot from ambiguity. The word ontology can be used and has been used with very different meanings attached to it. The authors will introduce in this chapter two faces of ontologies in computer science: (1) the ontology object: focusing on the nature and the characteristics of the ontology object, its core notions and its lifecycle. (2) ontology engineering: the branch of knowledge modeling that develops ontology-oriented computable models of some domain of knowledge, focusing here on the design rationale and the assisting tools.

INTRODUCTION

"When I use a word," Humpty Dumpty said in rather a scornful tone, "it means just what I choose it to mean - neither more nor less." – Lewis Carroll, *Through the Looking-Glass*, Chapter VI

Knowledge engineering is a broad research domain where the core issues include the acquisition and the modeling of knowledge. Modeling knowledge, consists in representing it in order to store it, to communicate it or to externally manipulate it. Automating the manipulation of knowledge leads to the design of knowledge-based systems *i.e.,* systems which behavior relies on the symbolic manipulation of formal models of knowledge pieces in order to perform meaningful operations that simulate intelligent capabilities. As such, knowledge engineering is a branch of artificial intelligence.

Knowledge representation raises the problem of the form *i.e.,* the choice of a representation formalism that allows us to capture the semantics at play in the targeted pieces of knowledge. One approach that emerged in the late 80s is based on the concept of ontologies. An ontology, as we shall see, is that part of the knowledge model that captures the semantics of primitives used to make formal assertions in a domain of application. Computer science ontologies

DOI: 10.4018/978-1-61520-859-3.ch001

are children of Artificial Intelligence that recently came to maturity and powerful conceptual tools of Knowledge Modeling. Ontologies provide a coherent base to build on, and a shared reference to align with, in the form of a consensual conceptual vocabulary on which one can build descriptions and communication acts.

In this introduction chapter, we shall focus on the branch of knowledge modeling that develops ontology-oriented computable models of some domain of knowledge.

The word *ontology* can be used and has been used with very different meanings attached to it. Ironically, and as we will see, the ontology field suffered a lot from ambiguity. We will introduce here two faces of ontologies in computer science:

- The *ontology object*: focusing on the nature and the characteristics of the ontology object, its core notions and its lifecycle.
- The *ontology engineering*: focusing on the design rationale and the assisting tools.

FROM ONTOLOGY TO ONTOLOGIES

The term "ontology" was constructed from the Greek Ontos ("what is", "what exists") and Logos ("the discourse", "the study"). It first appeared in 1606 in "Ogdoas scholastica" from Jacob Lorhard. In philosophy, ontology is a fundamental branch of metaphysics, concerned with the concept of existence, the basic categories of existing and the most general properties of being. As a branch of philosophy, Ontology is the metaphysical study of the nature and relations of existence.

As computer scientists, at first sight this term and its definition might not seem very useful for our daily work. However when one considers the work a software engineer performs when designing a class hierarchy in an object-oriented programming language or a database schema and especially the questions this engineer has to answer for such a task, then the ontological questions such has "what are the categories of the things existing around us?" appear to be very much closer than expected. Software engineers designing object-oriented models are wondering about: the objects that their applications will handle, the classes that combine the characteristics common to all these objects, the relationships that may exist between these objects, etc. In other words, these engineers are questioning what defines these classes of objects, which characteristics can ensure that a given object belongs to a class, what this membership means in terms of content or possible manipulations. Like Molières' character Mr. Jourdain who is amazed to discover that he has been speaking prose all his life without knowing it, we, computer scientists, could be amazed to see how close our modeling rationale can be to ontological questioning. When we question the existential definition of classes of objects used in the scenarios of the applications we develop, we are sometimes the "Mr. Jourdain" of Ontology.

Computer scientists borrowed the term "ontology" from the philosophers in the early 80s: it can be found for instance in an article by McCarthy (McCarthy, 1980) and in the book of John Sowa (Sowa, 1984) before it became famous with the article by Thomas Gruber (Gruber, 1993). To be more precise, the reason why object-oriented programming presents such a resemblance to the notion of ontologies in computer science is that they have common ancestors: the early systems of symbolic artificial intelligence. The beginnings of this branch of artificial intelligence are intertwined with the beginnings of computer science, because since its beginnings, computer science has perpetuated the dream of the designers of automata to simulate or exceed human intelligence with artificial systems.

A NOTION LOOKING FOR A NAME

The branch of artificial intelligence on which we just focused is said symbolic because it is