A FRAMEWORK FOR ANALYZING CHANGES IN HEALTH CARE LEXICONS AND NOMENCLATURES

Arash Shaban-Nejad

A thesis
in
The Department of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy

Concordia University
Montreal, Quebec, Canada

April 2010

Your file Votre référence
ISBN: 978-0-494-67378-2
Our file Notre référence
ISBN: 978-0-494-67378-2

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Arash Shaban-Nejad**

Entitled: **A Framework for Analyzing Changes in Health Care Lexicons and Nomenclatures**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair
Dr. Brigitte Jaumard

______________________________ External Examiner
Dr. Reda Alhajj

______________________________ Examiner
Dr. Greg Butler

______________________________ Examiner
Dr. Olga Ormandjieva

______________________________ Examiner
Dr. Justin Powlowski

______________________________ Supervisor
Dr. Volker Haarslev

Approved ______________________________
Chair of Department or Graduate Program Director

__________20______ ______________________________
Dr. Robin A.L. Drew
Faculty of Engineering and Computer Science

ABSTRACT

A Framework for Analyzing Changes in Health Care Lexicons and Nomenclatures

Arash Shaban-Nejad, Ph.D.

Concordia University, 2010

Ontologies play a crucial role in current web-based biomedical applications for capturing contextual knowledge in the domain of life sciences. Many of the so-called bio-ontologies and controlled vocabularies are known to be seriously defective from both terminological and ontological perspectives, and do not sufficiently comply with the standards to be considered formal ontologies. Therefore, they are continuously evolving in order to fix the problems and provide valid knowledge. Moreover, many problems in ontology evolution often originate from incomplete knowledge about the given domain. As our knowledge improves, the related definitions in the ontologies will be altered.

This problem is inadequately addressed by available tools and algorithms, mostly due to the lack of suitable knowledge representation formalisms to deal with temporal abstract notations, and the overreliance on human factors. Also most of the current approaches have been focused on changes within the internal structure of ontologies, and interactions with other existing ontologies have been widely neglected.

In this research, after revealing and classifying some of the common alterations in a number of popular biomedical ontologies, we present a novel agent-based framework, RLR (Represent, Legitimate, and Reproduce), to semi-automatically manage the evolution of bio-ontologies, with emphasis on the FungalWeb Ontology, with minimal human intervention. RLR assists and guides ontology engineers through the change management process in general, and aids in tracking and representing the changes, particularly through the use of category theory.

Category theory has been used as a mathematical vehicle for modeling changes in ontologies and representing agents' interactions, independent of any specific choice of ontology language or particular implementation. We have also employed rule-based hierarchical graph transformation techniques to propose a more specific semantics for analyzing ontological changes and transformations between different versions of an ontology, as well as tracking the effects of a change in different levels of abstractions. Thus, the RLR framework enables one to manage changes in ontologies, not as standalone artifacts in isolation, but in contact with other ontologies in an openly distributed semantic web environment. The emphasis upon the generality and abstractness makes RLR more feasible in the multi-disciplinary domain of biomedical Ontology change management.

ACKNOWLEDGEMENTS

I wish to express my heartily gratitude to my supervisor, Dr. Volker Haarslev, whose patience, guidance, comments, and continuous support enabled me to complete my study. I appreciate his confidence in my work, which encouraged me to pursue my research goals and overcome several obstacles.

I would like to thank my friends and colleagues for their comments, discussions, encouragements and friendship.

My deepest appreciation goes to my family for their unconditional support and love. To my parents who waited so long for this moment. To my lovely wife who bore the burden of managing everyday life with patience. To my brother who supported me in many ways and made me feel strong. To my adorable sisters who constantly cheered me up and gave me confidence and pride.

Furthermore I am deeply indebted to people who taught me anything at any stage of my life, people who gave me assistance in different situations, as well as people who gave me wounds, pain and sorrow, which in some cases was a prime motivation.

To

My Son

Nickan

- *Is it true, said Cocles, what you say?*
- *What?*
- *That you have killed him [the eagle]?*
- *And that we are going to eat him? ...*

Do you doubt it? said Prometheus. Have you looked at me? – When he was alive, did I dare to laugh? – Was I not horribly thin?

- *Certainly.*
- *He fed on me long enough. I think now that is my turn.*
- *A table! Sit down! Sit down! Gentlemen!*

[...]

If he had made me suffer less, he would have been less fat; less fat, he would have been less delectable.

André Gide (1869-1951), "Prometheus Illbound" translated by L. Rothermere, London, Chatto and Windus 1919

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

In alphabetical order:

A-Box: Assertion Box (assertions about individuals)
AI: Artificial Intelligence
API: Application Programming Interface
CDA: Clinical Document Architecture
CDR: Communicable Disease Reporting
COTS: Commercial Off-The-Shelf
CUI: Concept Unique Identifier
CVS: Concurrent Versions System
DAG: Directed Acyclic Graphs
DL: Description Logic
ELR: Electronic Laboratory Reporting
FMA: The Foundational Model of Anatomy
FOWL: Fuzzy OWL
FR: Functional requirement
FWOnt: FungalWeb Ontology
GALEN: Generalized Architecture for Languages, Encyclopedia and Nomenclatures
GO: Gene Ontology
GUI: Graphical User Interface
HD Graph: Hierarchical distributed graph
HL7: Health Level 7
HLR: High-Level Replacement
ICD: The International Classification of Diseases
KR: Knowledge Representation
LIMS: Laboratory Information Management System
MAS: Multi-agent system
MeSH: Medical Subject Headings
NCIT: National Cancer Institute Thesaurus
NFR: Non-Functional Requirement
NLM: The National Library of Medicine
NLP: Natural language processing
OBO: Open Biological Ontologies
OWL: The Web Ontology Language
RACER: Renamed Abox and Concept Expression Reasoner
RDF: Resource Description Framework
RLR: Representation, Legitimation and Reproduction
SKDON: Skin Disease Ontology
SPARQL: SPARQL Protocol and RDF Query Language
SWRL: The Semantic Web Rule Language
TA: Terminologia Anatomica
T-Box: Terminological Box (axioms about class definitions)
UMLS: The Unified Medical Language System
W3C: The World Wide Web Consortium
WWW: World Wide Web
XML: Extensible Markup Language

I. Introduction and Thesis Statement

This introductory chapter presents a general overview of the thesis organization and the proposed approach. In addition, it explains the motivation, the main problems, objective of the research and major contributions and questions addressed in the thesis.

I.1 Introduction

Nobody steps into the same river twice. The same river is never the same, because that is the nature of water.

"Heraclitus on Rivers," Derek Mahon, 1991, quoted Jaybook, August 1997

Using clinical vocabularies and lexicons has a long history in medicine and life science. However, a new trend is emerging to use ontologies, as defined by Gruber [Gru93] ("specification of a conceptualization") to provide an underlying discipline of sharing knowledge and modeling biomedical applications by defining concepts, properties and axioms. Ontologies are widely used as a vehicle for knowledge management in current biomedical applications, for sharing common vocabularies, describing semantics of programming interfaces, providing a structure to organize knowledge, reducing the development effort for generic tools and systems, improving the data and tool integration, reusing organizational knowledge, and capturing behavioral knowledge.

The main components of ontologies are concepts (classes), relations (properties), individuals (instances) and axioms. Concepts represent a set or class of entities within a domain. Relations describe the interactions between individuals of those concepts. Individuals are the "things" that exist in the real world, represented by a concept. Axioms are being used to constrain values for concepts or individuals. Ontologies capture knowledge from a domain of interest in order to share it between both machines and humans. When the knowledge changes, then definitions will be altered. A formal ontology is dynamic such as a living organism. It is evolving over time in order to fix errors, reclassify the taxonomy, add/remove concepts, attributes, relations and instances. As ontologies are changing over the time, one of the most challenging issues in ontology change management is keeping ontologies consistent when changes occur. The topic of change continues to be a source of much debate, as it brings together various issues that are central to philosophy, logic, cognitive science, neural networks, linguistics and physics including identity, persistence and time [Was06].

This research aims to provide an answer to the following questions: what is actually changed during the evolutionary process of a biomedical ontology? How this non-stop evolution can be controlled and managed with minimum human intervention? What formalisms are suitable to capture, represent and analyze the ontological alterations? To answer these questions, we present a novel multi-agent-based approach, RLR (Represent, Legitimate, and Reproduce) to manage the evolving structure of biomedical ontologies in a consistent manner. The RLR framework aims to assist and guide ontology engineers through the change management process in general, and aids in tracking and representing the changes, particularly through the use of graph transformation empowered with category theory as a mathematical notation, which is independent of any specific choice of ontology language or particular implementation.

As an application scenario, we consider the FungalWeb Ontology [BSS+06], an integrated formal bio-ontology in the domain of fungal genomics. The Fungal taxonomy is not stable. Most of the alterations are changes in names and taxonomic structure and relationships. Fungal names reflect data about the organisms; thus, as our understanding of the relationships among taxa improves, these names will need to be changed, as they will no longer convey the correct information to the user [Cro05]. Most fungi names are currently based on phenotypes (visible characteristics of an organism). These name changes may cause confusion and affect the validity of different queries. For example, eyespot disease in cereals and the issues related to naming its associated fungi are actually represented in [CGG03]. The morphological conceptualization of fungi is not sufficient, and will no longer work because all of the names based only on morphology must be re-evaluated. In addition, the phylogenetic-based conceptualization has its own limitations, since the decision of where to draw the line between different species is not always easy to make [Cro05]. To manage this process of continuous change, one needs to refer to the nature of ontological structure, where names in a taxonomy are only meaningful and valuable once linked to descriptive datasets, which are extracted and managed from various databases and literature in an integrated environment. Through advances in molecular biology, one can also expect changes in taxonomical structure and relationships. For example, by studying some molecular, morphological and ecological characteristics, *Glomeromycota* was discovered in 2001 [SSW01] as a new fungal phylum. Another example is the sedge parasite, *Kriegeria eriophori*, which has never been satisfactorily classified. As another example, ribosomal RNA gene sequences and nucleus-associated ultrastructural characters were analyzed separately and combined to define the new subclass *Microbotryomycetidae* [SFM99].

A small percentage of discovered fungi have been linked to human diseases, including dangerous infections. Treating these diseases can be risky because human and fungal cells are very similar. Any medicine that kills the fungus may also damage the human cells. Therefore, greater knowledge of fungi and correct identification of each species is crucial to improving the quality of fungal-based products and identifying new and better ways to treat serious fungal infections in humans. In order to update the ontological structure of the FungalWeb Ontology for the annotation of fungal genes, we need to improve the content of the ontology regularly. As an example, the older version of the FungalWeb Ontology did not have sufficient terminology to annotate genes involved in *Malassezia* infections. To meet this new requirement, the updated version of the ontology has gained 26 additional terms addressing these infections.

RLR takes a building blocks approach towards the development of a fully automatic ontology change management framework. What is presented in this thesis is a rather theoretical research, which uses insights and ideas from semantic web, software engineering, category theory, intelligent agents and the theory of graph transformation.

I 1.1 Motivation

After Implementing the FungalWeb Ontology we have reached a stage where we wish to develop a change management strategy to update ontological knowledge. Ontologies evolve all the time and each change in ontological structure or nomenclature can have crucial impacts on the inferred knowledge. Especially in a heterogeneous environment like the Web with vast amount of interdependencies, even simple changes on ontological elements can trigger a domino effect and sometimes it is really hard to guess all impacts of a simple change. Different versions of an ontology behave differently in response to the posed queries. If one works with a system based on frequently changing ontologies how one can even ask queries and be sure about the logical and scientific correctness of the answer. The issues arising from ontology evolution can affect validity of information in applications which are tightly bound to concepts in a particular ontological context.

The fact that the problem of ontology evolution has existed for the past decade in the field of knowledge representation and artificial intelligence and despite many efforts in this area, there are no trustable and widely accepted tools and algorithms available and also there is not any clear sign of progress in the attempts to solve the problem of changes in the conceptualization. These observations motivated us that there is a need to direct our attention to more diverse theories and disciplines which seem to propose an alternative set of concepts able to reveal and solve these fundamental problems.

I 1.2 Problem/Objective of Research

This study attempts to achieve the following objectives:

1. To identify the effects of changes in bio-ontologies in general and in the FungalWeb ontology and its dependent artifacts in particular (Section II.6, Section III.1, and Section IV.1);
2. To identify the factors influencing the consistency of evolving ontologies, and propose a method to deal with this issue (See Section II 3.3, Section II.4, Section II.6, Section III 2.3.3, Section III 3.5.5.2, Section III 4.5 (specifically III 4.5.4.2), and Section (V.3);
3. To analyze changes in distributed biomedical ontologies (See Section III 4.5, and Section IV.1);
4. To design an agent-based framework to capture, represent and analyze changes in bio-ontologies with minimum human intervention (See Section III.2);
5. To examine category theory as a formalism for ontological change management (See Section III.3);

6. To introduce a representation formalism to support agent interactions and analysis of evolving structures using graph transformation (Section III 3.5.6, Section III.4, and Section (III 4.5.5).

I 1.3 Research question

The major research question in this research is: "Which mechanisms and methods can be used to build a framework to handle changes in ontologies, especially the ones in the biomedical domain?" This general question can be detailed into some smaller questions:

1. What are the specific natures and characteristics of ontological changes? (see Chapter II (specifically Section II.2, Section II.3, and Section II.6))
2. What is actually changed during the evolutionary process of an ontology? (See Section II.2, Section II.3, and Section II.6)
3. How this non-stop evolution can be controlled and managed with minimum human intervention? (See Section II.4, and Section III.2)
4. What formalisms are suitable to capture, represent and analyze the ontological alterations? (Section III.3, and Section III.4)
5. How changes can be captured, tracked and represented and how a representation can be changed? (Section III.2, Section III.3, and Section III.4)
6. What variables determine the quality of the changed ontology, and to control consistencies during the evolution process? (See Section II 3.3, Section II.4, Section II.6, Section III 2.3.3, Section III 3.5.5.2, and Section III 4.5)
7. How can we manage and monitor the frequently changing ontologies in a distributed environment? (Section III 4.5, and Section IV.1)

8. Is it possible to extend the usage of the proposed framework into different domains? (Section IV.1)

In our research we attempt to explain how the state-of-the-art research and development in the Semantic Web and bioinformatics can help in addressing these issues.

I 1.4 Approach

By analyzing the context of the problem and reviewing other existing techniques for change management in some existing ontologies, we propose an agent-based framework for maintaining changes in bio-ontologies through the notions of graph transformation and category theory.

As an experiment we have focused on changes in the FungalWeb Ontology which can potentially alter the related artifacts in an integrated biomedical system. In contrast to some of the existing works on ontology evolution, we specifically focus on changes in distributed ontologies, not as standalone artifacts but in contact with other ontologies in an open Semantic Web environment. The introduced formal representation framework, based on hierarchical distributed graph transformation and category theory, is expressive enough to capture the evolutionary behavior of dynamic ontologies in a distributed environment. Our proposed method offers a multidisciplinary framework in which different approaches from various disciplines can be plugged in to define a comprehensive change management mechanism. To provide some evidence of the usability of our framework, we will consider some case studies to apply some of the proposed techniques to show the technical correctness and feasibility of our approach.

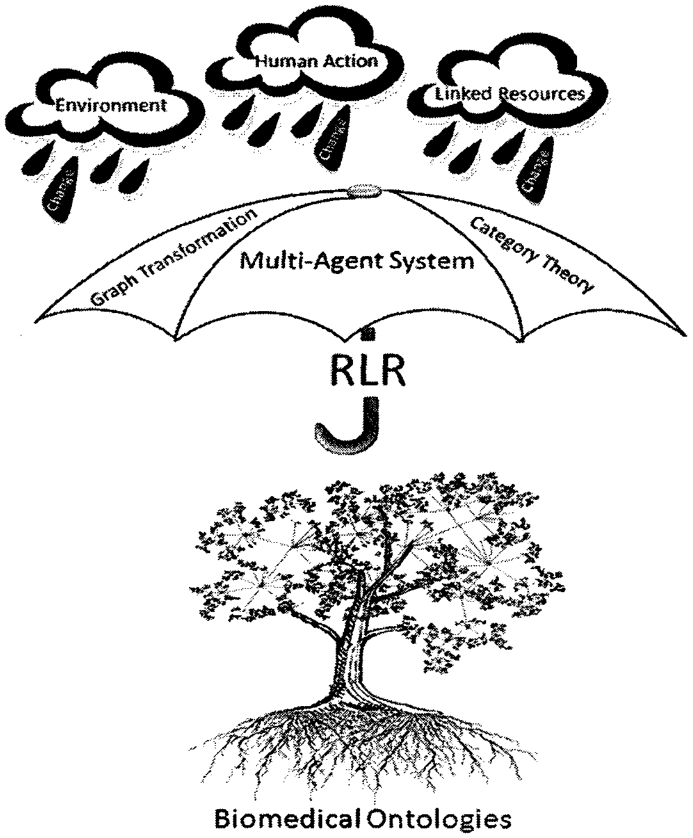


Fig. 1.1.[1] The RLR framework aims for protecting biomedical ontologies from the undesired effects of changes due to human actions, environments and alteration in other linked resources. RLR is an integrated multi-agent framework, which is formalized using category theory and graph transformation.

I 1.5 Contributions and Publications

In order to achieve the research objectives and answer to the questions raised in Section I 1.3, this thesis offers the following key contributions.

- Introducing and reviewing basic definitions (Section II.1), major tasks and challenges in ontology evolution from several perspectives including philosophical and linguistics (Section II.2), artificial intelligence (Section II.3), software

[1] This figure demonstrates our emphasize on a controlled method for applying changes in ontologies (analogous to water absorption of the tree roots rather than watering through uncontrolled scattered showers).

engineering and database (Section II.5), as well as issues relating to human intervention (Section II.4);

- Studying of change management in several bio-ontologies (Section II 6.1), along with revealing and classifying the most common alterations in their structure (Table 2.3), as well as reviewing the available tools and algorithms (Section II 6.2);
- Analyzing the FungalWeb Ontology and classifying the changes in its terminology and hierarchical structure, along with presenting actual examples of such changes (Section III.1);
- Modeling RLR, a cooperative Multi-agent framework, to capture, represent, track and analyze changes within ontological structures through a rule-based reactive and proactive behavior with minimum human intervention acting along with an integrated argumentation framework (Section III.2). RLR, with its associated formalisms, tends to provide a blueprint for modeling a realistic algorithm for managing changes in biomedical knowledge-based systems.
- Formalizing the RLR framework through category theory (Section III.3) and graph transformation (Section III.4);
- Employing category theory as a mathematical representation vehicle for analyzing changes within biomedical ontologies and performing a number of editorial operations in various abstraction levels, which can be used to address several problems including scalability and complexity issues in large biomedical ontologies through operations such as composition (Section III.3);

- Utilizing categories to support agents' communication, negotiation, state transitions, compositions and transformations in different levels of abstractions (Section III 3.5.6);
- Presenting an extended graph-oriented semantics for analyzing temporal distributed biomedical ontologies by means of hierarchical distributed graph transformation (Section III 4.5), which supports consistent transitions, and coordinates the communications and interactions between different agents to perform concurrent and parallel actions (Section III 4.5.4, and Section III 4.5.5);
- Focus on breadth of coverage to reflect the interdisciplinarity in our research as much as possible. To this end we have tried to address both computational and non-computational problems in ontology change management;
- Emphasis upon the generality and abstractness, which makes our approach more feasible in the multi-disciplinary domain of biomedical ontology change management;
- Demonstrating the applicability of our approach through a series of case studies in various domains, such as biomedical ontologies evolution (Section IV.1), requirement engineering for agile application modeling (Section IV.2) and exploring the evolutionary relationship between different species through phylogenetic analysis (Section IV.3).

The details of our contributions for each part of our research can be found at the end the related sections. Our efforts have been mostly reflected in our publications in refereed journals and conference proceedings [SH10a, SH10b, SOK+09, SH09, SH08a, SH08b, SH08c, SH07a, SH07b, SH07c, SH07d, SH06a, SH06b, BSS+06]. The following chapters and sections are partially written based on our published papers.

- Chapter II is partially based on [SH10a] (Section II.4), [SH09] (Section II.6 and Section II 2.1), [SH06a] (Section II.5), and [SH06b] (Section II.2, and Section II.3);
- Chapter III is partially based on [SH10b] (Section III.4), [SH08a] (Section III 2.2), [SH07a, SH07b, SH07d] (Section III 3.5 and Section III 1.1), [SH07c] (Section III 2.3), [SH06a] (Section III 3.5.5.2), and [BSS+06] (Section III.1);
- Chapter IV is partially based on [SOK+09] (Section IV.2), [SH08a, SH07b] (Section IV.1), [SH08b, SH08c, SH10b] (Section IV.3), and SH07c (Section IV.2).

I 1.6 Thesis Overview and Organization

This thesis proceeds as follows: Chapter II introduces the primary definitions, which are used throughout the thesis, of knowledge representation, the Semantic Web, ontologies with focus on ontologies in the domain of life science. We also look at the problem of change through the lens of other disciplines, such as philosophy and linguistics, with emphasize on the philosophical foundations for "change" "from ancient time till now. In addition we review some of the well known maintenance approaches in software engineering and database domains. Then we go on and will look at the major requirements and challenges in ontology change management, which need to be addressed in this field. We provide a comprehensive survey on the state of the art of change management in biomedical ontologies. We also address some of the issues related to human intervention in dynamic systems.

After a motivational scenario on the FungalWeb Ontology and its evolving structure, Chapter III will utilize our designed agent based ontology change management framework along with the categorical formalism needed to represent the agents' communication and analyzing changes in ontological structures. The discussion on the formalism will be continued in Chapter III by describing our graph oriented approach for representing model transformations and ontological transitions using hierarchical distributed graph transformation. The applicability of our approach will be shown in different application areas throughout a series of case studies in Chapter IV. Finally Chapter V concludes the discussion by giving a summary of our achievements and highlighting our scientific contribution and the plan for future work.

II. Ontology Maintenance: Scope, Requirements & Challenges

This chapter includes six sections, which respectively provide reviews on knowledge representation, the Semantic Web and ontologies, philosophical foundations for change mangement, general requirements for a successful ontology change management, challenges back to human factors, the established practices for change management in database and software engineering, and state of the art in biomedical ontologies maintenance.

II.1 Knowledge Representation, the Semantic Web and Ontologies

I know that you believe you understand what you think I said, but I'm not sure you realize that what you heard is not what I meant.

Robert McCloskey (1914-2003)

II 1.1 Knowledge Representation

Knowledge representation (KR) as a multi-disciplinary area in AI is concerned with formally representing and analyzing a meaning in a domain of discourse within the natural world by adding metadata to the content and using logical reasoning, which allows inference [DSS93]. In summary, KR, as stated by Sowa [Sow00], can be defined as an "application of logic and ontology to the task of constructing computable models for some domain". The term "computable model" in this definition is what distinguishes KR in computer science from philosophy[2]. A broad range of major knowledge representation frameworks have been modeled based on frames, rules, logics, semantic networks and graphs, Prolog, SQL, Java, Petri nets, and object-oriented languages [Sow00]. Sowa indicates [Sow00] four essentials for any knowledge representation language, namely vocabulary, syntax, semantics, and rules of inference. Since the development of the World Wide Web (WWW) and its advance as a core part of the daily lives of many people around the world, the way in which information is transmitted,

[2] http://www.formalontology.it/index.htm

stored, and accessed has been revolutionized. In order to effectively represent knowledge out of the huge quantity of available data in the Web, W3C supported what is called the Semantic Web—as opposed to the syntactic Web—to move the Web towards being both human and machine understandable. The primary idea behind the Semantic Web has been defined as an "extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is based on the idea of having data on the Web defined and linked such that it can be used for more effective discovery, automation, integration, and reuse across various applications" [HBM02]. The Semantic Web generally uses URIs (Uniform Resource Identifier) to represent data in triple based structures such as "Resource Description Framework" (RDF)[3] syntaxes, which were built for metadata modeling.

II 1.2 Semantic Web and Ontologies

To overcome the problem of miscommunication between humans and computers, ontologies have been employed as basic building blocks of the Semantic Web to reuse and share the common consensus of knowledge of a domain in the real world. The term ontology is originally borrowed from philosophy and, as stated by Smith in [Smi03.b], ontologies have been employed in computer science to solve the so-called "Tower of Babel" problem in databases, which refers to the lack of a standard (due to historical, cultural, technical, behavioral, or linguistic reasons) in representing information in different databases, where a unique concept may be represented with several dissimilar labels and vice versa. Gruber describes an ontology in the context of knowledge

[3]http://www.w3.org/RDF/

representation as "the specification of a conceptualization" [Gru93]. He defined "conceptualization" in his paper [Gru95] as "an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly". Later, to distinguish between ontology in philosophy and in computer science, the term "formal" was added to Gruber's definition to emphasize the computability feature: "an ontology is a formal, explicit specification of a shared conceptualization" [SBF98]. Stumme and Maedche [SM01] defined an ontology as a tuple $\mathcal{O} := (\mathcal{C}, \text{is\_a}, \mathcal{R}, \sigma)$ with $\mathcal{C}$ is a set of concepts, is_a as a partial order on C (i. e., a binary relation is_a $\subseteq C \times C$ which is reflexive, transitive, and anti-symmetric), $\mathcal{R}$ is a set of *relations*, and $\sigma: \mathcal{R} \rightarrow C^{+}$ is a function which assigns to each relation name its arity. Another algebraic definition of an ontology was presented in [KS03] as: "a pair $\mathcal{O} = (\mathcal{S}; \mathcal{A})$, where $\mathcal{S}$ is the ontological signature - describing the vocabulary- and $\mathcal{A}$ is a set of ontological axioms- specifying the intended interpretation of the vocabulary in some domain of discourse".

II 1.3 Biomedical Ontologies and Controlled Vocabularies

Biomedical informatics is an emerging multi-disciplinary field that aims to integrate computer science techniques with applications derived from medicine and biology. It talks about different computational problems in the integration of biomedical databases, spatial and temporal patterns of mRNA expression, protein structure, laboratory management, clinical outcomes, publication records, and so forth [SWL+03]. There are some issues in biomedical informatics that motivate us to use ontologies as the basic building blocks of knowledge representation methods in this area. Some of these issues are:

- **Multidisciplinary nature of the domain:** Needs a common shared language between different agents (human or machine). The ontology provides a shared understanding, so different parties using the same interoperable ontology can recognize the meaning of the same resource.
- **Mass production of data:** Biomedical applications usually produce and use massive quantities of data (e.g., all genes in a genome, all transcripts in a cell, all metabolic processes in a tissue, and all data involved in protein-protein interactions) [SWL+03], so we need some formal methods to deal with these data, and to annotate and process them for use by biologists.
- **Complexity of data**: Biological data are complex in terms of the types of data stored and the richness and constraints working upon relationships between those data [BBB+98]. Ontologies in these complex structures facilitate data, information, and knowledge exchange.
- **Distribution of data**: Bioinformatics is an inherently integrative discipline, requiring access to data from a wide range of sources and the ability to combine these data in new and interesting ways [AGM+90]. Hundreds of differet data resources and analysis tools are used in bioinformatics [CBB+00].
- **Volatility of data**: Biological data are not static. As knowledge about biological entities changes and increases, so the annotations of data resources will be changed [SWL+03].

- **Heterogeneity of data**: Most knowledge and data in the area of biology are both syntactically and semantically heterogeneous [Enz84]. Individual concepts, such as gene, have many different, but equally valid, interpretations.

These issues cause great difficulties for both curators of bioinformatics resources and their users. Some of the difficulties are knowing which resources to use in a task, discovering instances of those resources, and knowing how to use each of those resources, how to link their content, and how to transfer data between resources [SWL+03]. Therefore, computational support is required for storing, exploring, representing, and exploiting biological knowledge as well as knowledge in the minds of domain experts.

Biological classification has a long history, dating back to Aristotle's *scala naturae* [Ver08] (scale of nature), which was a very simple method of dividing organisms into groups, ranging from the simple species to more complex ones, based on their appearance. In the 17th century, Carl Linnaeus (1707-78), who is often referred to as the father of modern taxonomy, developed his classification system called *Systema Naturae*[4] for the naming and classification of all organisms. Linnaeus represented his classification method based on binomial nomenclature, "the combination of a genus name and a single specific epithet to uniquely identify each species of organism"[5] (e.g., humans are identified by the binomial *Homo sapiens*). In his system, all species were categorized in three kingdoms, namely Plantae, Animalia, and a group for minerals and organized based on their structural similarities in a five-rank hierarchy as Kingdom, Class, Order, Genus, and Species.

[4] http://www.linnaeus.uu.se/online/animal/1_1.html
[5] http://en.wikipedia.org/wiki/Linnaean_taxonomy

Later, as the understanding of the relationships between organisms changed, taxonomists converted the five ranks into the seven-rank hierarchy by adding the two ranks of "Phylum" (between Kingdom and Class) and "Family" (between Order and Genus). Change in the taxonomic ranks is still an ongoing process. Due to advances in knowledge and the influence of Darwinian evolution as the mechanism of biological diversity and species formation, taxonomists needed a new classification scheme to reflect the phylogeny of organisms. Also, recruiting new criteria other than structural similarities, such as genetic codes and molecular features, and advances in tools and techniques resulted in the discovery of various organisms, forming three new kingdoms, Archaea, Bacteria, and Fungi. These three kingdoms, plus Plantae and Animalia, formed the popular five-kingdom scheme. The biomedical classifications have been organized in several models as Controlled Vocabularies, Thesauri, Taxonomies, and Ontologies. According to Hedden [Hed08]:

- **Controlled Vocabularies:** are restricted lists of words or terms used for labeling, indexing, or categorizing and cross-referencing, which evolve under central control over the changes based on defined policies.
- **Thesauri:** are a more structured kind of controlled vocabulary, providing information about each term and its relationships with other terms.
- **Taxonomies:** are a type of controlled vocabulary that has a tree structure hierarchy (broader term/narrower terms), but not necessarily containing the related-term relationships and other requirements of a standard thesaurus.

Many of the so-called biomedical ontologies are in fact controlled vocabularies, thesauri, or taxonomies, as they do not follow the essential requirements of formal ontologies. Several efforts for migrating available biomedical terminologies to a formal ontological framework are still ongoing. In Section II.6, we will look at some of the popular existing ontologies and controlled vocabularies in the area of life science.

II 1.4 Formalisms for Ontological Knowledge Representation

There are different ontology languages [ZK05] for the representation of conceptual models, with varying characteristics in terms of their expressiveness, ease of use, and computational complexity [SGB00]. The current languages range from natural language-based representations to frame-based and logic-based languages. To support the available ontology languages, several tools and editors [CFG03] are available to aid the ontologist in building, editing, managing, querying, and visualizing ontologies, as well as checking their consistency and reasoning.

II 1.4.1 Description Logics

Description logics (DL) [BCM+03], as a family of knowledge representation languages, provide formal semantics and terminology for describing ontologies. DLs describe knowledge in terms of concepts and relations that are used to automatically derive classification taxonomies. Description logic is also being used for ontology validation. The validation of an ontology by a DL-based classifier such as RACER[6] [HM01], Pellet[7], and FaCT++[8] allows compliance with certain rules of classification, and it also brings other benefits in terms of coherence checking and query optimization. The basic building

[6] http://www.racer-systems.com/
[7] http://clarkparsia.com/pellet
[8] http://owl.man.ac.uk/factplusplus/

blocks used to represent knowledge in description logics are called Tbox (Terminological box: axioms about class definitions), Abox (Assertional box: axioms about individuals) and Rbox (axioms about roles).

II 1.4.2 The OWL Web Ontology Language

OWL is a W3C[9] recommendation and a de facto standard designed for use by applications that need to process the content of information instead of just presenting information to humans [OWL04]. In comparison with XML (Extensible Markup Language) and RDF, OWL adds more vocabulary [OWL04] for describing properties and classes, such as relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes. The OWL has three types: (i) OWL-Lite: supports basic hierarchical representation with simple constraints, which make it easier to provide tool support; (ii) OWL-DL: supports maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems [OWL04]; (iii) OWL-Full: supports maximum expressiveness and the syntactic freedom of RDF with no computational guarantees, which makes it difficult to be supported by reasoning tools [OWL04]. The rich expressivity of OWL and its ability to use description logics, which facilitate formal reasoning, make it a fine candidate to model the complexities of biomedical applications.

[9] http://www.w3.org/

II 1.5 Summary of Section II.1

In this Section, we have reviewed basic definitions, which we will use in the rest of the thesis, of knowledge representation, Semantic Web, and ontologies. In addition, the roles of ontologies and controlled vocabularies for sharing a common understanding between human and machines in computer science and biomedicine have been briefly introduced.

II.2 Philosophical Foundations

Artificial Intelligence cannot avoid philosophy.If a computer program is to behave intelligently in the real world, it must be provided with some kind of framework into which to fit particular facts it is told or discovers. This amounts to at least a fragment of some kind of philosophy, however naive.

John McCarthy, Mathematical Logic in AI. Daedalus 117(1): 297-310, Winter 1988.

This section discusses how we can gain valuable perspectives on our research by viewing it through the lens of other disciplines, such as philosophy and linguistics.

II 2.1 Change and Philosophy

Designing a framework for ontology evolution by using available methods in the area of knowledge representation (KR) is the main strategic plan in the Semantic Web community. However, since the problem of change management is not completely computational, it seems necessary to incorporate complementary techniques from other disciplines such as philosophy, mathematics, biology, neural networks, semiotics, linguistics, and psychology (to study the behavioral affects) for the ontology evolution process (cf. Figure 2.1). The topic of change, particularly changes in ontologies, brings together various issues that are central to philosophy, including identity, persistence and time [Was06].

Discussion about change is as old as philosophy itself. Heraclitus (535–475 BCE), for example, argued that "All is flux," and everything is changing all the time, so that it is impossible to step into the same river twice. Parmenides (b. 510 BCE) and Zeno of Elea (490–430 BCE) were not in agreement with Heraclitus's statement; they believed in the constancy and stability of the world. Parmenides had stated that "reality is one, and this one, which only is, is unchanging" [Mag99]. Zeno of Elea also believed all changes and motions are in fact illusions of the senses [HG04], and to show the paradoxical nature of change and motion, he summarized his philosophy into several paradoxes, including The Dichotomy, Achilles and the Tortoise and The Arrow [Kem06].

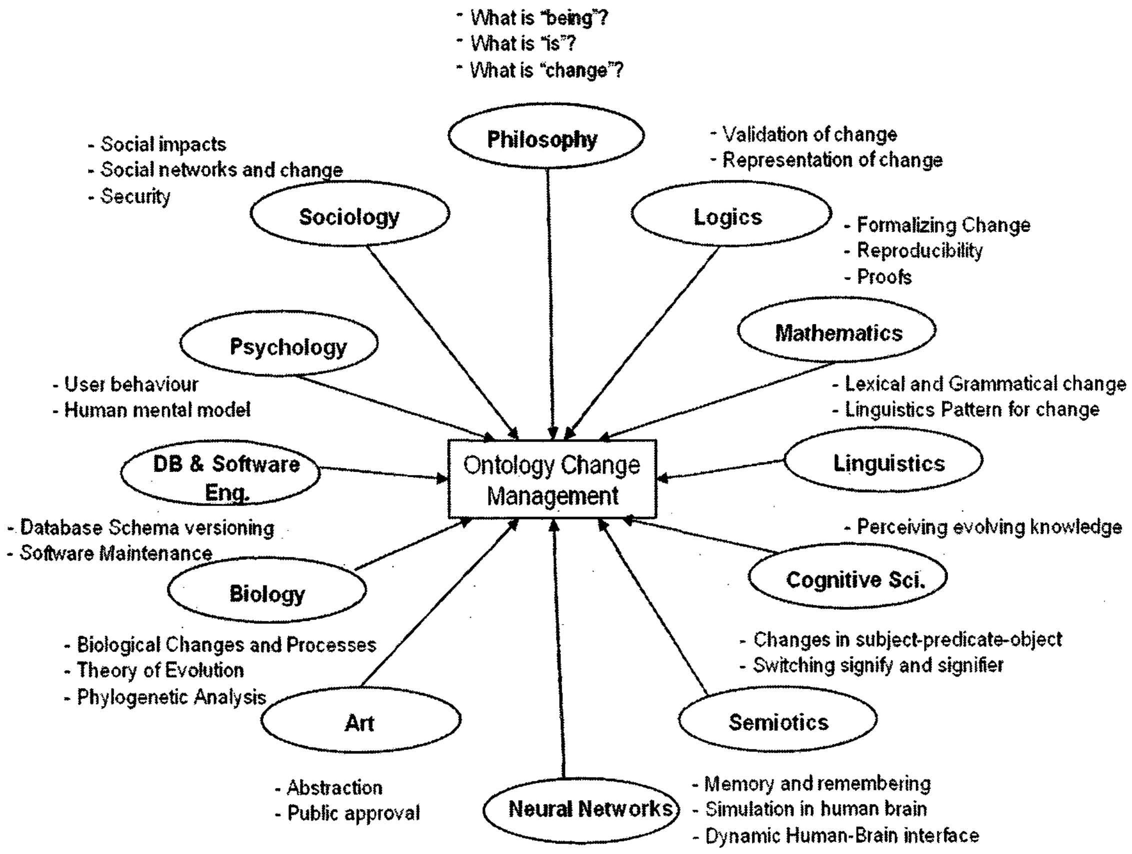


Fig. 2.1. Multi-disciplinary nature of research on ontology change management.

Plato (427–347 BCE) in his allegory of the Cave tried to overcome this issue by separating the world into the visible world, which is uncertain and changes frequently, and the intelligible or real world, which is stable, arose from reason and includes the timeless unchanging “Forms”. Husserl (born 1859) tried to define the concept of changes by considering the notion of time, saying, “Things are always intended *toward* something, and are always ‘about’ something,” which shifts the notion of ontology from studying “being” towards studying “becoming”.

It has been commonly acknowledged that a change happens in relation to time. However, Aristotle (384–322 BCE) in his book *Physics IV* (10) argued that since change, unlike time, occurs at different rates, it is distinct from time [HG04]. The nature of change may appear to be contradictory and a source of inconsistency, as “it requires both sameness and difference” in parts and attributes [Was06] and deals with contrary facts about the identity of things. Consider a cup of tea that changes from hot to cold as it remains on a table. The hot tea must be the same as the cold tea or else the tea does not change. The hot tea is also not exactly the same as the cold tea. More information on change, persistence, and identity can be found in Leibniz’s Law at [Was06], Theseus's paradox at [Coh04], and the heap paradox (Sorites) at [Zal05]. A classical example to demonstrate the change-driven issues of identity was described in the heap paradox. This paradox is usually presented as chains of conditions as following [Zal05]:

- *1 grain of wheat does not make a heap.*
- *If 1 grain of wheat does not make a heap then 2 grains of wheat do not.*
- *If 2 grains of wheat do not make a heap then 3 grains do not.*

……..

- *If 999,999 grains of wheat do not make a heap then 1,000,000 do not.*

1,000,000 grains of wheat do not make a heap.

Or more formally:

$$\begin{array}{lcl}
Fa_1 & & \\
\text{If } Fa_1 \text{ then } Fa_2 & & \\
\text{If } Fa_2 \text{ then } Fa_3 & \text{or} & Fa_1 \\
\ldots\ldots & & \forall n(Fa_n \rightarrow Fa_{n+1}) \\
\text{If } Fa_{i-1} \text{ then } Fa_i & & \\
\text{-------------------} & & \text{-------------------} \\
Fa_i \;\; \text{(where } i \text{ can be arbitrarily large)} & & \forall n Fa_n
\end{array}$$

Where "F" represents the predicate (e.g., "does not make a heap"), "a_n" (n is a natural number) represents a subject expression in the series with regard to which "F" is soritical (e.g., "n grain(s) of wheat"). Thus, the argument is that since one grain of wheat does not build a heap and adding one more grain does not make any difference for building a heap (for any number n, if n grains of wheat do not make a heap, n+1 grains won't either). These rules of inference are endorsed by modern and classical logic [Zal05]. The heap paradox can be applied to any situation that one can make minute changes to. Unger, in his paper entitled "I Do Not Exist" [Mac79], applied the heap paradox to himself, removing one cell at a time. This puzzle becomes very important once we try to apply meaning and semantics to the logical symbols because many frequently used words, such as few, a lot, big, small and like, as well as colors and sounds, may be used to generate a heap paradox [Wil94].

II 2.2 Identity, Change, and Time

Due to the paradoxical nature of change, change in a thing causes various problems, including the problem of the consistency of change. Some have said that the only way to make sense of change is through inconsistency [Var05]. Many philosophers believe that studying and reasoning about change only make sense when things extend through "time". This means the temporal parts of a changing "concept" can have different properties at different times [Var05]. In other words, one may think of time as another dimension along which objects extend, just as they extend across the three spatial dimensions.

For example, when we say a cup of tea is placed here but not there, by the passage of time and the changing tea temperature, we can say that the tea is cold now but it was not a couple of minutes ago, insofar as the current temporal part of the tea is cold but the previous part is not. We have got the same object (the same tea), but its temporal parts (as well as spatial parts) are not quite alike [Var05]. So for ontologies to capture the scientific picture of the real world, things should be studied in four-dimensional models [Miz04], considering time as the additional dimension to traditional three-dimensional models.

In order to talk about the identity of objects, ontologists need to distinguish between Continuants/Occurrents, Dependents/Independents, and Universals/Particulars [SWS03]. According to [SWS03], Continuants (objects) are things that continue to exist through time and their identities remain unchanged. Occurrents (processes) are time-dependent entities whose identities unfold at different points in time. The existence of a "Dependent" depends on the existence of other things (e.g., a bodily injury is dependent upon the injured organ), in contrast to an "Independent", whose existence does not necessarily depend on other things (e.g., atoms, molecules). Also, "Universals" can be considered classes or groups of things (e.g., "student") while "Particulars" are "instances" of those classes (e.g., a specific student). In Chapter III, we will consider "time" as a primary factor in our approach to analyzing changes in temporal biomedical ontologies.

In debates on distinguishing between "Dependent" and "Independent" entities in the real world, the two concepts of *Ontological Philosophy* and *Dialectic Change* attracted our attention. The concept of Ontological Philosophy [Scr99] focuses on the wholeness and unity[10] of the world and considers change as an aspect of substances in the real world. From the other side, the concept of Dialectical Change [Hol98] tries to represent a change as new forms built upon the old and by combining the new and the old without total replacement, implying both newness and continuity. In this theory, any change needs a cause and can be placed through a process. Holsti [Hol98] used the Marxist idiom, the synthesis, as a metaphor for this processes. However, unlike synthesis in Marxist vocabulary, which is defined as the process arising from the contradictions between old forms and always leads to a "higher" form, a change process can also denote reversal, corruption, or decline [Hol98]. Change also can be studied as a Transformation, which results from quantitative changes accumulated over a period of time and generates a new form out of old patterns (coexistence of both old and new) [Hol98]. It means a concept may remain structurally similar, but its semantic changes (e.g., the concept of monarchy in England has changed from ruling to symbolic) [Hol98].

II 2.3 Change and Philosophical Problems in Knowledge Representation

Hansson [Han03] described several philosophical problems in dealing with change and revision, focusing on the AGM model of belief change [AGM85]. Hansson classified these problems, which are mostly applicable in the areas of knowledge representation and semantic web, under ten categories [Han03]:

1. Can stricter cognitive limitations than finiteness be represented in an interesting way?

2. How can modal and conditional sentences be represented?

[10] "We live in exactly one world, not two or three or seventeen." [Sea95]

3. *What is the (formal and informal) relationship between the two notions of degree of belief: confidence and resistance to change?*
4. *What is the relation between vulnerability/resistance and justificatory structure?*
5. *Which is the best way to change the AGM model to achieve categorical matching?*
6. *To what extent are retrieval and change operations interchangeable?*
7. *How should ordinary, non-pure contraction be represented?*
8. *Are there atomic operations in terms of which all belief changes can be represented?*
9. *What are the roles of intermediate non-committed and intermediate inconsistent belief states?*
10. *What is the relation between decision+revision and expansion+consolidation?*

As can be seen, half of the problems are explicitly related to representation, while the rest of the problems are implicitly affected by the issues in representation. In Chapter III, we focus on the problem of representation in dynamic ontologies from two broad perspectives: how to represent a change and how to change the representation.

II 2.4 Philosophy, Linguistics, and Change

If words are not things, or maps are not the actual territory, then, obviously, the only possible link between the objective world and the linguistic world is found in structure and structure alone."

Alfred Korzybski (1879-1950)

Changes in all aspect of a language (words, syntax, grammar, meanings, and pronunciation) are constantly taking place throughout the passage of time. Sentences like "*I logged on to my account with my Blackberry and sent her an email!*" would have been incomprehensible nonsense only a few years ago. There is a famous issue in linguistics, known as the Saussurean paradox [Ferdinand de Saussure (1857-1913)], which states: "if a language is primarily an orderly system of relations, how is it that a language can change without disrupting that system?" [TM05]. In other words, "how can a language continue to be used effectively as a vehicle for expression and communication while it is in the middle of a change, or rather in the middle of a large number of changes?" [TM05]. Just imagine a court, where laws are changing during a trial; or a tennis match with frequently changing rules during a match [TM05].

In linguistics, there is still no consensus for using words like news, people, and law as plural or singular [FR98]. The answer may lies in "variation", which is "the vehicle of change" and means "all accepted forms of one word can be accepted and used side by side. When a change is in progress, the older and newer forms coexist, and almost all the users and applications are familiar with both forms, even if some people use only one or the other. Over time, the older form becomes less and less frequent, and the newer one becomes ever more frequent, until, one day, there is no one left alive still using the older form, and the change is complete" [TM05]. For example, one may choose between *telephone* and *phone*, between *gymnasium* and *gym*, between *omnibus* and *bus*, and after a while, one form is no longer used at all, as has now happened with *omnibus* [TM05]. The study of variation in language is called sociolinguistics [Cry97, Lab00].

II 2.5 Summary of Section II.2

Several sub-disciplines in artificial intelligence, software engineering, cognitive science, philosophy, and so forth have considerable overlaps in their outcomes, which should be considered for a successful ontology change management process. In summary, one can distinguish different kinds of problems related to changes in ontologies. Many of them are philosophical and linguistics problems. Inspired by the philosophical perspectives explained in this section, we ground our proposed techniques for ontology change management. One of the distinguishing features of our study is doing broad research in several interrelated domains on performing successful ontology change management.

II.3 Ontology Change Management - Requirements and Challenges

The fluttering of a butterfly's wings can effect climate changes on the other side of the planet.

Paul Erlich

II 3.1 Ontology Engineering and Maintenance

Knowledge engineering has been defined by Sowa [Sow00] as "the application of logic and ontology to the task of building computable models of some domain for some purpose". Ontology engineering, as an essential part of the knowledge engineering process, consists of ontology modeling (e.g., defining author concept descriptions, relations, and axioms), managing changes, refining the ontology, managing errors, and reusing and integrating different ontologies [Hor07]. Ontology maintenance is traditionally focused on two aspects of ontology engineering, namely ontology change management and integration in dynamic environments.

Due to the dynamic nature of biomedical knowledge-based applications, the need for change management can be seen in their entire developmental life cycles. For example, a typical clinical application must frequently deal with new information on a timely basis, such as drug-related and similar data from patients in a hospital setting, or in a biomedical research lab, where the knowledge essentially grows and changes over time. Capturing, representing, tracking, and applying the changes, along with discovering all the consequences of even small changes in such dynamic environment, are far from trivial.

II 3.2 Ontology Evolution and Change Management

Ontology change management can be studied as the process of changing an ontology in response to a set of particular requirements [FMK+08]. Considering the definition by Studer et al. [SBF98] of an ontology as "a formal, explicit specification of a shared conceptualization" in a domain of interest in the real world, researchers distinguish different rationales for changes in an ontological structure:

- Changes may happen in the formal representation (formalization) of the ontology from one version to another (e.g., from DAML+OIL[11] to OWL). These changes mostly affect the syntax of the representation of the ontological axioms, without altering the semantics or terminologies. Formalization change is the subject of "ontology translation" [DMQ05] studies.
- Specifications and granularities can be altered because of changes in the target application, changes in potential users' requirements [HS05], or changes in the original ontological structure by adding newly discovered knowledge or fixing errors [PT05] or inconsistencies [FHP+06].
- The domain of interest [SMS+03] as well as views on the domain may change [NK04].
- The conceptualization might also change if it cannot convey a shared consensus of meaning in the real world, which may happen due to changes in view of the

[11] http://www.daml.org/2001/03/daml+oil-index.html

world or in usage perspective [NK04]. In fact, the conceptualization changes as the knowledge about the domain grows [HHL99].

In distributed Semantic Web environments, where ontologies are developed based on several inter-related components [KN03] and meant to be reused as much as possible in a collaborative fashion [NCL+06], the high coupling between different ontologies can cause a domino effect (a chain reaction caused by a small initial change, which leads to a series of changes in the objects nearby (Wikipedia)) in dependent ontologies and knowledge sources. Also, reusing the ontologies gives rise to issues like ontology matching, mapping, merging, alignment, and integration [PGM99].

II 3.3 Ontology Change Management and Sub-Fields

As mentioned in the previous section, the iterative [HHL99], collaborative nature of an ontology development life cycle requires that ontologies go through one or more processes, such as matching, mapping, merging, alignment, integration, debugging, and versioning [PGM99], which often impose changes on one or more components of the ontological structure. Ontology change management consists of all activities and processes that are required for consistently maintaining an ontology in response to a particular change in the ontological structure. It may consist of several steps depending on the complexity of the ontology and its application, as well as the degree of coupling between the ontology structure and other dependent artifacts.

