Representation of Changes in Ontology Driven Object Oriented Software using Categories

Arash Shaban-Nejad, Volker Haarslev

Department of Computer Science and Software Engineering, Concordia University,
H3G1M8 Montreal, Quebec, Canada

{arash_sh, haarslev}@cs.concordia.ca

1 Introduction

In order to address certain aspects of representation of changes in an ontology driven object-oriented software application we propose a method based on category theory as a mathematical notation, which is independent of a specific choice of ontology language and any particular implementation. In our research, we have focused on ontologies not in isolation but as artifacts that are part of object-oriented software and used to specify, model or document software systems. An ontology captures concepts descriptions in a domain of interest into classes or concepts and defines relationships among instances of those classes. Ontologies can be used to define a common shared understanding about a software application domain and associated tasks. Despite many differences between ontology and object oriented modeling [1], in some sense, ontologies can be viewed as a hierarchical structure of classes and objects in a software conceptual design phase. Therefore, some rules and definitions are applicable for both.

2 An Ontology Driven Software Application

Emphasis on object-oriented and component-based architectures in software engineering allows for modularization, encapsulation, and distribution of units of program code [2]. Using ontologies that aims to provide a common vocabulary to represent useful knowledge for the software developers is a new trend to manage the inherent complexity of large software systems. Ontologies provide an underlying discipline of modeling software applications by defining concepts and properties. Ontologies can describe software architectures and requirements, which are difficult to model with object-oriented languages [3]. Ontologies are useful in software applications for: describing semantics of programming interfaces, providing a structure to organize knowledge, reducing development effort for generic tools, improving the data and the tool integration, facilitating requirements elicitation by providing a common vocabulary, reusing organizational knowledge to compose the requirements [4].

Ontologies could also be used for capturing behavioral knowledge [3]. In addition, ontological commitment in software plays an important role to increase accessibility, maintainability, integrity and transparency of the application software which based on the ontology [5]. An ontology driven object oriented application in our context is defined as an architecture created from a shared domain model which includes several interrelated knowledge sources which are connected with some object-oriented components for user interface and control components [6].

Due to reusability of ontologies, the overall cost and effort for creating and maintaining ontology driven applications will be reduced. Modifying and adjusting the ontologies in response to changing data or requirements can be handled using some mathematical operation such as category theory.

3 Category Theories and Ontology

A formal model of objects based on "category theory" is introduced in [7, 8] which allows analyzing of both static and dynamic properties of a design. Category theory is closely connected with computation and logic [9]. Using categories one can recognize certain regularities to distinguish a variety of objects and capture and compose their interactions and differentiate equivalent interactions, identify patterns of interacting objects and extract some invariant in their action, or decompose a complex object in basic components [10]. Categorical notations consist of diagrams with arrows. Each arrow $f: X \rightarrow Y$ represents a function. A Category C includes a class of objects, a class of morphisms ("arrows") and for each morphism f there exists one object as the domain of f and one object as the codomain. The graphical representation of a category can be formalized using the notion of a diagram.

$$A \circ \xrightarrow{\quad f \quad} \circ B$$

Category theory with its logical and analytical features has the potential to be considered as a vehicle for representation of ontologies. An ontology can be viewed in an interconnected hierarchy of theories as a sub-category of a category of theories expressed in a formal logic [11]. Ontological relationships represented using category theories are considered to be directed [12] to show the direction of information. These "relationships" are known as "morphism" in category context.

4 Managing Changes using Category Theory

After describing the ontological concepts within categories representing a modular hierarchy of domain knowledge, we deploy category theory to analyze ontological changes in the following ways:

I. **By comparing a previous state of a class with a later state:** An object oriented category model [9] able to describe the state space (set of all possible states for a given state variable set) for a class as a cross product of attribute domain and the operations of a class as transitions between states. It also allows the definition of message passing and method binding mechanisms. Category theory has a special type of mapping between categories called *functor*. Functors are defined as morphisms in the category of all small categories (where classes are defined as categories) [13]. The role of time is not usually taken into account in current ontology evolution studies. Considering time in ontologies can increase the complexity and needs a very expressive ontology language to represent it. As expressivity increases, the computability will decrease. In our approach, we represent conceptualization of things indexed by times, for example from FungalWeb Ontology [14]: "*enzyme* has_pH_optimum at *t*" is rendered as "*enzyme*-at-*t* has_ pH_optimum". Then we use a set of categories indexed by time using functors to capture different state of ontological structure in different times. The category O at the time t that is represented as O_t models the state of the ontologies and all the related interactions at this time. Using a functor allows us to represent the transition from O_t to $O_{t'}$ (Figure 1) when the time changes from *t* to *t'*. In addition, each sub ontology A can be modeled by the series of its successive states A_t from its '*Creation*' to '*Destruction*' [10].
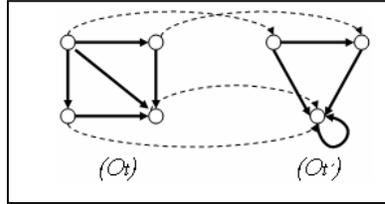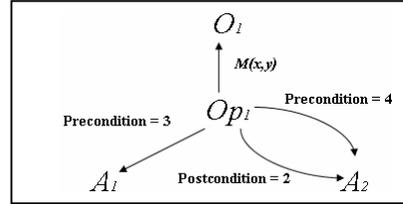