For example, this process has been described in six phases by [SMM+02] for iterative change management: (i) change capturing (determining the required changes), (ii) change representation (formally encoding the changes), (iii) semantics of change (analyzing the sources and effects of changes and resolving the problems caused by the changes), (iv) change implementation (applying the changes to the ontology), (v) change propagation (propagating the changes and the related consequences in the dependent artifacts), and (vi) change validation (assessing the target ontology for consistency).

The current state of ontology evolution, as well as a list of existing tools, can be found in [FMK+08] and [DM08]. Flouris et al. (2008) [FMK+08] presents a comparative survey for clarifying the borders for each of the mentioned ontology change management sub-fields. Despite their efforts, it is not always easy to draw a clear line between these fields. For example, defining where ontology mapping ends and ontology alignments start still seems far from trivial.

II 3.3.1 Ontology Mapping

Ontology mapping is defined [KS03] as "the task of relating the vocabulary of two ontologies in such a way that the mathematical structure of ontological signatures (the terminologies) and their intended interpretations, as specified by the ontological axioms, are respected". There are also less formal definitions, such as [ES04], which describes the mapping of a given ontology A to B as follows: "for each concept (node) in ontology A, we try to find a corresponding concept (node), which has the same or similar semantics, in ontology B and vice versa." The tasks of finding and measuring semantic similarities between the concepts in different granularities are the subject of several research projects (e.g., in biomedical ontologies, see [CSC07]).

II 3.3.2 Ontology Matching and Alignment

Ontology matching is described as "the process of finding relationships or correspondences between entities of different ontologies" [ES07], and its result, which can be used for purposes such as ontology merging, integration, translation, and interoperability management, is called ontology alignment, which expresses "with various degrees of precision the relations between the ontologies under consideration" [ES07].

An extensive list of ongoing projects and infrastructures for ontology matching can be found at http://www.ontologymatching.org/projects.html.

II 3.3.3 Ontology Translation

Translation takes place when an ontology or its parts need to be reused with a tool or algorithm that uses a language different from that of the ontology [Cor05]. In this situation, one must deal with several mismatches in language level (differences in ontology languages, syntaxes, and logical notations) and model level (differences in the way a domain is conceptualized and interpreted) [Kle01]. Several tools and techniques, such as OntoMorph [Cha00] and ODEDialect [CG07], focus on ontology translation.

II 3.3.4 Ontology Debugging

Ontology debugging is defined as the "process of identifying and removing undesirable logical contradictions (inconsistencies/incoherencies) from an ontology" [FMK+08]. Most of the existing ontology inference engines can report errors like unsatisfiable concepts or inconsistencies in ontologies without clarifying the reason and source of these errors. A prompt and precise debugging service is a vital part of a safe and effective change management system [KPS+06]. As an example, a debugging framework for OWL-DL ontologies using the Pellet[12] [SPG+07] description logic inference engine has been described in [Kal06].

II 3.3.5 Ontology Versioning

Ontology versioning has been defined [KF01] as "the ability to handle changes in ontologies by creating and managing different variants of it." In other words, ontology versioning [HHL99, HP04] deals with "the process of managing different versions of an evolving ontology, maintaining interoperability between versions and providing transparent access to each version as required by the accessing element (data, service, application or other ontology)" [FMK+08].

II 3.3.6 Ontology Integration

Ontology integration is defined as the process of "building an ontology in one subject reusing one or more ontologies in different subjects" [PGM99]. This process is often performed by the aggregation and combination of source ontologies, and usually involves changes, such as extension, specialization, or adaptation [PM01]. To reuse ontologies in one consistent integrated structure, they need to be aligned, which means that they have to be brought into mutual agreement, and then mapped by relating similar concepts or relations from different sources to each other by an equivalence relation [Kle01].

[12] http://clarkparsia.com/pellet

The integration process is usually done in two steps: data/semantic integration and reconciliation. The data integration is comparable to data integration as studied in databases, with the one major distinction being that while in database integration it is assumed that each source is basically a logical theory with a single model, such an assumption is not made in ontology integration, where an ontology is an arbitrary logical theory that may convey several models [CGL01]. The most common issue in ontology integration is mismatching between ontologies on language and model levels [Kle01]. The language level mismatches mostly deal with problems in syntax, semantics, and expressivity of different ontology languages. To fix this problem, one usually needs to utilize some translation techniques alongside the integration method. The model level mismatches involve interpretation and conceptualization mismatches, and differences in the way the conceptualization is specified [Kle01]. The second issue is much more challenging, since many of the effecting parameters cannot be fit in a computational model. Some of the available approaches in ontology integration that also deal with problems of ontology alignment and matching are FCA-MERGE [SM01a], COMA++ [ADM+05], ILIAD [UGM07], and DINO [NLH+08].

II 3.4 Challenges for Ontology Change Management

There are major challenges in this field of research, going back to the theoretical foundations and practical implementations as categorized by [Nov07b]. Lack of appropriate formalism for representation of ontology changes, tracking and analyzing logical consequences of different changes, analyzing semantic changes and the relation between syntactic and semantic changes, and consistency management in dynamic ontologies are all issues related to the theoretical foundations. From the implementation perspective, Novacek [Nov07b] highlights some of the main issues concerned. For example, most of the few ontology change management models are analogical to schema version management and software evolution, with little focus on ontological features. Efficient implementation of the existing methodologies to explicitly address ontology evolution is still challenging. Also, one must rely on advances in other related fields (e.g., NLP techniques for automatic ontology learning from text) for dynamic knowledge acquisition in evolving ontologies. In addition, any successful approach should address the human factor as an essential part of an interactive Semantic Web environment. Some other challenges in an ontology change management process are highlighted below.

II 3.4.1 Backward and Forward Compatibility

A major process in any ontology maintenance framework is managing different versions of an ontology and checking the compatibility between them to determine if one version can be used as an alternative to other versions in a consistent way. The compatibility can be analyzed based on a set of requirements that a version of an ontology should fulfill [Ple06] with regard to backward (or downward) and forward (or upward) directions.

Backward compatibility [Kle04] checks if the newer version of an ontology uses a data source that conforms to the older version and ensures that the changes in new version do not affect the existing definitions (e.g., monotonic additions of concepts or relations [HH00]). As an example, according to [HH00], the version management service in SHOE[13] can assist agents and query systems in discovering and specifying the divergence and backward compatibility between the versions of an ontology. The forward

[13] Simple HTML Ontology Extensions (SHOE): http://www.cs.umd.edu/projects/plus/SHOE/index.html

compatibility verifies that the data source in an ontology version can be used in a newer version of the ontology and the changes in existing version do not change the validity of the future, upgraded version (e.g., deletion of a standalone concept). Determining forward compatibility is not always possible, since foreseeing the complexity, semantic richness, users, and usages of the future versions might not be feasible.

II 3.4.2 Traceability

Traceability is another critical task in change management, which provides transparent access to different versions of an evolving ontology. Traceability also aids in understanding the impact of a change, recognizing a change and alerting upon occurrence, improving the visibility, reliability, auditablity, and verifiability of the system, propagating a change [SDK+03], and reproducing results for (or undoing effects of) a particular type of change. Advances in impact analysis gained by traceability facilitate predictability in the post-change analysis stage in an ontology maintenance framework.

II 3.4.3 Querying Over Multiple Versions

Queries over different versions of an ontology may return different results, which in many cases may not be desirable. Consider a court trial, for example: how could we try a case in court if the laws were constantly changing during the trial? For successful querying over evolving ontologies with multiple versions, we need an approach for unifying and filtering all data in different versions. In database schema management, one solution for this problem is following the "view approach" by creating a view per version that maps each version into a universal document, which can hold all the information from every version [BÖS+05]. However, the problem seems much more complicated in ontology evolution.

II 3.4.4 Metamorphosis

Metamorphosis is defined as a marked change in appearance, character, condition, or function[14], which often appears as a sort of radical temporal discontinuity in one species. For example, a caterpillar becomes a moth or a butterfly, or a tadpole becomes amphibious. In ontology engineering, dealing with metamorphosis gives rise to many issues relating to conceptual identity (recall Leibniz's law, Theseus's paradox, and Sorites in Section II.2).

II 3.4.5 Controlling Belief Revisions

The concept of belief revision [Dra97] refers to consistently changing a belief during the revision of a knowledge base in response to a change [KL07]. From the logical point of view (i.e., from the DL perspective [QY08]), this problem deals with detecting and resolving logical inconsistencies caused by a revision and providing necessary justification to maintain the "truth" [BH90]. According to [AGM85], belief changes can be found in three forms: (i) expansion (adding a fact and its logical consequences), (ii) contraction (deleting a fact, which may involve the elimination of other dependent elements), and (iii) revision (consistently adding a new fact and its logical consequences, and retracting the knowledge base in case of an inconsistency). Control over belief

[14] Online Free Dictionary: http://www.thefreedictionary.com/metamorphosis

revision guarantees that new information gained through the learning process does not contradict the conceptualizations and specifications associated with the existing knowledge base system [Gär90].

II 3.4.6 Structural and Semantic Dependency

Due to the interoperability of different ontologies and their versions and the tight coupling between their elements, there are usually dependencies (implicit or explicit) between the effects of a change. This issue is most challenging in the change propagation stage, and requires some synchronization processes [Oli00] to ensure that the chain of changes is maintained consistently and coherently.

Employing modularization techniques [WHB07] in ontology engineering aims to address some of the challenges related to unintended and unexpected domino effects due to dependencies between ontological elements. Therefore, analyzing the dependency graphs, which represent the dependencies between ontological elements, is a starting point in managing updates and revisions in modular ontologies [SK03].

II 3.5 Summary of Section II.3

Ontology maintenance and change management consists of several interrelated tasks for refining ontologies, managing the errors and inconsistencies, (partially) reusing ontologies, and performing mapping, translation, merging, matching, alignment, and integration on different ontologies. These tasks are extremely challenging and interconnected, and need comprehensive methods along with logics, formalisms, tools, and infrastructure support in a collaborative environment. We will look at some of the existing tools, methodologies, and practical solutions for ontology maintenance, as well as state of the art of change management in some popular biomedical ontologies in Section II.6.

II.4 Human Factors in Change Management Process

Metathesiophobia: The persistent, abnormal, and unwarranted fear of change. Symptoms usually include shortness of breath, rapid breathing, sweating, nausea, irregular heartbeat, and overall feelings of dread.

Phobia list, Wikipedia

II 4.1 Human Factors in Dynamic e-health Environments

During the last two decades, many advances in healthcare have required the development of artificial intelligence (AI) techniques in the biomedical domain. Several integrated health knowledge management systems, such as Acute Care Systems, Medical Decision Support Systems, Educational Systems, Quality Assurance and Administration, Laboratory Systems, Medical Imaging, and so forth, are recruiting large knowledge-bases and ontologies as their backbone to facilitate human-machine communication and capture knowledge from the domain of interest. Biomedical knowledge based systems, especially the ones dealing with human health, require fast responses and real-time decision-making. Human intervention can be seen in the whole life cycle of biomedical systems. In fact, relations between the system maintainers, patients, nurses, lab technicians, health insurers, and physicians are crucial in such systems, and should be encouraged when necessary. From the other side, many of the editorial decisions on performing a change in a system need to be made by humans. Man-machine interaction problems are not purely computational and need a deep understanding of human behavior.

As mentioned in Chapter I dealing with change is mainly a social, linguistic, and philosophical problem, rather than a computational one. A key issue in managing current dynamic biomedical systems relates to users' behavior and the cultural and disciplinary assumptions [For98], which can determine the success or failure of a system. The change management phase in current systems is largely addressed implicitly, and followed with human supervision and intervention. The human contribution improves rationality and plays an important role in controlling the quality of the results. However, there are several applications where human intervention is difficult, impossible, or simply undesirable [FPA06] (e.g., due to security issues). Also, differences in background knowledge, views, or preferences are other obstacles for consensus between people. In this sense, a result might not be accurate or reproducible. In addition, the system's outcome might be highly dependent on human behavior, which makes it difficult for evaluation in terms of efficiency or correctness.

The existing well-known biomedical systems and digital libraries usually affect large and heterogeneous groups of people, with different levels of background knowledge and dissimilar interests. Therefore, an efficient user-centered approach, along with psychological and organizational proficiency should be taken to reduce the behavioral side-effects and successfully manage changes in healthcare applications. An ideal e-health system should be able to automatically coordinate human factors, processes, tools and knowledge-bases while coping with different changes. There are some issues that affect the successful implementation of such infrastructures. In this section, we review and survey the potential issues related to the human factor in an integrated dynamic biomedical system composed of several interrelated knowledge bases, and bio-ontologies by looking at different theories in social science, psychology, and cognitive science, and we address the following issues:

- The organizational and social impacts of human-driven changes in e-health systems;
- Different sources of change;
- Human errors due to change and alteration;
- Responding to change in a dynamic e-health environment;
- Safety;
- User interface issues;

Lorenzi and Riley [LR00] presented an overview of change management efforts in information systems showing the roles of people and the organizational issues (i.e., the interruption of a known routine) that were counterproductive to the implementation and management of major information systems. Based on their research, the main reasons for system failure can be categorized under miscommunication, cultural barriers, underestimation of complexity, inadequate or low-quality training, lack of organizational change management strategies, and weak leadership. Considering the dynamic nature of current knowledge bases, which need real-time decision-making and proper action from human agents, the concept of change and the ability to cope with various alterations play important roles in biomedical knowledge bases. Lewin [Lew47], with his social psychology perspective, focused on the motivations for an individual's behavior. He believed that psychological needs in humans cause tension until they are fulfilled. Lewin indicated three major conflict situations: the choice between two positive goals of equal strength, two equally negative goals, or opposing positive and negative forces of different strengths. Lewin's field theory, commonly used in healthcare systems, allows one to identify different types of conflict situations and to analyze the effect of a change in a knowledge-based environment [LR03.a].

II 4.2 Types of User-Driven Changes

Watzlawick et al. [WWF74, LR00] used two theories to explain first-order and second-order changes, namely the theory of groups and the theory of logical types, from philosophy and logic. A first-order change (improving a system) is defined as the logical extension and incremental improvements of past and current practices in a given system, leaving the system's core belief relatively unchanged (Examples include recovery from system failure, and generating new reports). If a system itself is changed, then a second-order change happens (deep alteration in a system). This change usually "involves a redefinition or re-conceptualization of the ideas, tasks, domains, or roles in an organization" (i.e. the change from paper-based medical records to electronic medical records in biomedicine) [LR00].

For any alteration in a system, users, designers and developers can play various roles, which will influence their conceptualization about the change and their reaction to it [LR00]. So, in making decisions and taking action within dynamic biomedical systems, the users' behavioral aspects associated to each role should be controlled.

II 4.3 Human Error in Clinical Systems and Change Management

Studies [LR00] on people working with health-related systems imply that due to high stress and pressure in the field they are relatively more resistant to being confronted with changes. Changes can potentially increase the chance of errors in a system by routine disruption. One factor urging system change is the need to deal with human errors, present in all stages of a system's life cycle. Human error should be considered in clinical application development's life cycle, along with many other aspects of design. Studying human error provides valuable information for analyzing human behavior and reveals user requirements and misunderstandings. Human error is defined by Barfield [Bar93] as an error caused in some way by the user of the system, in contrast to a system error, where there is a physical fault in the system. Based on the user's mental model, he grouped the errors into two categories: errors of action (error in the translation between a user's intention and their action) and errors of intention (the user doing the wrong thing on purpose). This classification is comparable with Norman's categorization of errors [Nor88] into mistakes and slips: if a person has intent to act that is inappropriate, it is a mistake; if the action was not what was intended, it is a slip. In order to deal with human error, Norman highlighted the needs for better consistency in describing the errors and better feedback for capturing and reporting them [LR94]. In dynamic environments with several external and environmental parameters such as evolving e-health systems, the rates of unintentional errors can increase greatly. Bés in [Bés97] and Decortis in [Dec93] have worked on the effects of temporal characteristics on users' activities in dynamic environments. Decortis stated that temporal errors can originate from incorrect estimates about the sequence or duration of actions and/or failure in choosing the right time to act, in anticipation of an event or in synchronization of collective actions [Dec93]. In addition, De Keyser [Dek95] identified other sources of temporal errors, such as the absence of high-quality indicators to highlight the change, the presence of micro-changes too short to be received, and the existence of distracters capturing the users' attention

[Bés99]. [HL02] made the distinction between two methods for change management: the technical method that can be understood and addressed with available knowledge (mostly used for managing first-order change) and the adaptive method that is beyond the existing and available techniques of operation. Several efforts such as [For98, LR94, LR00, LR03.a] have been made for applying knowledge of human and organizational behaviors derived from psychology, sociology and cognitive science to the implementation and management of healthcare systems.

II 4.4 Safety

The six principles were defined by the Committee on Quality of Healthcare in America [Com01], to be followed by any e-health knowledge-based system to provide high-quality services, with focus on safe, effective, patient-centered, timely, efficient, equitable environments. User and patient safety is a challenging issue that needs to be addressed with proper real-time control and feedback mechanisms in the systems. User interfaces can play a vital role in this case by providing appropriate forms of messages and warnings in a timely manner. The number of potentially hazardous errors can be reduced by employing intelligent safety devices, accurate alerts, and effective user-friendly interfaces. To cope with changes in the constantly evolving knowledge-based e-health environments, one must have a formal model of human reactions to change, enabling cognitive error analysis. Beitler et al [BFK+95] designed an interface that provides a virtually simulated multimodal user control environment, based on the knowledge of a reactive planner to allow "autonomous planning as well as planning through human-machine interaction". The system acts like a human agent and can be used in situations unsafe for people. This approach is especially useful in assisting people to perform repetitive tasks, which potentially increase the chance of error for humans.

II 4.5 Trust and Security Issues

Kini et al. [KC98] observed various aspects of human trust in computer-dependent systems, according to personality theory, sociology, economics, and social psychology. They defined trust as "a belief that is influenced by the individual's opinion about certain critical system features". Their study relies only on human as the "truster" (instead of system) and does not support the problem of trust between humans and processes involved in knowledge-based interactions. Gambetta [Gam00] defined trust as an estimation that can be determined by the probability of an action being successfully performed. Jøsang et al. [JIB07] look at trust in a user-centered framework where '*one party is willing to depend on something or somebody in a given situation with a feeling of a relative security, even though negative consequences are possible*'. In this sense, human-agent interactions play important roles in the security process, which usually includes authentication, authorization, and confidentiality. Relying only on human factors in the security process, especially in complex health systems, may lead to unpredictable, inaccurate, and inconsistent results that often may not be reproducible. So, in modern e-health knowledge bases, security management must be carried out automatically, with minimal human intervention.

II 4.6 User Interface Issues

Since biomedical knowledge bases and applications are most often used by lab technicians, nurses, and physicians, a formal logical language is not well-suited for representing the interactions. Therefore, special attention is given to the design of the operational user interface, based on natural language processing and intuitive graphical representations. Currently available tools do not provide complete support for dealing with the complexity of evolving medical systems, which go beyond the capabilities of existing user interfaces. One method for dealing with the representation of changes in user interfaces is to employ ontologies in capturing the knowledge about evolving concepts. In this way, changes to the user interface can be made by changing the underlying ontology. [TMM+96] and [GMZ99] undertook two efforts devoted to modeling user interface for biomedical applications. Pohl et al. [PRW07], Leitner et al. [LAH07], and Carrigan et al. [CGC+07] also recently demonstrated their advances in the usability of user interfaces of available information systems in medicine and healthcare. In general, a user interface based on human factors is a key to the acceptance of a system [Nie93] in medicine. In creating a graphical user interface (GUI), the level of expertise and the operational habits of the medical staff should be considered.

Hartson et al. [HB93] specified behavioral and construction domains for implementing a user interface. The behavioral domain includes the design and development of the interactive part of an interface, and the construction domain includes the development of the graphical environment. The development process of a usable GUI is not possible without active participation of physicians, psychologists, and other end-users of an e-health system. It also requires the consideration of important human factors, such as intuitiveness, functionality, accessibility, flexibility, and adaptability of the user interface. However, design criteria based on human factors do not automatically guarantee a solid, usable interface [TMM+96]. As the GUI development for dynamic environments is always an iterative process [HB93], it requires the occasional modification of initial system specifications based on new requirements or newly obtained knowledge.

II 4.7 Participative Change Management

A dynamic health knowledge-base usually deals with spatial and temporal data, metadata, documents, and data warehouses while working in an integrated web-based system that includes databases, ontologies, and software agents. To overcome some of the existing challenges in current knowledge-based systems, researchers try to design systems based on human behavior and needs [BT94, DH96].

In our approach we emphasize on the role of human factor in maintaining changes in a consistent way. For detecting any behavioral change, we first need to specify behavioral patterns to capture current behavior, the behavior upon change, and the advantageous replaced behaviors. For this purpose, we introduce our agent-assisted framework (RLR), meant to assist humans in performing changes (semi)automatically. Figure 2.2 demonstrates the interactions between human user/administrator, intelligent agents, environmental parameters and existing knowledge bases involved in a decision making process for performing a change in our proposed RLR framework.

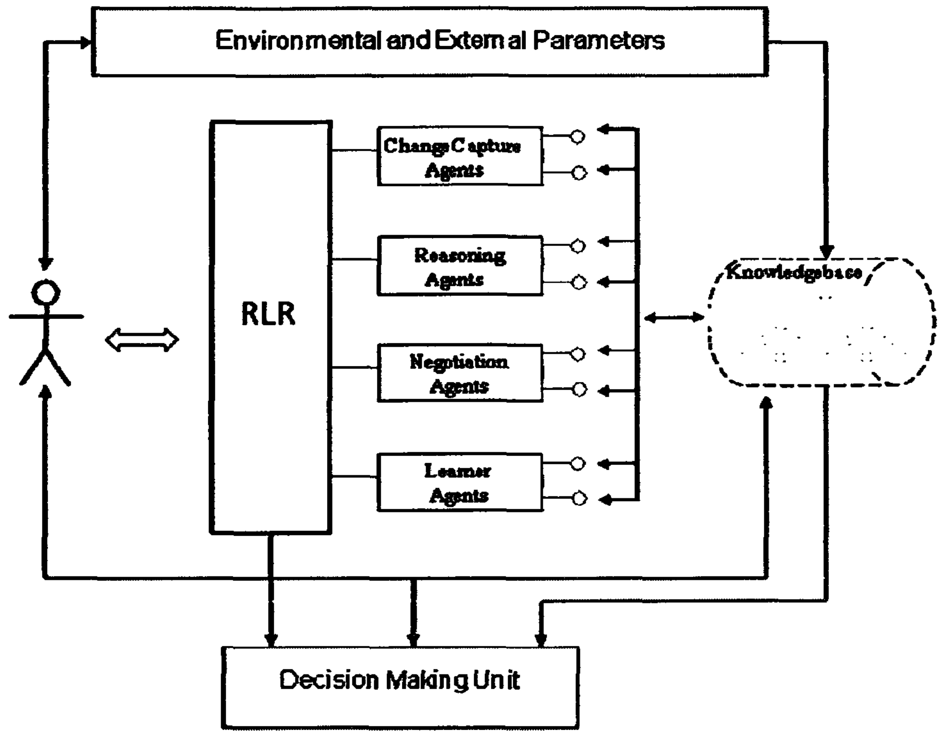


Fig. 2.2. The Decision making mechanism for user-centric change management.

The details on this framework and associated agents will be explained in Chapter III.

II 4.8 Summary of Section II.4

Hidetora: I am lost...
Kyoami: Such is the human condition.

Ran (1985) by Akira Kurosawa

A large body of literature exists on the importance of human-machine interactions in various domains of interest. Life science and biomedical fields are challenging domains in knowledge management. Biomedical data are highly dynamic, and the large biomedical knowledge sources contain complex interrelated elements, with various levels of interpretation. Considering the dynamic nature of current volatile digital libraries, which need real-time decision-making and proper action from human agents, the concept of change and the ability to cope with various alterations play important roles in biomedical knowledge bases.

In this section, we reviewed some of the issues relating to human intervention in maintaining biomedical systems and knowledge bases. Later we will investigate the potential of some advanced formalisms in the Semantic Web context (such as using intelligent agents to assist computational inferencing) to assist the human user in decision-making and dealing with changes. We will return to the concept of participative change management as the collaboration between human and software agents for (semi-) automatic ontology evolution in Chapter III, where we will see how an interactive diagrammatic formalism facilitates human-computer interaction, reasoning and problem solving.

II.5 Change Management in Database and Software Engineering

Management is doing things right; leadership is doing the right things.

Peter F. Drucker (b. 1909)

II 5.1 Database Schema Evolution

Since databases are characterized as one of the fundamental components in many software applications, experts in this field are faced with two issues: schema evolution and versioning. Software applications operate in a world of constant change. The changes particularly apply to the underlying schema, as it needs to be adapted to ever-changing requirements [BSH+06]. Dynamic schema evolution (DSE) is defined as the ability of the database schema to handle changes to its structure without losing the existing data and without interrupting the regular operations of the database [RS03]. While most of the popular database systems maintain a few simple change operations (e.g., adding/deleting) automatically, handling complex changes needs a precise, future-oriented strategy. A successful schema evolution process includes the study of the sources of change and the analysis of effects of different changes on the data and schema for coherent management of different versions [NK04].

The issues and potential of schema evolution are well studied and a large body of literature exists on the topic (for instance, see the surveys in [Rod95], and [RS03]). Generally, schema evolution consists of three [RS03] interrelated activities: core schema evolution (detecting and applying the changes while keeping the schema consistent), version management, and application management (keeping the applications that benefit from the database in working order). Some systems focus only on maintaining multiple versions [MS92, CLR04], while others consider all three aspects in their design model.

A comprehensive summary of the different research activities on schema evolution can be found in [RS03]. Most efforts on this topic have been focused on studying changes in single stand-alone databases, and evolution in distributed, heterogeneous sets of databases has not received enough attention. Another challenging problem in this domain relates to database integration issues, particularly semantic integration. One of the common operations during database schema evolution is the integration process, defined as "merging a set of given schemas into a single global schema" [DH05], which is usually performed in two phases: data and semantic integration. A brief survey on semantic integration research in the database community can be found in [DH05]. Comparing different types of database schemas, the XML databases, considering the semi-structured characteristics of XML, allow maximum flexibility in coping with schema changes and extensions[15] by enabling loose coupling through schema variation and evolution [BÖS+05].

II 5.2 Database Evolution vs. Ontology Evolution

Despite important differences between schema evolution and ontology evolution stemming from different usage paradigms, the presence of explicit semantics and different knowledge models [NK04], there are also similarities that allow some of the studied techniques to be reused for the ontology evolution process.

[15] The extendibility feature refers to the term "extensible" in E*X*tensible *M*arkup *L*anguage (*XML*).

Generally speaking, the content, structure, usage, and underlined semantics of ontologies are usually more complex than that of database schemas, and the set of potential alterations for ontologies is much more diverse than the possible set of changes in database schemas [BKK+87, Kle04]. In addition, the distinction between schema versioning and schema evolution, as described in [Rod95], is not fully applicable to ontologies because it is often far from trivial to find and capture similarities and differences between various ontology versions. Also, in ontologies, compatibilities between different versions are defined not only in terms of preservation of instance data (as it is with databases), but also in terms of preservation of the conceptual and ontological structure [NK04]. Conceptualization changes in ontologies, caused by alterations in perceived knowledge from the real world, are comparable [NK04] to changes in database schemas caused by changes in the real world [VH91].

II 5.3 Software Evolution and Change Management

A software application is continuously evolving to meet frequently changing requirements. Software maintenance and change management are crucial tasks in the software development life cycle and often take place after the application has released its first version. Improving the quality of the maintenance process reduces the associated costs. Software maintenance encompasses the contributions of human factors—for planning and scheduling—along with algorithms, heuristics, and formal methods to support the evolution process, while considering correctness to be the main concern [HKL05]. Software change management is a vital step in project management, which aims to maintain the reliability of the software products during their entire life cycle by deciding which changes to allow, support, or prevent, based on project goals, schedule, and budget [PCC+93]. Software change management processes have been traditionally studied under two general tasks, namely software maintenance [IEEE98, BBE91] and software configuration management (SCM) (i.e., handling changes during the software's life cycle) [Pre01, SN01].

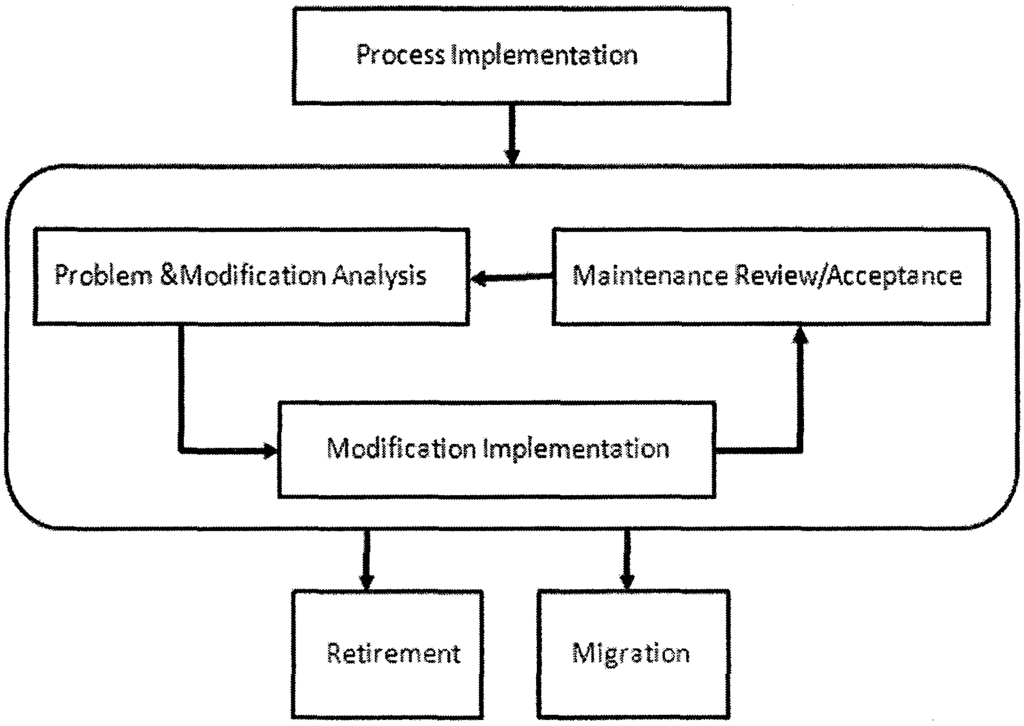

Process Implementation
Problem &Modification Analysis
Maintenance Review/Acceptance
Modification Implementation
Retirement
Migration

Fig. 2.3. ISO/IEC Maintenance Process Activities (adapted from [SN01]).

The maintenance process activities developed by ISO/IEC 14764 are illustrated in Figure 2.3 [SN01]. Each activity consists of several sub-actions. For example, Problem and Modification tasks can be broken down into these steps: performing initial analysis, verifying the problem, developing options for implementing the modification, documenting the results, and obtaining approval for the modification option. As another example, Software Retirement tasks include developing a retirement plan, notifying users of retirement plans, conducting parallel operations, notifying users that retirement has started, and ensuring that old data is accessible [SN01].

Olsen [Ols93] proposed a model for software change management based on considering the entire development process to be, metaphorically, "a dynamically overloaded queue, which can be described mathematically." In fact, Olson's model (Figure 2.4) is an abstraction that encompasses all activities performed by the software developer (i.e., enhancements like adding new features, revisions due to bug reports, filling out forms, etc.) as changes. Therefore, the model can be used for both software development and maintenance. Based on this model, change requests come from users, stakeholders, change managers, and test units in the forms of suggestions.

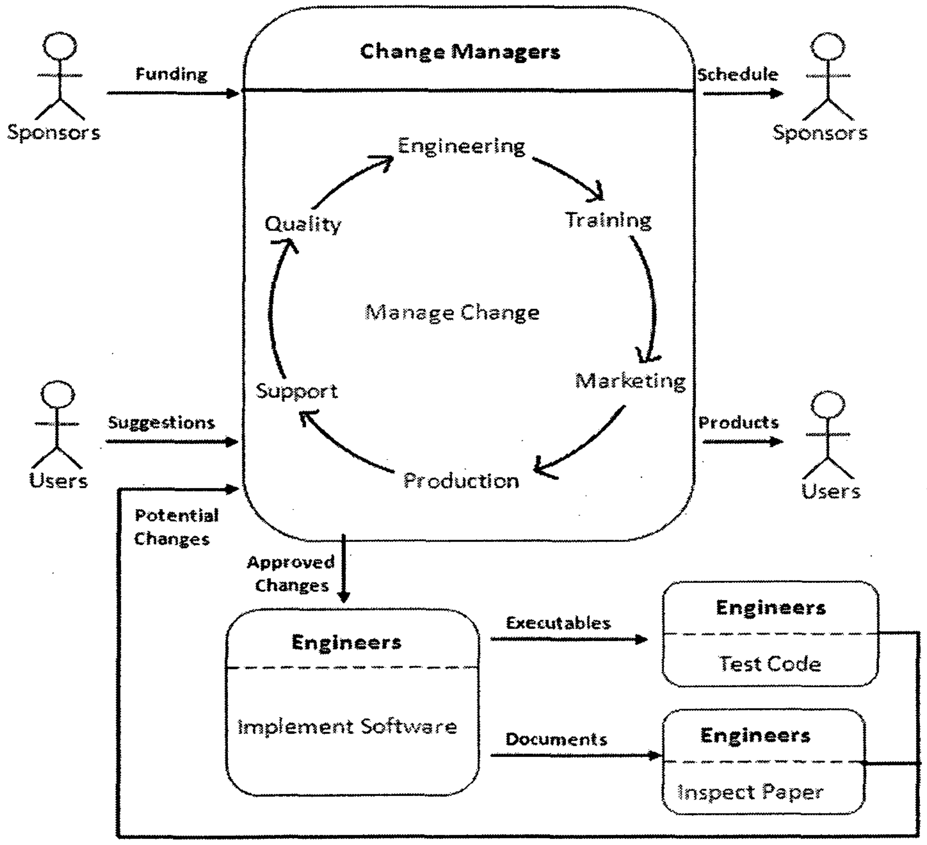


Fig. 2.4. Olson's proposed model for software change management (adapted from [Ols93]).

Changes in Olsen's model have been defined in a highly abstract manner, which makes it difficult to distinguish between different types of changes [Mäk00]. Lehman in [LRW+97] formulated the eight Laws for Software Evolution. In Lehman's context, software evolution is managed in a feedback-driven and controlled maintenance process [Leh96]. He believed that the functionality and quality of software applications need to be constantly improved over their lifetimes to meet users' needs and satisfaction [Leh96]. Lieberherr and Xiao [LX93] gave the motivation for using an ontological structure for managing changes in software systems by proposing propagation patterns—a set of programs wherein all class members are connected through part-of and inheritance relationships—for interpreting object-oriented applications at a higher level of abstraction.

The so-called AGILE software development methodologies [ASR+02], are another effort for developing software with futuristic perspective. Some of the main principles[16] behind an agile method are: (i) Incremental development (iterative, minimal planning, small releases in fast intervals); (ii) Cooperative and negotiative framework (strong collaboration and communication between designers, developers, customers and end-users along with contract negotiation); (iii) Accessible (well-documented, available, easy to learn and change); (iv) adaptive (can accommodate scheduled or non-scheduled modifications and changing circumstances); and (v) simplicity.

[16] Agile Manifesto principles: http://www.agilemanifesto.org/principles.html

II 5.4 An Ontology Driven Software Application

In our research, we have focused on ontologies not in isolation but as artifacts that are part of a software system and used to specify, model, or document these systems. Currently, there are some ongoing efforts in applying ontological concepts and concept-centric [HKL05] approaches to support software maintenance and evolution [DD04]. Emphasis on object-oriented and component-based architectures in software engineering allows for modularization, encapsulation, and distribution of units of program code [OHE96]. A vast amount of research [XS04, ACC01, XS06] in software evolution has focused on object-oriented systems. Using ontologies that aim to provide a common vocabulary to represent useful knowledge for software developers is a new trend to manage the inherent complexity of large software systems. Ontologies define a common shared understanding about a software application domain and associated tasks, and provide an underlying discipline of modeling software applications by defining concepts and properties. They can describe software architectures and requirements, which are difficult to model with object-oriented languages [DD04]. Ontologies are also useful in software applications for describing the semantics of programming interfaces, providing a structure to organize knowledge, reducing development effort for generic tools, improving the data and tool integration, facilitating requirement elicitation by providing a common vocabulary, reusing organizational knowledge [SVS04], and capturing behavioral knowledge [DD04]. In addition, ontological commitment in software plays an important role in increasing the accessibility, maintainability, integrity, and transparency of application software based on the ontologies [Gua98]. An ontology-driven object-oriented application, in our context, is defined as an architecture created from a shared domain model that includes several interrelated knowledge sources, which are connected with some object-oriented components for user interface and control components [KOT+06]. Due to the reusability of ontologies, the overall cost and effort for creating and maintaining ontology-driven applications will be reduced. Thus, consistently modifying and adjusting the underlined ontologies in response to changing data or requirements play significant roles in the maintenance of the knowledge-based systems.

II 5.5 Challenges in Software Change Management and Schema Evolution

Several challenges in software evolution and change management have been addressed in [MWD+05], including the needs for improved software quality to deal with software aging, common software evolution platforms, techniques to support higher levels of abstraction for supporting co-evolution between different representations of software artifacts, new theories, mathematical models, and formalisms for representing software evolution, a formal programming language to explicitly support software evolution, support for multi-language systems, evolution benchmarks, increasing managerial awareness, improving versioning systems, advanced predictive models, more comparative studies and empirical research, runtime evolution (maintaining evolution in continuously running systems), and advances in accessing, retrieving, integrating, and analyzing editorial data from various sources (i.e., historical data in change logs, bug reports, change requests, source code, versioning repositories, execution traces, error logs, documentation, and so on) [MWD+05].

These challenges are often interrelated and sometimes more than one problem can be addressed with the same proposed solution. For example, employing language independent methods for software change management can deal with several problems, including supporting model evolution and supporting multilingual systems. As another example, studies on evolution-supporting tools contribute to answering challenges related to empirical researches and theory of software evolution [MWD+05].

II 5.6 Summary of Section II.5

Despite many differences between ontology, database and object-oriented modeling [IBM], in some sense, an ontology 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, so we can benefit from the research in database schema evolution and software change management for managing changes in ontologies.

II.6 State of the Art and Related Works

"Criticism is an indirect form of self-boasting"

Emmet Fox (1886-1951)

Based on our recent literature review for ontology evolution, changes are being studied on three different levels: the domain, the specification, and the conceptualization [KF01]. The problems in the first level are partially similar to database schema evolution [VH91], and the second level mostly involves conversion and translation (of both syntax and semantics) of different ontology representation languages [CG00], but there is no clear detailed analysis of the effect of specific changes in conceptualization on the interpretation of data in the ontology evolution process [KF01]. This issue might lead to data and semantic inconsistencies. In our research, we have studied different editorial procedures for change management in existing biomedical ontologies, along with available tools and techniques.

II 6.1 Biomedical Ontologies and the Editorial Procedure – State of the Art

There are currently a growing number of ontologies and controlled vocabularies in various areas of life sciences. In this section, we review the state of the art of change management in some available bio-ontologies. It is not a surprise that many of them do not sufficiently meet the requirements to be considered a formal ontology [Gua95]. Most ontologies in the biomedical domain are recognized to be acutely defective from both terminological and ontological perspectives [KS03a, Smi03, KSS04, GSG04, CSK+04, SR04, CS06, SC06, Smi06]. A list of open-source ontologies used in life sciences can be found on the Open Biological Ontologies (OBO) website[17]. Many of the available ontologies are still under active development, revision and improvement, and are subject to frequent changes. The following ontologies and controlled vocabularies have been selected for a study of their change management mechanism based on several criteria, such as availability, popularity, and complexity of and accessibility to the source and documentation. The Gene Ontology (GO) [ABB+06] is a community standard and the Unified Medical Language System (UMLS) [HLS+98] is quite popular, with its rich collection of biomedical terminologies. Clinical Terms Version 2 [Cim96a, BR99] deals with actual patient care records and the Generalized Architecture for Languages, Encyclopedia and Nomenclatures in medicine (GALEN) [Bec] which is a formal description logic based ontology. We also look at HL7 [HLR], FMA [RM03], the NCI thesaurus (NCIT) [SCH+07], SNOMED [SCC97] and Terminologia Anatomica (TA) [Whi99] to see different examples of potential changes.

II 6.1.1 The Gene Ontology (GO)

The Gene Ontology (GO) is a collaborative project [ABB+06] that intends to provide a controlled vocabulary to describe gene and gene product attributes in existing organisms based on their associated biological processes, cellular components and molecular functions. The Gene Ontology has been modeled and implemented based on three distinct ontologies, represented as directed acyclic graphs (DAGs) or networks consisting of a number of terms, represented by nodes within the graph, connected by relationships that are represented by edges [LSB+03]. The current GO term count as of April 27, 2010 at

[17] http://obo.sourceforge.net/

14:00 (PST)[18] is 30350 terms with 1434 obsolete terms. The GO consortium makes cross-links between the ontologies and the genes and gene products in the collaborating databases [Skl00]. The Gene Ontology is currently available in Flat File, FASTA, MySQL, RDF-XML, OBO-XML and OWL formats. Members of the consortium contribute to updates and revisions of the GO. Changes in GO occur on a daily basis and a new version of GO is published monthly. As GO becomes larger and complexity arises, it also becomes more difficult to control and maintain. To ensure consistency of the modified ontology, all changes are coordinated by a few biologists in the GO editorial office staff, who have write access to the Concurrent Versions System (CVS) [Ced] repository in which GO files are maintained. The users can make requests for modifications through an online system that tracks the suggestions and manages the change requests. All tracking information about requests and changes are archived and several curator interest groups have been established with associated actively archived mailing lists [Har05]. The GO editorial staff notifies others of the changes via monthly reports[19] to the users (by email), or at the GO site. Different sources of suggested changes in GO, as described by [Har05], are advances in biology that alter the knowledge of gene and protein roles in cells; joining new groups that require new terms and relations; fixing errors; completing unfinished parts of the ontology; updating legacy terms and improving the formal representation of the ontology by identifying missing or misplaced relationships and terms. One of the problems in Gene Ontology maintenance is related to the versioning tool. CVS repositories, which currently handle versioning in GO, work based on syntactic differences between ontologies. For instance, CVS is not able to

[18] http://www.geneontology.org/GO.download s.shtml
[19] http://www.geneontology.org/MonthlyReports

differentiate class versions, being able only to differentiate text/file differences [VEK+05]. The research on conceptualization change over time [VEK+05] is still promising. The following statistics presented in [HKR08] show the average number of added/deleted/obsolete changed concepts per month in the period from May 2004 to Feb 2008.

Ontology	Addition	Deletion	Obsolete
GeneOntology	200	12	4
– Biological Process	146	7	2
– Molecular Function	36	3	2
– Cellular Components	18	2	0

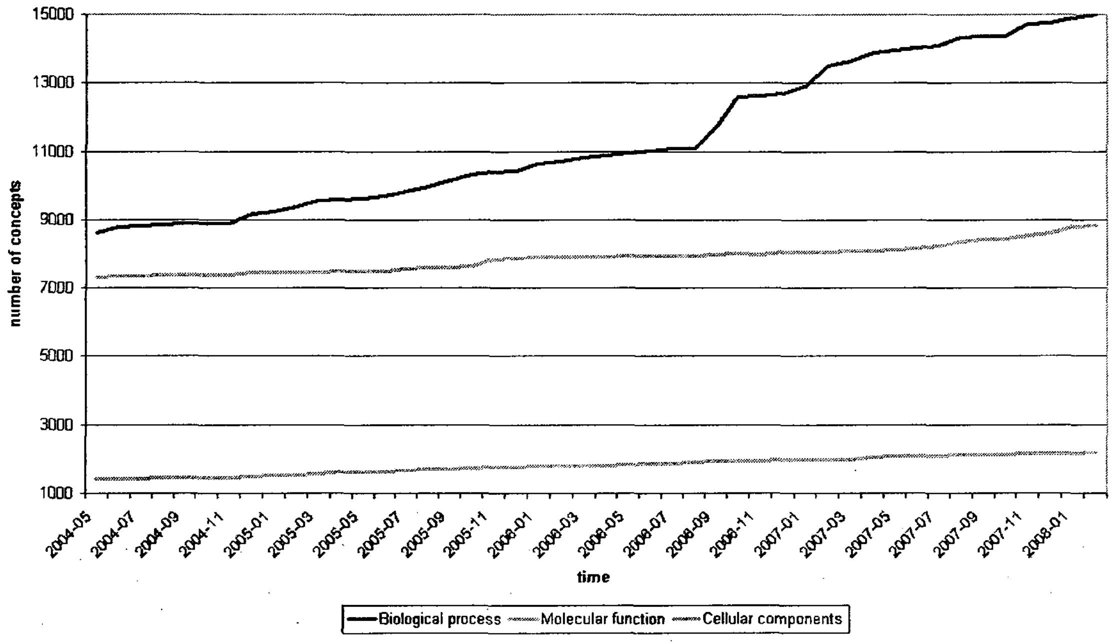


Fig. 2.5. Evolution chart in GO Ontology (Source: [DGL08]).

Also, some information about the rate of change in each one of the three sub-ontologies of GO has been provided by [HKR08] in the same period, and through 44 versions.

Ontology	|C|(start)[20]	|C|(latest)	grow
GeneOntology	17368	25995	1.50
– Biological Process	8625	15001	1.74
– Molecular Function	7336	8818	1.20
– Cellular Components	1407	2176	1.55

II 6.1.2 UMLS Semantic Network

The Unified Medical Language System (UMLS) [MN95] is a composite of about 100 source vocabularies that contain 870,853 concepts and 2.27 million terms [UML08]. It was created by the National Library of Medicine (NLM) to facilitate the development of computer systems that behave as if they "understand" the meaning of the biomedicine/health language. To that end, the NLM produces and distributes the UMLS knowledge sources (databases) and associated software tools (programs) to system developers for use in informatics research and in building or enhancing electronic information systems that create, process, retrieve, integrate, and aggregate biomedical/health data and information. The UMLS Knowledge Sources are multi-purpose, and can utilize a variety of data and information, such as patient records, scientific literature, guidelines and public health data [UML08]. Due to the popularity and multi-purpose nature of the UMLS, it seems to be a perfect candidate to study change management. The UMLS Semantic Network covers different levels of granularities, which have a key effect on interpreting the meaning that has been assigned to the Metathesaurus concepts [FSU06]. Changes in the UMLS are usually recommended by

[20] |C|(start) and |C|(end) are respectively indicating the number of concepts in first and last versions ; and "grow" denotes the ratio between them [HKR08].

the UMLS contractors and others who have experimented with the previous versions of the ontology. UMLS terms that share the same conceptual meaning are linked by a concept unique identifier (CUI) [COS+98]. Two files called DELETED.CUI, which lists deleted concepts, and MERGED.CUI, which lists all pairs of CUIs that were merged, are associated with each new release of the UMLS [OET+96]. These files help users to determine whether a CUI that is no longer present in the new version was removed due to a deletion of the concept, or due to a merger of the concept with another concept [OSS+99].

II 6.1.3 Clinical terms version 3 (The Read Codes)

The Clinical Terms Version 3 (CTV3)[21] [OPR95, NHS00a] or Read Codes are a set of coded terms arranged in a hierarchical structure for use in clinical practice, with such applications as viewing a patient's record from different perspectives (e.g., clinical audit, producing reports, meeting central returns, research, etc.). The CTV3 classifies chemicals by their name, i.e., alphabetically. The first version of Read Codes (CTV1) was initially developed to provide a terminology for describing relevant clinical summaries and administrative data for general practice. It is known as the 4-Byte Set since each code is four characters long. In the next version (CTV2), the codes were subsequently adapted for use in hospitals, and were extended to allow more detail. To hold more detailed information, a supplementary alphanumeric character was included in the Read Codes (5-Byte Sets) [NHS00b]. CTV2 uses the code to specify a class and its unique place within the taxonomy, which has a limited, fixed number of levels. The CTV3, with its flexible structure unlike the previous versions, allows more changes in terminology [JMY04].

[21] http://www.nhsia.nhs.uk/terms/pages/

The Read Codes have been changed in each version (based on strict protocol under central control of NHS) by adding terms and codes to fix the errors and reflect the newly discovered knowledge (mostly to enrich the descriptions). Further alterations include changes to qualifiers and atoms (semantic definitions), the hierarchical structure and the mapping files [NHS00a]. CTV1 and CTV2 changed relatively little between releases, due to their rigid file structure that was limited to five levels of offspring, and about 60 siblings. The CTV3 "Description Change File" (DCF) [NHS00a] shares the entire change management procedure between "terminology providers" and "terminology users" (i.e., clinicians). The DCF starts by recommending a new code for any terminology discovered to be incorrectly classified and suggesting that the user replace it. The process continues by labeling the obsolete concepts as "extinct". An example from [NHS00a] describes the deletion of the relation between the terms 'Cardiac rupture' and 'Myocardial infarct', which turned out to have the same code in CTV2, and the addition of a new code to 'Cardiac rupture' in CTV3.

We also consider some other popular controlled vocabularies in life science in the following.

II 6.1.4 GALEN

Generalized Architecture for Languages, Encyclopedia and Nomenclatures in medicine (GALEN)[22] [RN94] has been modeled to represent clinical information to support clinicians and is intended to "put the clinical into the clinical workstation" by generating a formal multilingual coding system for medicine [Bec]. It originally evolved from the Pen&Pad electronic medical record system [RNK91], which was modeled using

[22] http://www.opengalen.org/

Structured Meta Knowledge (SMK), in the way that terms were described through relationships to other terms. The core of GALEN is an ontology, the Common Reference Model, formulated in a specialized description logic, GRAIL, that does not support the use of disjunction or negation [RBG+97]. The GALEN community tries to enable the system to recognize concepts with different GRAIL descriptors that are equivalent in meaning. GALEN achieves expressiveness (the ability to represent the concepts formally) by providing a compositional representation of concept representations. It provides abstraction (defining generic categories of the concepts and the relations between them) by allowing formal logical classifications of the concepts and supports scalability and maintainability by using formal algorithms for consistency control [RR05]. GALEN has been employed as a basis for studying nursing terminologies [HR01], surgical vocabularies [TRR+00], anatomy [Don05], and decision support systems [Kar01].

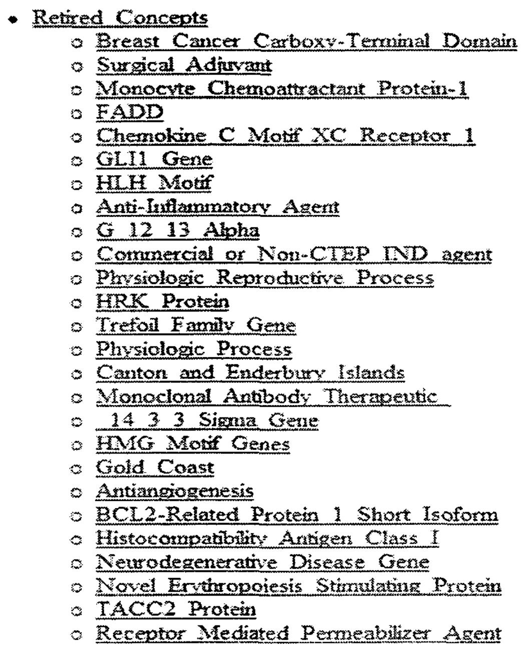
The major strengths of GALEN are the formal representation of clinical information and the use of a formal structure based on description logic. GALEN also allows "multiple views of relevant detail as needed" [Smi05]. From another point of view, GALEN is not fully developed and it is not a comprehensive, stable ontology. In its current state, GALEN contains some errors (e.g., Vomitus contains carrot [Smi05]), which are not prevented by description logics. Also, many of the relations in GALEN need to be reconstructed [RG04].

II 6.1.5 National Cancer Institute Thesaurus (NCIT)

The NCI Thesaurus[23] (NCIT) [SCH+07] is an integrated description logic-based terminology for supporting reliable coding and cross-translation research, based on cutting edge molecular and clinical cancer-related information. The NCIT contains about 100,000 terms (divided among several taxonomies), 34,000 concepts, and more than 50 types of role relationships for describing diseases, abnormalities, drugs, chemotherapy regimens, anatomy, gene, and proteins [CHS+04]. It was originally implemented using Apelon[24] and is now available in OWL (DL and Lite) format. The NCI uses the UMLS Metathesaurus as a basis for its NCI Metathesaurus (published monthly). It includes different cancer-oriented terminologies (prevention, treatment, and research), and assists users in finding appropriate terms and translations corresponding to related biomedical terminologies. A terminological and ontological analysis performed by Ceusters et al. [CSG05] revealed several inconsistencies in the terms and their definitions in NCIT. Some of the terminology errors have been inherited from the definitions in original sources, particularly some of the characteristic inconsistencies of the UMLS [CSG05].

The updates in NCIT take place weekly for internal and monthly for external baselines [CHS+04]. The editorial changes in NCIT are limited to the following actions: creation, modification (addition/deletion), splitting, merging, and retiring [HFO+03]. Some of the NCI's retired concepts can be seen in Figure 2.6.

[23] http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do
[24] http://www.apelon.com/



- Retired Concepts
  - Breast Cancer Carboxy-Terminal Domain
  - Surgical Adjuvant
  - Monocyte Chemoattractant Protein-1
  - FADD
  - Chemokine C Motif XC Receptor 1
  - GLI1 Gene
  - HLH Motif
  - Anti-Inflammatory Agent
  - G 12 13 Alpha
  - Commercial or Non-CTEP IND agent
  - Physiologic Reproductive Process
  - HRK Protein
  - Trefoil Family Gene
  - Physiologic Process
  - Canton and Enderbury Islands
  - Monoclonal Antibody Therapeutic
  - 14 3 3 Sigma Gene
  - HMG Motif Genes
  - Gold Coast
  - Antiangiogenesis
  - BCL2-Related Protein 1 Short Isoform
  - Histocompatibility Antigen Class I
  - Neurodegenerative Disease Gene
  - Novel Erythropoiesis Stimulating Protein
  - TACC2 Protein
  - Receptor Mediated Permeabilizer Agent

Fig. 2.6. Some of the NCI's retired concept[25].

NCI Thesaurus is maintained on a COTS (Commercial Off-The-Shelf) basis for terminology editing with public domain customizations as needed, mainly through the publishing tools from Apelon [CHS+04], including:

- TDE[26] (Terminology Development Environment): enables periodic exports of change sets, conflict resolution, and publishing of new baselines. It logs information related to creation, modification, and deletion. For managing changes in NCIT, the TDE has been extended to support split and merge, and deletion has been substituted with retirement [HFO+03].
- DTS[27] (Distributed Terminology Server): enables data normalization, code translation, comparisons of concept extensions, tracking, and localization (adding

[25] Resource: www.mindswap.org/2003/CancerOntology/htmls/retired_kind.html
[26] http://www.apelon.com/products/tde.htm
[27] http://www.apelon.com/products/dts.htm

concepts, synonyms, codes, etc.). It has been extended by including a DTS history API[28] to facilitate NCIT's history tracking [HFO+03].

The mechanisms for updating the NCI Metathesaurus and managing concept changes over time by history tracking in the NCIT has been described in [NCI06, HFO+03]. Here is the NCI's revisions statistics based on [HKR08] in the period of May 2004-Feb 2008.

Ontology	Addition	Deletion	Obsolete
NCI Thesaurus	627	2	12

Ontology	|C|(start)	|C|(latest)	grow
NCI Thesaurus	35814	63924	1.78

II 6.1.6 Health Level 7- Reference Information Model (HL7-RIM)

HL7[29]-RIM is a set of standard vocabularies that aims to provide a UML-based standard for the exchange, management, and integration of data to support clinical patient care and the management, delivery, and evaluation of healthcare services. HL7 was adopted by Oracle as basis for its Electronically health record (HER) support programs. It embraced as US federal standard and also considered as a central part of a multi billion dollars program for integration of all UK hospital information systems [Smi05]. HL7 has been also accepted as the mandatory standard[30] by Canada Health Infoway[31]. The relevant healthcare information in the RIM has been organized into the six classes, namely: Act, Entity, Role, Participation, Act-Relationship and Role-Link [HLR]. The ontological and logical analysis performed in [Smi05b] and [VSC04] address several problems in HL7-RIM, such as the problems of Circularity (some definitions fall into infinite regressive

[28] Application Programming Interface
[29] http://www.hl7.org/
[30] HL7 Canada: http://sl.infoway-inforoute.ca/content/dispPage.asp?cw_page=infostand_hl7can_about_e
[31] http://www.infoway-inforoute.ca/

loops)[32], logical incoherencies [VSC04], logical contradictions [Smi05], neglecting objective states of affairs and real processes, also the failure to distinguish properly between acts and documents [Viz04].

To be considered as a universal standard HL7 - with several known and unknown problems and incoherencies - needs to go through constant rigorous revisions. In respect to this issue, the HL7 standard includes a protocol version ID in all HL7 messages. The mechanism for controlling the changes in HL7 has been described [HLS] as: addition of new transactions or data elements to HL7, which are caused by changes in the Standard or due to legitimate changes in the local implementation. Considering some defined Encoding Rules, "new fields can be added first to the sending or source system; the receiving system will ignore the new fields until it has been updated to use them" [HLS]. Often, these rules also facilitate changing the receiving system first. Until the sending system is changed, the receiving system will find the new data field 'not present' and deal with this according to its rules for data not present. Similarly, the HL7 Encoding Rules support changes in data field sizes. Tables 2.1 and 2.2 demonstrate some of the new added features and changes in HL7 standards, in transition from version 2.1 to 2.2 and from version 2.2 to 2.3 respectively.

[32] For example defining "person" as "a person with document" (i.e. An A is an A which is B) makes it impossible to refer to As which are not Bs (e.g. to an undocumented person) [Smi05].

Table 2.1. Some changes in data elements of HL7 from version 2.1 to 2.2 (Source[33]: Health Level Seven Implementation Support Guide for HL7 Standard Version 2.3)

Segment/Seq	Name	New	Change	Description
MSH-5	Receiving Application		x	Length changed from 15 to 30
MSH-7	Date/Time of Message		x	Length changed from 19 to 26
MSH-9	Message Type		x	Datatype changed from ID to CM
MSH-12	Version ID		x	Datatype changed from NM to ID
MSH-15	Accept Acknowledgement Type	x		
MSH-16	Application Acknowledgement Type	x		
MSH-17	Country Code	x		
MSA-6	Error Condition	x		
ERR-1	Error Code and Location		x	Datatype changed from ID to CM
QRD-1	Query Date/Time		x	Length changed from 19 to 26
QRD-6	Deferred Response Date/Time		x	Length changed from 19 to 26
QRD-7	Quantity Limited Request		x	Length changed from 19 to 26
QRF-2	When Data Start		x	Length changed from 19 to 26

Table 2.2. Some changes in data elements of HL7 from version 2.2 to 2.3 (Source[34]: Health Level Seven Implementation Support Guide for HL7 Standard Version 2.3)

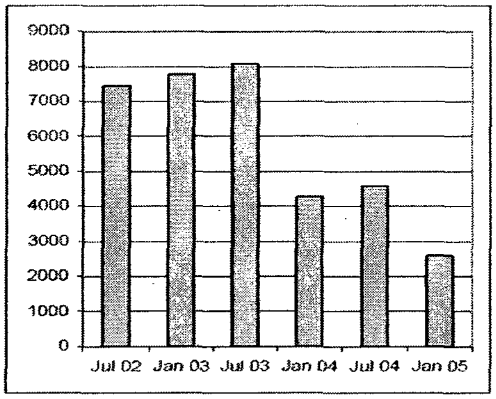
Segment/Seq	Name	New	Change	Description
MSH-3	Sending application		X	Length changed from 15 to 180. Data type changed from ST to HD
MSH-4	Sending facility ID		X	Length changed from 20 to 180, data type changed from ST to HD
MSH-5	Receiving application		X	Length changed from 30 to 180, data type changed from ST to HD
MSH-6	Receiving facility		X	Length changed from 30 to 180, data type changed from ST to HD
MSH-11	Processing ID		X	Length changed from 1 to 3, data type changed from ID to PT
MSH-18	Character set	X		
MSH-19	Principal language of message	X		
QRD-8	Who subject filter		X	Length changed from 20 to 60, data type changed from ST to XCN
QRD-9	What subject filter		X	Length changed from 3 to 60, data type changed from ID to CE
QRD-10	What department data code		X	Length changed from 20 to 60, data type changed from ST to CE
QRF-4	What user qualifier		X	Length changed from 20 to 60
QRF-5	Other QRY subject filter		X	Length changed from 20 to 60
QRF-6	Which date/time filter		X	Table - removed value CAN
QRF-9	When quantity/timing filter	X		
URD-3	R/U who subject definition		X	Length changed from 20 to 60, data type changed from ST to XCN
URD-4	R/U what subject definition		X	Length changed from 3 to 60, data type changed from ID to CE
URD-5	R/U what department code		X	Length changed from 20 to 60, data type changed from ST to CE

33 http://www.hl7.org.gr/assets/hl7implementationguide/HL7_implementation_guide.pdf

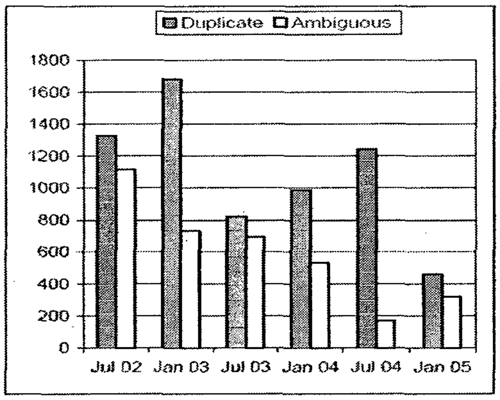
34 http://www.hl7.org.gr/assets/hl7implementationguide/HL7_implementation_guide.pdf

II 6.1.7 Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT)

SNOMED CT was generated by merging SNOMED Reference Terminology (RT) [SCC97] with Clinical Terms Version 3 (CTV3). According to [NS08], SNOMED CT includes 311,313 concepts (84% primitive and 16% fully defined) and 920,146 defining relationships. SNOMED CT can be used in various browsers[35] and is available in different formats[36], such as IHTSDO[37] support format, containing the original flat tables with information to concepts, descriptions, and relationships; description logic format [BSK+07]; and Metathesaurus format in UMLS (April 2009)[38]. A list of ontological and logical problems in SNOMED CT, which force the changes, can be found in [SSB07]. Spackman [Spa05] studied the rates of change in six subsequent releases over a period of three years (July 2002 to Jan 2005). The diagrams [Spa05] in Figures 2.7.a and 2.7.b illustrate the number of new active concepts added to each release and the number of duplicate and ambiguous concepts identified and retired in each release respectively.



(a)



(b)

Fig. 2.7. The number of (a) new active concepts added to each release; (b) duplicate and ambiguous concepts identified and retired (Source: [Spa05]).

[35] http://www.nlm.nih.gov/research/umls/Snomed/snomed_browsers.html
[36] http://www.nlm.nih.gov/research/umls/Snomed/snomed_faq.html
[37] The Int'l Health Terminology Standards Development Organisation (IHTSDO) (http://www.ihtsdo.org/)
[38] http://www.nlm.nih.gov/research/umls/licensedcontent/snomedctfiles.html

SNOMED is available as various snapshots of the current component status at a specific release date. The original SNOMED CT history mechanism could not support change tracking procedures for subsets and their membership [RefS06]. To solve this problem, a "Reference Set specification" (RefSet) has been defined, which is an extension of the original subset to enhance the change tracking mechanism, handle different user preferences, and use cases and issue recommendations for the evolution of other SNOMED CT elements [RefS06]. The ability to track each RefSet member and its property over time will improve "incremental updates of SNOMED's content since last synchronization, and facilitate time-sensitive queries for point in time retrieval of the status of each component" [RefS06].

Recently, a system called Terminology Version (TV) Manager [IB08] has been proposed for "searching and navigating in synchronized presentations of selected versions of SNOMED CT" based on comparisons of the sub-trees of interest.

II 6.1.8 The Foundational Model of Anatomy (FMA)

FMA[39] is a frame-based ontology (developed in Protégé) that represents an evolving source of explicit declarative knowledge about human anatomy and claims to be the most complete ontology of canonical human anatomy in a high granularity from the macromolecular to the macroscopic levels [RM03]. It primarily aims to expand the anatomical content of UMLS, by consisting of over 70,000 concepts and 110,000 anatomical terms along with 168 relationship types, which cover over 1.5 million relations between its concepts [CZ06]. FMA has been recently translated to OWL (DL

[39] http://sig.biostr.washington.edu/projects/fm/AboutFM.html

and Full) [NR08]. FMA includes three models, namely (i) the ontological model (represents classes); (ii) the structural model (describes spatial and topological relationships); and (iii) the transformational model (represents morphological changes) [CZ06].

The FMA has been recruited in applications such as the Biolucida system [WB05] to improve the capability of content authoring and knowledge presentation tools, functional computer-administered exam systems, study aids, and an injury propagation modeling environment, as well as haptic applications, such as surgery simulation [WB05].

II 6.1.9 Terminologia Anatomica (TA)

Terminologia Anatomica [Whi99] is a standard controlled vocabulary on human anatomical terminology, developed by the Federative Committee on Anatomical Terminology (FCAT). The TA's structure has been represented "through hierarchies of headings, varied typographical styles, indentations, and an alphanumeric code implies specific relationships between the terms embedded in the list" [Ros00]. All the changes in TA can be granted by decision and approval of the FCAT members [Ros00].

II 6.1.10 Different Types of Changes in Biomedical Ontologies

Based on our research of the literature, observing different releases of ontologies, surveys, and interviews with several domain experts and ontology engineers, we distinguished about 74 different types of changes that frequently occur in life cycles of existing bio-ontologies. These changes can be classified under 10 general terms: addition, deletion, retirement (obsoletion), merging, splitting, replacement (edit or rename), movement, importing, integration, or changes to file structure.

Table 2.3. Common changes in some of the existing popular bio-ontologies.

Type of change	Definition	Observed Ontology	Example
Addition	Improving ontological structure by adding one or more components to the available makeup. The most common additions in the observed bio-ontologies are of the following elements: Namespace, identifier code, concept, attribute, abbreviation, super-class, sub-class, attribute value, synonym, constraint (cardinality, type and min/max, inverse roles, default value), associative relationships (relationships to other individuals), annotation description, class-status (hidden/public), and instance.	Gene Ontology (GO)	The curators at MGI, who were reviewing the existing terms for comprehensive annotation of mammalian genes involved in the regulation of blood pressure, realized that the existing GO terms were not sufficient to annotate genes involved in the various processes that regulate blood pressure. They then proposed 43 new GO terms, which were discussed and refined with other GO curators through the GO discussion forum. They efforts yielded new annotations for mouse genes directly involved in the process of blood pressure regulation [GON06, GOB].
Deletion	Erasing the selected element(s) when it does not reflect the ontological 'truth' anymore. The most common deletions are of the following elements: Namespace, identifier code, concept, synonym, abbreviation, annotation (description), constraint (cardinality, type and min/max), attribute value, super-class, sub-class, constraint (cardinality, type and min/max, inverse roles, default value), associative relationships, annotation description, class-status (hidden/public), and instance.	Gene Ontology (GO)	The GO terms must characterize biological entities (i.e., functional activities that are catalyzed by enzymes). The terms classified as "Unknown" violated this principle, so the decision was made to delete the following terms: biological process unknown; GO:0000004, molecular function unknown; GO:0005554 and cellular component unknown; and GO:0008372 from the ontology. The new annotations signify that a given gene product should have a molecular function, biological process, or cellular component, but that no information was available as of the date of annotation [GON07b].
Retirement (Obsolescence)	Deprecating an older element when a newer, more functional element or meaning supersedes it. The older version can be kept somewhere for future use, but its usage will be discouraged [Cim96a]. The retirement can usually be seen for the concepts, attributes, identifier codes, instances and relationships.	Health Level 7 (HL7)	In the release 2.0 of HL7, the components: ClinicalDocument.copyTime, MaintainedEntity, CodedEntry, inkHtml.name,table.border, table. cellspacing and table.cellpadding are retained for backwards compatibility with HL7 Clinical Document Architecture (CDA), Release 1.0, and have been retired. Further use of these components is discouraged [DAB+04].
Merging	The process of creating a consistent and coherent ontological element that includes information from 2 or more basic elements. It can be seen as following: Merging two or more concepts into one of the concepts or into a new concept [Cim96a], two or more attributes into one of the attributes or into a new attribute, two or more associative relations into one of the relations or into a new relation, two or more identifier codes into one of the codes or into a new code.	Health Level 7 (HL7)	In HL7, the purpose of the header is to enable clinical document exchange across and within institutions, facilitate clinical document management, and facilitate compilation of an individual patient's clinical documents into a lifetime electronic patient record [DAB+04]. In HL7's Clinical Document Architecture (CDA), Release 2.0, two concepts in the header (service_actor and service_target) have been merged [DAB+04].

Splitting	An ontological element may be split into two or more new elements. This means that a concept can be split into two or more new concepts, an attribute into two or more new attributes, an associative relationship into two or more new relationships, or an identifier code into two or more codes.	Terminologia Anatomica (TA)	In TA, terms that share an id code are treated as synonyms. But, this does not hold for sexually dimorphic anatomical parts, such as 'Ovarian artery' and 'Testicular artery'. These two share the same TA code (A12.2.12.086) and therefore might be thought of as synonyms, but the two arteries are distinct and have different connections and other spatial relationships [Whi99]. So, they have to be modeled as two separated concepts, it means the code A12.2.12.086 can be split into A12.2.12.086-1 for 'Ovarian artery' and A12.2.12.086-2 for 'Testicular artery'.
Replacement (Edit,Rename)	This process is for editing available labels and values. This editing mostly happens to change namespace, concept name, concept definition, attribute value, attribute name, attribute definition, and concept role.	Health Level 7 (HL7)	A typical scenario [DAB+04] from HL7 Release 2.0 is a simple replacement of Clinical Document.id "1.2.345.6789.266" replacing ClinicalDocument.id "1.2.345.6789. 123"
Movement (Transition)	The transition of one or more ontological elements across the ontological hierarchy. This transition can happen to identifier *codes, concepts, attributes, super-class, sub-class,* associative relationships, and instances.	Gene Ontology (GO)	GO terms representing transporter activity in the Molecular Function are gradually being overtaken to better represent current scientific knowledge. A new high-level term called "transmembrane transporter activity" (GO:0022857) was introduced. So, the related child terms and sub-classes have been moved under GO terms that describe the activity of the transporters, such as channel activity, active transporter activity, and symporter, antiporter and uniporter activity [GON07c].
Importing	Importing refers to the process of bringing an existing ontology (a tree) or parts of an existing ontology (sub-tree) into another ontological structure.	Gene Ontology (GO)	In 2001, the GO developers imported the first pass[40] annotation from SWISS-PROT, trEMBL and Ensembl [GOM01]. Also, 7316 GO annotations were imported from Proteome and literature associations [GOM01].
Integration	In data integration, process data is extracted from different sources with different data formats, and then normalized into a consistent syntactic representation and semantic frame of reference [BCC+02]. The semantic integration is more complex than data integration.	Foundational Model of Anatomy (FMA)	In order to meet the need for an expressive ontology in neuroinformatics, the FMA developers have integrated the extensive terminologies of NeuroNames and Terminologia Anatomica into FMA. They have enhanced the FMA to accommodate information unique to neuronal structures, such as axonal input/output relationships [MRM+03].
Change to Release File (File Structure)	By the advancement of technology for storing and retrieving data files and the emergence of new standards, the format of file structures can be changed.	Read Codes	In Read Codes, Ver. 1.0 four character codes determined the position of a term in a hierarchy (4-Byte Set). The restrictions imposed by only 4 levels of hierarchy led to the development of a 5-Byte Set, which expanded the set to support secondary and tertiary care. This set was released in two structurally different versions. Ver. 1.0 has shorter terms and keys than Ver. 2.0. The more complex Ver. 3.0 structure is a superset of all old versions, and supports the character structures of both Ver. 1.0 and Ver. 2.0 [RCS+97].

[40] The annotations, which are derived with minimal human control and validation (e.g. initial results for a sequence similarity) and produced with various annotation programs such as tRNA Scan, Blast, etc.

After monitoring the alterations in several popular biomedical ontologies one can see that most of the changes are additions and deletions. For example, in the period May 2004 to Feb 2008, the changes in popular community standard Gene Ontology were almost 92.6% additions[41], and 5.6% deletions (see Figure 2.5), and for NCIT almost 97.8% additions, and 0.31% deletions (see Section II 6.1.5). The significant percent of additions is quite natural, since most of the biomedical ontologies are still under active development and ontology curators are adding new knowledge to their structure. These percentages may differ when the ontologies enter the maintenance phase.

II 6.1.11 Challenges in Maintaining Existing Bio-Ontologies

We found out from the current state-of-the-art of change management in existing ontologies in life science that formal change models with clear semantics are typically not employed. The change management in current systems is mostly addressed implicitly and takes place under human supervision. No matter how successful these change models are, for the purposes for which they were designed, they all have problems in maintaining their rapidly evolving structure because lack of formality and predictability. Most bio-ontologies that were built according to the existing formal knowledge representation models have not found widespread use in life science and health care applications [OSS+99]. Current bio-ontologies are built for a particular purpose, such as literature retrieval and there has been no goal to conform to a model that is useful for other applications. Therefore, due to inconsistencies among change models of different

[41] Although the given statistics is based on three types of changes in GO and NCIT, namely addition, deletion, and obsolescence, but it is a good indication to show the large number of additions in compare with other editorial activities in these ontologies.

ontologies, it is difficult to merge or share their content, therefore, it is not feasible to track the effect of changes in one ontology on other ontologies in an integrated system.

II 6.2 Existing Tools to Support Ontology Change management

There are a few tools [HS04, Sto04] to manage changes in ontologies. These tools include but are not limited to available ontology editors such as Protege [NFM00] and OntoEdit [SAS03], and TopBraid Composer [Top07]. Despite their differences, they all assist users in implementing, updating and managing elementary changes in ontologies. According to [Sto04, SM02], the most critical requirements for ontology editors in order to be more robust in a changing environment are related to *functionality, customizability, transparency, reversibility, auditing, refinement* and *usability*. Other available tools include but are not limited to Concurrent Version System (CVS) [Ced], CONCORDIA [OS00], KAON [MS03, GSV04] Ontology management tool, OntoView [KFK+02], OntoManager [SSG+03], TextToOnto [MV01], SWOOP [KPS+06b], DogmaModeler [Jar05], SemVersion [VEK+05], and DINO [NLH+08]. Table 2.4 represents some of the popular ontology editors and management tools with their descriptions.

Table 2.4. Some of the ontology editors and management tools.

Tool	Description
Protege [NFM00] (ver. 4.0 beta with web 2.0 support)	A popular ontology design environment with support for RDF and OWL ontologies. It provides some editing facilities such as: adding/deleting/renaming ontological elements, undo/redo of changes and version archiving [LAS05]. Protege also includes plug-ins such as PROMPT for managing multiple ontologies. It can compare versions of the same ontology, merge two ontologies into one and extract part of an ontology [NM03]. PromptDiff [NM04] also can determine the changes between two versions. Recently a new ontology reviser plug-in for Protege 4.0 has been introduced in [RW08], which helps performing some contraction and revision operations in DL ontologies. The reviser has been implemented using the OWL API[42] and the OWL DL reasoner Pellet [RW08].
TopBraid Composer [Top07]	A commercial ontology editor that supports editing RDF Schemas and OWL Ontologies, as well as executing rules and queries in the SPARQL Protocol and RDF Query Language (SPARQL) [Bec06] and the Semantic Web Rule Language (SWRL) within a multi-user environment. It manages multiple versions of ontologies by using the following set of rules. Any changes to the statements are written into the source ontology. If the change is "overtyping" an entry, it will be saved in the original ontology as an update. In case of the "deletion" of an entry and then the "addition" of a new one, the deletion would be done in the original file and the new triple would be saved in the existing file. Also, by changing any class, the composer scans to see if there are any other ontologies that import this class. It keeps a log of the changes that is accessible from the Change History view. Unsaved changes can be undone. To prevent accidental changes, a file can be defined as "read only".
SWOOP [KPS+06b]	A web ontology browser and editor, built based on the Model-View-Controller (MVC) paradigm [GHV04] for OWL ontologies. SWOOP consists of a version control unit, which aims for managing different versions by defining a set of annotation classes (i.e. ontology changes), logging all changes and processing the logs. Within the SWOOP OWL API each possible change type has a corresponding Java class, which is subsequently applied to the ontology and allow for the representation of changes, as well as metadata about the changes [KPS+06b].

[42] http://owlapi.sourceforge.net/

Concurrent Version System (CVS)[43] [Ced]	Supports basic version control functionality and maintains a history of the changes. CVS can reveal syntactical and textual differences between two files. It mostly works on the syntactic level. Since ontology versioning and change management need operations on the conceptual level rather than the syntactic level, CVS might not seem an appropriate tool for ontology change management [VG06]. However, CVS can provide basic support for managing structural changes in RDF and OWL files.
CONCORDIA [OS00]	A model for managing divergence in concept-based terminologies, developed to facilitate the study of synchronization in health care terminologies. CONCORDIA uses the models of Medical Subject Headings (MeSH) [NJH01], ICD-9-CM [Cim96a], and ICD-10. It enables one to manage 27 different kinds of changes, such as adding, deleting, retiring, or merging concepts, terms or attributes [OS00]. CONCORDIA does not provide any services to log motivations for the changes [CS06].
KAON[44] [MS03, GSV04]	An integrated open-source ontology management system targeted at semantics driven business applications, KAON components can be divided into 3 layers: (i) The applications/services layer realizes user interface applications and provides interfaces to non-human agents; (ii) The API, which is the major part of KAON, checks the validity of change sequences, and also requests user approval for performing a change, justifies the necessity of a particular change, executes the modifications, reverses the effect of some undesirable changes and keeps a history of changes; (iii) The data and remote services layer provides data storage facilities. See [GSV04] for more information.
OntoView [KFK+02]	A web-based system that assists users in handling ontology evolution. The system helps to keep different versions of web-based ontologies interoperable by maintaining the transformations between ontologies and the relations between concepts in different versions. OntoView was inspired by and can be considered a Web interface for CVS. OntoView compares ontologies at a conceptual level, analyzes effects of changes (e.g., by checking consistency and highlighting the places in the ontology where conceptually changed concepts or properties are used) [KFK+02]) and utilizes changes.

[43] www.nongnu.org/cvs
[44] http://kaon.semanticweb.org/

OntoManager [SSG+03]	Has been designed to assist ontology managers in managing ontologies according to the users' requirements. The technique used to evaluate users' needs depends on the information source by tracking user interactions with the application in a log file. The OntoManager consists of three modules: (i) The data integration module, which aggregates, transforms, and correlates the usage data; (ii) The visualization module that presents the integrated data in a comprehensible visual form; and (iii) The analysis module, as the major part of the change management, provides guidance for adapting and consistently improving the ontology with respect to the users' requirements. This module keeps track of the changes and has the ability to undo any action taken upon the ontology.
TextToOnto [MV01]	A tool suite built upon KAON in order to support the ontology engineering process by text mining techniques. Since TextToOnto does not keep any references between the ontology and the text documents it has been extracted from, it does not allow for mapping textual changes to the ontology. Therefore data-driven change discovery is not supported by this tool.
DogmaModeler[45] [Jar05]	DogmaModeler is an ontology modeling tool based on Object Role Modeling (ORM) [Hal01]. It is intended to be used for modeling, browsing, and managing domain and application axiomatizations, automatic composition of axiomatization modules, verbalizing application axiomatizations into pseudo natural language and other tasks described in [Jar05].
SemVersion [VEK+05]	SemVerion [VEK+05] is an RDF-based ontology versioning system that separates the management aspects of the problem from the versioning core functions [FMK+08]
DINO [NLH+08]	Dynamic INtegration of Ontologies (DINO) aims for integration of the knowledge in data-intensive and dynamic biomedical domains based on the negotiation of agreed alignments, inconsistency resolution and natural language generation methods. [NLH+08].

As can be seen from the current state-of-the-art change management in existing ontologies in life sciences, the current biomedical ontologies do not follow any standard, consistent, formal change models with clear semantics. Most of the available tools are just simple ontology editors with a few extended features. Some parts of ontology evolution, such as the change representation and conceptualization change, are not

[45] http://www.jarrar.info/Dogmamodeler/

satisfactorily managed by existing tools and they are left to be handled by the users. The major issues in available ontology management tools can be summarized as: (i) Too much reliance on human decisions due to lack of fully automatic ontology change management tools and too much dependency of the existing systems on the human factor [HS04], which both give rise to several issues relating to complexity, accuracy, security and reproducibility [Flo06]; (ii) Representation and tracking of complex changes using available technologies are limited; (iii) Lack of formal evaluation methods, which makes the comparison and evaluation of different algorithms extremely difficult [Flo06]; (iv) Little or no support for conceptualization change management; (v) Change models that have been designed based on time/space independent ontologies; and (vi) Lack of a precise benchmark forecast for anticipating future changes; (vii) Representing knowledge in dynamic environments is still challenging; (viii) The consequences of a change cannot be represented. An important open question about ontology evolution is: How can a machine decide on the best solution to implement a change from different available alternatives?

II 6.3 Employing Logics for Ontology Maintenance

Logics provide frameworks to describe the underlying semantics of ontologies. Two families of logic which are broadly being used in knowledge representation are Description logics and Fuzzy Logics. This section presents a quick review of these two and provides an introduction to the new compound logic, Fuzzy-DL and its relation to the ontology evolution tasks.

II 6.3.1 Description Logics and Ontology Evolution

In order to analyze effects of changes, one can use a DL reasoner such as RACER to automatically verify the changes and the specified conceptual relations between versions. RACER can help for checking the consistency of the ontology and look for unexpected implied relations. The authors in [RSS02] and [LLM+06] present interesting implications for updating dynamic DL-based knowledge bases.

II 6.3.2 Description Logics and Temporal Reasoning

Knowledge representation needs theories, applications and tools for expressing structured knowledge, accessing and reasoning with it [FVK+00]. In order to formalize time-based domains one can use description logics for temporal reasoning as proposed by Schmiedel [Sch90, Sch91]. The DL system BACK [Pel91] was inspired by this idea. Later, following the standard approaches in the representation of time, both interval-based and point-based approaches have been studied, specifically focusing on the decidability and complexity of the reasoning problems [BCM+03]. An interesting application of temporal description logics for reasoning about temporal conceptual models has been presented in [Art04]. Also a survey of temporal extensions of DL can be found in [AE01].

One of the main issues in temporal DLs is related to reasoning. Reasoning in temporal description logics that discriminate between past and future changes is generally undecidable [ALT07]. A multi-dimensional description logics has been proposed in [ALT07] by combining the modal logic with the description logic to support reasoning about change - without discriminate the past and future changes - by allowing to express the changes in concepts and roles over time [ALT07].

II 6.3.3 Fuzzy Logics: Towards Finding a Solution to the Old Puzzle

Recalling the discussion about Sorites in Section II.2 (Philosophy), philosophers tried to combine sets and logic in order to analyze language. One common idea is that the predicates of our language correspond to sets. So the predicate "is a heap" corresponds to the set of all heaps [Aub90]. What the Sorites tells us is that there will always be a questionable case about whether something is a heap. Apart from threatening the attempt to analyze predicates of a language, the Sorites throws a doubt on the ability of propositional and predicate calculus to describe the way the world is. The law of identity (a=a) and the law of non-contradiction $\neg(p\&\neg p)$ are two fundamental axioms of classical logic. The Sorites challenges both. It challenges the law of identity because it seems to come up with the result that something that is a heap is also not a heap. For the same reason it also challenges the law of non-contradiction [Rom99].

To answer this paradox, contemporary thinkers reconsider the classical logic's principle which says that truth is binary: true and false. Fuzzy set theory (and before, multiple-valued logics proposed by Lukasiewicz [Tom99] in 1918) has modified this rule by stating that a degree of truth is an abstract notion that cannot be directly measured as such [DP97]. Then, one can think of sentences as being very "true", "fairly true", "reasonably false", "completely false" and so on. Multiple-valued logics [Tom99] and fuzzy logics [Zad65] are created based on this new idea. The notion of a fuzzy set has been introduced by L. Zadeh [Zad65] in order to formalize the concept of gradedness in class membership, in connection with the representation of human knowledge. As an example in fuzzy logic it is true to say of an oval that "it is round" and to say the same of a rectangle, despite the fact that neither is really round. One of the challenges in conceptualization change management is comparing different versions of ontologies and finding similarities and differences. One way to do this is to get benefit from fuzzy logics to find the various degrees of similarities between new and old conceptualizations. In other words, one can find the "degree of truth" in ontologies represented by fuzzy propositions [DP97].

II 6.3.4 Fuzzy Description Logic

With advances in technology about fuzzy and uncertain knowledge management there are many efforts to apply these techniques in description logics [GL05, GL02, Str01, BDG06, Yen91] to represent uncertain and vague knowledge in the Semantic Web [LS06]. The main motivation of using fuzzy techniques in DL is to identify concepts and notions that cannot be properly defined with an "exact" numerical bound [BCM+03]. For example, the concept of "Acting in low pH" cannot be always defined with an exact boundary for low pH, but must be represented with a membership or degree function [BCM+03], which expresses low/high pH in a continuous way.

It seems an interesting initiative to extend OWL using fuzzy technologies. Ding et al. [DP04] also extends OWL using probabilistic knowledge. In fact, uncertain knowledge or vague concepts is as important as probabilistic knowledge in the real world. In [Str05] the authors try to extend OWL by encoding fuzzy constructors, axioms and constraints (denoted FOWL) and map semantics of new fuzzy terms to fuzzy description logic. The extended OWL can directly resolve fuzzy inference questions by a constraint propagation calculus. A *fuzzy description logic and constraint propagation calculus*, fuzzy constructors, axioms and constraints in RDF/XML and also a set of translation rules from

OWL to FOWL can be seen in [Str05, GL02]. Fuzzy description logic can present vague concepts and roles (from the point of fuzzy sets [DP04]) as well as interoperates in these concepts and roles. Reasoning algorithms are also provided for computing fuzzy subsumption within the framework of tableau-based methods [BCM+03].

A reasoner called FuzzyDL [Str] has been recently developed for *fSHIN(D)*. It is a free Java/C++ based reasoner for *fSHIN(D)* with concrete fuzzy concepts. FuzzyDL aims to provide a procedure to compute the maximal degree of subsumption and instance checking with respect to a general TBox and Abox [Str]. It supports Zadeh's semantics, Lukasiewicz semantics and is backward compatible with classical description logic reasoning [Sat]. The efficiency of FuzzyDL is still under investigation. For syntax and some examples of FuzzyDL one can refer to [Str].

II 6.4 Change Management for RDFS/OWL Ontologies

There are three main activities involved in managing ontology change. Firstly we need to identify changes, secondly describe these identified changes, and finally describe and implement the changes. Standard languages for encoding ontological knowledge on the web, such as the RDF schema (RDFS) [Bri04] and the Web Ontology Language (OWL) [BVH+04] provide some basic mechanisms for managing the evolving structure of ontologies. In [Kal06], a framework for change management in RDFS/OWL ontologies has been proposed. Also [Cha-1] and [Cha-2] provide studies for RDFS/OWL ontology evolution in two aspects: change in names and change in metadata with focus on OWL Full with maximum expressiveness but lack of full computational support.

II 6.4.1 Change in names

Currently changes in names for RDFS/OWL ontologies and ontology versions are handled by assigning a URI to the ontology, and also to each "snapshot" or "version" of the ontology. Two examples of name changes have been studied in the wine and the food ontologies [Cha-1].

II 6.4.2 Changes in Metadata

Metadata provide annotation for existing data. Creating metadata can support change management for RDFS/OWL ontologies [Cha_2] and control versioning. In current OWL ontologies two types of metadata are widely used:

I. **OWL Annotation Properties:** OWL facilitates ontology classes, properties and instances to be annotated with various pieces of metadata. These metadata are mostly being used to keep auditing or editorial information. For example, some predefined OWL annotation properties are comments, versionInfo, label, seeAlso and isDefinedBy. When we use a description logics based reasoner such as RACER all annotation properties are ignored and considered as comments by the reasoner. OWL-DL which is the selected language for The FungalWeb Ontology, supports maximum expressiveness without losing computational completeness and decidability of reasoning systems, but unlike OWL-Full it has some restriction for using annotation properties [BVH+04]. The sets of different properties (object, datatype and annotation properties) must be disjoint. It means, for example, owl:versionInfo is not allowed to be defined as a datatype and an annotation property at the same time. Also annotation properties must not be used in property axioms. So, specifying domain or range constraints and sub-properties for annotation properties is not allowed.

II. **Dublin Core Metadata:** The Dublin Core [Dub] can be used to specify a set of metadata elements that can be used to annotate various elements of an ontology with information such as 'creator', 'date', 'language', 'publisher', 'title', 'modified', 'issued'. These annotations can be use for change management purpose.

II 6.4.3 Dynamic OWL for handling the changes

A Dynamic OWL (DOWL) language [AY03] has been proposed for describing ontology changes. DOWL can be represented in the RDF abstract syntax which enables one to describe the effects of a change in a more formal manner. This formalism can provide the basis for an automated ontology change management system. It is claimed that DOWL provides a necessary and sufficient set of operators for expressing changes in an OWL ontology. DOWL formalism is set in the context of the OWL by extending the RDF compatible model theoretic syntax and semantics for OWL [AY05].

Despite all the efforts, creating a standard web ontology language to capture and represent the evolving structure of ontologies remains a difficult challenge [HVD02]. Another effort in OWL-DL ontology change management is OWLMeT (OWL-MetricTime) [KLG+07], which is grounded on the Metric temporal description logic with a temporal query language. It aims to trace the changes of each ontological element through time and determine the status of the ontological elements at a specific time point.

It introduces the special sort of a nominal (temporal nominal) for ontology versions [KLG+07] and uses an extended version of the DL-reasoner Pellet for temporal querying.

II 6.5 Summary of Section II.6

Biology and medicine are known as two fields with continuous evolution. Many healthcare applications must deal with the problem of change in order to keep their scientific knowledge up-to-date and valid. One of the important activities in knowledge representation and bioinformatics is properly responding to changes and coping with the ontological evolution. Research on ontology change management is an ongoing effort that is still in its early stages. In this section, we reviewed some of the available tools and techniques for maintaining biomedical ontologies and we have shown that they still have long road ahead to be considered for practical usage due to following issues:

- Lack of formal change models with clear semantics
- Inconsistencies among change models and log models
- Too much reliance on human decisions
- Reproducibility of the results cannot be guaranteed
- Little or no support for the representation of complex changes
- Lack of formal evaluation methods
- Little support for handling changes in conceptualization

In addition, we presented different types of potential changes in biomedical ontologies, and we tried to show actual evidence of these changes in some of the most popular ontologies in health science. Knowing different types of changes can help knowledge engineers model their ontologies accordingly. Through these insights into what can actually be changed in a typical bio-ontology, we begin to find an answer as to how we can manage and control this non-stop evolution. One of the issues in the ontology evolution process is the lack of formal change models with clear and comprehensible semantics. We will discuss this issue further in Chapter III.

III. The Framework for Change Management

The purpose of this chapter is to describe and analyze our proposed agent-based framework, namely RLR, for change management in biomedical ontologies. Moreover, in this chapter we explain the formalism chosen to support our framework and its potential to represent and analyze evolving ontologies in various levels of abstraction, independent of domain and implementation language. The use of category theory and hierarchical distributed graph transformation for realizing the semantics of evolving distributed ontologies in RLR will be utilized.

III.1 Evolutionary Taxonomy of Fungi: A Motivational Scenario

"The hierarchy of relations, from the molecular structure of carbon to the equilibrium of the species and ecological whole, will perhaps be the leading idea of the future."

Joseph Needham (1900-1995)

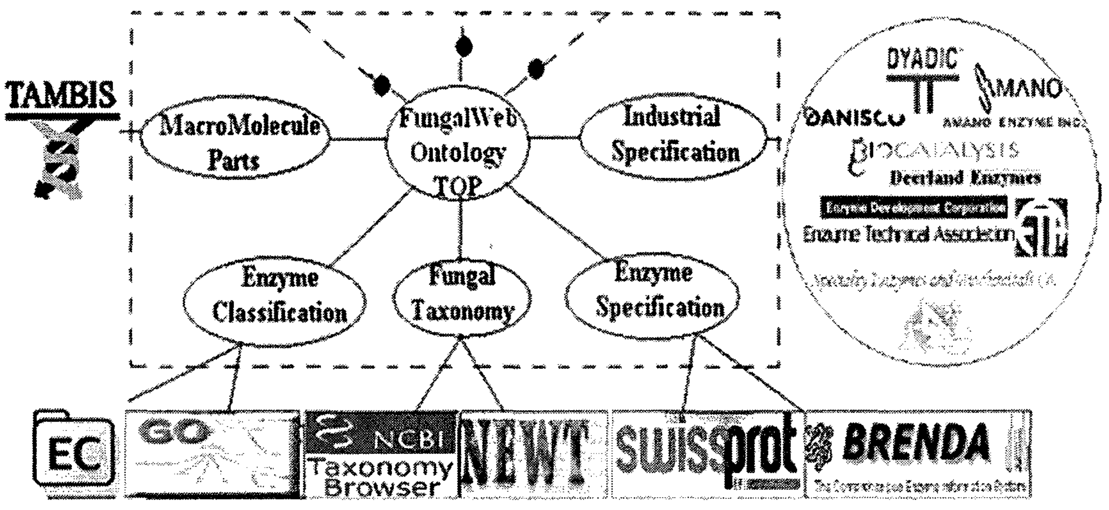
III 1.1 Fungi Phylogeny and Evolution

Fungi are widely used in industrial, medical, nutritional and biotechnological applications. They are also related to many human, animal and plant diseases, food spoilage and toxigenesis [BAP+02]. Fungi are also interesting because their cells are surprisingly similar to human cells [MRC06]. The reason for this is that fungi split from animals about 1.538 billion years ago—nine million years after plants did—therefore fungi are more closely related to animals than to plants [NHI94]. It is estimated that there are about 1.5 million fungal species [Hey95] on the earth, but only about 10% of those are known and only a few of the known fungi have an identified usage, such as yeast for making bread, beer, wine, cheese and some antibiotics [MRC06]. A small percentage of discovered fungi have been linked to human diseases, including dangerous infections. Due to the similarities between human and fungal cells, treating the fungal diseases can be risky. Any medicine that kills the fungus can also damage the human cells. Thus, knowing more about fungi and the correct identification of each fungal species is crucial, and can improve the quality of fungal-based products and help to identify new and better ways to treat serious fungal infections in humans. Fungi are also the main source of agricultural and plant diseases, so identifying them will aid us in tracking and controlling these diseases [MRC06].

Typically, fungal evolution studies have been based on comparative morphology, cell wall composition [Bar87], ultrastructure [Hea86], cellular metabolism [LéJ74], and the fossil records [HKS+95]. Recently, by advances in cladistic and molecular approaches, new insights have emerged [GGS99]. Some other new identification methods are based on immuno-taxonomy and polysaccharides [GGS99], which are highly suited antigens for the identification of fungi at the genus and species level [NDW+88]. The following fungal chemical substances are also used as complementary characters to the classical morphological taxonomy of fungi: proteins, DNA, antigens, carbohydrates, fatty acids and secondary metabolites. One can find a review of the methods for employing the substances in [FBA98]. These substances are very valuable at many taxonomic levels, and they play an increasing role in the clarification of the phylogeny (a classification or relationship based on the closeness of evolutionary descent) of fungi [NDW+88]. At the moment, the phylogenetic relationships between fungal taxa are still uncertain and controversial [GGS99].

III 1.2 The FungalWeb Ontology

For the application scenario, we have applied our method for managing changes to the FungalWeb Ontology [SBH+05]. The FungalWeb Ontology is a formal ontology in the domain of fungal genomics, which provides a semantic web infrastructure for sharing knowledge using four distinct sub-ontologies: enzyme classification based on their reaction mechanism, fungal species, enzyme substrates and industrial applications of enzymes. The ontology was developed in OWL-DL by integrating numerous online textual resources, interviews with domain experts, biological database schemas (e.g., NCBI [WCL+00], EC, NEWT [PPF+03], SwissProt [Bai00], Brenda [SCE+04]) and reusing some existing bio-ontologies, such as GO and TAMBIS [BBB+98].



Source name	Instances	Concepts
BRENDA & SwissProt	105	35
EC & GO	1308	211
NCBI taxonomy database	10926	3340
TAMBIS	0	22
Commercial Enzyme Vendors	401	6

Fig. 3.1. The FungalWeb Ontology and its major resources.

III 1.3 Name changes in Fungal Taxonomy

Most fungal names are not stable and change with time. Fungal names reflect information about organisms, and as our understanding of the relationships among taxa increases, names will be forced to change so that they do not implicitly contradict the data [Cro05]. Most names are currently based on the phenotype (visible characteristics of an organism).

As more data become available, however, we run into various problematic issues, such as convergent evolution, seen as the evolution of the same form in different families and even orders, so that similar anamorphs (the imperfect (asexual) state of a fungus) may have completely different, unrelated teleomorphs (the sexual stage in the life cycle of a fungus, considered the perfect stage). These names then have to change, as they no longer convey the correct information to the user [Cro05]. These name changes may cause confusion and affect the validity of different queries. Take for instance *Acremonium* Link, a simple anamorph morphology which is known to have affiliations to more than 20 different teleomorph genera [GBP+96], or as another example consider *Cladosporium* Link, which probably includes more than 20 different genera (Crous, unpublished data). *Verticillium* Nees [ZGC00], *Coniothyrium* Corda [LSG+04] and *Mycosphaerella* Johanson / *Sphaerulina* Sacc. [CGM+04] and a few more links [Cro05] are some other examples, which face with this issue. A more specific example about eyespot disease in cereals and issues related to naming its associated fungi has been described in [CGG03].

The morphological conceptualization is not sufficient, and will no longer work because all names based only on morphology have to be re-evaluated. In addition, the phylogenetic-based conceptualization has its own limitations, as sometimes the decision of where to draw the line between different species is not easy to make [Cro05]. Another issue in fungal taxonomies is dual nomenclature (two names for one organism) due to the anamorph/teleomorph debate [Cro05]. This is caused by the fact that it is frequently impossible to say when an asexual state belongs to a specific sexual state without the backup of molecular data. A study on revision of the fungi names [LSM+98] shows that between 1960 and 1975, 212 names of foliicolous lichenized fungi were described or used by A.C. Batista and co-workers.