Fig. 1. Using Functor

Fig. 2. Measuring Coupling

II. **By measuring coupling:** Coupling specifies the extent of the connections between elements of a system and it can identify the complexity of an evolving structure. Measuring coupling is useful for predicting and controlling the scope of changes to an ontological application. Often a change in one class can cause some changes to the dependent classes. When the coupling is high, it indicates existence of a large number of dependencies in an ontological structure which must be checked to analyze and control the chain of changes. Coupling for ontological elements can be described by number of connections and links between them. So, we focus on arrows in category theory to study these connections. For analyzing a conditional change we followed the formal model described in [9] by identifying three types of arrows in our category: precondition arrows, postcondition arrows and message-send arrows for an existing category [9]. The type of message is determined by the types of changes caused by a method. In the category shown in Figure 2, the coupling for the operation Op_1 is a nonnegative number which can

be calculated by the count of the three types of arrows (postconditions, preconditions and M(x,y)).

In order to handle ontology versioning we also use category theory to determine the degree of semantic similarity between different ontology versions. At this time, we are applying the proposed methods for managing changes in FungalWeb Ontology [14] and its associated applications. Bioinformatics is a challenging domain in knowledge management. Biological data are highly dynamic and bioinformatics applications are large and have complex interrelationships between their elements. In addition, they usually have various levels of interpretations for one particular concept.

To conclude incorporating ontology could highly improve software change management and maintenance and we believe category theory has potential to be considered as a supplementary tool to capture and represent the full semantics of ontology driven software applications and it can provide a formal basis for analyzing complex evolutive software systems.

References

1. What is ontology? IBM. (http://www.alphaworks.ibm.com/contentnr/semanticsfaqs)
2. Orfali, R., Harkey, D., and Edwards, J.,: The Essential Distributed Objects Survival Guide. New York: John Wiley & Sons, 1996.
3. Dirk Deridder, Theo D'Hondt.: A Concept-Centric Approach to Software Evolution, In proceeding of ACM Conference (OOPSALA 2004 workshop), Vancouver, Canada, 2004.
4. G. Santos, K. Villela, L. Schnaider, A. Rocha and G. Travassos.: Building Ontology Based Tools for a Software Development Environment. In proceeding of LSO04, Springer 2004.
5. N. Guarino.: Formal Ontology and Information Systems. In Proceedings of FOIS'98, Trento, Italy, pages 3-15. IOS Press, June 1998.
6. Holger Knublauch, Daniel Oberle, Phil Tetlow and Evan Wallace:. A Semantic Web Primer for Object-Oriented Software Developers, W3C Working Group Note 9 March 2006
7. S. Mac Lane, Categories for the Working Mathematician, 1971 (corrected 1994), Springer, New York, EUS, 1994.
8. B. Pierce.: Basic Category Theory for Computer Scientists, MIT Press, Cambridge, 1991.
9. Scott A. Whitmire.: Object Oriented Design Measurement, John Wiley & Sons, September 1997, ISBN: 0-471-13417-1.
10. A. EC. Ehresmann and J. P. Vanbremeersch.: The Memory Evolutive Systems as a Model of Rosen's Organism-(Metabolic, Replication) Systems, Springer 2006, 16:137–154
11. M. J. Healy, T. P. Caudell. "Ontologies and Worlds in Category Theory: Implications for Neural Systems", Axiomathes Journal (2006) Springer 16:165–214
12. M. Krötzsch, P. Hitzler, M. Ehrig, Y. Sure.: Category Theory in Ontology Research: Concrete Gain from an Abstract Approach. Technical Report, AIFB, U of Karlsruhe, Mar 2005.
13. Steve Awodey: Category Theory, Oxford University Press, 2006, ISBN-10: 0-19-856861-4.
14. Christopher J.O. Baker, Arash Shaban-Nejad, Xiao Su, Volker Haarslev, Greg Butler.: Semantic Web Infrastructure for Fungal Enzyme Biotechnologists. In: Journal of Web Semantics, (4)3, 2006.