Managing name changes: We are currently in the middle of a revolution in fungal taxonomy [Cro05]. Names are linked to data. Older names are mostly classified based on small data sets (largely phenotypic), and therefore they are subject to change. How can biologists deal with this process of continuous change? To answer this question, one needs to refer to the nature of ontological structure, where names in taxonomy are only meaningful and valuable once linked to descriptive datasets that were extracted from various databases and literatures and managed in an integrated environment. The incorporation of DNA data is also needed to ensure stability in names and reliable species recognition. Through future advances in the technology, biologists hope to preserve the fungal taxonomy from change by using unique DNA signatures and species identifier numbers to recognize the species rather than using the names [CG05]. There are currently databases such as MycoBank [CGS+04], which link fungi names to their DNA sequence data, pleomorphic states, herbarium specimens, descriptions, illustrations and related publications, etc.

By 2005 only about 16% of 100,000 known fungal species have been represented by DNA sequence data [Cro05], which is approximately 1.1% of the estimated 1.5 million species on Earth, thus it seems that a very low percentage of the already discovered fungal species are in fact being preserved from the change [Haw04]. The changing nomenclature of medically important fungi is often very confusing. Currently, some of the pathogenic fungi have a very unstable taxonomy. For instance, the name of the fungi *Allescheria boydii,* which can cause various infections in humans, was changed to

Petriellidium boydii and then to *Pseudallescheria boydii* within a short time [OAD+92]. Consequently, the infections caused by this organism were referred to as allescheriasis, allescheriosis, petriellidosis, and pseudallescheriosis in the medical literature [OAD+92].

In order to manage the changes in fungal names and clarify the ambiguities, the Nomenclature Sub-Committee of the International Society for Human and Animal Mycology (ISHAM) published its regulations for mycosis nomenclature [OAD+92, OR95]. Based on these regulations, a disease should be given a meaningful, descriptive name, while in the traditional disease taxonomies, the names "fungus+sis" indicate only a causative fungal genus that could be highly influenced by the taxonomic changes. Additionally, under the new regulations, the value of names of the "pathology A due to fungus B" construction was emphasized [OR95], e.g., "subcutaneous infection due to *Alternaria longipes*" [GGS99].

III 1.4 Changes and Revisions in Taxonomic Structure

Through advances in molecular biology and changes to the fungal nomenclature, one can expect changes in taxonomical structure and relationships. Here are some examples:

Example 1: *Glomeromycota* was discovered in 2001 [SSW01] as a new fungal phylum. The arbuscular mycorrhizal (AM) fungi and the endocytobiotic fungus, *Geosiphon pyriformis*, are analyzed phylogenetically by their small subunit rRNA gene sequences. By studying their molecular, morphological and ecological characteristics, it is discovered that they can be separated from all other major fungal groups in a monophyletic clade [SSW01]. Consequently, they are removed from the polyphyletic *Zygomycota*, and relocated to a new monophyletic phylum, the *Glomeromycota*, with four new orders: *Archaeosporales*, *Paraglomerales*, *Diversisporales* and *Glomerales* [SSW01].

Example 2: The sedge parasite *Kriegeria eriophori* has never been satisfactorily classified, because a number of its characters at the gross micromorphological and ultrastructural levels appeared to be autapomorphic [SFM99]. By advances in the nucleotide sequence data approach that provides more information than standard morphological approaches, some of the ultrastructural characters were discovered to be synapomorphies for a group containing *K. eriophori* and *Microbotryum violaceum.* These characters serve to define the new subclass Microbotryomycetidae [SFM99].

Figure 3.2 represents how the place of the concept "pH optimum" has been changed within the FungalWeb taxonomy (ver. 2.0) by adding the new concept "Functional Property".

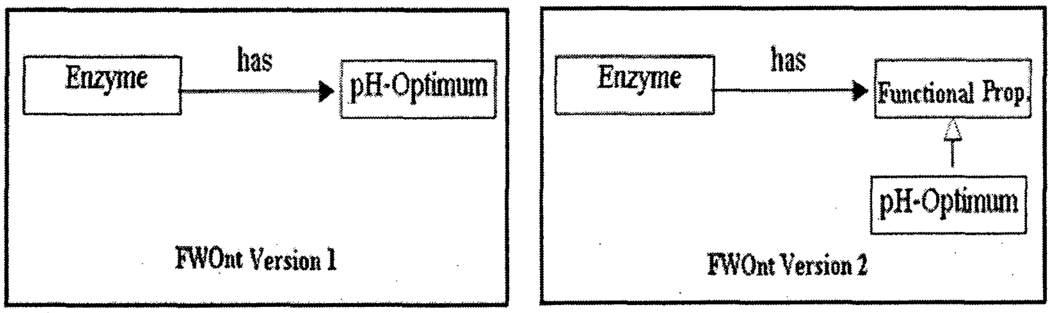


Fig. 3.2. A simple change in taxonomical structures of two consecutive versions of the FungalWeb Ontology (FWOnt).

The problem of Unspecified Fungi: As mentioned before only a small portion (around 100000) of 1.5 million fungi species are described. It means almost 1.4 million fungi are still unspecified due to the lack of knowledge. Clearly, as the knowledge about fungi species grows and new methods become available by discovering new species

[HR97], one can anticipate a fundamental change in the current fungal taxonomy structure. In the meantime using reliable approaches to ensure stability of fungal taxonomy by describing the names based on verifiable data and not on opinions and statements is still promising.

III 1.5 Summary of Section III.1

For the meantime, the categorization of fungi is controlled by the *International Code of Botanical Nomenclature* (ICBN) [GBB+94] as adopted by each International Botanical Congress. ICBN primarily aims to provide a reliable scheme for naming taxonomic groups, avoiding and rejecting names which may cause error, vagueness, or any confusion [GGS99]. Any proposed changes to the Code are published in *Taxon*, the official journal of the International Association for Plant Taxonomy, and then discussed in the Congress for approval [GGS99]. The strict application of the Code frequently leads to name changes for nomenclatural rather than scientific reasons [Haw93]. This causes confusion among users, who do not usually understand the reasons for the changes.

The changing nomenclature of fungi of biotechnological, industrial and medical importance is often tremendously confusing for workers in the applied field [Sam91]. Many of the pathogenic fungi have a very unstable taxonomy, which may cause fatal errors in highly critical medical knowledge based systems. In the rest of this chapter we will introduce our formal agent-based approach for consistently managing this non-stop evolution.

III.2. The Multi Agent Based Framework

It has been said that man is a rational animal. All my life I have been searching for evidence which could support this.

Bertrand Russell (1872–1970)

III 2.1 On the AI Completeness of Change Management for Biomedical Ontologies

The term "AI-complete"[46] (or AI-hard) [SA07] is commonly applied to certain computational problems in artificial intelligence whose difficulty is equivalent to solving the central artificial intelligence problem, i.e., making computers as intelligent as humans. Some such problems can be found in computer science when one deals with topics like computer vision, planning, natural language understanding, and so on. One of the classic AI-complete problems occurs when one needs to manage unexpected situations and deal with changes while planning for a real world critical system. Critical, in this case, means when the failure or malfunction of the system may result in severe loss [Ave09]. One may find excellent examples of life support critical systems based on massive integrated knowledge bases in health science, dealing with the health and life of a patient, the failure of which is intolerable.

As bio-ontologies are constantly being revised, each revision potentially makes the ontology more susceptible to future changes. Moreover, the biomedical knowledge bases are extremely dynamic [ECP+02], as they tend to be openly reused, and integrated by

[46] Wikipedia: http://en.wikipedia.org/wiki/AI-complete

other existing knowledge based systems in the distributed dynamic semantic web environment, where new pieces of elements connect and existing parts are removed, and the representation formalisms and the governing rules themselves are unpredictably volatile. Auditing and controlling all these change in large complicated biomedical ontologies, as seen in Section II.4, is simply beyond human ability. In this section, software agents are proposed as a remedy to assist the human factor (here, the ontology engineer) in overcoming this issue.

III 2.2 Multi-Agent Systems and Patterns of Change

According to Wooldridge [Woo09], agents act to meet their design objectives by carrying out autonomous actions in their environments. They achieve their goal through their actions: reactivity (perceiving the environment and responding in a timely fashion to changes that occur, in order to satisfy their design objectives); proactiveness (exhibiting goal-directed behavior by taking the initiative to satisfy their design objectives); and sociability (interacting with other agents and possibly humans to satisfy their goals). In addition, mobility and learning aptitude are other capabilities that are important for agents in several application areas. An integrated system consisting of several agents that are communicating and interacting with each other through a unified communication channel is generally referred as a Multi-Agent System (MAS). A multi-agent system is a network of multiple autonomous agents cooperating to solve a problem when each agent has incomplete and limited knowledge. Data is decentralized, there is no global system control, and the computation is asynchronous [JSW98]. A MAS can address some of the challenges in human-computer interaction mentioned in Section II.4 so that an intelligent environment supports collaborative maintenance and change management. Figure 3.3 represents a general overview of interactions between a typical MAS and the users.

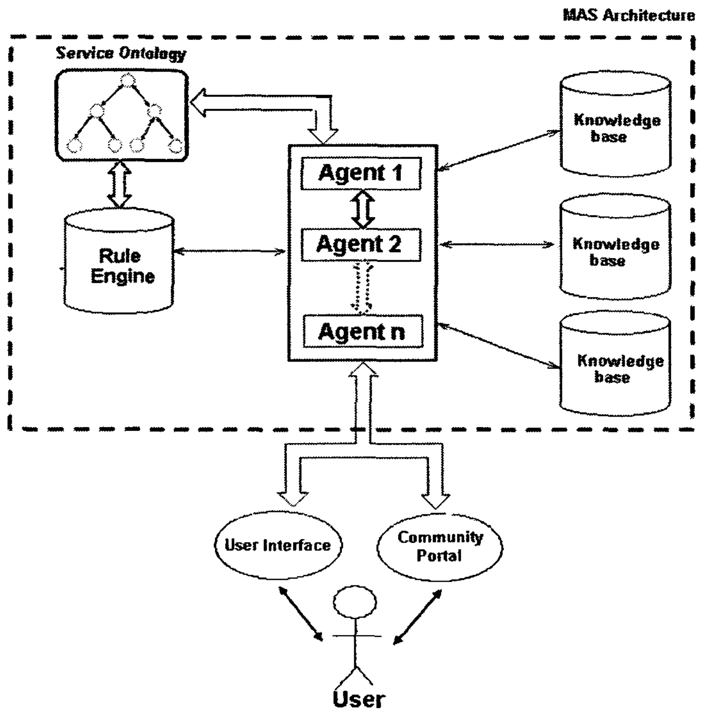


Fig. 3.3. An abstract view of the interactions between users and a typical multi-agent based framework. The MAS is capable of controlling the changes in the knowledge bases through a set of defined rules. A service ontology also provides sufficient knowledge for the interaction between the agents. Finally, the users can pose their query via a high-end user interface to communicate with the MAS.

Intelligent agents have the ability to perceive changes in the real world and find, identify, and collect desired information from multiple resources about various actions under changing conditions [Dev01]. Agents are also able to work rationally in order to capture changes in dynamic and heterogeneous environments, and to respond properly to these changes [LWY05], ideally in real time. Traditionally, agents in semantic web are classified under three categories [SWK+02], namely service providers (which present different kinds of services, such as searching, locating, and querying), service requesters (which ask the provider for a service), and middle agents (which help other agents perform their tasks). The middle agents seek out appropriate provider(s) to fulfill a particular request, issued by the requester(s), through the process called matchmaking. In a typical MAS, different types of links and relationships connect agents and represent dependencies, constraints, and dialogue paths between them. The research on intelligent agents and their interactions is already mature enough to be used and trusted in many autonomous systems, in areas such as medicine, supply chain management, auctioning, advertising, trip/vacation management, stock market analysis, and so forth.

III 2.3 The RLR Framework

The RLR framework aims to Represent, Legitimate, and Reproduce the changes and their effects (Figure 3.4). It helps to capture, track, represent, and manage the changes in a formal and consistent way, enabling the system to generate reproducible results.

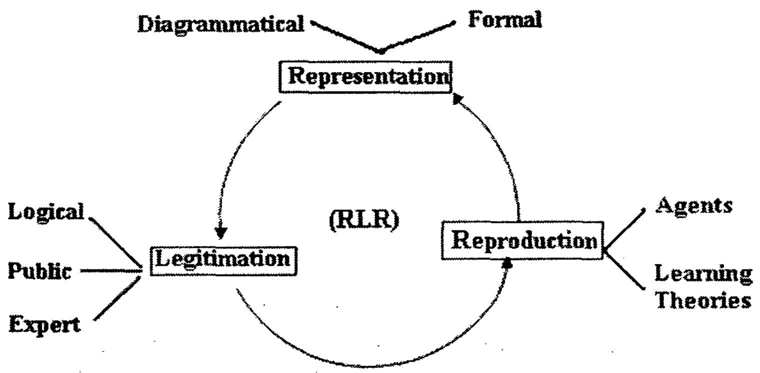


Fig. 3.4. The RLR framework: The arrows in the diagram denote the iterative nature of change management process. The representation of changes can be done through formal representation languages or via diagrammatical (semi-formal) representation methods or combination of both. The legitimation can be performed by experts and by public users. Also, logical validation is carried out using a logical reasoner. Intelligent agents with their learning ability contribute to reproduce the results of changes, when necessary.

- **Representation:** This phase is responsible for consistently updating the representations of new knowledge. Many of the problems in ontology evolution are basically problems about the nature and representation of change. The concerns about the problem of representation in dynamic systems seem to be twofold

[Hey90]: firstly, how the changes can be represented, and secondly, how the representation can be changed. For the formal representation of changes, we use description logics, and for diagrammatical representation, we employ a method based on discrete state model and category theory [SH07b]. Since a representation has been defined as "an abstract structure which is related through certain operations with external, physical phenomena" [Hey90], the abstractness of categories can help us to represent the dynamic interactions that happen in ontological structures through a set of operations in various discrete states.

- **Legitimation**: in our context, is defined as the verification of the legitimacy and consistency of a change in the domain of interest. This phase assesses the impact of a potential change before the change is actually made. Experts and logical reasoners should study a change based on its consistency with the whole design, including changes in the inferred assertions, in various degrees of granularity. Then, the final approval is needed from end-users. Logical legitimation can be obtained by the reasoning agents, which work in close collaboration with the negotiation agents.
- **Reproduction**: Overreliance on human factors is a problem in current change management methodologies. Despite the advantages of maintenance, including higher rationality, human intervention does not guarantee the reproducibility of results of a change [Flo06]. To overcome this issue, we propose using intelligent agents that discover patterns for different forms of changes and their consequences.

The final outcome, which has been generated through a rigorous argumentation process over generally accepted arguments, has an implicit link to the archived historical processes that can be reused to choose a proper pattern in the reproduction phase (Figure 3.5).

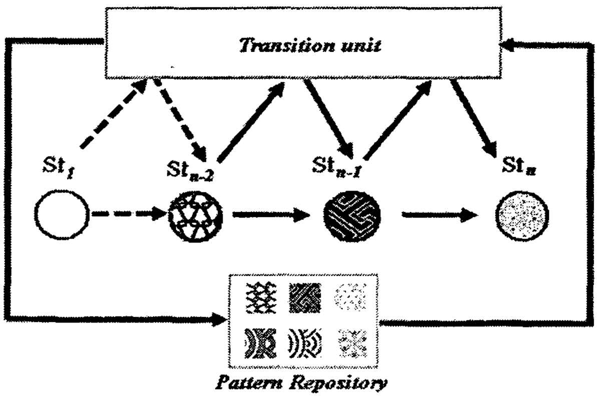


Fig. 3.5. A generic transition system in a multi-agent system. A system changes its state from St_1 to St_2 via a transition unit and a rigorous argumentation process between the agents to choose proper patterns from the change pattern repository for implementing a certain type of change.

In RLR framework, various ontological changes can be represented in either formal or diagrammatical ways. Each change will be legitimated and validated logically, then approved publicly and by experts. To reproduce the results of changes and automate the change management process, agents are recruited to learn change patterns (the pattern of change of ontological elements and constraints during the certain period of the ontology life cycle) and their consequences. The change patterns depict editorial activities, assist consistency control, and help predict the system's behavior and consecutive feedbacks. Several studies on change pattern have focused on representing change patterns to automatically infer likely changes [KNG07], revealing error patterns [LZ05] and aspect patterns to identify cross-cutting changes [BZ06], and extracting change patterns. One of the techniques for discovering and extracting change patterns is through hierarchical clustering with a sample change history [FGG08], which considers transformations in a matrix with the change types as the rows and method versions (extracted by

ChangeDistiller [FWP+07, GFP09]) as the columns. This process continues by dividing the change history of the system into fractions (e.g., yearly quarters, months, weeks, etc.) and creating a matrix for each of these fractions. The final step includes analyzing and comparing the change type patterns of each of the fraction clusters among each other and with those of the full cluster [FGG08]. The change patterns will be employed later as the basis for detecting and identifying the editorial activities and making automatic recommendations for performing different actions to deal with the applied changes.

RLR recruits four types of agents that act in a collaborative environment, namely: Change Capture Agents (CCA), Learner Agents (LA), Reasoning Agents (RA), and Negotiation Agents (NA). Figure 3.6 demonstrates the interactions between these agents.

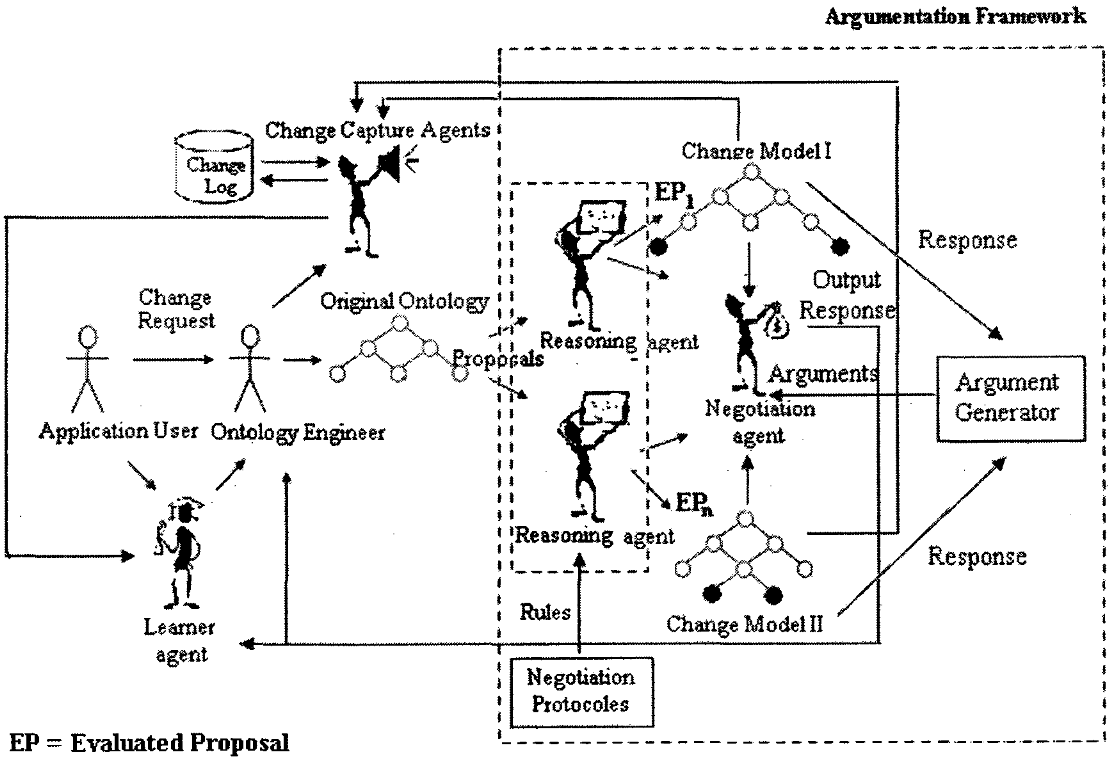


Fig. 3.6. The change management process using agents through an argumentation framework.

III 2.3.1 Change Capture Agents

Having the ability to detect and capture a change or any stimulation indicating an alteration in an ontological structure is not trivial; this is confirmed by the fact that existing change management approaches have so far managed to detect, capture, and represent only a small portion of ontological changes, mostly at the syntactical level. The change capture agent family in RLR is responsible for discovering, capturing, and tracking the changes in ontology, by processing one or more change logs. They detect real-world alterations and report them as new facts with which to update the knowledge base of an agent. Changes can occur on a random or scheduled basis. The change capture agents act like triggers in a database. We have defined the following three different types of change-capture agents:

- **Action Control Agents (ACA):** The action control agents consist of user activities and legal operations, which together capture changes such as deletion, insertion, and updates to ontology elements, and can store all the data related to different types of changes in change logs.
- **Explorer Agents (EA):** The explorer agents capture changes by processing and reading change logs in parallel, in a specified time range. By logically determining transactions, the explorer agents generate the appropriate messages for the corresponding services. They also assist in extracting a pattern of changes by exploring whether a particular change or a category of interconnected changes appears frequently, and whether it implies specific actions.

- **Log-Reading Agents (LRA):** The log-reading agents read the log files in a specified time period. This information will be passed on to a learning agent in order to create patterns for different changes. Later, the information can be used to Undo or Redo a change.

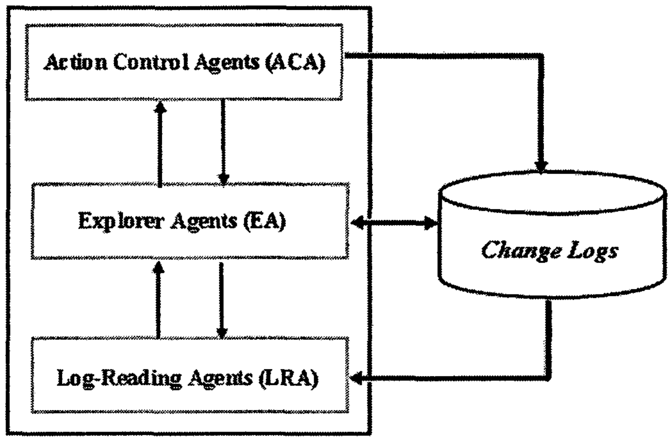


Fig. 3.7. The cooperation between the change capture agents

Together, these agents (Figure 3.7) monitor all the alterations and determine which ontological elements have been changed. To capture ontological changes, we also use annotation properties such as: Timestamps, Version and Status on ontological elements.

Moreover, since the popular biomedical ontologies have been organized in a hierarchical manner, it would be reasonable to employ the change capture agents to compute the changes by comparing old and new versions of the knowledge source and reducing the problem to that of finding a "minimum-cost edit script" [CRG+96] that gives us the necessary operations for transforming one hierarchy to another, or using the "fixed-point algorithm" presented in PROMPTDIFF [NM02]. As an example, in Figure 3.8, consider two taxonomies related to ontologies O_1 (source ontology) and O_2 (target ontology), where each node represents a concept, which is identified with a label along with a set of corresponding attributes. After discovering similarities and differences between these two taxonomies, we need to find a proper transformation that has been transformed O_1 to O_2. To start this procedure, the two taxonomies need to be aligned and brought into a mutual agreement, based on the matching concepts (the ones that affected less in the transformation) within the ontologies. The matching will be computed based on the degree of similarities between two concepts.

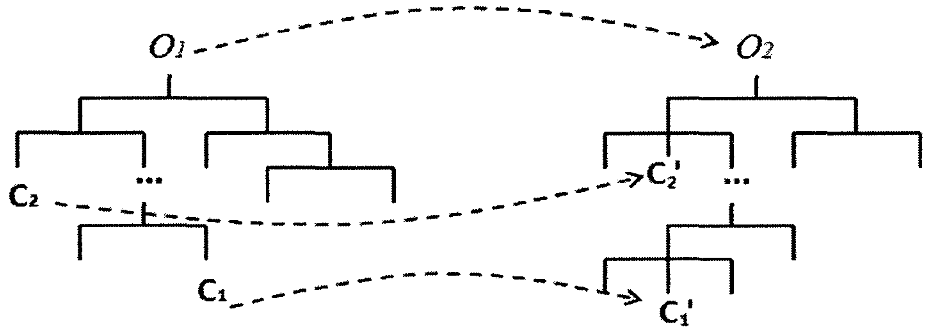


Fig. 3.8. The alignments between some concepts in two ontologies O_1 and O_2.

Detecting changes by comparing the old and new versions can also be performed by some available tools, such as PROMPDIFF [NM02]. The problem of comparing two hierarchical structures will be redefined in Section III.4 while exploring isomorphisms in their structures. We will also show how the use of graph transformations helps us discover the set of operations that transforms the hierarchy indicating the old version of an ontology into the hierarchy indicating the new one.

III 2.3.2 Learner Agent

As an application is used and evolves over time, the change logs can accumulate invaluable data and information about various types of changes. A learner agent can use these historical records of changes that occur over and over in a change process to derive a meaningful pattern. After several changes, possibly from various releases, it would be feasible to estimate the rate and direction of possible future changes for a system by generating rules or models. In RLR, the reasoner and negotiation agents can change the generated rules, and send modifications to the adaptive learning agent. Changing the rules is a main adaptation principle [RL04] for learning in RLR framework. The learning agent starts with limited, uncertain knowledge of the domain, and tries to improve itself, relying on adaptive learning based on semantics provided by the ontological backbone. The adaptive learner agent plays an important role in the reproduction phase, where we look for patterns to bootstrap the process of change management. The discovery of temporal patterns for event-based data is addressed by P.S. Kam, et al. [KF00], while Höppner tackled the problem with the discovery of informative temporal rules for defining temporal patterns in [Höp03]. Learning rules for discovering temporal patterns is described by L. Sacchi, et al. [SBL+05, SLC+07] for extracting temporal rules to learn patterns of evolving ontological data [SBL+05]. In RLR based on the extracted rules, we use a mathematical model to assist users in anticipating certain actions when the agents are faced with a specific type of changes in the knowledge based system.

III 2.3.2.1 Models of learning

By determining the tradeoffs between losses and benefits that can result from agents' actions, we will be able to have a mathematical model to foresee the agents' (software or human) behavior. A state of "Nash equilibrium" [Osb03] is one of the popular approaches in evolutionary game theory for modeling the most beneficial (or least harmful) set of actions for a set of intelligent agents. For the sake of prediction, Nash equilibrium can be understood as "a potential stable point of a dynamic adjustment process in which individuals adjust their behavior to that of the other players in the game, searching for strategy choices that will give them better results" [HR04]. Nash's theory has been found applicable in several dynamic domains, such as climate change [DR04], explaining economical and biological evolutions, where there is always the need to make a choice during a set of repetitive events and actions until agents reach an equilibrium.

Intelligent agents decide on the proper actions and are able to change and improve their decisions based on what they learn. Based on [Wan06], as shown in Figure 3.9, for each learner agent, we define an internal state b; a function *f* that shows how an agent decides and chooses actions based on its internal state (decision-making); the functions showing the payoff dominance (loss/benefit); and a state update function g, specifying how an agent updates its state based on the payoff received from previous iterations. The state of each agent depends on the probability distribution over all the possible situations [Wan06], and the one with the highest probability can specify the final decision. The update function can be computed based on different loss/benefit algorithms.

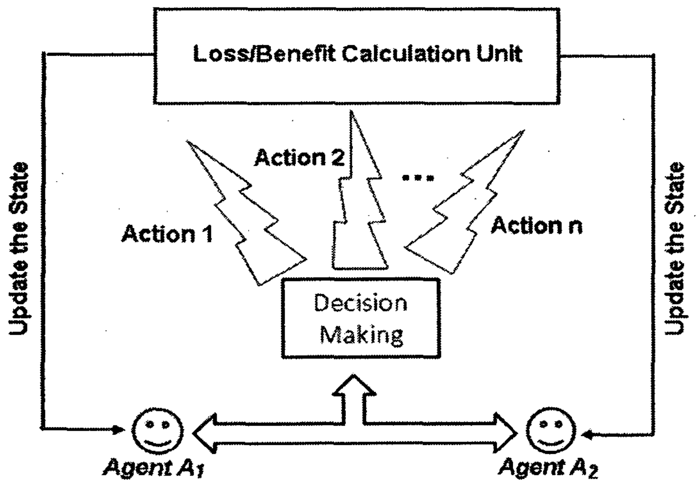


Fig. 3.9. A simple learning model for agents based on Nash equilibrium.

Another technique for automating the learning process is through inductive bias. The inductive bias of learning [Mit90] in neural networks is a set of assumptions, given as input, that the learner uses to predict and approximate the target outputs (even for unseen situations) through a series of training instances and their generalization. As stated by Mitchell [Mit90], in order to describe and represent the inductive bias learning, there is the need for a generalization language, so that each generalization denotes the set of its related instances (e.g., in Figure 3.10, g_1 and g_2 are two generalizations and each matches a different subset of the instances). The language that "allows describing every possible subset of these instances" is called an unbiased generalization language [Mit90].

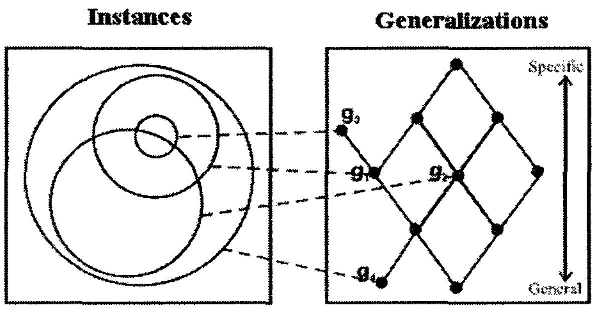


Fig. 3.10. Relationships among Instances and Generalizations (adapted from [Mit90])

III 2.3.2.2 Anomaly Pattern Analysis

Intelligent agents also detect and generate patterns of anomalies, either syntactic or semantic, by assessing and analyzing consistent common errors that occur through different revisions. After the anomalies have been flagged by change capture agents, the learner agent can then be taught the proper route for performing the revisions through a set of pattern mining algorithms (see [CM05] as an example of techniques for mining dynamic patterns). This task is crucial in a wide variety of applications, such as biosurveillance for disease outbreak detection [WMC+03] using Bayesian network analysis and cancer diagnosis. The learner agents not only enable the RLR framework to manage potential, expected, and prescheduled changes, but also prepare it for dealing with random and unexpected alterations. However, human supervision and participation will be anticipated for the former case.

III 2.3.3 Reasoning Agent

A reasoning agent is a software agent that controls and verifies the logical validity of a system, revealing inconsistencies, hidden dependencies, redundancies, and misclassifications. It automatically notifies users or other agents when new information about the system becomes available. We use RACER [HM03] as a description logic reasoner agent, along with other semi-formal reasoners in the RLR framework. When the agent is faced with a change, it ought to revise its conceptualization [CCS05] based on the new input by reasoning about the consistency of the change using both prior and new knowledge. Several attempts [Poi86, GLT89, Pav96, KKR06] have been made, to provide reasoning services for category-based systems. We also use a semi-automated reasoning system for basic category-theoretic reasoning based on a first-order sequent calculus [KKR06]. It captures the basic categorical constructors, functors, and natural transformations, and provides services to check consistency, semantic coherency, and inferencing [KKR06]. The reasoning agent in this framework uses the predefined constraints and axioms, given as input, to reason about the possible states of a certain ontology.

Another face of the reasoning agent in RLR will be revealed when it acts as a supplementary query engine (in cooperation with negotiation and learning agents) to reason and assess how the change in an ontology affects the state, quality, range, and depth of possible answers to some queries, which are posed at different time points. Just recall the incomplete nature of ontological knowledge that usually unfolds through the time. We may need to make some assertion about temporal situations without specifying the exact time (e.g., in response to the question, "Is the patient's heart rate at rest less than some value x?" one may expect an answer like, "No, should I notify you when it is?").

III 2.3.4 Negotiation Agent

Negotiation happens when agents with conflicting interests desire to cooperate [RRJ+03]. In the RLR framework, the negotiation agent acts as a mediator allowing the ontology engineer and other autonomous agents to negotiate the proper implementation of a specific change while maximizing the benefits and minimizing the loss caused by such change. A human expert may then browse the results, propose actions and decide whether to confirm, delete, or modify the proposals, in accordance with the intention of the application. In our framework, negotiation is defined based on the conceptual model of argumentation [VGH96]. In this context, an argument is described as a piece of information that allows an agent to support and justify its negotiation stance or influence that of another agent [RRJ+03, JPN+98] through a negotiation protocol, which formally provides necessary rules for negotiation dialogue among participants. These rules may include rules for admission, withdrawal, termination, proposal validity, or commitment [JPN+98]. In our approach, we adapted the architecture of the argumentative negotiating agent described at [ARL07]. We also assume the argumentation process is performed in a tree-like structure within the so-called "argumentation tree" [OT09].

Employing argumentation to analyze belief revision [FKS02, PC04, OT09] with the intention of updating an agent's knowledge has been studied in [CCS05] based on dialectical databases. Belief revision commonly refers to the situation where agents change their initial positions and statements because of a new conceptualization achieved by new inferred knowledge. To reach an agreement among the agents and provide a common understanding, a service ontology (Figure 3.11) is needed, so that updating this ontology generates a new understanding for the software agents, which can then update and adjust their beliefs based on new knowledge.

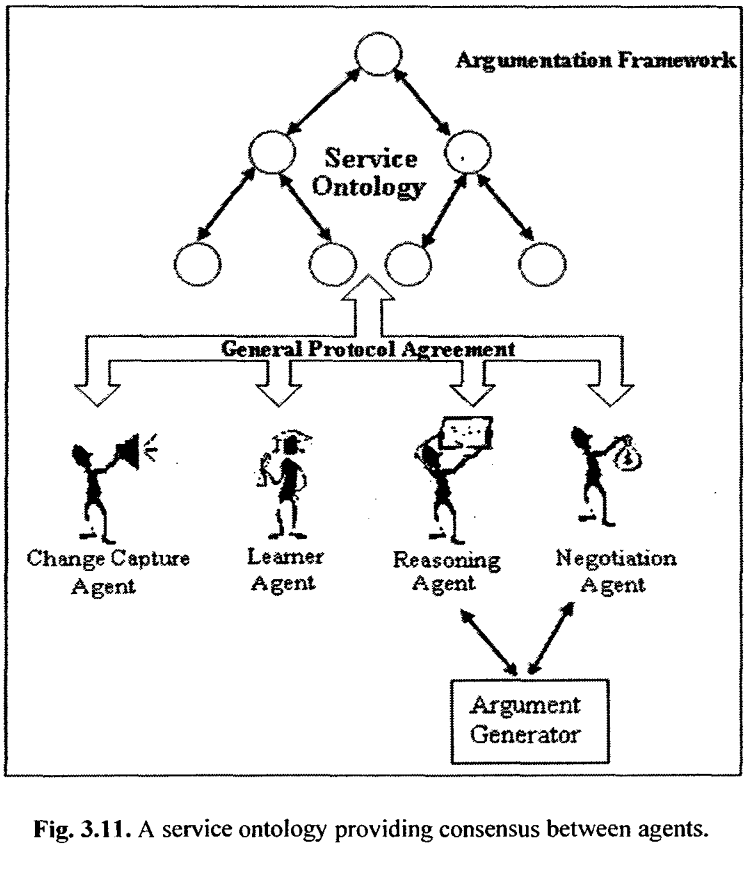


Fig. 3.11. A service ontology providing consensus between agents.

Employing service ontologies to automatically provide a service profile to describe the supported services and the related communicative transactions and invoke the services for service-seeking agents is currently being considered as a solution to overcome some of the issues related to overreliance on human intervention. However, these ontologies will not remain static and unchanged throughout their life cycle, and managing their dynamic structure would be part of the whole problem itself.

A software agent (Req-A) sends a request to the change capture agent (CCA-B) to check for the possible changes in an ontology while Req-A is interacting with other agents and the CCA-B responds to the Req-A by sending the list of changes (Figure 3.12).

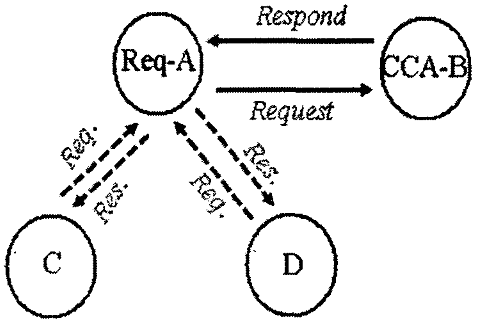


Fig. 3.12. Interactions between different types of agents for capturing changes. The solid lines represent the *main interactions and the dotted line denote the marginal interactions.*

III 2.4 Agent communications

Using a common language (syntax) is a necessary condition for communication and knowledge exchange in an MAS, but not sufficient by itself. The agents should also use a common semantics, using a generic consensus ontology. The consensus between the agents can be achieved either through a negotiation process, which supports future changes, or by determining a pre-consensus ontology for cases where changes to the core-ontology have been limited. Two standards, both founded on speech act theory [Sea72], are more commonly used for creating communication channels between intelligent agents, namely FIPA-ACL (standardized by FIPA[47]) and KQML[48] [FFM+94]. However, there are other communication languages offered by organizations such as KIF[49] (based on first-order predicate calculus) and OMG[50] (and its agent working group)[51] that are less popular in the field. In addition, to adapt agent communication languages to industrial needs, several attempts have been made to combine the aforementioned standards, i.e., the cooperation between FIPA and OMG to adapt the communication language with an object-oriented modeling paradigm. Following the FIPA+OMG approach, by extending UML, a formalization called "Agent UML" [BMO01] was proposed to describe interactions within an MAS. This formalism uses UML diagrams such as interaction diagrams (sequence and collaboration diagrams), state diagram and activity diagram to model dynamic behavior of agents[52]. It also benefits from the object constraint language (OCL)[53] to add constraints (i.e., pre- and post-conditions of operations) to the UML models. The "Agent UML" combines features of sequence diagrams with state diagrams to describe the interaction protocols [BMO01] and generate communicative patterns. The interaction protocols consist of "agent lifeline" (determines the time frame for the existence of an agent), several agent-roles (which satisfy certain properties and service descriptions, and assist in dynamic classification in

[47]The Foundation for Intelligent Physical Agents: http://www.fipa.org/repository/aclspecs.html
[48] Knowledge Query and Manipulation Language
[49] KIF: Knowledge Interchange Format: http://www.ksl.stanford.edu/knowledge-sharing/kif/
[50] Object Management Group: http://www.omg.org/
[51] http://www.objs.com/isig/wg-agents06-minutes.html
[52] Interaction diagrams are more appropriate to model how several objects collaborate and behave without representing the behavior's details. The state diagrams are more suitable to monitor a specific object's behavior [FS00].
[53] For more information on OCL specifications see: http://www.omg.org/docs/ad/97-08-08.pdf

a way that an agent can change its role and place in the UML classification), and proposed semantics for UML messages to define the agents communication patterns in a more efficient way through parameters, cardinalities, and so on [BMO01].

In RLR each agent has been defined to have a lifeline indicating its existence from creation to destruction (e.g., the Action Control Agents (ACA) for each session can be created upon an alteration in a system and can be destroyed after storing the change in the change logs). A lifeline may split into two or more lifelines to express the different alternatives that an agent has for responding to the received messages, or different lifelines may merge together at some point to represent an agreement or concurrency [BMO01]. Ideally, an agent communication language must allow flexible message exchanges with abstract semantics. In our approach (Section III.4), we extend the existing semantics by incorporating concepts from category theory in order to define more formal, reusable communicative patterns for agents' communications (i.e., message exchange[54]). This expressive categorical framework enables us to describe the interactions within an MAS and impose several restrictions and constraints, which are essential for reproducing agents' actions and responses using the defined rules. By this method, the ontology engineers can model the system with insights gained from foreseeing the changes and possible confrontations.

III 2.5 The Change Analysis Model in RLR

Our change analysis model is composed of a set of states that are linked to their predecessors and successors through some defined relationships. This allows us to check

[54] In classic UML-based agent communication formalisms, the message flow between agents can usually be represented using protocol diagrams [ODB00].

backward and forward compatibilities for one specific ontological structure from a given state. This is determined by defining various conditions and constraints for an event. The conditions can later be used to restore the previous state based on the insights gained for each event. Somehow it means a revision or review of the past, or an attempt to define an alternate (parallel) past [May83]. Since ontological assertions are based on open world assumptions, neither past nor future knowledge about the world is complete. One can always ask questions (e.g., "Could that mutation, under those circumstances, lead to the species X or Y?") and draw a different path from the previous states to the subsequent states. This iterative process of switching between the future, current, and revised past states has been regarded in [May83] as the process of "rolling back to some previous state and then reasoning forward" in the form of queries such as, "Is there some future time in which p is true?" [May83].

To deal with forward and backward compatibility, in our research we have employed graph transformation techniques, which enable us to analyze different states of the graphs based on the given initial states and the transformation rules. If the framework remained limited to only traditional graph transformation, no significant improvements would have been accomplished. Indeed, graph transformation offers many benefits, as will be outlined in Section III.3, but lacks sufficient expressivity and semantics to deal with all aspects of ontology change management. Our approach for this issue can be improved by recruiting a formal mathematical representation such as category theory. The enhancement can be done in two aspects: 1) the rules can impose restrictions on ontology transformation in the way that, for example, some alteration can be prohibited, or some changes, which have less impact on ontological elements, can be excluded in the related change analysis (e.g., the transition of a fungus from one genus to another does not affect its physical appearance); 2) the changes in states can be scheduled to occur simultaneously, sequentially, or in parallel.

III 2.5.1 The RLR Dialectic Change Management

Recall the concept of "dialectical changes" from Section II.2, where a change is defined as new forms built upon the old. Using this concept as a metaphor, we have introduced our formal agent-based argumentative framework, where "synthesis" takes place, for studying ontology evolution and shifting as model transformation. This transformation results from quantitative changes accumulated over a period of time and generates a new form out of old patterns ("coexistence of both old and new") [Hol98]. In fact, most of the changes that occur in an ontological structure, which lead to a new state, emerge from the preceding states[55]. In other words, the change lies within the system [Gil06]. Therefore, "learning" about different actions in different states of a system seems to be a key factor for starting a successful change management mechanism.

In a typical scenario within the RLR argumentative architecture, a user (human or agent) initially sends a request to an ontology engineer for a particular change in the ontological structure. Based on the system's background knowledge and the choice of the ontology engineer, various options are available to implement a change. The negotiation agent, along with the reasoning agent, provides arguments for the acceptance or rejection of a change proposal. The "Argument Generator" (Figure 3.6) determines appropriate responses based on the negotiation rules. Different arguments attack each other to enforce

[55] A "state" in this manuscript is being used to express a situation describing a part of the real (dynamic) world in a specific instance of time.

their rules and defeat their peers by sending counter-arguments. The inferred arguments can increase the possibility of higher quality agreements [CCS05, ARL07]. The Negotiation Protocols in the RLR architecture contain the rules that dictate a protocol. As the knowledge base is used and evolves, the historical information about different changes will be accumulated in the change logs. This information will be used by the learner agent, which acts as a basis for a recommender system[56], to propose different alternatives for the implementation of future changes.

The reasoning and negotiation agents can change the rules if necessary and send modifications to the learning agent. In order to maintain agents' argumentation for automation of ontology evolution, we employ the "dialectical databases" [CCS05]. In argumentation-based multi-agent systems, a dialectical database tends to improve the speed of inference responses by storing pre-compiled knowledge about potential dialectical trees [BK08]. The dialectical trees represent sets of possible dialectical confrontations between the arguments to accept or deny a proposal to deal with a particular change [CCS07].

III 2.5.2 Identity Preservation in RLR

The identity of a concept can be determined by those properties and facts that remain stable through time, even during multiple ontological changes. If ontologies are able to maintain their conceptual stability, they can better preserve their intended truth. To this end, the RLR framework employs a defensive mechanism to prevent harmful changes and reduce the risk of potentially dangerous actions by incrementally adapting to the

[56] The ability to generate (infer) appropriate recommendations is considered as one of the key functionalities in RLR. The level of the system's automaticity is highly depends on the quality of these recommendations.

changes at different levels. If a destructive change is about to happen in the ontology (e.g., deleting a concept, such as "fungi", when other dependent concepts, such as "fungal infection", exist), a warning signal will be sent to the agents based on the knowledge within the ontology (e.g., "fungi are the cause of fungal infections") to infer the potential threat and prepare them to plan for a proper action. This mechanism works much like the self-awareness system inside rational animals, which helps them avoid possible dangers without actually experiencing their life threatening influences. For example, as pointed out in [Hey90], a person who is confronted with fire does not have to experience the burning and can run away as a counteraction, since the person has been taught that smoke indicates fire and that fire can kill humans.

III 2.5.3 The Rule-based Recommender System for Change Management

As mentioned, RLR is applied to capture and describe changes (syntactical, semantical, or environmental) and respond promptly by generating adequate knowledge for other agents involved to propose recommendations or by making decisions about actions based on a set of pre-defined rules. For example, consider the deletion of a concept, C, from ontology O, which can be done using the RLR framework with various degrees of effort depending on the location of C (e.g., terminal concept (leaf), a parent concept with children and with or without siblings, a top concept (root)). As another example, we have defined the following rules for adding a concept to an ontology structure with a pure subsumption taxonomy:

Rule 1: Check whether the concept to be inserted is an initial concept (the only one) in the ontology. In this case, just add the concept and associate its attributes.

Rule 2: If the concept is not an initial concept, add is-a relationship to its parent concept, which is usually determined as the one with the most similar derived and primitive properties, to form the hierarchy (Figure 3.13).

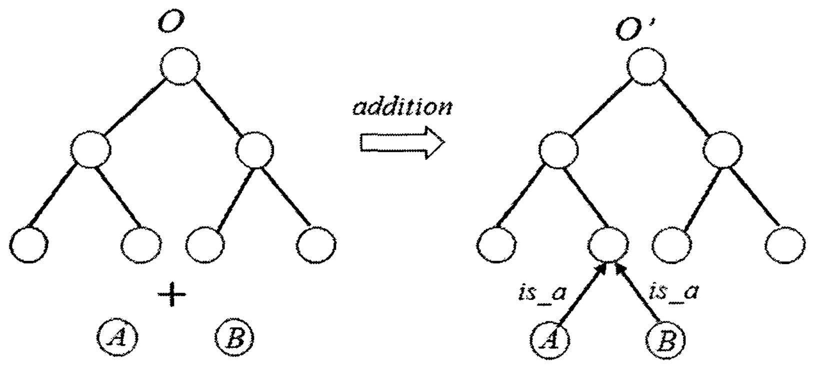


Fig. 3.13. Adding new concepts to an ontology.

There are of course cases where we want to replace a concept with a new one. In this case:

Rule 3: Check whether the change only affects the concept's name or not. If the old and new concepts follow the same semantic (same definition, attributes, and relationships) but carry two different names, the replacement task will be reduced to editing the old concept's name and the related offspring can stay the same or its name can be changed accordingly (in the case of a dependency between the names of parent and child).

Rule 4: If the old and new concepts are not equivalent, the old concept should be deleted and then the new one must be added.

To delete a concept from a hierarchy, several cases can be anticipated:

Rule 5: If the concept is the only concept in the ontology, it can be safely removed.

Rule 6: If the concept is a terminal concept (leaf) within the hierarchy, the deletion can be done by removing the concept and the is-a relation that connects it with the parent concept.

Rule 7: If the concept has offspring, they should be deleted first, along with their taxonomical relationship (Figure 3.14).

Each of the above rules may be decomposed into several simpler rules.

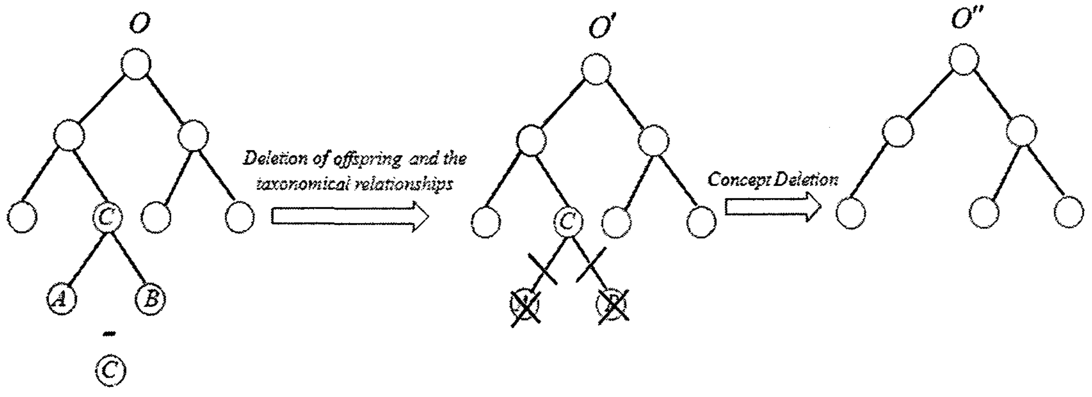


Fig. 3.14. Deleting a concept from an ontology.

One way of studying the process of merging between two ontologies from the same domain is through the union of their algebraically represented hierarchies [LM04]. Figure 3.15 demonstrates the partial merging between ontologies O and O'. We will model ontology merging in Section III.4 using the notion of co-product in category theory.

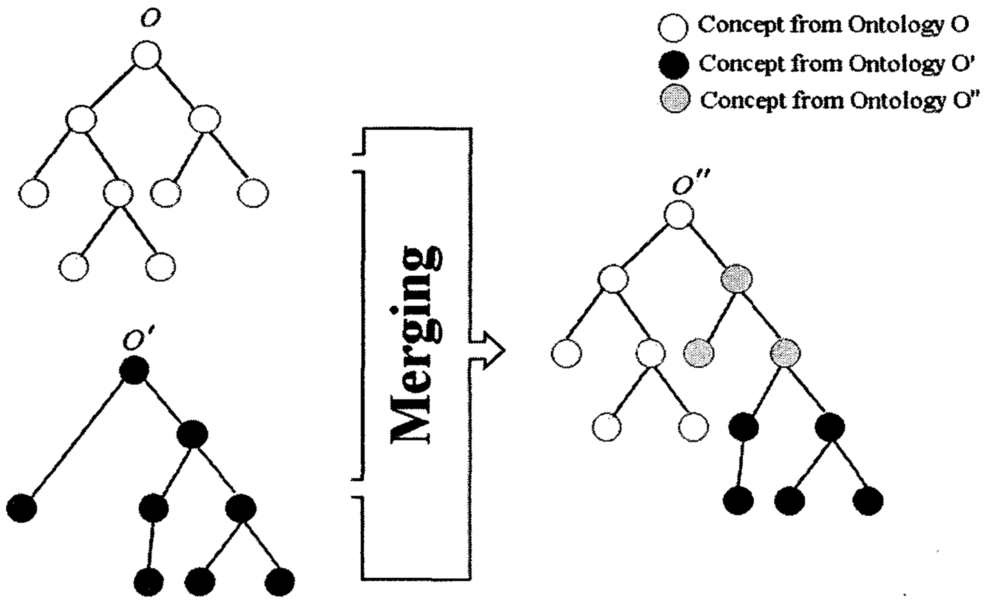


Fig. 3.15. The partial merging between ontologies *O* and *O'*.

III 2.6 Summary of Contributions in Section III.2

As it has been pointed out in Section II.4, and Section II.6, the overreliance on human factors is one of the challenges in current change management practices. Despite the advantages of human intervention in the process of ontology maintenance, including a relative increase in the overall rationality of the system, it does not guarantee reproducible results of a change. Also, it is far beyond the capability of a human to deal with all changes and their impacts in large complex biomedical knowledge-based systems, which are usually integrated from several knowledge sources. Another issue that we mentioned in Section II.6 is inconsistencies among different change models, which is largely originated from miscommunication, and lack of proper conflict resolution mechanism.

In order to address these issues we have made the following contributions in this Section.

- Modeling RLR, a Multi-agent framework, to capture, represent, track and analyze changes through a rule-based reactive and proactive behavior with minimum human intervention;
- Proposing an integrated argumentation framework that enables the different types of agents in RLR to communicate with each other within a dialectic environment to manage the changes and resolve the conflicts.
- Defining a set of evolution rules for generating patterns, which increase the learning capacity, assist in estimating the direction of potential changes, and thus improve the ability for reproduction of the results

In Section III.3, and Section III.4 we will describe how we employ category theory and graph transformation for representation and analysis of the changes in biomedical ontologies, modeling agents' dynamic behavior, and providing a formal semantic for communications, interactions, and operations within the RLR framework.

III.3 Category Theory as Knowledge Representation Formalism

"We often forget that we just made the categories up. Then we treat them as though nature created them with such specificity. Nature didn't."

Curran J., and Takata, S.R., Categorical Thinking, 2002[57]

Several attempts have been made in last two decades to provide a formal foundation for conceptual representation and modeling. In this section, along with some terminological clarification, we discuss the appropriateness of category theory with its mathematical and logical basis for representing dynamic knowledge and tracing changes in ontological structure. In order to orient the reader with a precise definition of categories and some important introductory definitions, we refer to [AL91][58] for additional information.

III 3.1 The Problem of Representation of Change

Knowledge Representation (KR) as a discipline within Artificial Intelligence is generally concerned with the representation and management of knowledge. The existing knowledge representation languages have not been properly adapted to respond to the interactivity and evolvability requirements. Many biomedical ontologies and controlled vocabularies face various challenges when it comes to changing their compositional terminologies and expressions [EBL+03] that usually describe a time-dependent event or

[57] Available at: http://www.students.uwp.edu/academic/criminal.justice/catthink01.htm
[58] Readers can access the entire book freely at: ftp://ftp.di.ens.fr/pub/users/longo/CategTypesStructures/book.pdf (Accessed on 10 March 2010)

process (i.e., the term consuming_medicine_x_after_meal in a drug-food interaction knowledge base).

Set theory, being a powerful, significant, and flexible mathematical formalism, has been widely used for conceptual modeling. However, sets are abstract entities, which exist beyond the realms of time, space, and causality [DHH+01]. Therefore, in order to deal with objects in the world of flux, sets should be accompanied by other complementary frameworks [DHH+01].

A diagrammatic representation is a possible alternative for capturing the behavior of dynamic systems. Diagrams have the ability to intuitively resemble a structural correspondence with the fact (entity or event) they represent, be it visual, propositional (only describes the domain model), or analogical (mimics the domain model). In diagrammatic representations meaning can be conveyed via the diagrams' shape. Diagrammatic representation and reasoning as surveyed in [AB09] have also been used extensively in various application domains, such as: arrow diagrams in algebra and category theory [Pie91]; Euler and Venn diagrams in set theory and logic; circuit, state, and timing diagrams in hardware design [JBA96]; UML diagrams in software modeling; higraphs in specification [Har88]; visual programming languages [Cha90] and visual logic and specification languages [APR98], [HTI90], [OT00]; transition graphs in model checking [BBF+01]; ER-diagrams and hyper-graphs in databases [FMU82]; semantic networks in AI [RN02]; and icons and other pictorial devices in GUIs and information visualization [MS94, Tuf90, War04]. The problem in diagrammatical representation languages is that they are not expressive enough to represent all the behaviors of dynamic ontological structures.

Therefore, due to the limitations of the set theory-based knowledge representation formalisms (including the popular web ontology languages RDFS and OWL) for dynamic conceptual modeling, we have decided to use another type of formalism based on category theory, which is a powerful vehicle to model abstract systems, yet expressive enough to demonstrate their evolutionary behaviors.

III 3.2 Categorization and Categorical Representation

The idea of categorization is central to many disciplines in AI, machine learning, cognitive science, and so on. Categorization is defined in cognitive science as "the process of dividing the world into categories, and usually involves constructing concepts that provide mental representations of those categories" [TF05], and can be done for both observable concepts (e.g., humans, limbs) and non-observable concepts (e.g., genes, disease agents, a process such as injection). In the case of categorizations for non-observables, the process also involves creating concepts for unambiguous rationalization of the real world [TF05]. More formal categorization is also referred to as "any systematic differential interaction between an autonomous, adaptive sensorimotor[59] system and its world" [Har05b]. In this definition, the term "systematic" has been used to exclude arbitrary interactions (e.g., the effects of the wind blowing on the sand) and an "autonomous, adaptive sensorimotor system" means a dynamic system that interacts and changes in time through adaptive changes in the states of the system. "Differential" implies that the categorization process generates a different kind of output with a different kind of input [Har05b].

[59] For more information on sensorimotor activities and systems, see: Rowlands, M. (2006) Sensorimotor Activity. Psyche 12 (1), March 2006. http://psyche.cs.monash.edu.au/symposia/noe/Rowlands.pdf

As can be seen in this definition, categorization has to deal with adaptive state changes across time[60]. In the real world, all the perceptions, experiences, and beliefs, which may be categorized in several specific domains, link up with one another and together shape the webs of our beliefs. In other words, different categories interact with each other firstly because they exist as parts of a single, seamless world view, and secondly, due to reciprocal interaction between the categories, it is not practical to reduce either type to the other [Bev03]. From this insight, one can see that categorization is a natural way to deal with conceptual changes.

III 3.3 What is Category Theory?

Category theory is a relatively new domain of mathematics, introduced and formulated in 1945 [EM45]. Employing formalisms based on logics and mathematics in order to move the Web from being only human understandable to being both human and machine understandable is the known goal of Semantic Web, defined by W3C [CCV+04]. Category theory is closely connected with computation and logic [Whi97], which allows an ontology engineer to implement different states of design models to represent the reality. Categorical notations consist of diagrams with arrows. Each arrow $f: X \rightarrow Y$ represents a function. A Category C includes:

- A class of objects and a class of morphisms ("arrows"), and for each morphism f there exists one object (A) as the domain of f, and one object (B) as the codomain (Figure 3.16 (a)).

[60] To put it simply, the exact same input will not produce the exact same output across time, every time, the way it does in the interaction between wind and sand ("whenever the wind blows in exactly the same direction and the sand is in exactly the same configuration"). Categorization is accordingly not about exactly the same output occurring whenever there is exactly the same input [Har05b].

- For each object A, an identity morphism, which has domain A and codomain A ("ID_A") (Figure 3.16 (b)).
- For each pair of morphisms *f*:A→B and *g*:B→C, (i.e., cod(*f*) = dom(*g*)), a *composite morphism, g o f*: A→C exists (Figure 3.16 (c)).

Representation of a category can be formalized using the notion of a diagram.

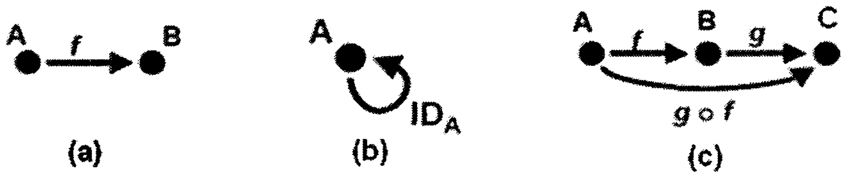


Fig. 3.16. Categorical concepts representation

Category theory has been also defined as:

- A branch of abstract algebra devoted to investigating transformations and compositions of transformations in a highly abstract form [Sym08].
- A toolbox of techniques for illuminating relationships between distinct domains of mathematical investigation [Sym08].

Moreover, the categorical representation of sets unifies the two ancient philosophical problems of continuity and discreteness [Bel06], by offering a deep insight into the shared features of different phenomena. Here are some examples[61] of categories:

- ***Set:*** *the category of sets and set functions.*
- ***Graph:*** *the category of graphs and graph morphisms.*
- ***Cat:*** *the category of categories and functors.*
- *The category of stateful objects and dependencies (object diagram).*
- *The category of states and messages (state diagram).*

[61] The examples are taken from: Category Theory in Haskell theoretical foundations Wiki: http://www.haskell.org/haskellwiki/Category_theory

Some of the primitive constructors of category theory [Mac71] that we use in our framework for ontology change management are as follows: Products, Co-products, Functors, Natural Transformation, Pushout and Pullback. More information on these categorical notions can be found in [AL91].

III 3.3.1 Category Theory, Logic, and Set Theory

Based on [LS81], the traditional "development of logic in an elementary course proceeds with (i) the propositional calculus; (ii) the predicate calculus and (iii) the theory of identity"; however, this definition has been open to criticism [LS81]. There are tight connections between logics and category theory, as studied by Lambek [Lam89] and others [Poi86b, Gol06], and many categorical structures can be studied under logical interpretations. From the logical perspective, a category can be studied as "a deductive system of the objects as formulas and of the arrows as deductions". Today, the study of categorical logic [LS86, Pit00] is quite common between logicians. The categorical framework offers a rich conceptual background for logical and type-theoretic constructions, for representing both syntax and semantics by a category, and a semantic interpretation by a functor [Awo09]. Jacobs also presented some of the relations between categorical logics and equational logic and first order and higher order predicate logic [Jac99]. In addition, many basic concepts of category theory are comparable with the set of notions in set theory. Table 3.3.1 shows some of these pairs of concepts [Gra84].

Table 3.1. A partial list of the pairs of concepts in category theory and set theory (adapted from [Gra84]).

Category Theory	Set Theory
Object	Set
Morphism	Function
Monomorphism	One to one function
Epimorphism	Surjection
Isomorphism	Bijection
Product	Cartesian product
Co-product	Disjoint Union

Figure 3.17 (adapted from [Che04]) demonstrates the world from different perspectives of category and set theories.

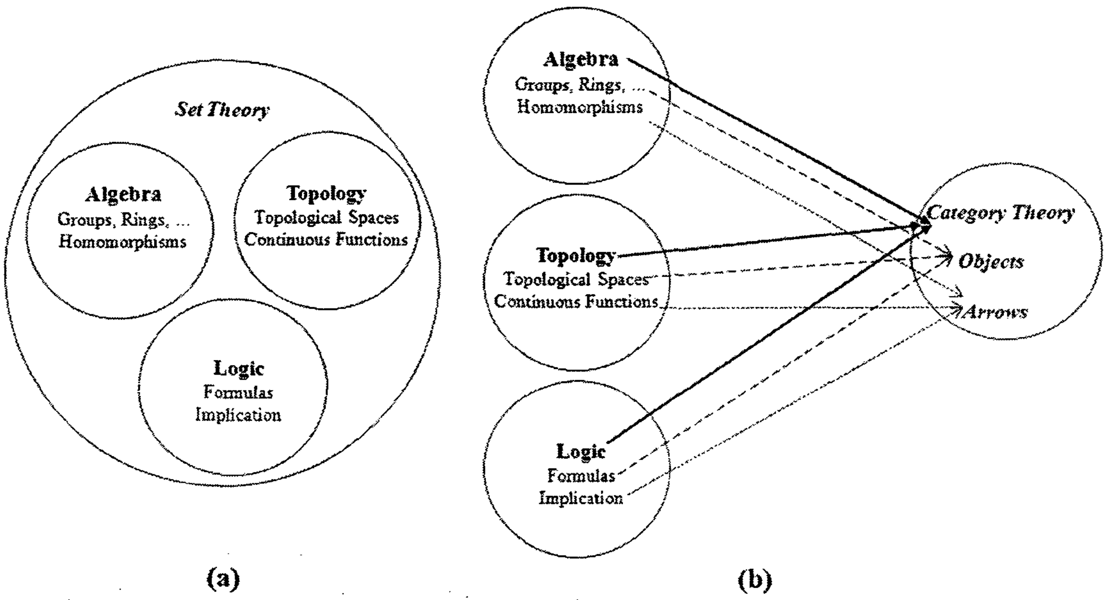


Fig. 3.17. (a) The world from the set theory perspective; and (b) The world from the category theory point of view (adapted from [Che04]).

The declarative approach offered by category theory describes objects only in terms of their relationships and interactions with other objects, without the necessity of knowing about the internal structure of objects. This is one of the distinct features of categories in comparison with sets or logic theories [Gog91, DC94]. For more information on the interaction between category theory and set theory, one may refer to [Bla84].

III 3.3.2 Why Category Theory?

Using categories, one can recognize certain regularities to distinguish a variety of objects, capture and compose interactions and identify patterns of interacting objects in a declarative way and extract some invariants in their action, or decompose a complex object into basic components [EV06]. They offer a graphical yet formal notation for knowledge representation. Categories are also able to identify patterns that recur over and over in a changing system [KKR06]. Some other reasons for using category theory in our framework, as stated by Adamek, et al. [AHS90], are abundance, precise language, and convenience of symbolism for visualization. Categories can be found in many places in mathematics (e.g., sets, vector spaces, groups, and topological spaces all naturally give rise to categories). It also provides a language to precisely describe many similar phenomena that occur in different mathematical fields with an appropriate degree of generality. For example, it allows one to precisely make distinctions via the notion of natural isomorphism. It also provides a unified language to describe topological spaces via the notion of concrete isomorphism [AHS90]. In addition, Categorists have developed a symbolism for visualizing complicated facts by means of diagrams.

In a category, one can only have access to the processes, the arrows (similar to an API in software engineering terms), and it is not necessary to know what the available objects are made of or how they have been created [Alp07]. This is important if operating in an interactive semantic web (or Web 2.0) environment, where the potential users do not usually have direct and transparent access and control over the existing objects. In fact, categorically, the behavior of the objects is much more important than their identities; this is why definitions in category theory are usually very abstract and conveyed through isomorphisms (if two objects behave the same way in an API, they must be considered similar based on the given definition of similarity). The abstractness of the definition can facilitate reusing the definitions in different contexts. In addition, employing the concept of isomorphism enables us to generalize the definition of similarity [Alp07]. As well, categorical entities are "subject to a constant process of enrichment, which bears a certain resemblance to evolution" [Kai05].

For these reasons, category theory has great potential to be used as an abstract mathematical vehicle to represent, track, and analyze changes in ontologies, without considering the type of underlying knowledge representation formalism or any implementation language (representation independence [Gog91]).

III 3.3.3 Applications of Category Theory

Category theory has been extensively used in a wide range of applications. It is already being applied in physics, linguistics, philosophy, and different disciplines in computer science, including XML semantic analysis [CD02] and XML database engineering [Tot08]; object databases and the Semantic Web [Güt04]; conceptual modeling [HLW97, WH99, CHR08]; ontology and knowledge-base modeling [HC06, KHE+05, JR08]; designing multi-agent systems [Pfa07b]; neural networks' architecture [HOY+09]; knowledge engineering and cognition [HC04]; analyzing living systems [MCF81, Kai05]; biology [Ros58, MCF81, Mac01, EV06, LSA+06, Din08]; theoretical neurobiology [Pfa07a]; neural modeling and graphical representations [Hea00]; philosophy [Per06]; linguistics [Van06]; software engineering [WH99, Fia04]; object oriented visual modeling [DW08]; managing software specifications [WE98]; managing software component dependencies [Guo02] and model merging [SNS+07]; cognitive development [HW80]; data refinement [JNP09]; machine semantics [Hin08]; and so forth.

III 3.3.4 Tools Supporting Category Theory

As mentioned, category theory provides an abstract formalism, which does not pay much attention to the operational details and internal interactions of a system. This feature is one strength of this formalism, but the high level of abstractness makes the actual usage of category theory and its constructors in software tool applications tricky [Men99].

There are some tools, however, such as Specware[62] [MA01], which has been used in [WH00] for software maintenance at the requirements level, and GDCT[63] [BRG+06], that are available to study a category and answer queries about isomorphism, product, coproduct, pushout, pullback, creating sum and product, checking the equality of arrows, testing whether an object is initial or terminal, and so on (Figure 3.18). Also, there are software packages for implementing categorical concepts and structures in Haskell[64] (an advanced, purely functional programming language) [HHJ+07]. For instance, category-extras[65] [Men04] offers a collection of modules implementing various constructors inspired by category theory.

[62] http://www.kestrel.edu/home/prototypes/specware.html
[63] http://mathcs.mta.ca/research/rosebrugh/gdct/
[64] The Haskell Home Page: http://haskell.org/
[65] http://hackage.haskell.org/cgi-bin/hackage-scripts/package/category-extras-0.1

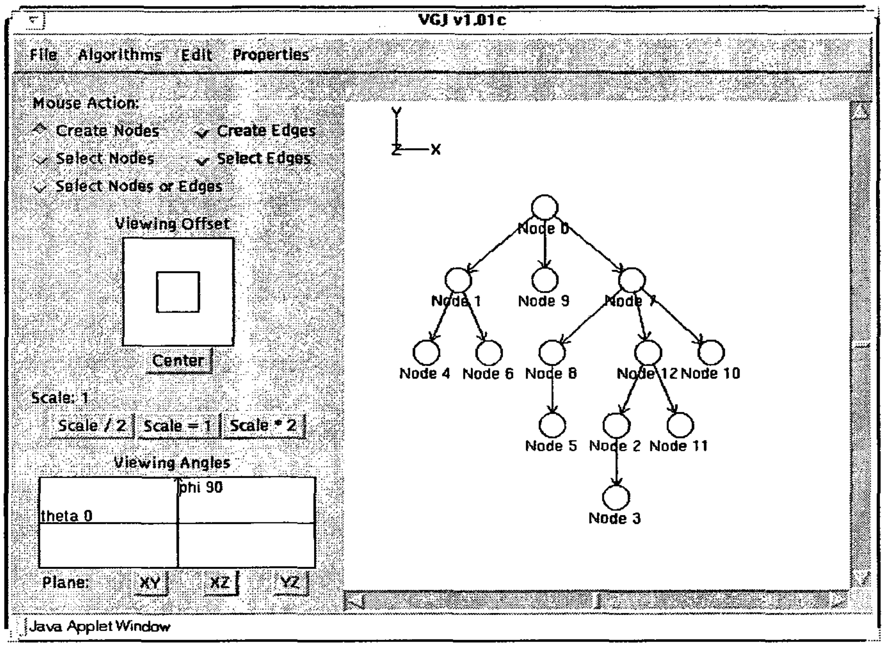


Fig. 3.18. A Screenshot representing a hierarchical tree structure [Source: from the introduction to Graphical Database for Category Theory (GDCT)[66]].

III 3.3.5 Categories, Conceptual Data Modeling, and Ontologies

The concept of ontology is based on the categorization of things in the real world. 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 [HC06]. In fact, we use category theory to represent ontologies as a modular hierarchy of domain knowledge. Ontological relationships represented using category theories are considered to be directed [KHE+05] to show the direction of information. These "relationships", which preserve the conceptual hierarchies and the relations, are known as "morphisms".

[66] http://www.eng.auburn.edu/department/cse/research/graph_drawing/manual/tree.gif

The research presented in [BM99] employed categories for algebraic specifications and the representation of ontologies via morphisms. The authors in [CDJ01] described a categorical method for formalizing the relationship of abstraction and refinement for abstract models of enterprise information systems and for managing databases (e.g., through view updates [JR01]). Kent [Ken04] presented a categorical axiomatization of the first-order model theory[67] for representing ontologies as hypergraphs with respect to formal concept analysis (FCA)[68]. Hitzler et al. [HEK+06] proposed an approach for analyzing the alignment between ontologies using category theory. Johnson and Rosebrugh [JR08] recently applied their method based on universal algebra and category theory to the analysis of interoperability between ontologies using the notions of "view" and "view update".

We now present the basic ideas concerning the generic ontological representation in a categorical frame. Here is a simple intuitive example: consider a world consisting of categories of families, with persons as objects and the family relations that exist between them as morphisms. One may use family.owl[69] knowledge base for the purpose of initial conceptualization. Figure 3.19 shows the related T-Box, A-Box along with the set of role assertions for this example (adapted from [HMW04]).

[67] See: First-order Model Theory. Stanford encyclopedia of Philosophy. First published Sat Nov 10, 2001; substantive revision Tue Apr 28, 2009. http://plato.stanford.edu/entries/modeltheory-fo/

[68] http://www.upriss.org.uk/fca/fca.html

[69] http://www.owldl.com/ontologies/family.owl

T-box:

$$\begin{aligned}
has\_child &\sqsubseteq has\_descendant\\
inv\_has\_child &\doteq inv(has\_child)\\
has\_father &\sqsubseteq inv\_has\_child\\
has\_mother &\sqsubseteq inv\_has\_child\\
man &\sqsubseteq person\\
woman &\sqsubseteq person\\
brother &\sqsubseteq man\\
parent &\doteq person \sqcap (\exists has\_child.person)\\
mother &\doteq woman \sqcap parent\\
grandmother &\doteq mother \sqcap\\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}$$

Role Declarations:

$transitive(has\_descendant)$
$attribute(age, integer)$
$feature(has\_father)$
$feature(has\_mother)$

A-box:

$woman(alice)$, $woman(betty)$, $brother(charles)$,
$(\leq 1 has\_sibling)(charles)$, $has\_sister(eve, doris)$,
$has\_child(alice, betty)$, $has\_child(alice, charles)$, $has\_child(betty, doris)$,
$has\_child(betty, eve)$, $has\_sibling(charles, betty)$, $has\_sister(doris, eve)$

Fig. 3.19. A knowledge base representing the domain of family using DL axioms (adapted from [HMW04]).

The categorical representation for the Smith family by considering people as the objects and the family relationships as morphisms can be illustrated as shown in Figure 3.20.

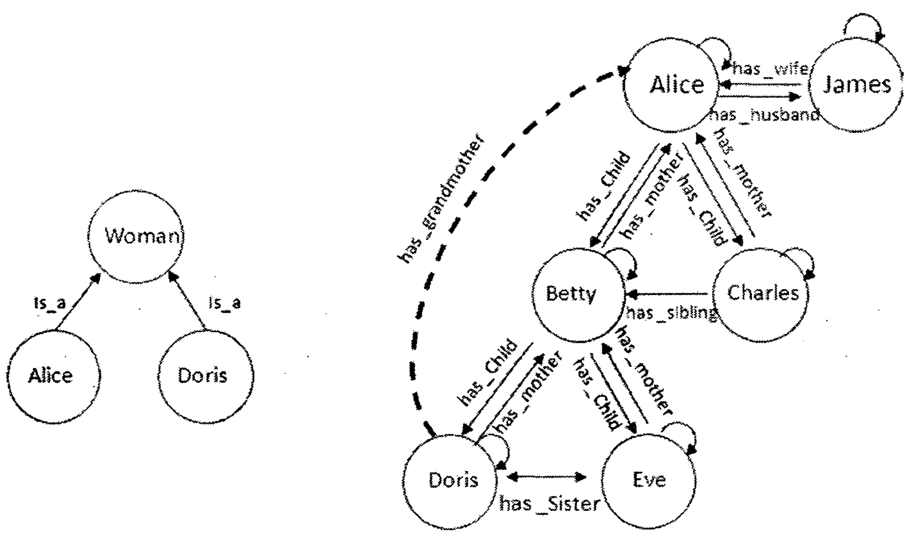


Fig. 3.20. The categorical representation of the family knowledge base.

As it can be seen in Figure 3.20 both identity and composition laws are valid; for example there is an identity morphism for object Doris such that Doris $\longrightarrow$ Doris, or

"Doris is Doris", which is a true statement. Also the composition for the following diagram for example:

$$\text{Doris} \xrightarrow{has\_mother} \text{Betty} \xrightarrow{has\_mother} \text{Alice} \text{ yields to } \text{Doris} \xrightarrow{has\_grandmother} \text{Alice}.$$

As you may noticed by now, this representation resembles the A-box diagram in description logics sense, which enables one to do some sort of assertions. In dealing with internal structures of the objects, categories might not fully reveal their capabilities however, category theory, as we will see throughout this chapter, has a set of universal constructors that help us in dealing with more general and abstract problems.

III 3.4 Categories for Dynamic Systems: The Birdwatching Approach

"When you know what the habitat and the habits of birds are watching them is so much more interesting."

The Beginners Guide to Bird Watching[70]

Since the existing biomedical knowledge bases are being used in various organizational and geographical levels (i.e. institutional, local, regional, national and international), any change management framework should be able to address this decentralization and distribution nature. As mentioned in Section II 3.4.2, one of the critical tasks in any change management framework is traceability. To explain our proposed method for change management in RLR more intuitively we use a conceptual metaphor based on Birdwatching activity. Birdwatching as a recreational and social activity is the process of observation and study of birds through a particular time frame using different auditory

[70] http://birdwatchingforbeginners.info/

devices. Figure 3.21 shows a sequence[71] of typical activities recommended for Birdwatching:

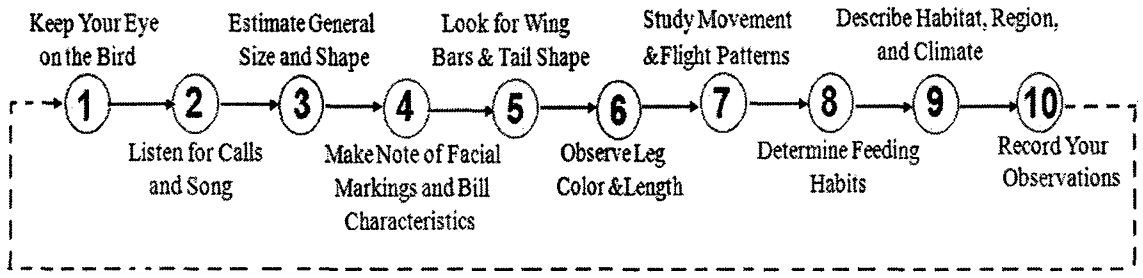

Keep Your Eye on the Bird
Estimate General Size and Shape
Look for Wing Bars & Tail Shape
Study Movement &Flight Patterns
Describe Habitat, Region, and Climate
1
2
3
4
5
6
7
8
9
10
Listen for Calls and Song
Make Note of Facial Markings and Bill Characteristics
Observe Leg Color &Length
Determine Feeding Habits
Record Your Observations

Fig. 3.21. A series of activities in Birdwatching.

Looking at the above list one can discover that the central idea of Birdwatching, which is tracking the position of the birds at different time points and predicting their path by deriving a flight pattern based on recorded observed information, is quite close in spirit to monitoring any dynamic spatial-temporal system. Inspired by this metaphor we can explain how the functionalities within the RLR framework can assist to fulfill the Birdwatching's goal. In RLR the change capture agents are responsible for tasks 1 and 2 (in Figure 3.21), the changes logs store the information about the changes (task 4), the learning agents starts with limited knowledge (task 5 and 6) and tries to improve itself by gaining inferred knowledge (tasks 8 and 9) based on the semantics provided by the ontological backbone. Moreover the learning agents along with negotiation agents and reasoning agents can derive a pattern of changes using the information stored in the change logs and the background and derived knowledge (task 7). Using this pattern one can achieve a practical estimate for expected changes (task 3). Finally the result of the observation will be stored to be used for future inferencings (task 10), and to choose an appropriate pattern (task 7) in the reproduction phase.

[71] Bird Watching Tips for Beginners: http://animals.about.com/od/birding/tp/birdidtips.htm

The Galileo's dialogue[72] for explaining motion for the first time stated that for capturing and tracking a moving object one needs to record the position of that object in each instance of time. Categorically speaking [LS09], studying any motion and dynamism needs an analysis on mapping from a category of times to a category of spaces. Figure 3.22 demonstrates such mappings.

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. In our approach, as we will show in Section III 3.5.5.2, we represent conceptualization of things indexed by time and we use categorical constructors for capturing the states of ontologies at different time points.

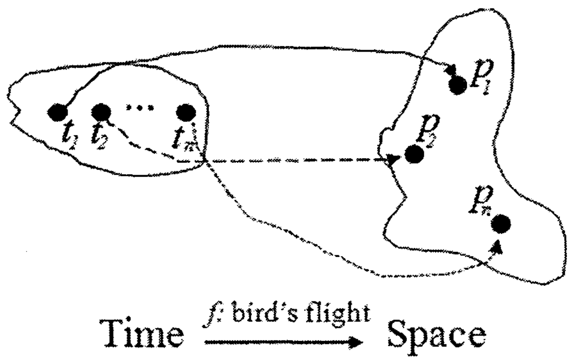


Fig. 3.22. A map from category of time points to category of positions in space for describing a bird's flight in categorical perspective [LS09].

Similarly, the behavior of an individual ontological element (state) can be monitored by function g, which maps the time points to the set of positions for the element in the ontology. $\text{Time} \xrightarrow{g:element's\_behavior} \text{Ontology}$

Moreover an ontology has different states and behaves in a distributed semantic web environment. $\text{State} \xleftarrow{i:has\_state} \text{Ontology} \xrightarrow{h:behavies} \text{Semantic Web}$

[72]Galileo, G. (1632) Dialogue Concerning the Two Chief Systems of the World - Ptolemaic and Copenican. http://www.gap-system.org/~history/Extras/Galileo_Dialogue.html

Composing these diagrams one can see that a behavior of an individual ontological element should be studied in close relations with time, the state and the behavior of the whole ontological structure in a semantic web environment (Figure 3.23).

$$\begin{tikzcd}
 & \textbf{Time} \arrow[d, "g{:}element's\_behavior"] & \\
\textbf{State} & \textbf{Ontology} \arrow[l, "i{:}has\_state"'] \arrow[r, "h{:}behaves"] & \textbf{Semantic Web}
\end{tikzcd}$$

Fig. 3.23. A temporal diagram for studying the behavior of ontologies.

III 3.5 Category Theory as an Algebraic Formalism for the RLR

Category theory facilitates representing, tracking, and analyzing changes in ontologies. It can also be considered as a supplementary formalism alongside other formalisms to capture the full semantics of evolving bio-ontologies. Categorical constructors allow one to describe different relationships between the entities of a dynamic system, as well as offering a formal ground for representing various changes, actions, and operations, such as addition/deletion, merging/splitting, mapping, alignment, and integration. Category theory puts most of its effort into describing the relations between elements of a dynamic system (morphisms) rather than the system's elements (objects). Depending on the level of abstraction, different types of categories (i.e., categories of classes and properties in the lower level of abstraction, and categories of ontologies and contexts in the higher level) can be defined for modeling ontological structures.

III 3.5.1 The Category *Class*

Classes can be defined as a set of properties (attributes and methods) shared by a set of individuals within an equivalence class. Whitmire [Whi97] was one of the few who identified a model based on category theories for object oriented applications measurement. Here we follow his approach for demonstration of ontological elements. We can define the category Class with attribute domains as objects and set-theoretic functions as arrows. We can also define some operations for a class. In ontology, a concept or an instance can transit from one state to another based on its behavior in response to a change. An event can be formally modeled as an ordered pair $E = <St_1, St_2>$ [EV06]. St_1 is the start state and St_2 is the end state. St_1 and St_2 are not necessarily distinct and they might refer to the same state [Wan89] (when an event does not change state). The category *Class* is defined with three types of objects and three types of arrows. The three types of objects are [Whi97]:

1- The state space for the class, labeled with the name of the class.

2- The domain sets for the attributes in the class, labeled with the name of the domain.

3- The steady states (a situation in which the relevant variables are constant over time) for objects of the class, labeled with the name for the state used in the domain.

Three types of arrows are: projection (π), selection (σ), and operation arrows.

$$\begin{tikzcd} & A2 & A3 & \\ A1 & & & A4 \\ & A^n = C \arrow[ul, "\pi_1"] \arrow[uu, "\pi_2"] \arrow[uur, "\pi_3"'] \arrow[urr, "\pi_n"'] & & \end{tikzcd}$$

(a)

$$\begin{tikzcd} St \arrow[d, "\sigma_j"'] \arrow[dr, dashed, "\sigma_{ij}"] & \\ St_j \arrow[r, "\pi_{ij}"'] & a_i \end{tikzcd}$$

(b)

Fig. 3.24. (a) Representation of the n attribute domains, and the state space of class C (A^n), when π_n determines the value of nth attribute (adapted from [Whi97]); **(b)** σ_j has been defined to select a state (here jth state) from the state space [Whi97]; π_{ij} retrieves the value of ith attribute in state j; which also can be inferred directly from $\sigma_{ij} = \sigma_j \circ \pi_{ij}$. As it can be seen this inference causes the triangle at the right side to commute.

The projection arrow for each attribute is drawn from the state space to the attribute domain and labeled with the name of the attribute. The value of the *i*th attribute is provided by π_i. A selection arrow for each state is drawn from the state space to the state and labeled as σ_x where x is the name of the state [Whi97]. An operation arrow for each event $E = <St_1, St_2>$ drawn from St_1 to St_2 and labeled with the name of the method to which the operation corresponds (Figure 3.25). One can select a state using the selection function σ_i which gives the *i*th state.

$$\begin{tikzcd} St_1 \arrow[r, "op_1"] & St_2 \end{tikzcd}$$

Fig. 3.25. Operation arrow op_1 denotes a valid operation in the defined category and demonstrates a transition of an object from one state to another (e.g. from St_1 to St_2). This operation is only valid within the determined state.

III 3.5.2 Operations on the *Class*

Most common operations during ontology evolution are adding a class, deleting a class, combining two classes into one, adding a generalization relationship, adding an association relationship, adding/deleting a property, and adding/deleting a relationship.

Figure 3.26 represents adding a class to our available structure and Figures 3.27 (a) and Figure 3.27 (b) demonstrate adding and deleting a relationship respectively.

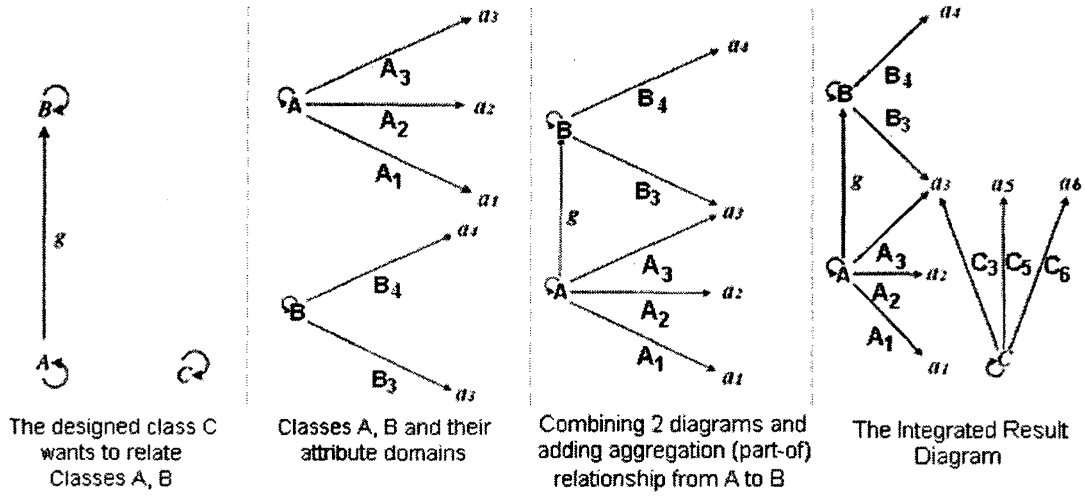

The designed class C wants to relate Classes A, B
Classes A, B and their attribute domains
Combining 2 diagrams and adding aggregation (part-of) relationship from A to B
The Integrated Result Diagram

Fig. 3.26. Adding a class to the available structure based on categorical operation following Whitmire's approach (adapted from [Whi97]). The represented aggregation[73] relation between the classes A and B implies the part-of relationship between them. The classes A, B, C in the left hand side has been represented in the higher level (external view), while during the rest of the operations their attributes and internal structures have been demonstrated.

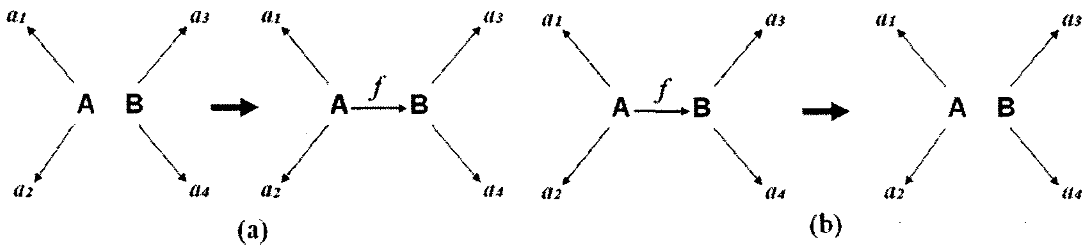


Fig. 3.27. (a) ADD a Relationship between two classes A and B (b) Drop a Relationship (adapted from [Whi97])

III 3.5.3 Categories Operation and States

We define the category Operation with the set of defined operations and attributes as objects and the relationships between them as morphisms. The morphisms can be

[73] There are different types of relations in an ontological structure such as subsumption (parent-child relationship), association (relationships between individuals of different classes), and aggregation (a type of association relation, which causes the semantic enrichment of the related classes; i.e. part-whole relation).

considered as pre/post conditions, which allow an operation to be executed. For example, as demonstrated in Figure 3.28, an object Op_1 can be related to other objects A_1 and A_2 (indicating attributes) through morphisms pre_1 and pos_2 (indication pre- and post conditions). It is also useful to add other morphisms such as message links, for communication and comment exchange purposes.

$$\begin{tikzcd}
 & Op_2 & \\
 & Op_1 \arrow[u, "\textbf{Message}"'] \arrow[dl, "\textbf{Post}"'] \arrow[dr, "\textbf{Pre}"] & \\
A_1 & & A_2
\end{tikzcd}$$

Fig. 3.28. Category operation with operation/attributes as objects and messages/conditions as morphisms.

The message links may also pass parameters to other operations and therefore constribute in the definition of pre/post conditions for that operation. In addition, we define the category State with states[74] of ontologies as objects and the operations, which determine the behavior of an evolving structure, as morphisms.

$$\begin{tikzcd}
 & \begin{matrix} C_1 = X \\ R_1 = Y \end{matrix} & \begin{matrix} C_1 = X \\ R_1 = Y' \end{matrix} & \begin{matrix} C_1 = X' \\ R_1 = Y' \end{matrix} & \\
\cdots & St_1 \arrow[r, "op_1"'] & St_2 \arrow[r, "op_2"'] & St_3 & \cdots
\end{tikzcd}$$

Fig. 3.29. Ontology O(C, R)[75] transits to different states due to the different operations. One specific set of concepts and relationships from this ontology may have different values in different states.

Figure 3.29 represents an example of transition of an evolving ontology through different states. Since our primary purpose for defining this category is to trace an impact of a change, through different versions of one ontology, here we only consider consistent

[74] By state we mean the situation, in which a system is consistent and stable.
[75] C and R are representing classes and relationships respectively.

states; however we can extend our definition to cover both consistent and inconsistant states for the sake of conflict detection and inconsistency resolution. Another extension is also possible through defining functors, which let us analyze the behavior of a system while for example mapping two different categories of state. More on functors can be found in Section III 3.5.5.2.

The introduced categories (operation and state) together assist us to analyze the behavior of an evolving structure, and monitor the impact of one particular change based on the complexity and coupling of this structure.

III 3.5.3 The Category Ontologies

The category Ontologies can assist in analyzing different behaviours and interactions with other ontologies, be they independent or various versions of the original ontology. The category Ontologies can be represented either by simple categorical notation, with ontologies as objects and the links between them as morphisms, or in a nested fashion, using a special categorical constructor called "functor".

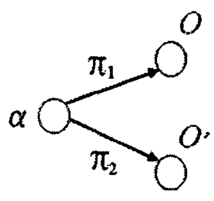
III 3.5.4 Operations on Ontologies

As we noted in Section II 3.3, ontology change management is composed of several sub-fields, including ontology alignment, mapping, merging, and integration, which are inevitable in a distributed environment. In this section, we discuss how category theory can be used as a visual formalism to model some of these processes. However, it is not so easy to generalize the categorical approaches, whose descriptions are mostly mathematical, in the context of practical ontology change management, and one needs to have a preliminary familiarity with this formalism. We try to limit our approach to the most common categorical notions from a computer science perspective rather than purely mathematical techniques. The reader can refer to [AL91, Awo06, and LS09] for more information on category theory. For this section, we consider the category $\mathcal{C}$ representing the ontologies[76] (O_1, O_2, ..., O_n) as the object and the transition functions between these ontologies as the morphisms. In this setting, categorical composition and identity morphisms can be understood through the notions of transitivity and reflexivity, respectively. The internal structures of objects are entirely ignored in this categorical representation. In the rest of this section, we will show that despite the simple appearance of a category, the semantics and derived results of categorical elements employed for ontology change management are amazingly rich, and often can represent the entire knowledge about a set of its defined objects (here, ontologies).

III 3.5.4.1 Alignment and Mapping between Ontological Structures

Basically, two ontologies can be aligned by first specifying the most similar (syntactically and semantically) components in both ontologies, through a binary relation. A categorical framework has been proposed by [BEE+04] to describe alignments in ontologies. Categorically, their analysis began with the assumption that there exists an object α and a pair of morphisms to the two ontologies O and O' in such a way that α is the most specific ontology that approximates both O and O'. Then the morphisms, representing binary relations that describe an alignment, are defined as a set of pairs of entities, which represents one entity from ontology O and another from ontology O' via a pair of projection functions (π_1, π_2) [ZKE+06].

[76] This approach is not limited to formal ontologies and we can use any hierarchical controlled vocabularies as object.

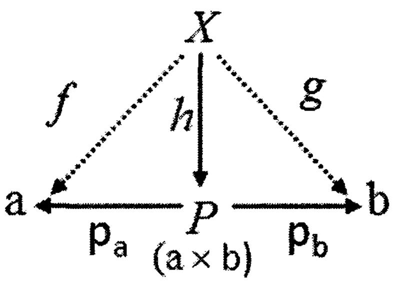


When analyzing two ontologies, two types of taxonomical relationships may be seen in their subsumption structures, namely the parent-child relationship and equivalency (or isomorphism). A map *f*: A $\rightarrow$ B is called an isomorphism [LS09] if there exists *g:* B $\rightarrow$ A (inverse of *f*) for which $g \circ f = \mathrm{Id}_A$ and $f \circ g = \mathrm{Id}_B$. In this case *f*: A $\rightarrow$ B is called invertible [Mac71]. The objects A and B are called isomorphic (equivalent) if there is at least one isomorphism *f*: A $\rightarrow$ B. The functions that change one ontology into another can be considered morphisms between these two ontologies. Regardless of the type of these change functions, the identity function and the composition of the change function can always be defined. The isomorphism between ontologies can be studied by applying the knowledge, implied by the change functions, backward and forward between the ontology versions.

In [JPV+98] and [ZKE+06], ontology alignment has been addressed through Cartesian products (resembling the intersection between two structures). Products [AL91] in category theory generalize the notion of a Cartesian product of sets, but unlike the sets, they focus on morphisms and their properties rather than the internal structure of the objects. Products in category theory are generalization of the notion of Cartesian product of sets, and are defined [AL91] as follows:

Let *C* be a category, and consider a and b as two objects in this category. The *product* of a and b is an object *P* representing (axb) together with two morphisms p_a: *P* $\rightarrow$ a and p_b: *P* $\rightarrow$ b, such that for any object $X \in C$ and each pair of maps *f: X* $\rightarrow$ a and *g:*

$X \rightarrow b$, there exists exactly one (unique) map $h: X \rightarrow P$ for which both $f = P_a h$ and $g = P_b h$ holds (means the following diagram in Figure 3.30 commutes). Two maps p_a and p_b are called projection maps for the product and we may refer to them as π_a and π_b respectively.



$$\begin{tikzcd} & X \arrow[dl, dotted, "f"'] \arrow[d, "h"] \arrow[dr, dotted, "g"] & \\ a & P \; (a \times b) \arrow[l, "p_a"] \arrow[r, "p_b"'] & b \end{tikzcd}$$

Fig. 3.30. A diagrammatical representation of categorical product.

As an example of product in the category of sets, assume two sets a: {x, y, z} and b: {1, 2} based on the definition. The following diagram (Figure 3.31) represents the categorical product P: a×b.

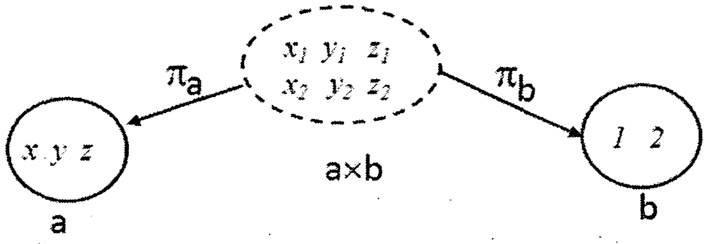


Fig. 3.31. An example, demonstrating the categorical product in the category of sets.

The product is useful for analyzing the alignment of two ontological structures, but it is not fully appropriate for ontology merging. It seems that another categorical constructor called *Coproduct*, which performs a sum (or union) operation between ontological structures, would be better suited for merging.

Definition (Coproduct) [AL91]: Let C be a category, and consider a, b to be two objects in this category. The *Coproduct* of a, b is an object q together with two morphisms q_a: a $\rightarrow$ q, q_b: b $\rightarrow$ q such that for any object $X \in C$ and each pair of maps *f*: a $\rightarrow$ X and *g*: b $\rightarrow$ X, there exists exactly one (unique) map *h*: q $\rightarrow$ X in the way that the following diagram commutes (Figure 3.32).

$$\begin{tikzcd}
 & X & \\
a \arrow[ur, dotted, "f"] \arrow[r, "q_a"'] & \underset{(a+b)}{q} \arrow[u, "h"] & b \arrow[ul, dotted, "g"'] \arrow[l, "q_b"]
\end{tikzcd}$$

Fig. 3.32. A diagrammatical representation of categorical coproduct.

The categorical coproduct is also unique up to isomorphisms[77]. As an example, for obtaining coproduct in the category of sets one can consider the disjoint union between the sets.

Mapping by means of binary relations can be achieved for ontologies O_1 and O_2. As shown in [ZKE+06], categories can be recruited for ontology alignments on the abstract level in two forms, V-alignment (for simple alignments) and W-alignment (more expressive for more complex alignments). A V-alignment between two ontologies O_1 and O_2 has been defined as a triple $<O, p_1, p_2>$ such that O is an ontology, and p_1: $O \rightarrow O_1$ and p_2: $O \rightarrow O_2$ are two refinement functions. In a W-alignment, a set of bridge axioms [BEF+06], for defining a bridge ontology between the two ontologies to be aligned, and

[77] "An object A having a certain property $\varphi(A)$ is unique upto isomorphism if given any other object B such that $\varphi(B)$, there exists an isomorphism *f* between A and B" [Enc04] (e.g., in category theory terminal objects are unique upto isomorphism).

two V-alignments[78], for aligning each of the two ontologies with the bridge ontology, have been employed to cover more types of relationships [ZKE+06].

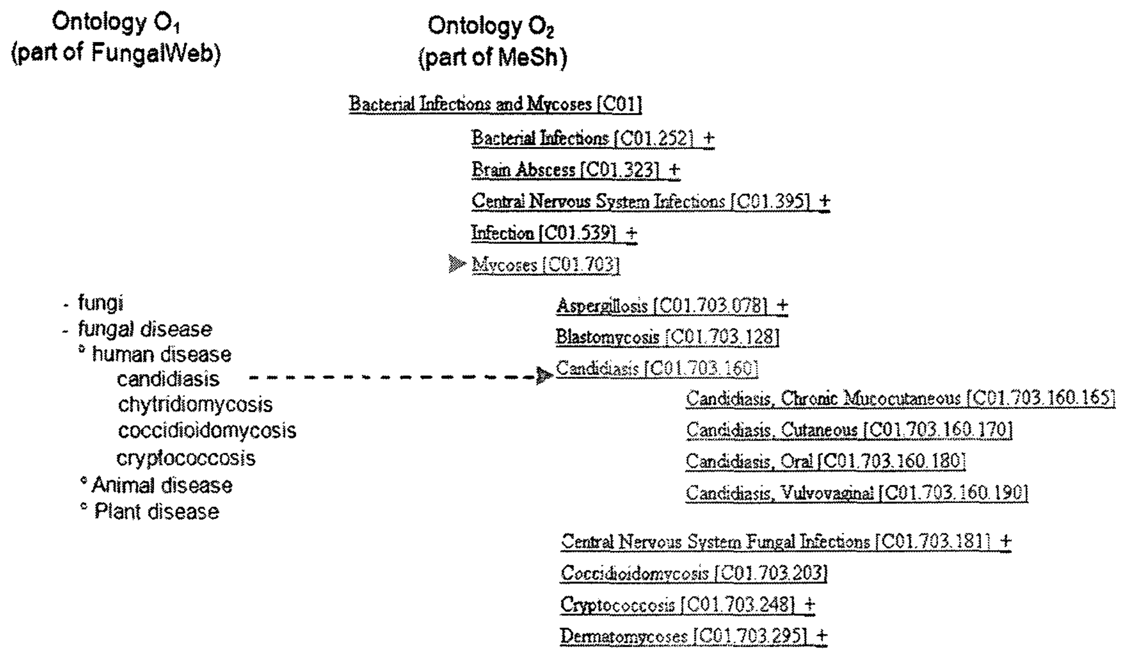

Ontology O_1
(part of FungalWeb)
Ontology O_2
(part of MeSh)
Bacterial Infections and Mycoses [C01]
Bacterial Infections [C01.252] +
Brain Abscess [C01.323] +
Central Nervous System Infections [C01.395] +
Infection [C01.539] +
Mycoses [C01.703]
Aspergillosis [C01.703.078] +
Blastomycosis [C01.703.128]
Candidiasis [C01.703.160]
Candidiasis, Chronic Mucocutaneous [C01.703.160.165]
Candidiasis, Cutaneous [C01.703.160.170]
Candidiasis, Oral [C01.703.160.180]
Candidiasis, Vulvovaginal [C01.703.160.190]
Central Nervous System Fungal Infections [C01.703.181] +
Coccidioidomycosis [C01.703.203]
Cryptococcosis [C01.703.248] +
Dermatomycoses [C01.703.295] +
- fungi
- fungal disease
human disease
candidiasis
chytridiomycosis
coccidioidomycosis
cryptococcosis
Animal disease
Plant disease

Fig. 3.33. Two ontologies covering a specific domain with different granularities.

Figure 3.33 demonstrates two ontologies O_1 and O_2 that are simply representing part of the FungalWeb and MeSH ontology, respectively. These two taxonomical structures can be linked together via a mediator ontology, which is built based on the similarities in both ontologies O_1 and O_2. As noted before, the two projection mappings $\alpha \rightarrow O_1$ and $\alpha \rightarrow O_2$ give us the intended alignment in its simplest situation (V-alignment). When there are matching concepts in both ontologies, for example, the existence of two synonymous concepts, "*haole rot*" in one ontology and "*tinea versicolor*" in the other, each of these concepts can be considered as a gluing point between two ontologies. When there is no exact match for an entity from ontology O_1 into ontology O_2, we may need to consider

[78] If one needs to obtain the alignment between more than two ontologies, the number of simpler alignments (i.e., V-alignments) and composed alignments (i.e., W-alignments) would increase accordingly.

other closely related elements that share the most similar properties with that entity. For example, the concept "*tinea versicolor*" in the extended version of the FungalWeb Ontology may not have an exact equivalent in the "Human disease ontology"[79], but there is a concept "*fungal infection*" in the "Human disease ontology" that can be considered a parent class for the concept "*tinea versicolor*" in the FungalWeb ontology. This subsumption (parent-child) relationship between two concepts would be later considered as one of the major gluing points for merging (partially) the two ontologies. One possibility for merging these ontologies is through an artificial gluing concept (e.g., the concept "*fungal infection* v *tinea versicolor*") in the mediator ontology. By creating the mediator ontology (O_m) and employing the so called W-alignment [ZKE+06], we can use the composition condition in category theory to generalize the notion of alignment in such a way that if there are alignments between O_1 and O_m and between O_m and O_2, then one can get the alignment between $O_1$and O_2[80].

Here let us look at two important categorical notions called Pushout and Pullback. The *pushout* for two morphisms *f*: A→B & *g:* A→C is an object D, and two morphisms i_1: B→D & i_2: C→D exist such that the square commutes (Figure 3.34 (a)). D is the initial object in the full subcategory of all candidates D' (i.e., for all objects D' with morphisms j_1 and j_2, there is a unique morphism from D to D'). The *pullback* (also known as "Cartesian square") for two morphisms *f:* A→C and *g:* B→C is an object D, and two morphisms i_1: D→A and i_2: D→B, such that the square commutes. Here D is the terminal object in the full subcategory of all such candidates D' [Eas98] (Figure 3.34 (b)).

[79] http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/phenotype/human_disease.obo
[80] Categorically speaking, consider α and β as the alignments between O_1 and O_m and between O_m and O_2 respectively, then the alignment between α and β can be described as $\gamma = \alpha \circ \beta$.

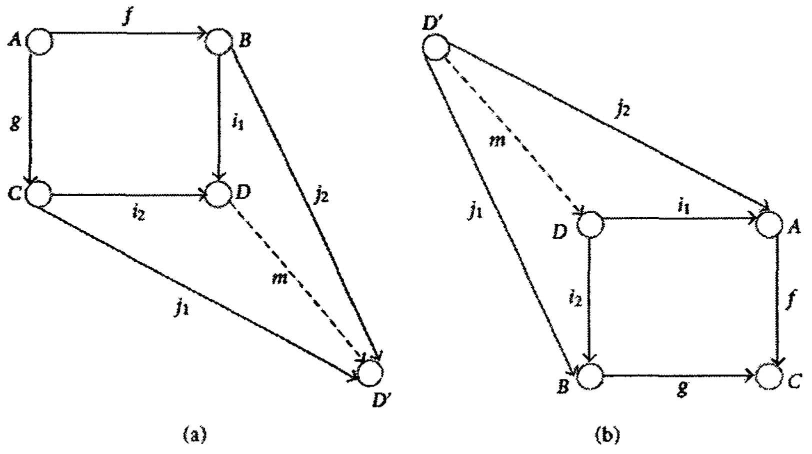


Fig. 3.34. Two categorical constructors (a) Pushout, (b) Pullback.

For example as represented in Figure 3.35 (a) in the category of sets pushouts can be defined as union of pairs of elements from B and C that are the images of the same element in A, plus the rest of the elements of B and C. The pullback of can be defined dually[81] (Figure 3.35 (b)).

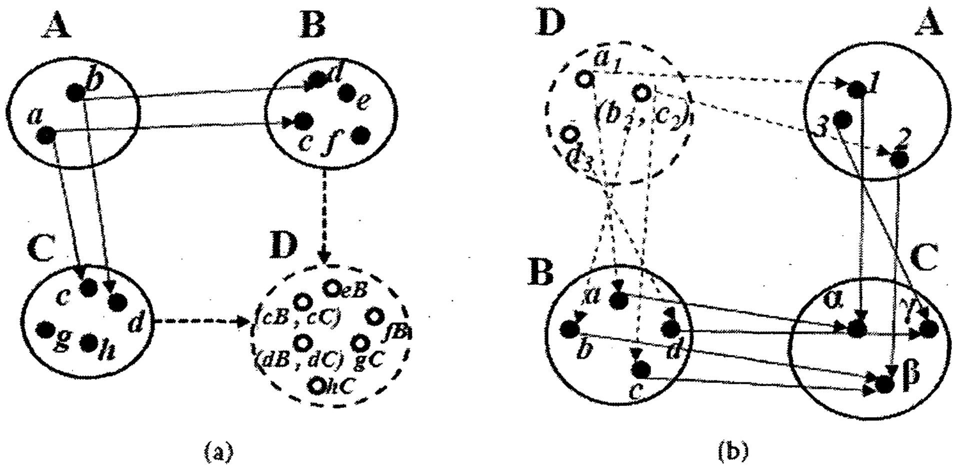


Fig. 3.35. An example, demonstrating (a) the pushout for two morphisms A→B and A→C in the category of sets (adapted from [Eas98]); and (b) the pullback for two morphisms A→C and B→C.

[81] The dual notion for a theorem can be achieved by reversing the morphisms.

If a given diagram composing *f*: A→B and i_1: B→D can be completed such that diagram represented in Figure 3.34 (a) is a pushout diagram, then we call C (in Figure 3.34 (a)) together with morphisms *g*: A→C and i_2: C→D a pushout complement of $i_1.f$.[82] In many cases, where one has to deal with composition and decomposition of different evolving structures, finding a pushout complement for a given state is a primary task. One can find details on using pushout and pullback for ontology alignment and merging [HKE+05, ZKE+06]. Specifically, the V-alignment and W-alignment approaches in [ZKE+06] have been described in terms of the pushout construction. Several researchers, including [JPV+98, HEK+06, ZKE+06], employed categorical pullback to model the composition of alignments. Also, to obtain different types of alignments (based on the level of granularity) and to ensure the minimality of the results, one can use the intersection or union (achieved by pushout of the intersection [Sol06]) of different alignments or their compositions.

III 3.5.4.2 Categorical Constructors for Ontology Merging and Integration

The ontology merging process transforms two or more ontologies into a single ontology. Ontology merging in its simplest situation (when ontologies are totally separate) can be represented by their disjoint union, but in real world applications, the ontologies to be merged usually have some elements in common and overlap (syntactically or semantically) in some areas. As a result the merging process can be seen as gluing the non-aligned part of one ontology to the aligned subpart of another one. Therefore, the pushout operator in category theory, which resembles the merging operation, can be

[82] For simplification C is usually refer to as the pushout complement without mentioning the morphisms.

employed [SRP02, BEE+04, HKE+05, ZKE+06]. Pushouts model the merging between two aligned structures without any restrictions due to dependency on any implementation language. The initial attempt for merging and integrating two ontologies (see Figure 3.36) in existing approaches starts with creating a mediator ontology using the notion of approximation[83] and entailment between the two ontologies [HP04b, Ken04].

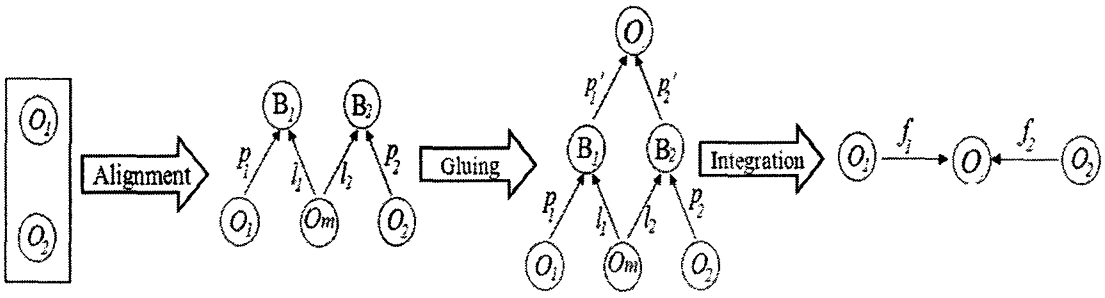


Fig. 3.36. Ontology integration process (Adapted from [Ken04]). In the first step two ontologies O_1 and O_2 are being aligned using a bridge ontology O_m and a set of refinement morphisms and bridge axioms. Then two mediator ontologies B_1 and B_2 are merged[84] into the ontology O. The final integration phase consists of deriving two direct morphisms from the two initial ontologies by composing the morphisms in the previous states.

There are several possibilities to use categories as a basis for merging and integrating ontological elements. Besides pushouts and pullbacks, other categorical notations, which are commonly employed for performing integration, are limits and colimits. Before defining these notions here we need to present some introductory definitions of other categorical constructors such as initial and terminal objects, diagrams, and cones.

Definition (initial and terminal objects): an initial object of a category C is an object I of this category such that for every object O in C, there exists exactly one morphism I $\longrightarrow$ O. In another words I is an initial object if for each object O there is exactly one map

[83] As defined in Wikipedia, approximation “is an inexact representation of something that is still close enough to be useful”. For more information on approximation in OWL-DL ontologies, we refer the reader to [PT07].
[84] In the set theoretical sense, one may describe this process with a special sum of B_1 and B_2.

from I to X. For example, in the category of sets an empty set is an initial object. In the following diagram *I* denotes an example of initial objects (Figure 3.37)

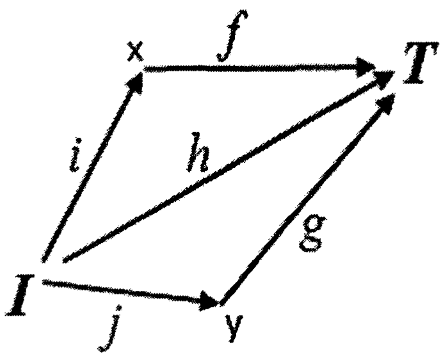


Fig. 3.37. A diagrammatical representation of initial (*I*) and Terminal (*T*) objects in category C.

The terminal object (*T* in Figure 3.37) is defined dually as follows: T is terminal if for every object O in C there exists a single morphism O → T[85]. As an example a singleton set is a terminal object. If an object is both initial and terminal, it is called a zero object or null object.

Definition (diagram) [AL91]: A diagram *D* in a category *C* is a directed graph whose vertices $i \in I$ are labeled by objects d_i and whose edges $e \in E$ are labeled by morphisms f_e. Later on in Section III.4 we will see that we can define a categorical diagram as a graph homomorphism.

Definition (cones and co-cones) [AL91]: Let **C** be a category and D a diagram with objects d_i, $i \in I$. Then as represented in Figure 3.38 a cone to D is an object c and a family of morphisms $\{f_i \in C[c, d_i] \mid i \in I\}$[86] such that $\forall i,j \in I$, $\forall e \in E$ $f_e \in C[d_i, d_j]$ then $f_e \circ f_i = f_j$.

[85] This definition of terminal object closely resembles the conception of "Thing" in OWL.
[86] The brackets in $C[c, d_i]$ are representing the domain and co-domain of a morphism in *C* (*i.e.* $c \rightarrow d_i$).

$$\begin{tikzcd}
 & & c \arrow[dll] \arrow[dl, "f_i"'] \arrow[dr, "f_j"] \arrow[drr] & & \\
\cdots d_{i'} \arrow[r] & d_i \arrow[rr] & & d_j \arrow[r] & d_{j'} \cdots
\end{tikzcd}$$

Fig. 3.38. A representation of a cone to a diagram D, which is an object c $\in$ *C* and arrows f_i: $c \rightarrow d_i$, $d_i \in C$ such that for each arrow $d_i \rightarrow d_j$ the diagram, commutes.

Co-cones are defined dually, such that a co-cone for a diagram D is an object c and a family of morphisms $\{f_i \in \boldsymbol{C}\,[d_i, c] \mid i \in I\}$ such that the generated diagram commutes.

Definition (limits and colimits) [Awo06]: A limit for a diagram D in category C is a cone $\{f_i$: c $\rightarrow d_i\}$ such that if $\{f'_i$: c' $\rightarrow d_i\}$ exists, then there a unique map u: c' $\rightarrow c$ exists that causes the following diagram for every d_i in D commutes (cf. Figure 3.39 (a)).

$$\begin{tikzcd}
c' \arrow[rr, dashed, "u"] \arrow[dr, "f'_i"'] & & c \arrow[dl, "f_i"] \\
 & d_i &
\end{tikzcd}
\qquad
\begin{tikzcd}
c' & & c \arrow[ll, dashed, "u"'] \\
 & d_i \arrow[ul, "f'_i"] \arrow[ur, "f_i"'] &
\end{tikzcd}$$

(a) (b)

Fig. 3.39. Diagrammatical definition of (a) limits and (b) colimits.

Colimits are also defined as dual of limits. A colimit of D is a co-cone $\{f_i$: $d_i \rightarrow c\}$ such that if $\{f'_i$: $d_i \rightarrow$ c'$\}$ exists, then there is a unique map u: c $\rightarrow$ c' exist cause the following the diagram in Figure 3.39 (b) commutes for every d_i in D. It is some how inferable [LS09] that limits as a universal construction generalize notions such as terminal objects, intersections, products, pullbacks; while colimits generalize notions such as initial objects, sums, coproducts, disjoint unions, and pushouts[87].

After aligning the ontologies, using a bridge ontology, a set of bridge axioms, and the mediating ontologies (the initial ontologies after applying the bridge axioms), a categorical colimit can be used [ZKE+06] to model the merging[88], through a series of successive pushouts. For example if two ontologies are aligned using W-alignment then three pushouts can compute the merging between the ontologies (Figure 3.40). By increasing the number of ontologies and composing different alignment methods together, the number of pushouts, which are used for computing the merge increases accordingly.

$$\begin{tikzcd}
 & & M & & \\
 & O'_1 \arrow[ur, dashed] & & O'_2 \arrow[ul, dashed] & \\
O_1 \arrow[ur, dashed] & & B \arrow[ul, dashed] \arrow[ur, dashed] & & O_2 \arrow[ul, dashed] \\
 & A_1 \arrow[ul] \arrow[ur] & & A_2 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Fig. 3.40. Integration of two ontologies O_1 and O_2, which are aligned via W-alignment technique, using colimits and three pushouts (adapted from [ZKE+06]). A_1 and A_2 are the alignments and B is the bridge ontology and O'_1 and O'_2 are the initial ontologies plus the added bridge axioms and finally M represents the final integration result.

[87] For example, initial objects can be defined as colimits of empty diagrams, coproducts are colimits of diagrams indexed by discrete categories, and pushouts are colimits of a pair of morphisms with common domain [Lim09].

[88] For example, categorically speaking one can describe the integration described in Figure 3.32 to be the colimit of the specified alignment diagrams.

III 3.5.5 Category Theory for Representing and Tracking Changes

Categorical representation enables the progressive analysis of ontologies. After describing the ontological concepts within categories representing a modular hierarchy of domain knowledge, we employ category theory to analyze ontological changes in the following ways.

III 3.5.5.1 Exploring the Similarities

One of the major tasks in performing ontology alignment and mapping is finding similarities (structural or semantical) between the ontologies. Finding semantic similarities in a network structure gives rise to several computational, psychological [Tve77], and philosophical issues, including the problem of identities and essence. As we discussed in Section III 3.5.4.2, similarity checking in ontology engineering can be studied under the notion of approximation [MME+06, PT07]. An approach for measuring semantic similarity that generates similarity scores based on trees [GGW03] and a graph-based algorithm [MME+06] for managing semantic similarities in ontologies are examples of some of the efforts in the area of similarity measurement for hierarchical structures. Recently, research [AN09] on measuring semantic similarity for concepts within biomedical ontologies and a review of different approaches in this domain [PFF+09] has been conducted.

The semantic similarity can be studied as finding logically equivalent classes and relationships that may differ in name while performing the same function. In fact, one of the significant uses of categories is analyzing different objects with some degree of similarity in their underlying structure[89]. Employing category theory enables us to deal with this problem of logical equality in evolving hierarchies using isomorphic reasoning [Maz07]. In set theory, two ordered sets are defined to be equivalent[90], iff there exists a third set, the members of which being ordered pairs such that *(i)* the first member of each pair is an element of A and the second is an element of B, and *(ii)* each member of A occurs as a first member and each member of B occurs as a second member of exactly one pair. In summary, a bijective order-preserving (monotonic) function should exist between A and B. These structure-preserving functions are a typical form of morphisms in category theory [HKE+05]. In categories, we do not focus on the internal structures of categories (i.e., the names of elements of a set are not important in the categorical approach) and instead all attention will be focused on the morphisms (representing the relations between objects), the composition of morphisms, and the cardinalities of categories of sets.

The definition of equivalence of categories has been given in [Sel05, AL91], and we will return to this problem in Section IV.3 on the case studies.

III 3.5.5.2 Tracking the Changes and their Impacts

The tracking mechanism keeps track of ontological structures over time. A chosen ontological structure (or element) can be monitored in a certain time interval, and its behavior in response to various changes can be captured and marked. In this way, after a while, the elements with a high chance of alteration will be highlighted and can be used

[89] As an example described in [Hea07], one may consider an architectural plan of a building that includes several details about forms. The shape and measurements of the building may exist in different forms, such as a hand-drawn or printed form on a paper, or a digital version in a computer.
[90] Adapted from: "set theory." Encyclopaedia Britannica. 2009. Encyclopaedia Britannica Online. 12 Nov. 2009 <http://www.britannica.com/EBchecked/topic/536159/set-theory>.

for detecting the possible change couplings[91] [GHJ98, DLR09] through backward tracing. In our approach using categorical morphisms, we make an explicit connection between different versions of an ontological structure, which enables us to analyze and generalize dependencies and monitor the impact of different operations on the parts affected. The categorical representation enables the progressive analysis of ontologies. Category theory is being used to represent the evolutionary structure of ontologies and provides facilities for tracking changes and analyzing the impact of these changes as follows.

I. **Comparing a previous state of a class with a later state:** A categorical model [Whi97] is 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 domains 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) [Awo06]. In other words they are structure-preserving maps between categories. As defined in [Oos02], we assume A, B are two categories, so a functor $F: A \rightarrow B$ is a pair of mappings (Figure 3.41) that associates to each object x in A an object F(x) in B; and also maps each morphism of A onto a morphism of B, such that the identities and composition are preserved. The preservation of identities means if, for example, x is an A-identity, then F(x) is a B-identity; and the preservation of composition means that considering f and g as two arrows in A, then one can find the following statement valid in B [Eas98].

[91] Change coupling in an application can be defined as the implicit relationship between two or more components that frequently change together during the systems' evolution [DLR09].

$$\forall f \circ g \in A \Rightarrow F(f \circ g) = F(f) \circ F(g) \in B$$

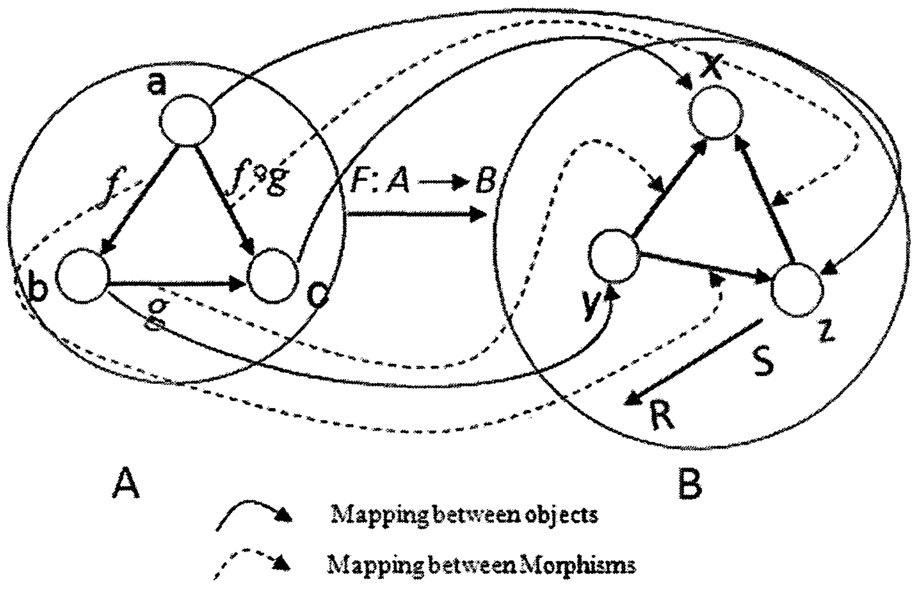


Fig. 3.41. A diagrammatical representation of a functor for two categories A, and B (adapted from [Jao06]).

As it can be seen in this definition, functors not only transform the objects but also represent an associated transformation of the structures (morphisms) [Ryd85]. A categorical model can represent transitions between different states of an ontological structure. As mentioned in Section III 3.4, following our Birdwatching allegory to capture and track this kind of transition, we represent the conceptualization of things indexed by time. For example, from the FungalWeb Ontology, "*enzyme* has_property_x at *t*" is rendered as "*enzyme*-at-*t* has_property_x". As another example, in the higher level, we can consider that an ontological structure O at time t has a certain feature. Then we represent a set of time-indexed categories using functors to capture different states of the ontological structure at different points in time. The category O at time t (O_t) models the state of the ontologies and all related interactions at this time. A functor can represent the transition from O_t to $O_{t'}$ (Figure

3.42) where the time changes from t to t'. In addition, each subontology A can be modeled by the series of its successive states A_t from its '*Creation*' to '*Destruction*' [EV06].

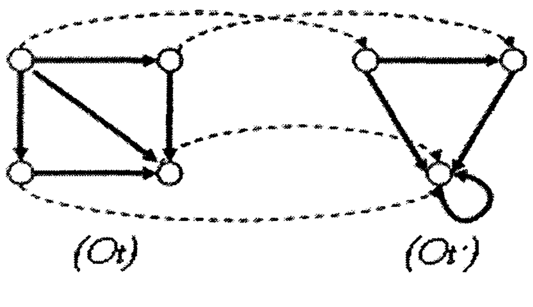


Fig. 3.42. Using Functor

It is quite common in software engineering to represent the relations between different versions of an application through a version graph [MDS00] consisting of nodes and arrows representing a version of the application pointing towards the successor versions (Figure 3.43).

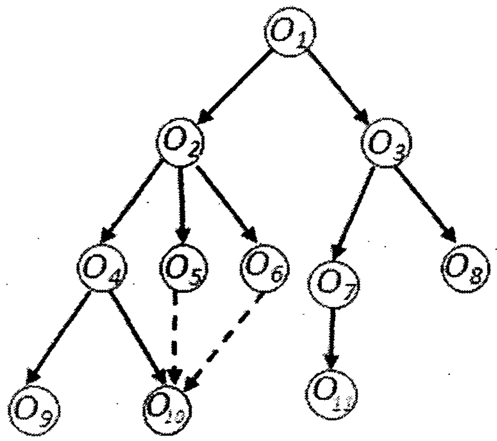


Fig. 3.43. A typical version graph [MDS00] composed of different branches representing the relationships between different versions and revisions of an ontology through a set of solid (shows a direct offspring and successor version) and dotted (shows the inheritance of some features) arrows.

It would be also natural to use categorical functors to represent and analyze the relationships between different versions of ontologies, organized in a version graph, within a specific life cycle. Here we extend the use of functor in ontology change management by introducing another categorical constructor called Natural Transformations, which describes the maps between functors (morphism of functors). Given two functors *S*, *T*, which represent two different transformations from category A into category B ($S,\ T: A \rightarrow B$), a natural transformation between these two functors ($S \rightarrow T$) is a morphism t which assign to each object x of A a morphism $t(x): S(x) \rightarrow T(x)$ of B in such a way that every morphism $f: x \rightarrow x'$ yields the following commutative diagram (Figure 3.44).

$$\begin{tikzcd}
x \arrow[r, "f"] & x' & S(x) \arrow[r, "S(f)"] \arrow[d, "t(x)"'] & S(x') \arrow[d, "t(x')"] \\
 & & T(x) \arrow[r, "T(f)"'] & T(x')
\end{tikzcd}$$

Fig. 3.44. Diagrammatical representation of a Natural Transformation between two functors S, T.

In fact natural transformation acts as a vehicle to represent transformation of one structure (modeled by a functor) into another structure (represented by another functor) within a temporal environment. So, it makes it feasible to model and track the relations between different revisions of one model. For example, considering another functor H in the natural transformation, depicted in Figure 3.44, which represents a map to the second revision of an evolving structure, one can obtain a composition of natural transformations [BW05].

II. Measuring coupling: As knowledge based systems become more complex, the new trends lean towards describing their architecture and behavior with more abstract representations. Category theory not only supports high-level abstraction, but also treats these complex interconnected infrastructures with a more intuitive style. Categorical representation also improves the readability of the complex ontological structure by omitting some of the irrelevant details of the internal structures. 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 the existence of a large number of dependencies in an ontological structure, which must be checked to analyze and control the chain of changes. Different types of couplings can be defined for ontologies, e.g. structural coupling, semantic coupling, message coupling and so forth. Especially structural coupling for ontological elements can be described by a number of connections and the links between them. Therefore, we focus on arrows in category theory to study these connections.

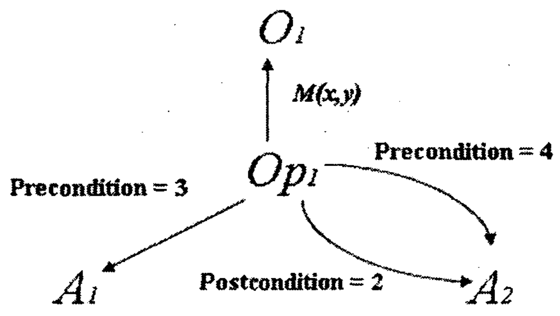


Fig. 3.45. Measuring Coupling as defined by [Whi97].

For analyzing a conditional change, we followed the formal model described in [Whi97] by identifying three types of arrows in the category operation: precondition, post-condition and message-send arrows. The type of message is determined by the types of changes caused by a method. In the category shown in Figure 3.45, the coupling for the operation Op_1 is a nonnegative number ($\geq$ zero) that can be calculated by counting the three types of arrows (post-conditions, preconditions, and M(x,y)). The message-send arrows can be excluded from this calculation, if they do not pass any parameters, thus do not have any operational affect on other ontological structures, or on other operations.

III. **Analyzing dependencies to control co-evolution:** Dependency analysis generally means exploring and tracing the dependencies and couplings between different units in a system. Analyzing ontological dependencies [DMM07], ranging from an individual concept to an entire ontology, facilitates the study of potential relations between an ontological element and its context through a set of constraints. When a change occurs in an ontological element, the other dependent elements will be changed accordingly to keep the ontology valid and consistent. This leads to a new version of the ontology. Similarly, when the ontology O evolves into the new version O', this evolution should be reflected in the other interconnected ontologies as well. These reflections – or co-evolution – should be formalized and supervised in a consistent way. In our approach, this problem will be addressed in the next sections by defining different levels of abstraction (micro and macro) in our analysis. Dependency analysis has been studied in [WH92] to maintain object-oriented programs and change impact analysis [AB96] by means of external dependency graphs (EDGs) and clustering methodologies. A classification of different dependencies in object-oriented programming, which is organized in a dependency graph, has been introduced in [WH92] as: Class-Class dependencies (e.g., C_1 is a direct parent of C_2); Class-Method dependencies (e.g., class C inherits method M); Class-Message dependencies (e.g., C understands message); Class-Variable dependencies (e.g., i is an instance of class C); Method-Variable dependencies (e.g., V is a parameter for method M); Method-Message dependencies (e.g., method M sends message M'); and Method-Method dependencies (e.g., method M_1 invokes method M_2). The nodes and arcs in the dependency graph may represent, respectively, ontological elements and different types of dependencies between these elements. Using category theory as described in Parts I and II helps not only in tracking changes but also assists in tracing the dependencies between ontological elements. Tracing the dependencies provides more information for agents and makes the negotiation process more realistic, the conflict resolution more effective, and the outcome more consistent with the intended purpose of the ontology.

III 3.5.6 Category Theory for Representing Agents' Interactions

"The meaning of things lies not in the things themselves, but in our attitude towards them"

Antoine de Saint Exupery (1900-44)

One of our primary research objectives in the RLR framework is to reduce human intervention in ontology change management life cycles. To this end, a mathematical knowledge representation formalism is necessary to support agent communications and interactions. As highlighted in [RM07], despite worldwide efforts in this domain, no proven formal frameworks, methods, and tools for modeling automatic agent interactions and argumentation yet exist. The interaction protocols, which consist of a set of steering rules to manage the interactions, are commonly represented using UML [Lin01], Petri net [PCN+04], State-charts [DCP05], state-transition diagrams, or finite state machines [FC03]. A key feature of our contribution has been the extension of existing agent modeling techniques using category theory to provide a formal yet intuitive diagrammatical representation for RLR. RLR employs categorical notions as a basis for modeling an agent communication language. The categorical framework is expressive enough to model the agents' behaviours, yet abstract enough to represent the generality of the protocols. RLR benefits from the algebraic power gained by using an abstract categorical representation of agents' interactions to increase the autonomy of argumentations in the change management framework. Category theoretical representation, with its ability to derive formal inference out of a diagrammatic representation, is independent of the type of interactions and their details, so its generality can be used to describe different types of protocols, study a MAS framework in different levels of abstraction, analyze rule transformations (yielding a practical image of adaptive learning agents and their semantics), and formalize dialectic trees for argumentation.

III 3.5.6.1 Analyzing a Multi-Agent Framework in Different Levels of Abstraction

Recalling the zoom-in and zoom-out notions in conceptual modeling, we define different types of categories based on different local and global perspectives. Each agent can be considered a category, with states of the agent as objects and the actions that cause an agent to change its state as morphisms. More generally, we can define a category of agents, with agents as objects and the different types of communication and interaction channels between agents as (functor) morphisms. In the same way, one can for example define the services given by agents as a category, with agents as objects and the composition relations between the agents (representing different interactions, communications, message passing, or sharing attributes between agents) as morphisms, or, alternatively, the category of services, with agents' services as objects and the mapping between the services as (functor) morphisms. Moreover, by changing the level of abstraction, we define a multi-agent system as a category[92] consisting of services as objects and the relations between them as morphisms, as well as the category of multi-agent systems, with each system composed of several agents providing different services as an object and the different communication channels between two or more distributed multi-agent systems as (functor) morphisms. This viewpoint about the categorical conceptualization of MAS structures in different levels of abstractions leads us towards

[92] Based on different conceptualizations, one may consider a multi-agent system to be a category with agents as objects and the relations between them as the morphisms.

defining a formal semantic for various interactions occurring between agents in evolving environments.

In an integrated multi-agent-based framework such as RLR, functors and their compositions are powerful abstraction mechanisms for analyzing the relations between different categories (e.g., relations between categories of agents, relations between a category of agents and a category of services, or a category of multi-agent systems and a category of states, etc.). As an example, consider a scenario for the alignment of two ontologies, by considering state as a category with the different states of an agent (i.e. initial state (Ini_S); requesting merge (Req_M); receiving the merge result (Rec_M) and checking for validity (Chk_V)) as objects and the message passing between the states (i.e. issuing alert, change notice, ontology ID, and so on) as morphisms (Figure 3.46).

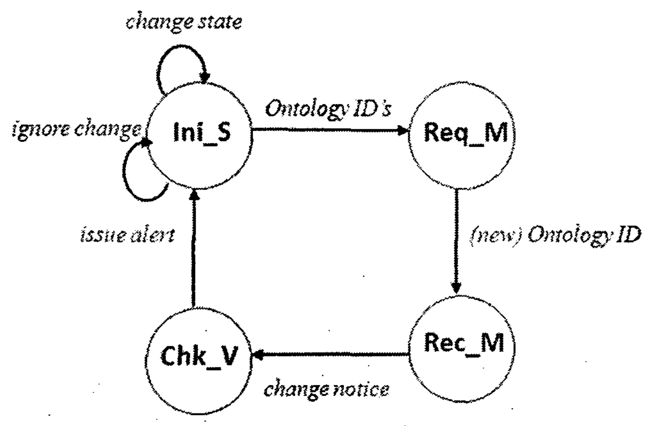


Fig. 3.46. The categorical illustration of states for the ontology merging scenario.

The above categorical diagram might be changed to demonstrate different options for performing ontology merging. For example one may want to check whether two ontologies are from the same domain or not (Chk_D) and if they are not from the same domain, cancel the merging and move back to the initial state (Figure 3.47).

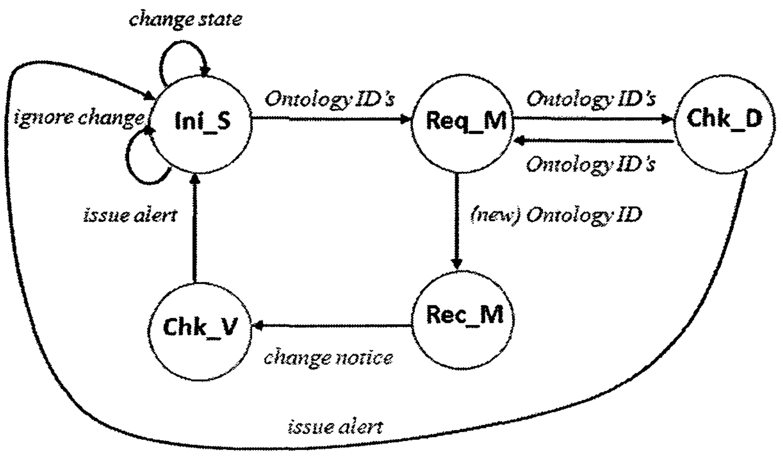


Fig. 3.47. The categorical illustration of states for an alternative ontology merging scenario.

Considering the first model represented in Figure 3.46 as Category OSt_1 and the modified version represented in Figure 3.47 as category OSt_2 in the category of states using the composition law and a functor, we are able to represent the transition between St_1 and St_2 through the functor F representing "Check Domain": F: $St_1 \rightarrow St_2$.

In a similar way different associations between different types of objects (e.g. various cognitive units [And81] described in the categorical sense) can be modeled. For example[93], one can describe a set of prepositions as objects within the category of prepositions and the relations between them as the morphisms in this category. Figure 3.48 represents a typical diagrammatic representation of such interactions for the following prepositions.

1. Agent **AG_1** received a message.
2. Agent **AG_1** has perceived a change request through the message.
3. The perceived change request is a delete request.
4. The delete request is issued to be performed on ontology $\mathbf{O_1}$.
5. The target for the deletion is concept $\mathbf{C_x}$ within Ontology $\mathbf{O_1}$.
6. Ontology $\mathbf{O_1}$ is currently being used by $\mathbf{KB_1}$.
7. The concept $\mathbf{C_x}$ is being reused in a process **Pr_1.**

[93] Oue example is inspired from the communication between the cognitive units presented in [And81].

8. The concept C_x has three sub concepts C_{x1}, C_{x2}, C_{x3}.
9. Two concepts C_{x1} and C_{x2} are currently being used in a process **Pr_1.**
10. The controller agent of KB_1 should be notified about the request.
11. The negotiation for loss/benefit has been performed.
12. Based on the negotiation outcome the delete request is postponed
13. The notification to the agent **AG_1** is sent.
14. **AG_1** ignored the change request.

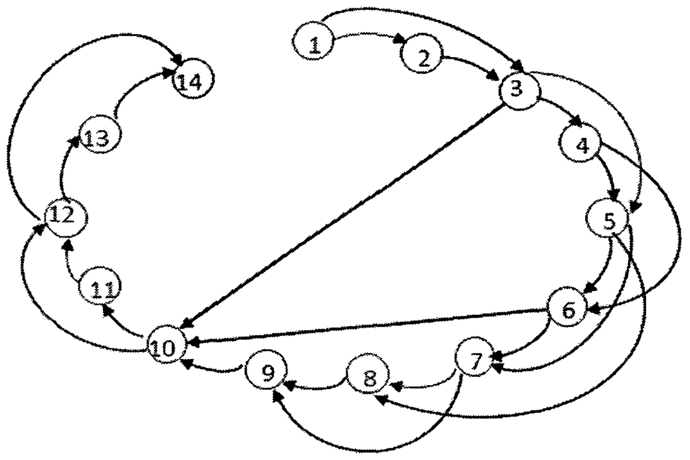


Fig. 3.48. A generic categorical representation of different prepositions in an agent based framework dealing with a "delete request" message.

As can be seen in Figure 3.48 several concurrent interactions may be performed through the compositions between the morphisms. Also several inferred knowledge can be gained[94] through this categorical approach, which can later on be used in the learning phase. Upon successful completion of the negotiation process in RLR, the ontology will either remain unchanged or be modified to convey the new knowledge based on the outcome.

III 3.5.6.2 Representation of Agents' Rule Compositions and Transformations

Intelligent agents perform actions in a context by using rules that guide interactions. In order to perform an action, which may lead to a state transition, often two or more sets of

[94] As an example of this inferred knowledge, one may notice the simple composition of morphisms in Figure 3.3.27 such that for instance: $1 \rightarrow 2 \rightarrow 3$ implies $1 \rightarrow 3$.

rules may be combined and integrated, ideally in an automatic fashion. The manual combination of rules is neither desirable nor feasible in many circumstances (i.e., when dealing with large sets of rules). The mathematical power of categories can deliver a formal guidance for combining these sets of rules, which are usually described in a diagrammatic representation[95]. For example, in the RLR framework, the agents follow certain rules, some simple and some complex ones (in the case of multiple options leading to different decision points, e.g., adding concepts, which needs the combination of several rules to find a place, check the validity, and so on). As shown in Figure 3.49, the two graphs 1 and 2, respectively denoting the (partial) state diagrams of agents A_1 and A_2 with nodes, represent the state and edges symbolizing the transitions. These two agents have their own opinions about the set of states in a change management process, which may differ with each other in some particular cases. To achieve the compositions of the two agents' views on performing a task, one can follow several options including conjunction or adjunction [CS01].

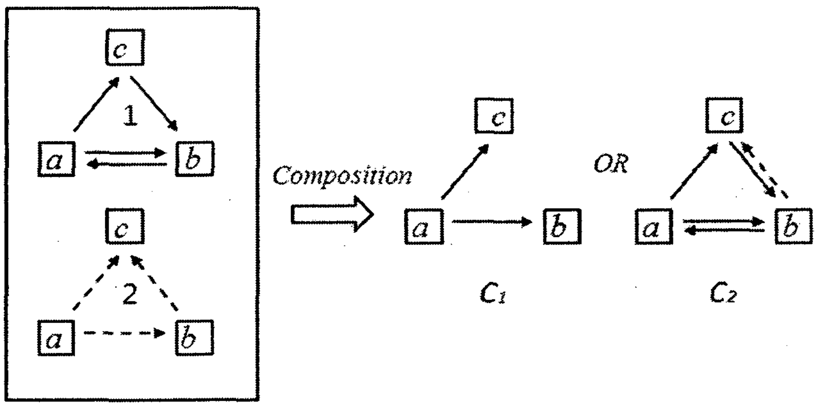


Fig. 3.49. The composition of two initial agents' action graphs through conjunction (C_1) or adjunction (C_2). As can be seen in C_1 emphasizes are on common paths within the two action graphs, while in C2 the focus is on sum of the available paths.

[95] They may be represented by UML, state transition diagrams, Petri nets, or finite state machines, to name a few possibilities.

As the above figure shows in these types of compositions, the origin of arrows might not be preserved. As another example, Figure 3.50 demonstrates the merging of two simplified transition diagrams D_1 and D_2, respectively corresponding to the rules specifying state spaces S_1 (location finding) and S_2 (adding an object), into the diagram D, which can be used in a typical algorithm for finding the shortest path [Fra08] (determining the closest node accessible from a particular node) in an agent based system.

$$\begin{tikzcd} D_1: & a \arrow[r, "f_1"] & b \arrow[r, "f_2"] & c \\ D_2: & d \arrow[r, "g"] & e \end{tikzcd} \Longrightarrow D: \begin{tikzcd} a.d \arrow[r, "f_1"] \arrow[d, "g"'] & b.d \arrow[r, "f_2"] \arrow[d, "g"'] & c.d \arrow[d, "g"'] \\ a.e \arrow[r] & b.e \arrow[r] & c.e \end{tikzcd}$$

Fig. 3.50. The integration of rules described in two transition diagram D_1 and D_2 using the categorical product $D_1 \times D_2$ to obtain one compound state diagram, which can be used in a typical shortest path algorithm (adapted from [Fra08]).

This approach can also be generalized for merging more than two rules with more complex structures through bridge-rules, which glue the rules based on their common features (similar to bridge axioms described in Section III 3.5.4).

As mentioned in Section III 2.3.2, RLR considers the change of the rules as a primary adaptation principle for learning. For describing our adaptive agents, we follow the formalization method used by G. Resconi in [RL04]. Each rule includes a finite or infinite semantic unity[96], which can be symbolized as S_1, IN, P_1, and OUT. These symbols represent the input statement, the domain of the rule, the rule, and the range of the rule (denoting the value of an agent's action), respectively. Generally, when we work in a static environment, we deal with only one family of rules for each context. However,

[96] Semantic unities represent the conceptual map between a set of concepts within a specific context through rules [RL04].

when the environment is dynamic, it is very likely that these rules change into other rules. Therefore, a single change in an ontological element triggers other changes in rules and contexts. As an agent gradually learns the different rules for various contexts, there is the need for a communication channel between these rules, as well as between different agents. Such changes are demonstrated in [RL04] as follows (Figure 3.51).

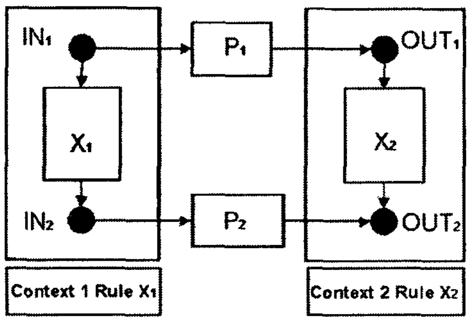
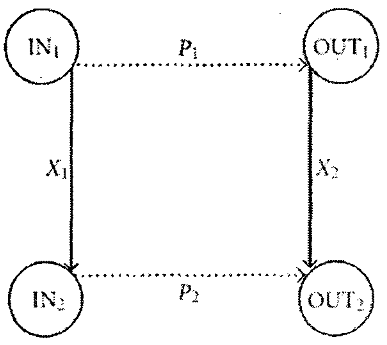


Fig. 3.51. Demonstration of the semantic unity of the changes of the rule X_1 in the context 1 into the rule X_2 in the context 2 (adapted from [RL04]).

From the point of view of category theory, we consider the category of rules with semantic unities as objects and the mappings between them as morphisms. We then use category theory, along with General Systems Logical Theory (GSLT, described in [RH96]), to describe agents' communication. For example, the communication between different semantic unities can be represented as follows:



$$\begin{tikzcd}
IN_1 \arrow[r, dotted, "P_1"] \arrow[d, "X_1"'] & OUT_1 \arrow[d, "X_2"] \\
IN_2 \arrow[r, dotted, "P_2"'] & OUT_2
\end{tikzcd}$$

Fig. 3.52. Categorical representation that demonstrates how rules P_1 and P_2 enable the transformation of the rule X_1 into the rule X_2 (adapted from [RL04]).

Using categories to enhance the learning process has been also addressed in [FFG+95] by measuring and comparing the relative sizes of classes of inferable sets of functions based on inductive inference. To define the semantics of agents' protocols, we describe a set of pre- and post-conditions that need to be satisfied before/after the occurrence of a particular action or actions. Then the categorical semantics can be used to model different interaction protocols within a general dialectic framework. Few approaches attempt in defining categorical semantics for agent interactions including the one that can be seen in [JMP05], where they focus on denotational semantics, considering the protocols abstracted away from the type and the nature of the interaction results. Pfalzgraf [Pfa04a] has proposed a distributed logical ground based on category theory, the concept of logical fiberings [Pfa04b], and many-valued logics [Got07] for modeling multi-agent communications. In summary, the idea in this approach is to allocate a local logic (logical fiber) to each agent and make the fibering (global logical state space) out of the group of all the fibers over the base space of agents.

In the RLR framework the semantics of an evolving agent-based system can be captured through a category of states and a set of operational transitions $Op\colon St_m \rightarrow St_n$, representing that the state St_m can change into St_n by performing an operation Op. As illustrated in Figure 3.53 each individual agent (e.g. $A_1 \ldots A_n$) can make a transition using a function (e.g. f_n, g_n, ...), which force the transition of MAS to the new states through the operation arrows (e.g. f or g).

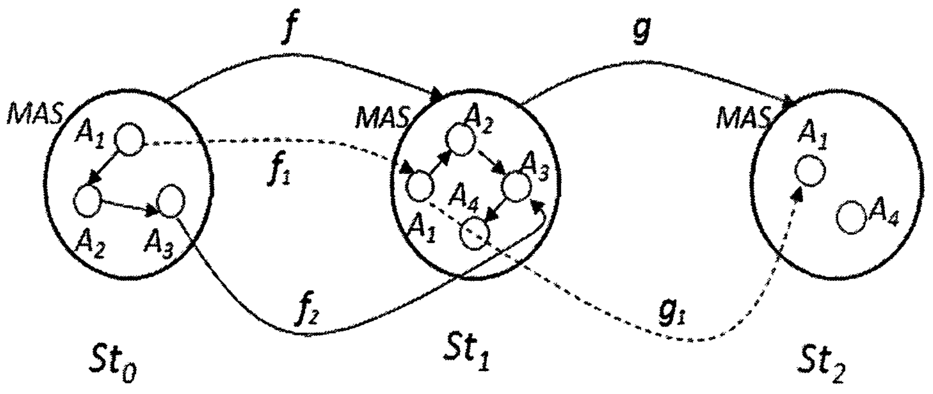


Fig. 3.53. The representation of a multi-agent system (MAS) transitions to different states using different operations and in different levels of abstractions.

The interactions in RLR can be studies through a category with a set of states (*St*) denoting the points, a set *M* of possible message expressions, and a transition morphism T (product of states and massages). The current existing formalisms seem sufficient to model the interaction protocols for a relatively small set of interactions, but as the number of messages, exchanged expressions and potential interactions between multiple levels of nestings increase, it is far from trivial to manage all the prospective arrangements.

Categories support the agents' rule interactions with no need for deep architectural and procedural nesting. As a simple example, let us once again look at the composition operation (o), which can be used to formalize the declarative rule interactions for agents. For instance, one may need to define a situation in which an agent should decide about the deletion of a node in an ontology. Since the rules are not isolated in RLR by using the composition operation (o), we can represent: $R_S \circ R_P \circ R_R \circ R_D$, where R_S is a morphism denoting "select node command", R_P is "parent checking condition", R_R is "remove child morphism", and R_D is the action (i.e., deletion) to be taken in the next move.

III 3.5.6.3. Modeling Argument Trees

Analyzing the dependencies and legitimacy of a claim in an argument should be performed within a logical structure. Toulmin [Tou58] described an argument based on the Claim and Data supporting this Claim, a Warrant to infer the Claim from the Data and Backing to support the materials that support the Warrant, a Qualifier to represent the soundness of an argument with uncertainty, and a Rebuttal (Reservations) to represent the exceptional cases (Figure 3.54).

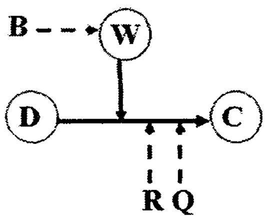


Fig. 3.54[97]. The Toulmin's layout for argumentation, with C, D, W, B, R and Q denote respectively Claim, Data, Warrant, Backing materials, Rebuttal, and Qualifiers.

Since Toulmin's description of argumentation trees have been adopted as one of the preferred vehicles for representing an argumentation framework through two or more contradicting structures where the roots, the nodes, and the edges respectively denote a claim, the grounds (supporting information), and the warrants (rules). Many of the uncertain and arguable grounds can be considered sub-claims, which are supported by a set of nodes (grounds). Figure 3.55 represents an example of such a tree.

[97] Adapted from: "A Description of Toulmin's Layout of Argumentation" http://www.unl.edu/speech/comm109/Toulmin/layout.htm

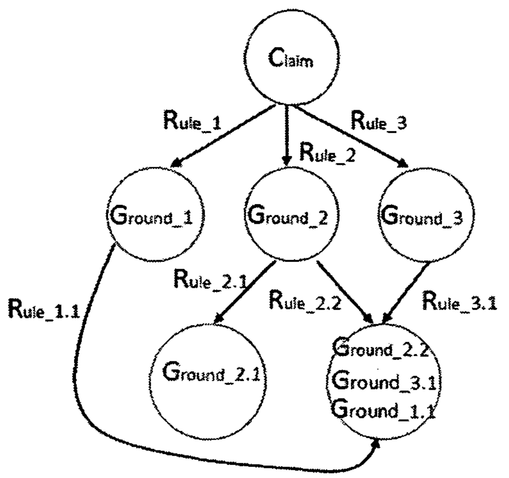


Fig. 3.55. A partial representation of a tree-like dialectically grounded argumentation structure. In this structure C represents the claims (e.g. Ontology O is a formal Ontology); G denotes grounds (e.g. Ontology O is written in OWL-DL); and R represents the warrants or rules (e.g. OWL-DL ontologies are formal ontologies).

As it can be seen in the tree shown in Figure 3.55, each branch has been associated with an argument about the claim (root) and its interactions with other branches (other arguments) form the argumentation structure. Currently several tools are available for creating such argument diagrams (e.g., Araucaria [RR04]). Toulmin argumentation diagrams mainly focus on the static representation of arguments, but they have been also extended to reflect the evolving nature of argumentations in various domains (e.g., the dialogue game [Ben98, BGL00]). In RLR, we also define categories of arguments with each category including the arguments as objects interacting within an argumentation framework (Figure 3.56) and the interactions between them as morphisms. Our initial plan to design a categorical model for RLR agent protocols starts with creating a graph for potential messages exchanged by the agents. Consider the category $C_{Communication}$ with a set of *time points* as objects and *message expressions*, usually placed in argumentation protocols, as morphisms. The morphisms represent the expression needs for argumentation between two time points (simply denoting the start and end of an argumentation). Thus, a communication for a protocol in the argumentation framework can be simply modeled by a sequence of morphisms and their compositions. Recall the use of functors and natural transformations to define different assignments between various categories. Here we are also able to generalize the communications between different protocols (e.g., two categories C_{Comm}, $D_{Comm.}$) using functors (i.e. F: $C_{Comm.} \rightarrow D_{Comm}$). In existing agent languages such as FIPA-ACL (see Section III 2.4), the messages exchanged between the agents may consist of requests and notifications, for example, without the possibility to define any combination rules; while in our approach we can define the rules' compositions in various levels of abstractions. This observation has been also studied in [PS08] from a different angle[98], where every MAS diagrammatic topology has been interpreted as a category PATH where the nodes are the objects and every sequence of consecutive arrows (a path which may include more than one single arrow) in the diagram is a morphism[99]. Based on [PS08], a base diagram, which is a category PATH, has been associated with each MAS to represent the general attributes and organization of related communication channels (arrows) for that MAS.

As mentioned in the introduction, the arrows in the category of agents (morphisms) convey a communicative operation of forwarding a message from one agent to another. A category of such arrows together implies an argument framework starting with an initial action and ending with a final decision (i.e. one may consider the classical example of auctioning). The set of rules provides sufficient expressiveness for the argument framework (e.g., winning_bid $\geq$ starting_bid). Each communication protocol can be considered a reusable pattern, which is "formally defined and abstracted away from any

[98] In the study performed in [PS08] the authors focused on the communications between different MASs rather than the dialogues between individual agents.
[99] This actually implies the composition of morphisms.

particular sequence of execution steps" [WJK00], and can be applied to other frameworks with different purposes. The categorical representations along with the graphical transformation greatly resemble UML representations (specifically state and activity diagrams) while providing more expressivity in terms of the underlying semantics.

For agent negotiation, we also assumed that one may consider two options for merging two ontological elements A and B: simply by the product A×B (all possible pairs <element from A, element from B>) or the co-product of the objects A+B (all elements from A and all elements from B). The negotiation agent can select the best method of merging and integration out of several alternatives for both categorical objects and arrows (denoting ontological elements). Assume we define following arguments for the integration and merging of ontological structures:

$$a_1\text{: A×B},\quad a_2\text{: A+B},\quad a_3\text{: A},\quad a_4\text{: B}$$

"a_1 defeats a_2" can be represented by an arrow from the domain a_1 to the co-domain a_2 (Figure 3.56). By following the categorical representation, an argumentation network will be generated that can be used to formally describe negotiations and speed up inferences.

$$\begin{tikzcd}
 & a_1 \arrow[dl] \arrow[dd] \arrow[dr] & \\
a_2 \arrow[dr] & & a_n \arrow[ll, bend left=90] \\
 & a_3 \arrow[ur, dashed] &
\end{tikzcd}$$

Fig. 3.56. Categorical representation of the argumentation network.

The categorical representation focuses on the behavior in which arguments interact (i.e. the argument a_1 defeated the argument a_2) instead of focusing on details of their internal structures. This categorical formalism can be used as a basis for conflict resolution in a recommender system based on dialectical databases [CCS05]. An algebraic semantics based on category theory has been also introduced in [Amb96] for argumentation, which provides proof of soundness and reliability for the structures based on "Logic of Argumentation" [KAF92].

III 3.6 Summary of Contributions in Section III.3

Category theory facilitates the analysis of the process of structural relationships and structural change in living and evolving systems. Categories have been extensively used in mathematics and theoretical computer science to assist in separating the levels of abstractions and integration of generic components. The categorical method to study and measure changes and to test several hypotheses and certain effects on developmental change between two time points has been applied in biological and social analysis as well as in psychological [AAG05] domains. However, the applications of category theory in biomedical Ontology change management are extremely rare. Category theory provides a universal algebra for the representation of highly abstract concepts. We use category theory to explore systematic changes in ontologies and study various dependencies between the ontological elements, as well as formalizing agents' interactions and communications in the RLR framework. The following is a summary of our main contributions in this section.

- Defining a method based on a metaphor taken from a recreational activity, Birdwatching, to highlight the temporal aspects of ontologies by representing conceptualization of things indexed by times, which enables one to control forward and backward compatibilities for taxonomic revisions.
- Introducing the potential of category theory as a formal representation vehicle for analyzing changes within biomedical ontologies in different levels of abstraction.
- Utilizing different categorical constructors and notations to assist in different tasks for the ontological change management process such as: performing change operations on ontological structures (e.g. add/delete, merge); exploring the similarities between different versions; tracking an ontological structure through its different states to monitor changes; measuring coupling and analyzing dependencies to control co-evolution. To this end, we have defined several categories to analyze classes, ontologies, operations, and states.
- Extending the semantics for change management process within RLR, by defining a categorical framework to support agents' communication, negotiation (i.e. formalizing dialectic trees), state transitions, compositions and transformations (i.e. rule transformation) in different levels of abstractions (agents and MAS). For this purpose we have defined several categories including categories of agents, multi-agents systems (MAS), services, states, rules and prepositions.

In the next section we extend our use of categories for managing evolving biomedical ontologies in the context of graph transformation.

III.4 A Graph-Oriented Formalism for Change Management

"I love fishing. You put that line in the water and you don't know what's on the other end. Your imagination is under there."

Robert Altman (1925-2006)

Advances in the World Wide Web, leading to the Semantic Web, Web 2.0, and Web 3.0,[100] have made a considerable impact on almost everything. The Semantic Web has been known for its complex and heterogeneous environment with highly volatile and non-deterministic interactions between its components, which are tightly coupled to each other. All of these features make the change management process for ontologies as basic blocks of the Semantic Web far from trivial. An enormous number of components, connected semantically and syntactically, are interacting with each other via several available knowledge bases, ontologies, databases, tools, and applications within the open distributed heterogeneous web environment. In such a situation, analyzing various changes requires a formalism with higher abstraction levels, which can simplify complex notions and representations to allow the study of changes in various levels. Despite the fact that employing the power of mathematical notation and mathematical proofs of formal methods has been studied in computer science for a long time [Wor99], in general, the current formal methods do not offer sufficient support for change management in terms of representation and verification (i.e. it is not yet formally verifiable that an

[100] Shannon, V. (26/06/2006). "A 'more revolutionary' Web". Int'l Herald Tribune. http://www.iht.com/articles/2006/05/23/business/web.php

implementation satisfies the initial defined specification). According to [MWD+05], "The formal methods need to embrace change and evolution as an essential fact of life. Besides the need for existing formal methods to provide more explicit support for software evolution, there is also a clear need for new formalisms to support activities specific to software evolution."

In order to deal with several issues of ontology change management, we chose to use several areas to reflect the interdisciplinary nature of the topic. In this attempt, after considering graphs as a generic notation for information representation and link data [BHB09] in the web, we employ graph transformation [EPT04] and category theory to study ontological transitions and changes in different levels of abstractions. Using graphs enables researchers to study structural evolution and changes in a rule-based manner. The transformation rules assist in modeling the change operations. Moreover, we use graph transformation to support dependency analysis through structural and semantical changes. We then proceed by using graph transformation to propose more specific semantics for ontology change management in the context of distributed hierarchical systems. Because of the tight coupling between ontological elements within typical biomedical ontologies and the sophisticated, complex relationships between dependent ontologies in Semantic Web, the change management strategies that mainly focus on changes in individual ontological elements might not seem to be very realistic or appropriate. In order to increase the flexibility and practicality of our approach, we consider the representation of change, independent of any implementation language, and defined algorithms. In fact, our method mostly focuses on the representation of changes in the distributed ontological compositions in different topological models rather than changes in an individual ontological element.

After applying the categorical concepts, some of the consequences will be formally derived and their formal interpretations will be given. Subsequently, we present a report on our approach towards categorically modeling the RLR multi-agent communication channel. In Section III 4.3, the description of graph transformations along with the categorical double-pushout method is given. In Section III 4.5 we represent our approach based on hierarchical distributed graph transformation as an extension of traditional graph transformation.

III 4.1 Graphs and Ontology Research

Graph representation has been used extensively to build formalisms and algorithms for supporting different change management tasks such as dependency analysis [Mos90, WH92, AB96], traceability analysis [Boh95, LWS+00], and impact analysis [AB96, Lee98]. [Men99] also employed labeled typed graphs and conditional graph rewriting [Hec95] technique formalized with categories to represent the evolution in software components. Graphs enable us to model the dynamic behavior of a system (e.g., UML state diagrams) in terms of transformations, in a wide variety of application domains. Petri nets, Entity-Relationship (ER), UML, and flow diagrams are all examples of graphs, some of which have been employed extensively for semi-formal ontology modeling. Some of the preliminary concepts and definitions in graphs and graph transformation as introduced in [KKK06] and [Rei05] are as follows:

- **Directed Graph (Digraph):** A directed graph is a pair $G = (V, E)$ of sets of vertices (nodes) V and edges (arcs) $E \subseteq V \times V$. A pair $(v, v') \in E$ is called an edge from v to v', with v named the initial node and v' the terminal node. If a graph has attributed nodes and edges, it is called an attributed graph [Rei05].
- **Graph Morphisms:** A morphism m between two graphs G and H, represented as m: $G \rightarrow H$, is a pair of structure-preserving mappings (m_V, m_E) where m_V: $V_G \rightarrow V_H$ and m_E: $E_G \rightarrow E_H$. The image of G in H from the morphism m is called a match of G in H, which means the match of G is the subgraph $m(G) \subseteq H$ with respect to the morphism m [KKK06].
- **Graph Homomorphisms:** A homomorphism between two graphs G and H is a mapping f: $V_G \rightarrow V_H$ such that for any $(a, b) \in E_G \Rightarrow f(a)f(b) \in E_H$. Given a graph TG, called a type graph, TG-typed (instance) graph G can be defined if there is a homomorphism g: $G \rightarrow TG$ [HC04b].

Graphs have proven themselves to be an appropriate formalism for representing network of hierarchical structures. They not only represent the relationships between ontological elements in a natural and diagrammatic fashion (see [MGH+09] for some examples), but also enable us to intuitively describe key concepts in ontology evolution, such as dependencies, couplings, transformations, traceability, and impact analysis.

Considering the graphical representation of ontologies, we study ontological structures at two levels, namely Micro-level (zoom-in approach), for analyzing the changes in internal structure of an ontology (Figure 3.57 (a)), and Macro-level (zoom-out approach), for exploring changes in a world consisting of interrelated ontologies (Figure 3.57 (b)).

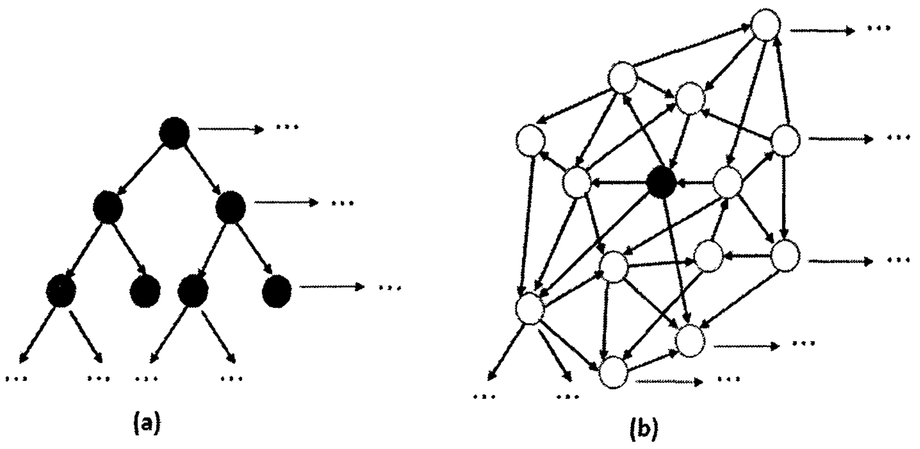


Fig. 3.57. (a) The evolving structure of a standalone ontology; (b) An evolving arbitrary lattice-like structure consisting of several interconnected ontologies.

At the first level, we consider an ontology consisting of several related RDF (OWL) graphs represented in a formal framework.

III 4.1.1 RDF Graph Representation (Micro-Level)

Following the RDF graph-based assumption, the digraph representation of an RDF[101,102] triple, consisting of the predicate that relates the subject to the object, has been demonstrated in Figure 3.58. For the sake of flexibility, the subject and object can be left unspecified, indicating the blank nodes. A typical ontology consists of several collections of related RDF triples that form a generic graph-like structure as well.

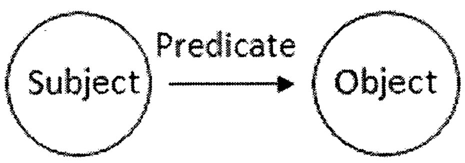


Fig. 3.58. A directed graph representing an RDF triple.

[101] http://www.w3.org/TR/rdf-syntax-grammar/
[102] http://www.w3.org/TR/rdf-mt/

For example, assume we have the following information: there is a fungus identified by a particular URI, its name is "*Aspergillus nidulans*", its NCBI *Taxonomy ID is* "162425", and it has a synonym "*Emericella nidulans*". The corresponding graph-like structure for these triples can be visualized as Figure 3.59.

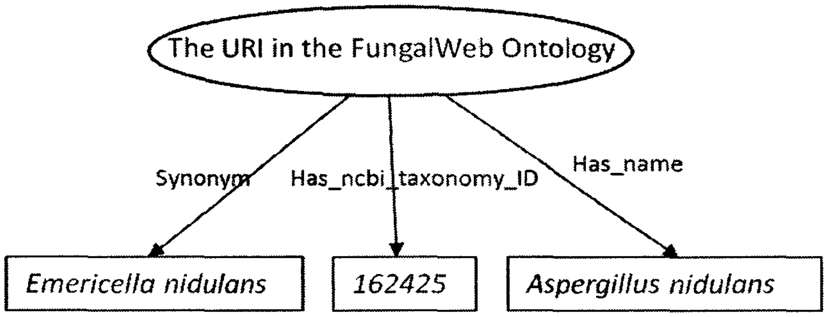


Fig. 3.59. An illustrated example of an RDF graph describing a fungal species from the FungalWeb Ontology.

Graphical representation of RDF and its associated operations (e.g., union, intersection, merging, mapping and so on) has been discussed in literature. According to [GHM04], the mapping between two RDF graphs G_1 and G_2, is defined as: $G_1 \rightarrow G_2$; the union of the graphs ($G_1 \cup G_2$), is defined as the set theoretical union of their sets of triples; and the merging of the graphs ($G_1 + G_2$) is the "union $G_1 \cup G'_2$, where G'_2 is an isomorphic copy of G_2 whose set of blank nodes is disjoint with that of G_1, and $G_1 + G_2$ is unique up to isomorphism".

An OWL graph is a subset of RDF graphs, however, the reverse is not always correct. The W3C OWL working group[103] [PH04] proposes a set of transformation rules for mapping and translating the abstract ontological syntax to OWL (with an emphasis on OWL DL) and RDF triples. RDF is considered the exchange syntax for OWL [PH04], thus the semantics of OWL ontologies in RDF can be determined from the corresponding

[103] http://www.w3.org/TR/owl-semantics/mapping.html#transformation

RDF graph organized by the collection of triples, obtained from the parsing of related documents. An RDF graph is an OWL-DL ontology (in graph structure) if it is equal to a result of the given transformation to triples and satisfies certain conditions (see [PH04] for more details on definitions of OWL-DL and OWL-Lite ontologies in RDF graph form).

Because RDF and OWL graphs are naturally attributed graphs[104] [TFH03], it is feasible to adapt AGG [Tae04] to perform the graph transformation [EPT04] for RDF/OWL Ontologies.

III 4.1.2 Lattice-Like Graph Representation (Macro-Level)

Instead of an individual analysis of ontologies to find out the changes in their internal structure, we use categories to study changes in different linked ontologies (as objects within the category) algebraically. In this view, ontologies are specified in an abstract way based on their relations to other ontologies. As mentioned in Sections III.2 and III.3, we are able to identify various types of categories in different levels of abstraction. As well, using the functor (a structure-preserving mapping between categories) facilitates the modeling of nested structures and the coexistence of several complex structures.

There are both differences and similarities between the ways we deal with objects and morphisms on the micro and macro levels, but the difference of terms primarily reflects the changes in our perspective. There is also another possibility to define an intermediate level between these two levels to analyze ontologies and their relevant segments [SR06] in a modular manner.

[104] An attributed graph is usually made based on a graph structure and the data about this structure, which makes it comparable with ontologies hierarchical structures with related set of attributes and cardinlaities.

III 4.2 Incorporating Time within RDF Structures

As mentioned before, considering time as an important factor in change management has several benefits for dealing with chronological data and knowledge scattered in different log files and for accessing different versions of ontologies. Upon changing a specific element in an ontology, several changes in the related triples in the RDF graph can be foreseen, which leads the graph to change its state through time frequently. To incorporate temporal reasoning, several frameworks including [GHV07] have been proposed for analyzing temporal RDF[105] graphs, which allow metadata description, navigation, and querying across time.

As an example (based on the approach given in [GHV07]) to reflect the temporal feature in the FungalWeb ontology, consider t_1 as the initial time when the ontology is in its initial state St_1 and assume we add a new type of enzyme as concept C_{enzyme} at time t_2. This newly added concept has its own properties that affect several related concepts in the ontology, beginning at time t_3 when the service is offered by the new concept, and at time t_4, when new relations will be offered based on the newly added concept. By deleting a concept at time t_n, the associated properties and relations would be removed as well. There is a problem in traditional ontological modeling, namely that when an ontology goes through a sequence of different changes (e.g., insertion, deletion, and/or replacement of concepts or properties), the answer to queries regarding the previous state might return no valid answers. Despite the fact that many approaches are available for different query languages for RDF [HBE+04], the temporal aspects of RDF graphs have not been sufficiently studied. Visser in [Vis04] reviewed some of the requirements that

[105] Recall that the RDF is naturally built in an extensible format.

are necessary for annotating and querying temporal knowledge bases, and then he described their approach by representing the so-called "qualitative abstraction of time". Several issues related to the temporal extension of RDF have been discussed in [GHV07], including decisions about different mechanisms for incorporating time (e.g., time labeling, snapshot capturing, considering time points or intervals, etc.) into regular RDF graphs, constructing temporal query languages, and temporal entailment[106] (logical implication).

In our approach, based on different levels of abstraction, we use timestamping (indexing the ontological structures with the time at which a certain event/change occurred) and snapshotting (denoting different states of the ontology) methods for temporal analysis of our hierarchy. To represent the temporal triples, we can index each triple with time. A series of related temporal triples form a temporal ontological graph. The time index can either be defined as constant or variable (to represent unknown or incomplete temporal information) [GHV07]. In this way, we can offer bitemporal[107] data analysis, which allows the query agents to perform temporal rollbacks and chronological information retrieval. According to the ordinary temporal knowledge-based systems, time itself can be studied in points or intervals (e.g., an axiom about ontology is legitimate in specific time period $[t_1, t_2]$ when $t_1 \leq t_2$). In the FungalWeb Ontology, the initial graph at time t_1 can go through a series of changes in different timestamps, therefore any query would only be meaningful during a particular time range (e.g. $[t_3, t_{n-1}]$). For example, the period $[t_3, t_6]$ may indicate that the triple graph (Enzyme, has, property_x) is only valid

[106] The entailement between RDF grphs is indicated by $\models$, and we can say $G_1 \models G_2$ iff there is a map from G_2 to a closure of G_1 [GHV07].
(see also: W3C on RDF Semantics: http://www.w3.org/TR/rdf-mt/#entail_)

[107] In temporal databases, bitemporal tables support both "valid time", capturing the history of a changing reality, and "transaction time", capturing the sequence of states of a changing table [Sno99].

during the time period between t_3 and t_6, and in t_7 it might not be valid any more. Accordingly, the ontological elements can be indexed with their period of validity, which allows movement between the time periods and access to the different states (past and present) of the system. Since in temporal ontologies, unlike temporal databases [JS95], the union of all of the corresponding snapshots (taken at different time points) does not always yield the whole ontology, a check for the logical implication (entailment) (which in RDF/OWL graph sense may be reduced to satisfiability checking) [HP04b] would be necessary. Considering the graphical structure of ontologies, we continue our study of transitions and changes in ontologies in the context of graph transformation, which has a great potential to deal with temporal graphs [Kos09] and their transformations [GPS98, YTT+05].

III 4.3 Graph Transformation

In order to overcome some of the limitations of the traditional rewriting methods (i.e., Chomsky grammars and term rewriting) for expressing the non-linear structures, graph transformation has been proposed [PR69] in the context of web grammars. There are various types of graph transformation methods, which can be classified in two general categories: the methods, which use the gluing condition [EKL90] and pushout constructor, and those based on nodes and subgraph replacement [ER97]. There are also two major formalisms for describing graph transformations based on category and set theories.

In this section, we briefly describe the rule-based graph transformation, and introduce some of its important notions along with some of the existing formal methods, such as single- and double-pushout approaches for graph transformation. The rule-based graph transformation can be studied based on the following three activities [Hec06]:

- Creating the conceptual generalizations of the reality and transferring them from "reality" to its representation in a model;
- The definition of rules as specifications of state transformations;
- Using graphs as a means to represent snapshots, concepts, and rules.

Generally, as shown in Figure 3.60 applying a transformation rule (production) p: (L, R) denotes finding a proper match of L (Left hand side) in the source graph and replacing L by R (Right hand side), leading to the target graph of the graph transformation.

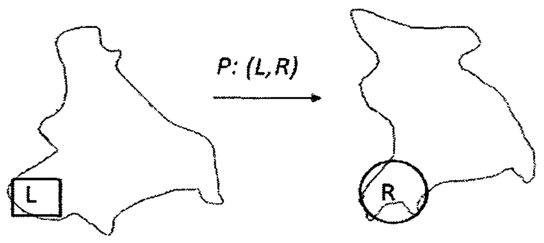


Fig. 3.60. A rule based graph transformation for a dynamic system (adapted from [EP05]).

The major question in graph transformations is how to delete L from a source graph and connect R with the context in the target graph [EEP+06]. Following the double-pushout approach [EPS73] (see Section III 4.3.1.1), a transformation rule (or production) is defined [DHP02] as a pair $t: L \leftarrow I \rightarrow R$ of morphisms $l: I \rightarrow L$ and $r: I \rightarrow R$ such that l is injective[108], where the graphs L and R are called the left and right-hand sides respectively, and I is called the interface or gluing graph. It is not necessary for the morphism $r: I \rightarrow R$ to be injective, which allows one to identify different nodes or edges in various transformations. Also, the injectivity of $l: I \rightarrow L$ ensures the uniqueness of the

[108] One to one: every unique argument produces a unique result.

results in backward tracing in a transformation. The rule t transforms a graph O_G into a graph O_H, denoted by $O_G \Rightarrow_t O_H$ if there is an injective occurrence morphism $m: L \rightarrow O_G$, and two pushouts of the following form:

$$\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "(I)"] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] \arrow[dr, phantom, "(II)"] & R \arrow[d] \\
O_G & K \arrow[l] \arrow[r] & O_H
\end{tikzcd}$$

Fig. 3.61. The Double Pushout approach for graph transformation.

The morphism m, which models an occurrence of L in $O_{G,}$ is called a match. The transformation, which is performed by the specified rule, represents the change of the graph O_G to the graph O_H. In more complex transformations we usually see a sequence of simpler transformations and a set of several transformation rules. As stated in [EKL90], by considering the dangling points (those points in L, a subgraph of $O_{G,}$ that are the source or target of arcs in O_G minus L) and the identification points (those points in L that are identified in O_G) in the transformation of O_G, the gluing points of L (identified by K_L) can be identified if both dangling and identification conditions are satisfied. These two conditions together form the gluing condition, which ensures the transformation is valid.

Dangling condition $\cup$ *Identification condition* $\subseteq$ *Gluing Condition*

Based on the previous definitions, the pushout exist iff m satisfies the dangling condition with respect to l, and in this case O_G, t, and m determine O_H uniquely up to isomorphism. A graph transformation system is usually defined as a set of transformation rules (productions) *P*. Graph transformation deals with the rule-based modification of graphs in such a way that "the core of a rule p = (L, R) is a pair of graphs (L, R) known as the left-hand side L and the right-hand side R. Applying the rule p = (L, R) means to find a match L in the source graph and replacing L by R, thus leading to the target graph" [EP05].

There are currently some available tools and programming languages to perform and visualize graph transformation such as: AGG[109] [Tae04], Fujaba[110], Grace[111] [KBK01], and Progres[112] [HJK+95]. To find more information on some preliminary definitions and terminologies of graph transformation for the readers, we refer to [KKK06, EEP+06].

III 4.3.1 Graph Transformation and Category Theory

Several formalisms have been used to represent graph transformations, including set theory, algebra logics, and category theory [Roz97]. Category theory along with channel theory and situation theory are the most popular mathematical theories for representing semantic information flow (IF) [SK03] in dynamic systems. The concept of "homomorphism", borrowed from abstract algebra, means the transformation of structure (as morphism) and composition (as object) [Bel01]. Several research attempts have been inspired by this embedded ability for representing transformation. Category theory has been used to represent the semantic backbone for graph transformation since 1979 [Ehr79]. It provides an abstract framework for efficiently generalizing and transferring conceptual structures with the ability for reasoning about basic concepts in different

[109] The Attributed Graph Grammar System: http://user.cs.tu-berlin.de/~gragra/agg/
[110] http://wwwcs.uni-paderborn.de/cs/fujaba/
[111] GRAph and rule CEntered specification language
[112] PRogrammed Graph REwriting Systems

levels of abstractions. Using the abstraction power of categories considerably reduces the proofs [Men99] and facilitates the parallel representation of different behaviors (either as a whole or in part) in complex and nested structures. The categorical method to graph transformation is a prominent generic method for studying the behavior of a dynamic system through modern representation languages. It is highly generic because all the proofs and constructions are valid and applicable for different kinds of graphs (e.g., node/edge, labeled/unlabeled) [Sch08a, Sch08c]. To maintain categorical graph transformations, Schneider [Sch08c] recently proposed a roadmap for implementing some of the categorical constructors in Haskell[113], to support functional programming [Bir98] through interactive categories of sets and of graphs. Generally, two categorical methods are frequently used for graph transformation based on single-pushout (SPO) [Rao84, Löw93] and double-pushout (DPO) [EPS73]. One of the differences [EHK+97] between these two methods is this requirement for DPO to have additional dangling and identification conditions. Moreover other categorical constructors such as pullbacks [Bau95, BJ01a] (as the dual construction of pushouts) can be used for modeling the transformations in the context of double-pullbacks [HEW01].

III 4.3.1.1 Double-Pushout Approach for Graph Transformation (DPO)

The double-pushout approach (DPO) [EPS73] and its variations, defined by different researchers, are among the most common methods for modeling graph transformations. We also follow this approach for studying graph transformation in our framework. As mentioned in Section III 4.3 in this method, any legitimate transformation should satisfy the gluing condition, composed of identification and dangling conditions, so it can assist

[113] http://www.haskell.org/

in identifying which component is changed and substituted by which other component and whether a transformation is valid or not. In DPO the source graph G of a graph transformation G $\Rightarrow$ H via the rule $L \leftarrow I \rightarrow R$ is given by the gluing of L and an intermediate graph K via I, written G = L $+_I$ K (pushout I in Figure 3.61), and the target graph H is given by the gluing of R and K via I, written H = R$+_I$ K (pushout II in Figure 3.61). As shown in Figure 3.61, applying graph morphisms I $\rightarrow$ L, I $\rightarrow$ R, and I $\rightarrow$ K shows how I is included in L, R, and K. In summary DPO should be performed through the following steps when a rule $L \leftarrow I \rightarrow R$ is given.

1. Find the elements of L in the given graph G, i.e. a match m: L $\rightarrow$ G.
2. Delete from G all the elements specified in L, which are not in the gluing graph I. This means to find a graph K and graph morphisms K $\rightarrow$ G, and $I \rightarrow$ K such that the square is a pushout.
3. Add to graph K all the elements of R, which are not in the gluing graph I and create the second pushout and obtain a derived graph H.

As an example, consider the following graph transformation, which transforms graph G to H using the rule p: $L \leftarrow I \rightarrow R$ (Figure 3.62).

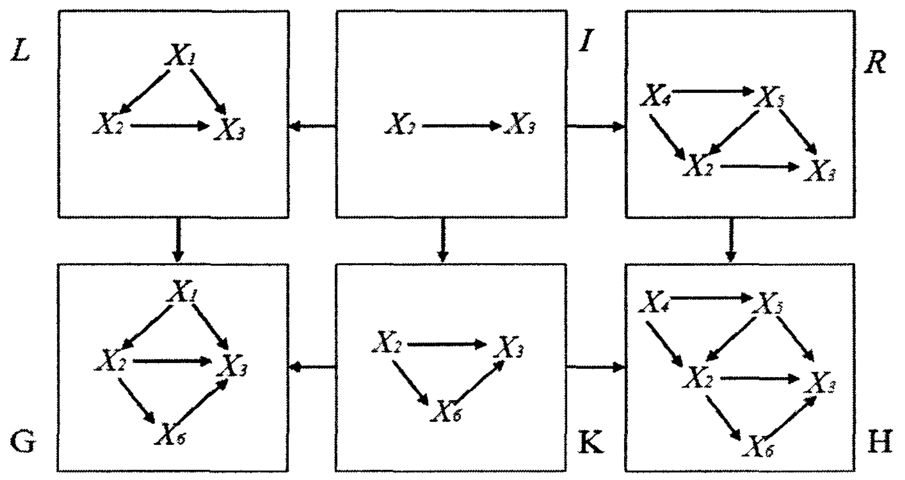


Fig. 3.62. An example representing a graph transformation using DPO. The upper part represents the transformation rule (L ← I → R), and at the bottom left there is a given graph G. Graph H is the result of applying the transformation rule on the given graph G, which has been obtained by following the three steps in DPO.

III 4.3.1.2 Single-Pushout Approach for Graph Transformation (SPO)

The single-pushout (SPO) [Rao84, Löw93] is another categorical approach for graph transformation that, unlike the DPO, has a single morphism in the transformation rule (production) *p*, which is a morphism in the category of graph with partial graph morphisms as arrows $p: L \longrightarrow R$. In contrast with DPO, the transformation can be represented by a single-pushout diagram and there is no interface between the source graph G and the target graph H.

III 4.3.2 Ontological Transitions in the Shade of Graph Transformation

Specifying of the transformations between different versions of an ontology is one of the primary concerns in ontology change management research, which can be gradually analyzed through the changes between different versions of an ontology since its creation. Studying the rationale behind these transformations can reduce some of the evolution's side effects (e.g., divergence and loss of information through different versions). Graph transformations have been efficiently used for describing dynamic changes of networked and hierarchical structures [KP02]. Some efforts to specify conceptual model transformations using the notion of type graph and conditional graph rewriting has been presented in [DM07] with emphasis on critical pair analysis for conflict detection [LEO06]. Recalling the graph-based origins of ontological models, the graph transformation techniques can offer several benefits in managing ontologies including: representing the operational semantics of evolving ontologies via an intuitive visual graphical syntax; offering a means for studying states of concurrent and distributed systems [KKK06]; providing a clear realization of complex context dependency operators and coupling between different components, which facilitate their comparison, matching, and alignment; and providing reasoning facilities for conflict detection and resolution [DM07] as well as modularization frameworks [ADM+07] for capturing knowledge.

Most of the graph transformation languages support the basic operations for node/edge addition and deletion. Also many available transformation approaches are highly application-dependent and informal, and have been proposed for specific purposes. Tools such as OwlDotNetApi[114] can assist us in creating a directed-linked OWL graph for a given ontology file. More information on the classes and interfaces available in this tool can be found in its web site[115]. COE[116] [HES+05] is another RDF/OWL ontology viewing/composing/editing tool that has been built on top of the IHMC CmapTools[117] concept mapping package. Users can use COE for importing OWL/RDF ontologies and rendering them into graphical representation, as well as

[114] http://users.skynet.be/bpellens/OwlDotNetApi/owldotnetapi.html
[115] http://mach.vub.ac.be/~bpellens/OwlDotNetApi/index.html
[116] http://coe.ihmc.us/groups/coe/
[117] http://cmap.ihmc.us/conceptmap.html

performing regular editorial operations such as adding/deleting/moving nodes and edges, and dragging, navigating, exporting, and so forth.

In our model, changes in ontologies can be performed through a set of consecutive transformations (ideally autonomous) via transformation rules, performed on the initial graph representation of an ontological structure. We start our analysis by considering changes in a single ontological element, which can shift the state of the ontology to another state. By considering graphs and the associated formalism for abstract syntax representation of ontologies, we represent the changing ontological structure within the RLR framework through the graph transformation process, operating on the source ontology (initial graph) along with a set of rules or productions (operations) transforming the initial ontology to its target version. An ontology O_{t1} can be transformed into another conceptual framework O_{t2} through the transformation *T*, shown as: $T: O_{t1} \rightarrow O_{t2}$, where O_{t1} and O_{t2} are ontologies at times t_1 and t_2 respectively, represented as an OWL graph, and the arrow indicates a transformation, which may consist of a set of simple transformations of the graph's elements. For example, adding a node to the graph is a composite transformation, which consists of several elementary operations such as adding corresponding edges for that node or assigning matching attributes and characteristics for satisfying the ontology axioms and facts. When dealing with ontologies conveyed in very expressive languages with rich semantics, one should always keep in mind that the complete transformation of all elements to the graphical representation might not be straightforward.

By defining a set of constraints within the transformation rules one can differentiate between different relations between ontological elements (e.g. subsumption relationships (hierarchical relations) and association relationships (non-hierachical relations)). The dependencies between elements and editorial operations can be specified in the transformation rules by monitoring the states before (L) and after (R) applying the rule *p*: L→ R [DM07]. When an ontology is being implemented collaboratively and used by different users and groups, there may be cases where different editorial activities can cause inconsistencies and conflicts (syntactical or semantical). The conflict arises in cases of incompatible modifications of a component (e.g., nodes or edges) through different transformations. Graph transformation along with some techniques such as critical pair analysis [LEO06, DM07] and tools such as AGG [Tae04] can be beneficial in automatically detecting possible conflicts for each of the defined transformation rules.

III 4.4 Change Analysis during Conceptual Model Transformation

Graphs are a powerful vehicle for analyzing model transformations. Various types of model transformations have been surveyed in [MG06], including horizontal versus vertical [HCE+96], endogenous versus exogenous, and syntactical versus semantical transformations. In a "horizontal" transformation, the source and target models stay at the same abstraction level (e.g., refactoring, migration*)* while in a "vertical" transformation, they reside at different levels (e.g., incremental refinement). If the source and target models have been expressed in the same language, the transformation is called "endogenous"; otherwise, it is "exogenous". In the syntactical transformation, only the syntax will be transformed (e.g., model import or export), unlike the semantical transformation, which also take the semantics of the model into account [MG06].

Different changes that occurred to each ontological element can be passed along the chain of dependent elements. Our proposed framework supports the conceptual transformations between different versions of ontologies, as well as maintaining the links and relationships between the versions. In fact, sequences of horizontal and vertical transformations in ontological structures occur during the evolution process. Several ontology transformations can be studied during ontology evolution. These transformations include, but are not limited to, transformations in relationships and properties (data type or object) (see Figure 3.63 and Figure 3.64), concepts, domain, cardinalities, and constraints. In addition, the transformation can be partial, which affects only a limited part of the ontological element or structure (i.e., part of the taxonomy) or complete (e.g., in the case of metamorphosis).

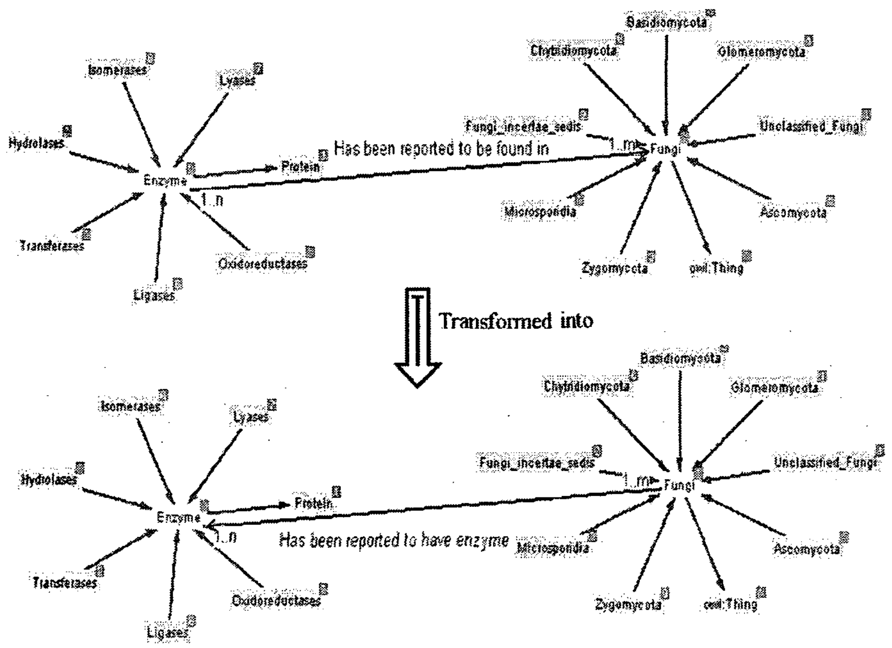


Fig. 3.63. Transformation by means of switching the domain and range of an associative relationship.

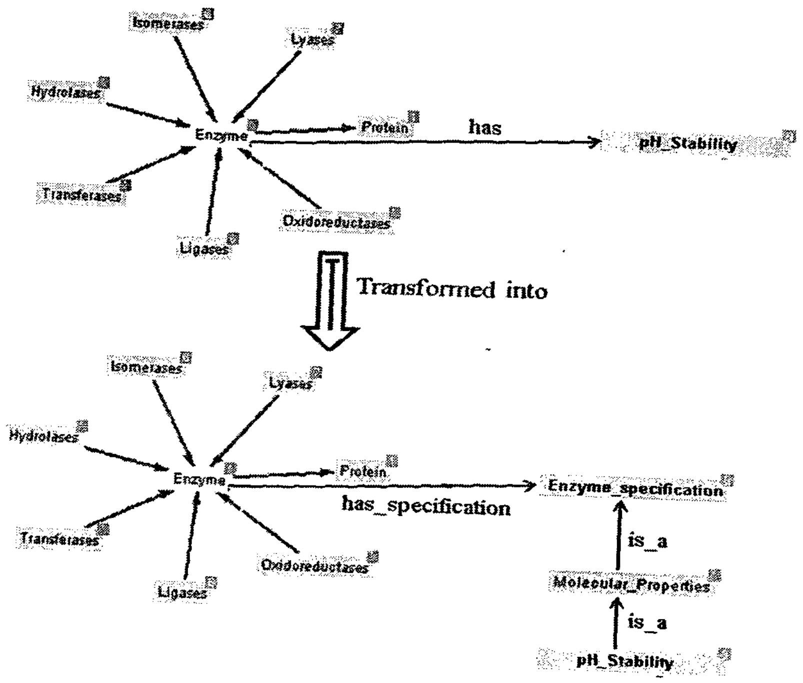


Fig. 3.64. Transformation by decomposing a property.

We consider "transformation" as a specification of the series of actions and operations that make an alteration in an ontological structure and cause the ontology to change state. For example, a change in a constraint or the addition/deletion of a concept or a property can be shown as action A on ontological element *a*, action B on ontological element *b*, and so on. A change can be defined as the mapping between two specific definitions from one ontology or different ontologies, or from different versions of one ontology. For example, $O_1(a) \rightarrow O_2(b)$ may be read as ontological element *a* from ontology 1 has been replaced by *b* from ontology 2. RLR traces the changes within an ontology and its transformations, and uses the information employed for each transformation for the reproduction phase, if necessary.

III 4.5 The Transformation in Action

Here we employ an adapted type of graph transformation, namely a hierarchically distributed graph transformation to maintain the hierarchically structured knowledge in the Semantic Web environment. In this framework, the graph transformation rules can describe the structural changes placed during a knowledge base operation.

III 4.5.1 Employing Hierarchically Distributed Graph Transformation

Changes in an area due to technical, industrial, cultural, or social matters force the existing systems and applications to adapt themselves to the new state. Particularly, large systems and knowledge bases built upon smaller reusable sub-systems are in greater danger and should be continuously monitored to ensure the correctness and consistency of the entire infrastructure. Graphs can be seen as appropriate vehicles to represent such hierarchical systems with nesting and layered relationships. In an ontological sense, concepts in an ontology naturally match with nodes of a graph, while the relationships in an ontology correspond to edges. Several biomedical systems and applications currently deal with complex graphs, with millions of nodes and edges and likely a large number of different rules. These graphs need to continuously evolve and transform to supply the revised and valid knowledge for the systems. The graph-based representation of the biomedical ontologies has a great tendency to become large, complex, and hard to grasp, understand, or maintain in a very short time. In applications dealing with compound graphs in layered organizations, the notion of graph can be extended to hierarchical graph. Hierarchical graphs attract broad attentions in theoretical computer science (e.g., object oriented design [EJ03], database [EN07], and computational molecular biology

[MV08]), mostly for representing semantically complex and interrelated network structures. Despite the popularity of hierarchical graphs in different domains, there is no common data model available; however, most of the existing models support treelike structures. This is one of the factors that make the hierarchical graph transformation techniques an appropriate option for analyzing hierarchically organized ontologies.

Different models, including the ones in [ES95, ER00, DKK+99, BKK05], have been studied concerning the issue of hierarchical transformation of dynamic complex graphs, and several models ([Hof99], [DHP02], [Pal08]) have been implemented using the rule-based approaches.

In order to mimic the actual nested hierarchical structure of the Semantic Web, where information is distributed in the nodes (graphs) and edges (relations between the graphs), we employ hierarchical distributed graphs [Tae99] for our approach. The hierarchical graphs have richer semantics and are more expressive in comparison with regular flat graphs. In addition, they reduce the complexity of representation of large interrelated systems by allowing one to describe a system on a more abstract level through hiding the irrelevant details in encapsulated sub-graphs [ES95]. Hierarchical graph transformation can be performed along with the algebraic and categorical graph grammars, using the extended double-pushout notion to represent various aspects of dynamic structures (e.g., the rearrangements of some temporal parts, describing the changes in relations, creation/deletion of communication channels, and performing operations such as "splitting" a graph into two or more graphs or "joining" distributed graphs into one graph [Tae94]). The categorical graph grammar [Sch89] supports the flexible change of complex interrelated compositions while providing explanations for corresponding actions performed by graph transformation. Various states can be produced by internal or external actions, and their communications can be modeled and simulated using graphs and state transitions, then represented and described by means of graph transformation. The double-pushout technique has been extended from flat to hierarchical graphs [DHP02], where the associated transformation rules can be applied at all hierarchical levels. This facilitates changes of the graph's entries (i.e., by insertion or deletion) regardless of their size and configuration, with adaptation of the "dangling condition" from the flat graphs transformations [DHP02].

We use the concept of hierarchical distributed graphs to be able to perform graph transformation on different levels of abstraction. As defined by [TKF+99] distributed graphs distinguish between two levels, namely local (internal), and network (external or lattice) (Figure 3.65).

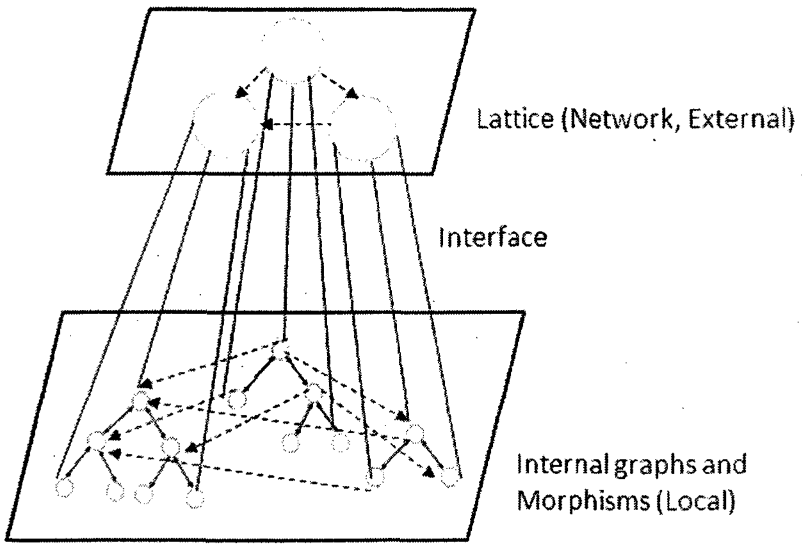

Lattice (Network, External)
Interface
Internal graphs and Morphisms (Local)

Fig. 3.65. A schematic representation of a distributed graph.

In our model, the hierarchical graph (the lattice) consists of a set of internal graphs (which may be hierarchical graphs as well), the root of the hierarchy, and a set of edges that relates the internal graphs to each other. Each editorial action is expressed through a graph transformation and every state of the ontological structure is modeled in a graph with the nodes denoting objects and the edges representing the connections linking them. The compound state of the entire system can be known by analyzing several other internal graphs, each having an internal state and behavior. There are also lattice-like dependency graphs representing the dependencies between different internal graphs. In the process of change management for the lattice-like structure, several concerns related to sequential, parallel, or concurrent evolution of its components arise.

Different ontologies in Semantic Web are usually connected in a lattice-like structure and interact with each other through one or more interfaces. This lattice can be modeled as a directed graph with individual ontologies (internal graphs) as its nodes and the links between these ontologies as its edges. The described configuration is analogous to what is called a hierarchically distributed graph (HD-graph) [Tae94], where each of the links connecting the internal graphs contains a graph morphism specifying the relation between two internal states. When the internal graphs are faced with any change (e.g., adding/deleting a concept or relation), their state would be changed, which would affect other dependent graphs, and a synchronization unit within the RLR framework, which stores all the states in the change logs, forces the lattice-like structure and the mediator interface to change their states accordingly. Following the approach given in [Tae94], this structure can be modeled in two different but related planes, namely conceptual (shows all existing and potential relations, paths, and their revisions) and operational (shows only actual existing nodes and relations).

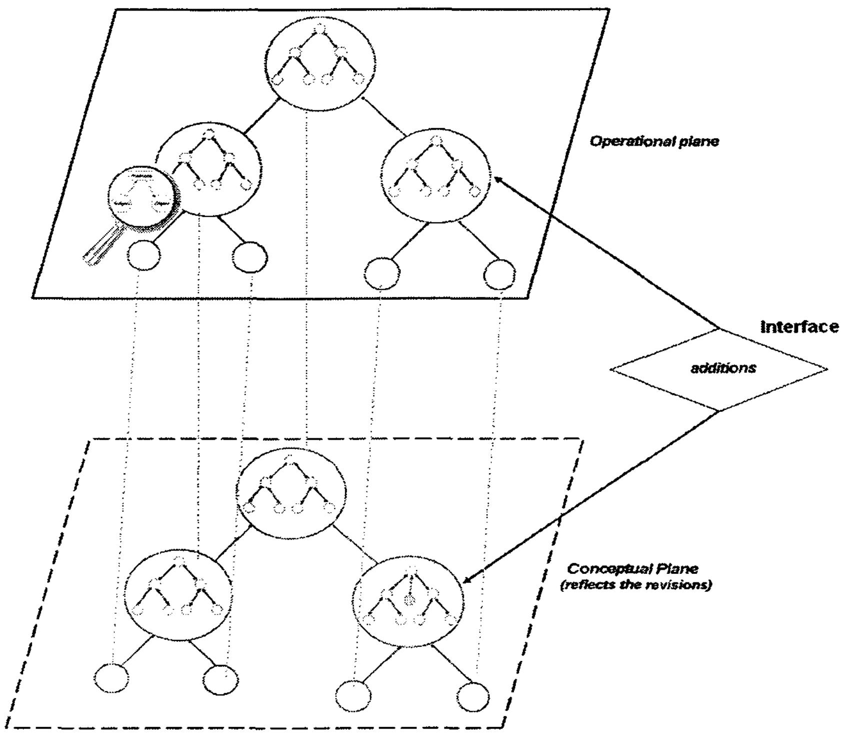


Fig. 3.66. A hierarchical graph for managing distributed ontologies representing the relations between different states of a lattice-like structure consisting different distributed ontologies. The changes can be performed in an interface graph that consists of all the nodes which have a matching node in the related internal graphs. In this way, the transformation of objects and morphisms allow the change of an evolving structure by changing its interfaces.

The synchronization unit needs to check certain conditions, namely connection and network conditions, to ensure the consistency of distributed graph transformations. The connection condition [TKF+99] determines that: i) the objects of source graphs should not be deleted without first deleting the related local mappings into target graphs; ii) the local nodes and edges, which have corresponding elements in interfaces should not be deleted; iii) to extend source graphs (upon insertion) first the new graph objects need to be mapped to the related target graphs; and iv) The attributes of graph objects, which have correspondence in interfaces should not be changed. The network condition [TKF+99] regulates that: i) if a network node needs to be removed, its associated local graph has to be completely known by the rule; ii) if a network edge needs to be removed, the local graph of its source node and the local morphism have to be completely known by the rule; and iii) if a new network edge needs to be added, the local graph of its source node has to be completely known by the rule.

In order to categorically analyze the distributed transformations we employ the category of distributed graphs DGRAPH with distributed graphs as objects and distributed graph morphisms as arrows[118] to define a transformation using an adapted version of double pushout approach described in [Tae99]. For the details of proofs and other related categorical notions in distributed graph transformation one may refer to [Tae99, EOP06].

III 4.5.2 Analyzing Events and Actions in Rule-Based Model Transformation

In order to analyze different events that trigger actions during the ontological evolution process, we consider events as part of the rule condition in a graph transformation. The actions as mentioned before (Section III 4.4) are described by productions and the events will occur if certain predefined conditions are assessed to be true. To formalize graph transformation, we employ the notion of double-pushout from category theory, which needs certain requirements to compute production (describes actions in graph grammar) and its corresponding element in other graphs. One of the requirements is satisfying the gluing condition to derive a new graph by finding a match of the left side of the rule in the given graph, then deleting it (except the gluing point) and adding the right side of the rule (see [EKL90] for the details).

By following the approach proposed in [Tae94], we use hierarchical distributed graph rules covering both internal and external production describing the internal and

[118] Notice the similarities with the functor categories.

external actions respectively. Since the lattice-like structure covering the internal graphs is less likely to be changed by internal actions, which affect mostly internal graphs, the external graph is transformed through an identical production that preserves the external graph nodes. A typical example, illustrated in Figure 3.67, is the addition of an ontological element (i.e., a concept) to an existing ontology, which causes the state of the ontological structure (internal graph) to be changed. This action does not have a significant effect on the lattice-like structure (external graph).

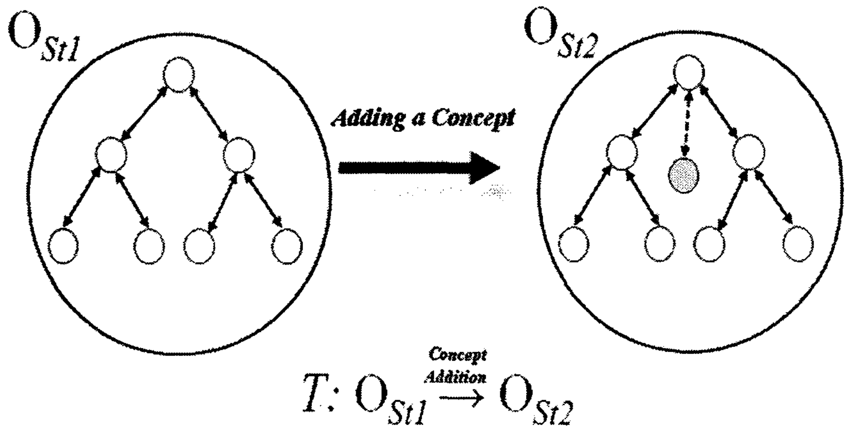


Fig. 3.67. Adding a new concept to an individual ontology that is part of a lattice made from several interconnected ontologies.

As represented in Figure 3.67, the hierarchical graph production "concept addition" demonstrates an internal action that transforms the ontological structure O from state St_1 to state St_2. This production will not alter the external graph represented in Figure 3.66. If one wants to delete an ontological element that has referenced a relation from other distributed ontologies in the lattice, then an external action needs to be performed. The external actions are capable of transforming the external graph. Controlling these transformations is a central task in the ontology engineering domain, since they can easily give rise to different types of inconsistencies, especially in cases that involve several parallel actions and transformations.

As long as the actions (e.g. deletion, insertion) do not violate the defined conditions in the production rules several actions can be executed in parallel at the local level (e.g deletion/creation of internal elements). As mentioned, the external lattice production describes the structural changes of the external graph, and we can model the external actions using a hierarchical distributed graph production in such a way that a unique production for the internal graphs of every node of the external graph (individual ontological structures) must be performed. If the stated predefined conditions for insertion/deletion of the nodes in the internal graphs are satisfied, then the hierarchical distributed graph production can be applied at the external (lattice) level (for adding/deleting edges, a set of morphisms will be described instead).

An example of alterations in the lattice is the insertion of connective internal graphs (nodes) between two or more other internal graphs (nodes). For instance, it is known that "a daily cup of yogurt significantly reduces the risk of candida infection and colonization" [HIA92], but this diet might not seem appropriate for lactose intolerant patients. Also, some studies show that some nutrition is beneficial to reduce the risk and severity of candida infections if consumed in a proper diet.

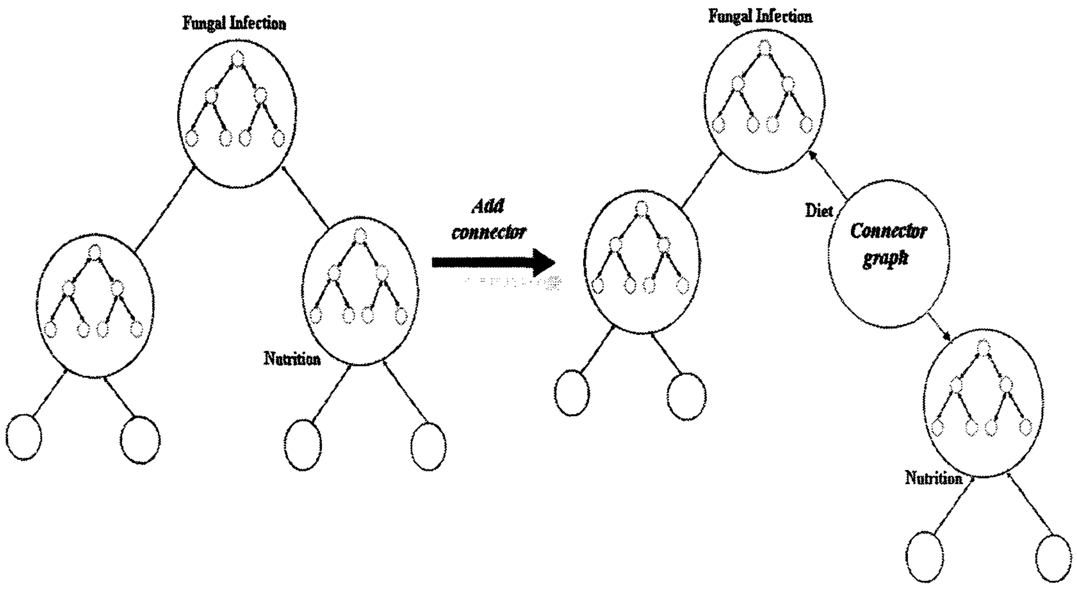


Fig. 3.68. The hierarchical distributed graph production "add connector" is represented in a way that the state of the graph "fungal infection" is now related to the graph "diet", rather than "nutrition".

Some of the examples[119] are Probiotics (up to 900 mg daily of beneficial bacteria), Fructooligosaccharides (up to 4 g daily), Goldenseal (250 to 750 mg daily), Lactoferrin (300 mg daily), Topical tea tree oil (based on the prescription), Oil of oregano (460 mg daily), Garlic (600 mg daily), and Boric acid (600 mg daily for 2-3 weeks, shown effective in 65% of women with vaginal candida infections [SCN+03]). In order to conceptualize these facts in an ontological framework, we use a connecting node (internal graph) "diet" to connect two structure fungal infections and "nutrition" through the hierarchical distributed graph production "add connector" (Figure 3.68).

III 4.5.3 Transformation Rules for Changes in Ontologies

The transformation rules in ontology evolution determine what types of changes are allowed and can be performed on the ontological elements and axioms. Padberg [Pad08]

[119] Fungal Infections (Candida). Life Extension Electronic Magazine: http://www.lef.org/protocols/infections/fungal_infections_candida_01.htm

describes the notion of rule-based refinement as an extension of transformations with added refinement morphisms alongside the rules, which can be applied for maintaining component-based applications. We found that the ontology evolution process, through subsequent refinements, is generally analogous and compatible with rule-based hierarchical graph transformation and refinement. Generally, in a double-pushout approach, a rule-based transformation indicates the changes of O_G to O_H based on the defined rule (see Section III 4.3). The rules can be atomic[120] or compound[121] and will be examined to ensure the compatibility and consistency[122] of the transformations.

Our proposed rule-based transformation method for ontologies determines the circumstances under which an ontological element can be changed or refined. Table 3.2 represents some examples[123] of graph transformation rules, which can transform a typical graph such as Industry (Diagram 2). Diagram 3 represents the establishment of the relation “is being used in” to connect two graphs, “Fungi” and “Industry”. Diagrams 4 and 5 show the rules that specify the internal structure of the food industry. By applying these transformation rules, Diagram 6 is obtained, which gives us two potential matches (baking and wine industry) on the left.

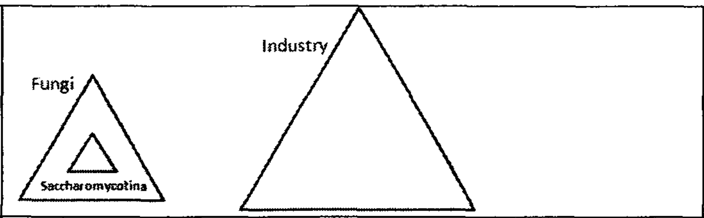
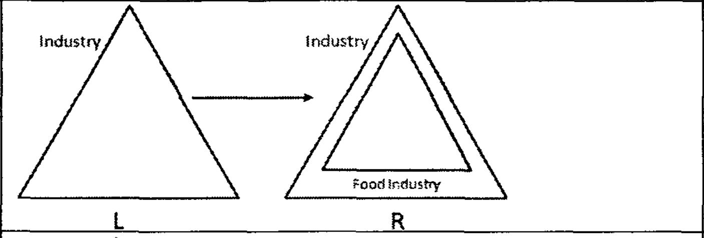
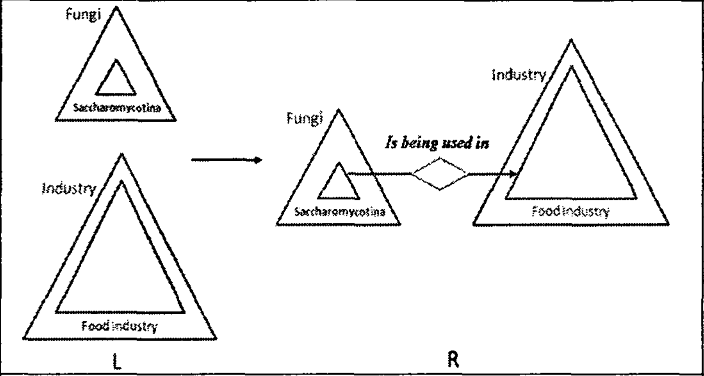
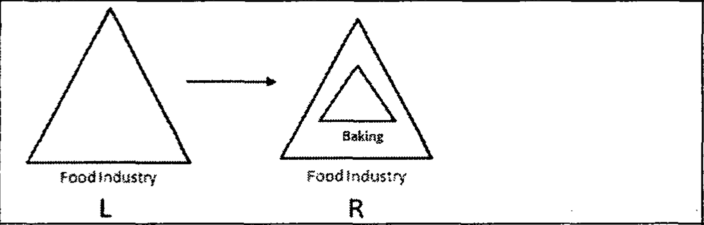
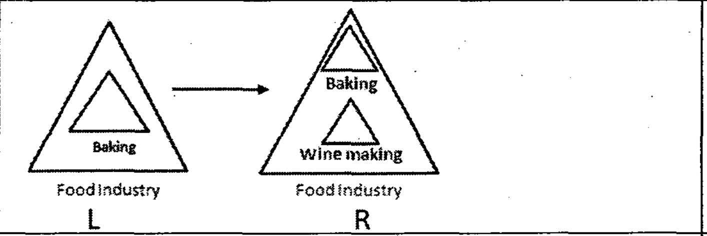
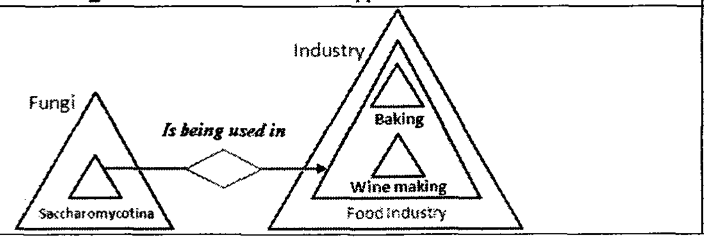
[120] For example, in a DL sense, a rule with a single literal in the head can be counted as atomic [FT05].

[121] A compound rule is made by combing the effects of two or more rules (atomic or non-atomic).

[122] In fact using graph transformation as the underlying formalism can guarantee the consistency of the results [TGM98]. This is an important point, since the distributing nature of evolving structures gives rise to different types of inconsistencies.

[123] For demonstrating the transformation rules in our model (Table 3.2), we employed the diagrammatical notions introduced in [Pal04].

Table 3.2. Some examples of the graph transformation rules for part of the FungalWeb Ontology.

1		Two individual graphs Fungi and Industry are in their initial state
2		Transforming the Industry graph (R) to the new version (L) to cover more detailed information (adding child)
3	 Fungi Saccharomycotina Industry Food Industry Fungi Is being used in Saccharomycotina Industry Food Industry L R	Defining the relation *"is being used in"* to connect the two graphs Fungi and Industry.
4		Adding a child node to specify the internal structure of the food industry.
5	 Baking Food Industry Baking Wine making Food Industry L R	Adding another child node to specify the internal structure of the food industry.
6	 Fungi Saccharomycotina Is being used in Industry Baking Wine making Food Industry	The two potential matches (baking and wine industry) can be chosen from the left hand side.

A graph transformation can be defined to be conditional [HHT96] in such a way that under certain conditions, the graph production (rules) transform a source graph into the target graph. These conditions, which impose a set of restrictions on the transformation processes, can help one to avoid inconsistencies and conflicts (e.g., the conflicts due to dangling edges).

III 4.5.4 Formalizing the Ontology Change Model in Distributed Environments

The hierarchical distributed graph can be used for analyzing dynamic distributed models and their transitions by describing the initial state, internal and external actions and defining communicating channels for synchronization. Category theory can be used as a complementary formalism for supporting graph grammar describing the initial graph and a set of all hierarchical graph productions modeling various actions (e.g., additions, modification of relations, and so on) in a distributed system. The double-pushout approach to graph transformation as a constructor within the categorical framework is comprehensively described in [Ehr79, EOP06] for directed and labeled graphs. This method has been generalized to so-called high-level replacement (HLR) systems in [EHK+90, EEP+06] by abstracting the results into arbitrary objects and morphisms[124]. It has been proven [Tae94] that the hierarchical distributed graph transformation is a highly appropriate scenario for HLR systems. Reflecting this approach into our framework, we consider the lattice L consisting of all interacting ontologies as a hierarchical distributed graph, with a set of transformation rules (e.g., rules for node addition/deletion), which is defined [Tae94] as a functor HD: $L \rightarrow \mathrm{G}$, where G is the category of all labeled graph

[124]The theory of HLR has been developed for different graphs, e.g. hyper-graphs, attributed and typed graphs, various Petri net classes, elementary nets, place/transition nets, and Colored Petri nets [Pad08].

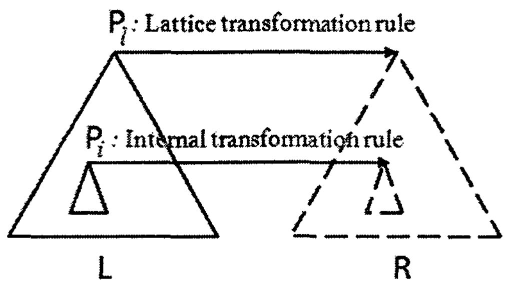
and L∈G. To define the HD-morphism we can use natural transformations, which are simply the morphisms in the category of functors.

A transformation rule can determine conditions such as: 'the deletion of a lattice node should be performed after deleting its corresponding internal graphs'. The hierarchically distributed graph transformation provides a means for dynamically analyzing model transformations in a distributed environment that consists of several hierarchically organized ontologies. Categorically speaking, the ontological structure can be considered as objects and the links between them, which shape the lattice structure, as morphisms. This approach allows one to study the behavior of evolving categorical systems in different layers (analogous to the modular definition of ontologies) and different levels of abstraction.

III 4.5.4.1 Distributed Change Management within the RLR Framework

In our approach, we adapted the graph transformation methods for realizing the problem of specifying changes in distributed ontologies in two levels of abstractions, namely micro level (changes in internal structure of an ontology, e.g., adding/deleting a concept to/from an ontology) and macro level (when the internal changes spread out to an interrelated ontological organization, e.g., changing the state of an ontology or adding/deleting an ontology to/from interrelated system). The propagation of changes may need to be performed during the runtime of many critical systems (e.g., knowledge bases supporting robotic surgeries or aviations); therefore, these two levels always need to interact closely to ensure the success of a change management strategy. We use distributed graph transformation to represent the dynamic nature of distributed ontologies, and to model a framework for describing the changes in an ontological structure and their effect on the other dependent artifacts organized within a lattice-like environment, such as the Semantic Web. The distributed graph transformation can act on different levels of abstractions, ranging from explaining the details of local actions to the rule-based analysis of different interactions and operations (e.g., inter-communication, migration, and synchronization) [TGM98] before or after a transformation. In order to successfully manage changes in a specific dynamic system, it would be essential to know, or at least have a reasonably accurate guess, about all the possible states of that system at different times. The fact that the dynamic system acts in a distributed environment makes this need more vital. Several studies [KM90, KM98, TGM98] have been done on managing the coordination between structural and state changes in software engineering.

The concept of distributed graphs has been defined in [CMR+97, TGM98] as networked compound graphs with a set of internal graphs as the nodes expressing a internal state of the system, and a set of graph morphisms as the edges connecting the nodes (internal graphs) to each other. Distributed graph transformation aims to mediate between these two levels of abstractions (networks and nodes) and can be used to model many different types of dynamic network reconfiguration [TGM99] by applying a set of rules for each of the levels (Figure 3.69) The rules contain the instructions for performing different changes (either in the network topology or in the nodes) and transformation in a dynamic system via defined actions at different levels of a distributed graph. The rules also determine whether or not a change operation is eligible to occur.



P_l : Lattice transformation rule
P_i : Internal transformation rule
L
R

Fig. 3.69. P_l and P_i respectively specify sets of lattice and internal transformation rules.

The communication between lattice and internal rules performed within a coordinated channel can be used to synchronize different actions in node and lattice levels.

III 4.5.4.2 Synchronization and Coordination

Managing several concurrent internal and external actions is also vital in the Semantic Web domain. Considering the Semantic Web as a hierarchically organized graph-like structure, each action on a graph has consequences in its modified consecutive version, which helps in tracing the events while preserving the reference state, or in some cases reconstruction of the past, if it has been removed from the original version. A hierarchical distributed graph production can be used for synchronization purposes by checking whether the external production is identical (or compatible) with what is performed by internal actions [Tae94]. More precisely, it checks if the lattice nodes and edges, in coordination with internal actions, have been identically replaced in the interface with respect to the gluing condition. For example, a graph production can describe a synchronous communication channel [Tae94] between two different versions of an internal graph by highlighting the revisions in the original state and the current state through the use of an interface graph. Later on, the action that causes a change in the internal graph needs to be synchronized with other actions on dependent internal graphs and finally with the actions that alter the external graph. In real world applications, this synchronization usually results in a series of mappings between the previous and current states. To manage the interaction between the actions on different levels, we generalize the change model proposed in [KM90, KM98, TGM98] for the software engineering domain to classify the changes in a dynamic network at nodes and network levels. The distributed Semantic Web environment can be conceptualized in a hierarchical lattice-like structure, composed of several ontologies as nodes and the links between them as edges. The changes in a lattice-like structure can be performed at the nodes (e.g., replace/rename a node), edges (e.g., replace an edge) or hierarchical structure (e.g., adding/deleting one or more nodes).

The agents in the RLR framework interact with each other through a set of communication channels to control actions at different levels. This control assures the consistency and integrity of changes by defining quiescent[125] nodes and states. The nodes are assumed to be in a quiescent state (non-active/passive state) when changes occur at the lattice level. According to [KM90], a quiescent state for a node is a state wherein the whole system is consistent and no active communication exists between the nodes or within their environment. The notification for changing the node's state from active to passive (and vice versa) is given through the established communication channel between the defined abstraction levels. In RLR, upon detection of the alterations by the set of change capture agents, the current state of the system would be assigned to the newly

[125] This strategy is similar to "locking" in database research.

affected elements (e.g., newly added nodes) and an alert would be sent to the other involved components to inform them about the latest state of the system.

The state of a system should be determined and declared by an agent to allow some actions to be performed in a proper state of the system, to postpone them for later states, or to prevent them from acting on some of the preserved elements. For example, in the case of deleting or splitting a node, it acts like the lock mechanism in the database. The synchronization begins with assigning the states to each element, starting with the initial state upon its creation and continuing until the final state is assigned upon its termination. RLR controls the changes by incorporating the transformation rules (at different levels) along with other pre-defined consistency conditions. The synchronization of two different nodes (internal graphs) in a distributed graph can be performed through an interface [TGM98] that connects these nodes together. The transformation is performed by a sequence of simpler transformations, each meeting certain conditions to ensure the target graph is still a distributed graph and to avoid any side-effects (explicit or implicit) on the graph structure. Some of these conditions are as follows [TGM98]:

- Gluing condition of the double-pushout approach for the rules at different levels;
- Connection condition, which prevents the deletion of the nodes and the edges if they are being used by other components.

Also some other conditions and restrictions may be applied to each distributed rule, depending on its function. The main context conveyed by the lattice may be defined as protected to keep it unchanged. If the different actions and changes that are executed at the node's level have minimal or no interference with each other, they can operate in parallel. Assume a set of related ontologies, each with the ability to manage the changes in its own structure and each change potentially affecting other ontologies. An agent can initiate an action for changing each ontology in the lattice, based on imposed rules. This action can then be spread throughout the entire lattice. The distributed graph transformation can be used to model real-time changes, such as the insertion or deletion of ontologies. This is important since many changes and updates, unseen in the design phase, can be applied when the system is in operation if they do not cause any interruption. If we consider changing a node, it should be flagged as an inactive state, so it will not update the system's knowledge upon a change (neither initiate an update nor service any update request [TGM98]).

III 4.5.4.3 Rule-based Patterns for Transformations

After each change, the system needs to be verified for consistency. In order to preserve the ontological elements' identities and guarantee the consistency and integrity of the changes, we can define a set of pre- and post-conditions to be satisfied. If all the conditions within a distributed graph transformation rule are satisfied, then the result of transforming an initial distributed graph would be a legitimate distributed graph as well. Consider the three ontologies (O_1, O_2, and O_3), connected to each other in a lattice-like structure. Each node of the lattice represents an ontology and each edge signifies a graph morphism. The information about the state of each ontology and its relations with other ontologies in the lattice is stored in an interface node. The diagrams in category theory intuitively reflect the feasibility of our method, by demonstrating the interactions between the states and the information related to the changes. By following the method given in [TGM98], Figure 3.70 demonstrates the changes in industrial applications within the

FungalWeb Ontology (as an internal graph in a whole integrated lattice), which consists of the concepts "enzyme" and "product", with the relation "uses". The figure depicts the effect of changes and the state of the ontology (starting from initial inactive state) in the lattice-like environment, along with its predecessor and successor versions, using the following distributed graphs:

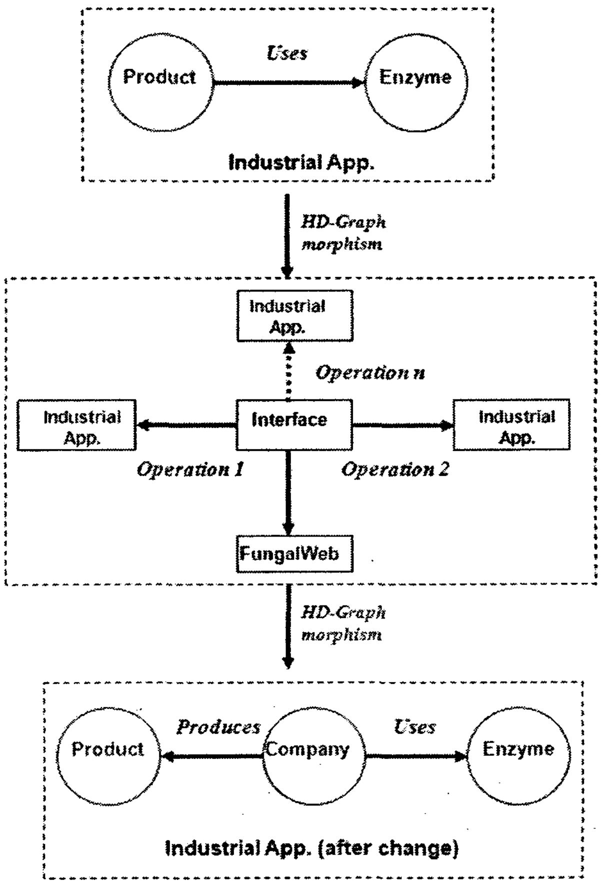


Fig. 3.70. Representation of a change in a part of the FungalWeb Ontology using graph transformation.

In the Figure 3.70, assume an update (internal action) starts at the FungalWeb Ontology to delete the existing relation "Uses" and add the new concept "Company" and the new relations "Uses" and "Produces" to relate the newly added concept with concepts

"Enzyme" and "Product" respectively. We apply the following rules to perform this update:

Add interface node ("FungalWeb Interface"),

Operation 1: *Add ontological element _Concept (FungalWeb, "Company");*

Operation 2: *Delete ontological element _Relation (FungalWeb, "Uses");*

Operation 3: *Add ontological element _Relation (FungalWeb, ""Company", "Product", "Produces");*

Operation 4: *Add ontological element _Relation (FungalWeb, ""Company", "Enzyme", "Uses").*

To hide unnecessary details, the change processes and related interactions are performed via interfaces[126] (cf. Figure 3.70). As mentioned in Section III.2, in using category theory, we focus on the interactions between objects rather than their internal structure. In summary, in our categorical representation of a hierarchical graph organization, anything other than nodes and edges (e.g., attributes such as data type properties for ontologies) are supposed to be marginal and not essential [BKK05]. Thus, the notion of graph transformation can be defined [BKK05] as $G,R \Rightarrow C,E$, with G, R, C, E respectively indicating a category of graphs, a category of rules, a category of control conditions, and a category of graph expressions (cf. [BKK05] for more information). Modeling the notion of graph transformation in an abstract way is significant in the sense that it hides the marginal information, which does not explicitly contribute in the transformation process. As an example, a transformation using the double-pushout (DPO) has been shown in Figure 3.71 for part of the FungalWeb taxonomy. The transformation rule determines a condition for a consistent deletion operation within an ontology by specifying that if a parent-node has to be deleted its children should be deleted as well.

[126] "Interface generally refers to an abstraction that an entity provides of itself to the outside. This separates the methods of external communication from internal operation, and allows it to be internally modified without affecting the way outside entities interact with it." [MVM10].

The double-pushout approach, constructed based on categorical pushout, in our example has been generally represented as the gluing of two graphs via a common interface.

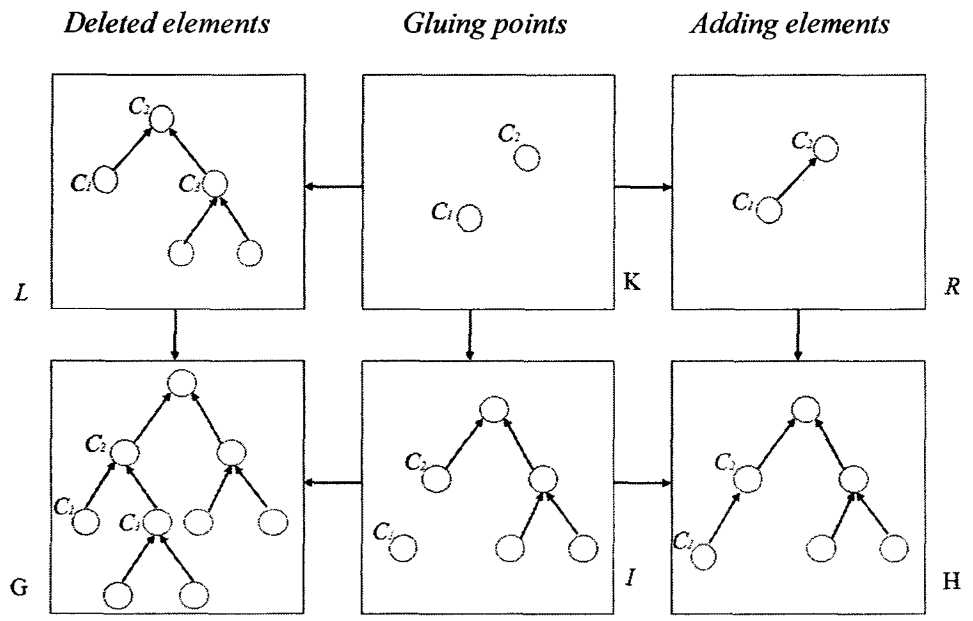


Fig. 3.71. The transformation of part of an ontological structure following the rule "deletes a parent node". The upper part represents the transformation rule, and the bottom left shows a given graph and the bottom right demonstrate the result of the transformation, which has been obtained by following the three steps in DPO (see III 4.3.1.1)

As shown in Figure 3.71, the left side indicates a pattern[127] to be located in the original graph (G); the right side represents the requested transformation, which transforms the original graph (G) to the transformed graph (H); and the middle section represents the gluing point(s) (C_1 and C_2), which are identified by L ∩ R.

In the RLR Framework the agents generalize the behaviors by systematically monitoring the transformations and encapsulating the changes from one point to the subsequent position to extract rules and generate the patterns. The patterns can be

[127] In order to define a pattern to be always applicable it would be sufficient to leave the left side of the associated rule empty.

repaired, improved, and evolved through an intensive didactic teaching[128] process, which enables the agents to derive rules from a sequence of trial state changes[129].

III 4.5.4.4 Similarity Checking and Traceability

A graph comparison methodology has been presented in [DHP02] to compare the contents of two graphs by considering the number of nodes and edges. The comparison has been performed based on applying the rules while considering the hierarchical dangling condition, to check whether a specific sub-graph exists or not[130]. This approach has been later used to perform hyperedge replacement and substitution. RLR intends to audit and monitor very large, heterogeneous, evolving biomedical ontologies and nomenclature scattered across the Web by highlighting changes between different versions of an ontology. In order to facilitate the change tracking process, we employ diagrammatic features on graph representation along with category theory, which enable us to represent the system's activity in different levels of abstraction. Our approach is similar to the tracking graph transformation approach [BKK05], which models the rules' internal structure by means of LHS (left-hand side) and RHS (right-hand side) graphs and a partial morphism between them, which facilitates the tracking of preserved graph components between two versions of a graph through a set of consistency constraints to check matching morphisms.

[128] Coleman, A. Didactic Teaching. http://www.resus.org.uk/pages/IDnpP_AC.pdf
[129] The idea of extracting rules as general behavior descriptions from sample state transformations is called *programming by example* and represents the main didactic tool of the *Stage-Cast* environment [Hec06].
[130] This can be performed when one attempts to delete a graph.

III 4.5.5 MAS and Graph Transformations

The transformation rules can be used to determine and model agents' behaviors and operations in MAS [KK99, DHK00]. They also capture the effects of different agents' actions and operations on local or network levels, thus as a representation method, these rules enable modeling the agents' cooperation and interactions. When we consider graph transformation for formalizing agents' interactions and cooperation by means of communication with the other agents within a specific MAS or between different MAS systems and with their environment, it can be used for representing the transformation of the agents' communication network. To analyze changes in relations between a set of cooperative agents within a generic multi-agent system (MAS), considering the category of MAS, a transformation mechanism based on DPO can be defined by finding a pushout complement for a particular state through examining the gluing condition.

Considering the challenges for modeling changes in distributed systems, which involve several issues including traceability and synchronization, RLR utilizes a distributed graph transformation technique, which explicitly supports the synchronization and concurrency processes. For the sake of consistent change management, a process within the RLR model needs to be synchronized with its adjacent processes in order to evolve coherently. For example, if two operations want to act on a common ontology through specific actions and conditions, these actions should act under a consensus agreement so they can both perform and evolve coherently. To coordinate the potential changes in the processes, a set of synchronization requests are issued at each abstraction level. These requests need to follow certain transformation rules and conditions, which are compatible with each other[131], in such a way that they support the concurrent evolutions of different parts of the system autonomously based on the consensus agreement. To ensure the consistency of the transformations, we enforce certain types of reactions and behaviors (preferably among several options) for agents in certain states, when the conditions are applicable (determined by L in the rule $L \leftarrow I \rightarrow R$). The overall effect of an action within a scenario (e.g., select a node to be deleted) is described by a pair of instance diagrams[132], modeling the before/after states [DHK02]. Sequences of transformations represent the changes in the states' agents and their behavior, and model their interactions within the communication channels in a MAS[133]. For example using the method presented in [DHK02], we consider the communications between the Explorer Agent (EA) and the Log-Reading Agent (LRA), in RLR (described in Section III 2.3.1) to capture the type of change operation (Figure 3.72).

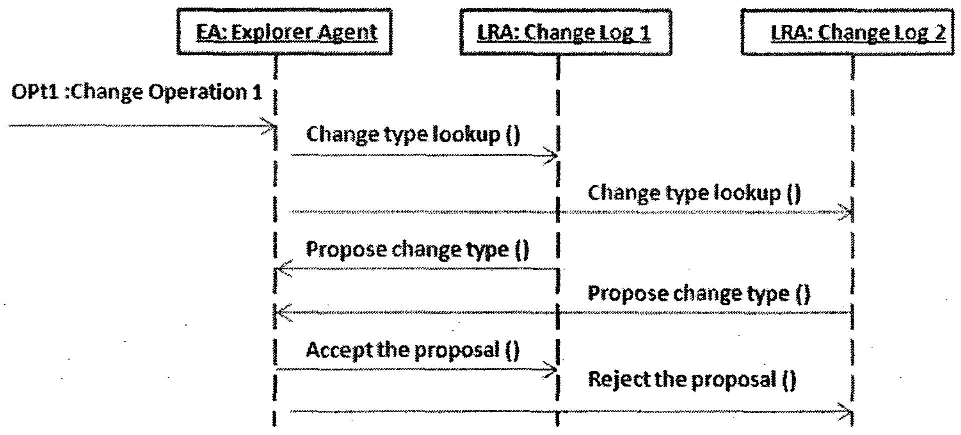


Fig. 3.72. The communication between explorer and log reading agents to specify the type of a change operation. These communications can be placed during the negotiation phase.

[131] The compatibility here refers to this fact that the combinations of these transformation rules should keep the entire system in a consistent state.
[132] In UML, instance diagrams (object diagrams), are useful for exploring "real world" examples of objects and the relationships between them, while the type diagram reflects a given Use Case. [UML2]
[133] In agile object oriented modeling , This usually is represented by UML sequence diagrams [UML3].

Due to the ability of graph transformation for handling temporal representation [GVH03], we have used the rule-based graph transformation [DHK02] to describe the pre- and post-states of an agent-based model (Figure 3.73), grounded on the communication diagram demonstrated in Figure 3.72. In this figure, the Explorer Agent (EA) reacts to alterations that appear in the environment (e.g., a change operation) and tries to affect the environment by locating the change and determining its type based on different proposals. In the same way, one can define other rules for rejecting the proposals, storing the proposals for future decisions, or aborting all the communications.

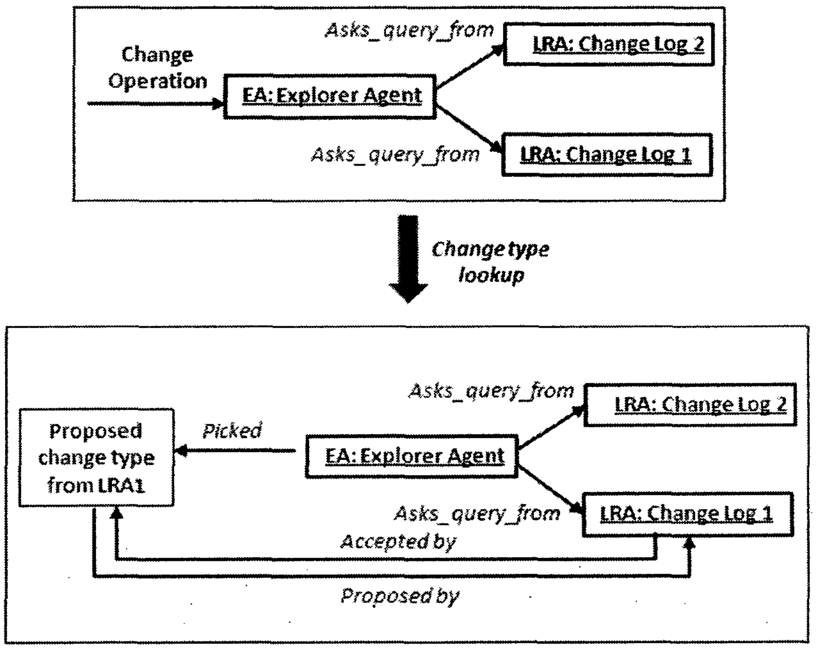


Fig. 3.73. A generic transformation rule for describing the pre- and post states in an agent-based model transformation based on the communication diagram demonstrated in Figure 3.72.

By noticing the fact that many of the current dynamic agent models are represented by sequence and state diagrams, which have been studied here under a graph-oriented approach as well, we can extend our approach to study agents' model transformations in more complex situations. For example, by following some of the object-oriented principles like differentiation between instance and type graphs (diagrams)[134] ([CMR96], [BH04]), we can model the typed graph transformations [HCE+96, GPS98] by means of refactoring [SPL+01, MED+05, Men05] the state diagrams (adding/removing, merging, or decomposing the states) and conceptual models of ontological structures.

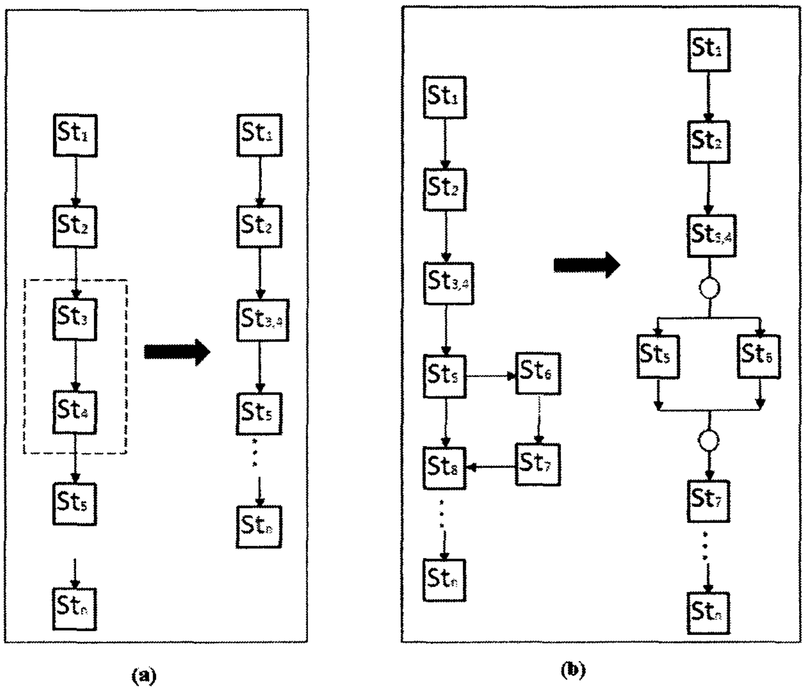


Fig. 3.74. (a) pre/post state representation before/after merging two states; (b) the representation of concurrency of two parallel states.

As an example, following the approaches presented in [BSF02] and [Men05], we may merge the two states St_3 and St_4, which respectively represent the state of the RLR system after querying to determine the type of change and receiving the proposed answers, into one merged state $St_{3,4}$ (Figure 3.74 (a)). As another example, Figure 3.74 (b) demonstrates the transformation of a state diagram to the new diagram, representing

[134] In conceptual modeling a type graph models a class diagram and an instance graph models the objects (instance) diagram.

the concurrencies between the two states St_5 (validation of the accepted response), and state St_6 (ask permission to put the results into an action). In fact, we would be able to model various aspects of agents in the RLR model, such as agents' networks, topology, properties, interactions, and cooperation based on the agreed goal in the negotiation process. Also, due to the rule-based nature of this framework, we can formally model the structure and behavior of an evolving system and anticipate certain types of transformations and re-configurations upon future changes.

III 4.6 Summary of Contributions in Section III.4

In Section III.3 a formal framework for managing changes in ontologies based on category theory has been defined. On top of this formalism, we defined a graph transformation approach to manage ontological changes by means of model transformation. In this method, graphs correspond to the evolving ontological structures and graph transformation has been employed to model their evolution. Semantic web is considered as hierarchical graphs with the ontologies composed of RDF/OWL triples' graphs as its nodes and the relations between these ontologies as its edges[135]. Therefore, we can naturally use graph transformation to define the changes in an ontology (or a series of related ontologies) and control the consistency of the result by imposing the rules and conditions to guarantee that the transformation result is a valid hierarchical graph as well.

Graph transformation offers the means for analyzing updates and changes in graph-like structures. As well, there is a vast amount of theoretical studies with promising

[135] The relation between the OWL/RDF triples also represent edges for the ontology graphs.

outcomes readily available. The transformation rules can be used to describe merging and integration of internal graphs (analogous to the concatenation operation presented in [DHP02]). An approach for verification of whether a transformation indicated by a double-pushout rule is consistent or not has been shown in [BKK05], by demonstrating that every rule in a rule-based graph transformation can satisfy the path-checking and root-checking (due to root-level morphisms) conditions. In contrast to other existing methods, we do not limit ourselves to the specific type of implementation language. Moreover, our model is equipped with a category theory formalism and rule-based transformation mechanism, which enables us to represent the dynamic nature of ontological elements not only in isolation but also considering their interactions with other dependent components and artifacts in a distributed Semantic Web environment. Also the purpose and domain of our approach differs from other currently ongoing efforts in this area. In summary in this section we have presented the following major contributions.

- Providing a graph-oriented semantics for analyzing temporal biomedical ontologies;
- Extending the existing graph-based analysis for RDFS/OWL ontologies, by means of hierarchical distributed graphs, which enables one to deal with nested distributed ontologies in the real world applications;
- Employing category theory along with graph transformation to represent, and analyze changes in distributed biomedical ontologies in different levels of abstraction, independent of any implementation language;

- Defining transformation rules for evolving ontologies that ensures the consistency of the results and coordinates the communications and interactions between different agents for concurrent and parallel actions.

In the next chapter we demonstrate the feasibility of our approach through a series of experimentations on different application scenarios.

IV. Application Scenarios & Case Studies

The applicability of our proposed method for managing change in ontologies has been already demonstrated throughout several examples in Chapter III. In Chapter IV we represent that the techniques presented in our proposed RLR framework can be joined together to serve as a blueprint for designing practical algorithms for maintaining changes in several domains. With the extensive popularity of biomedical ontologies in modern knowledge bases in healthcare, we believe our method is not only applicable for managing evolving biomedical ontologies, but also appropriates for many other topics, including requirement engineering and model analysis, and phylogeny evolution, where formal representation and analysis of changes are key to overcome parts of the big problem of bootstrapping the evolution process.

IV.1 Case Study 1: Managing the Evolving Structure of an Ontology for Clinical Fungus

There's a tiresome young man of Bay Shore
When his fiancee cried: "I adore
The beautiful sea!"
He replied, "I agree;
It's pretty, but what is it for?"

Morris G. Bishop (1893-1973)

Life sciences constitute a challenging domain in knowledge representation. Biological data are highly dynamic, and bioinformatics applications are large and there are complex interrelationships between their elements with various levels of interpretation for each concept. At this time, we are applying the proposed methods for managing changes in the FungalWeb Ontology which is the result of integrating numerous biological databases, web accessible textual resources and interviews with domain experts and reusing some existing bio-ontologies. To use the FungalWeb framework more practically in the medical domain to support dermatological practice and enhance the accuracy of clinical knowledge management, we have also modeled the SKin-Disease ONtology (SKDON), an integrated OWL-DL ontology with focus on medical mycology for dermatologists. In our work, we have concentrated on disorders of the skin and related tissues, such as hair and nail due to fungi. SKDON is created from several distributed resources, including structured/unstructured texts, online databases, and existing controlled vocabularies, such as MeSH [NLM94], ICD-9[136], SNOMED[137] and Disease database[138]. Cross referencing between the FungalWeb ontology, SKDON and MeSH "Chemicals & Drugs" category

[136] http://www.cdc.gov/nchs/icd9.htm
[137] http://www.snomed.org/
[138] http://www.diseasesdatabase.com/

provides valuable information about the disease, the involved fungus and the drugs prescribed. Change in any of the resources can alter the definitions in the target ontology.

Recalling our discussion in Section III.1, as the knowledge about fungi species grows and new methods become available one can anticipate a fundamental change in the current fungal taxonomy structure. From the other way since skin disorders have been historically categorized by appearance rather than scientific and systematic facts [PCB+04], the existing taxonomy of fungal diseases must be also modified based on the new knowledge to update the ontological truth. Many terms in current medical mycology vocabularies describing skin disorders originate as verbal descriptions of appearance, foods, people, mythological and religious texts, geographical places, and acronyms [AAR+03]. Many names and terms are highly dependent on individual or regional preferences, causing redundancy, vagueness, and misclassification in current vocabularies. Thus, we study various alterations in both fungal taxonomy and fungal disease classification. As an example of changes in fungal terminologies, one can see several changes in the name of pathogenic fungi *Trichophyton* family (i.e. *Trichophyton Soudanense*, *Trichophyton megninii,* and *Trichophyton equinum*) in relatively short period of time. As another example, the pathogenic fungus *Candida glabrata* is now called *Torulopsis glabrata* [CS05b]. Usually changes in fungi taxonomy alter the related disease name and description (Figure 4.1). For instance, the name of the fungus, *Allescheria boydii* which can cause various infections in humans, was changed to *Petriellidium boydii* and then to *Pseudallescheria boydii* within a short time [OAD+92]. Consequently, the infections caused by this organism were referred to as *allescheriasis, allescheriosis*, *petriellidosis*, and *pseudallescheriosis* in the medical literature [OR95].

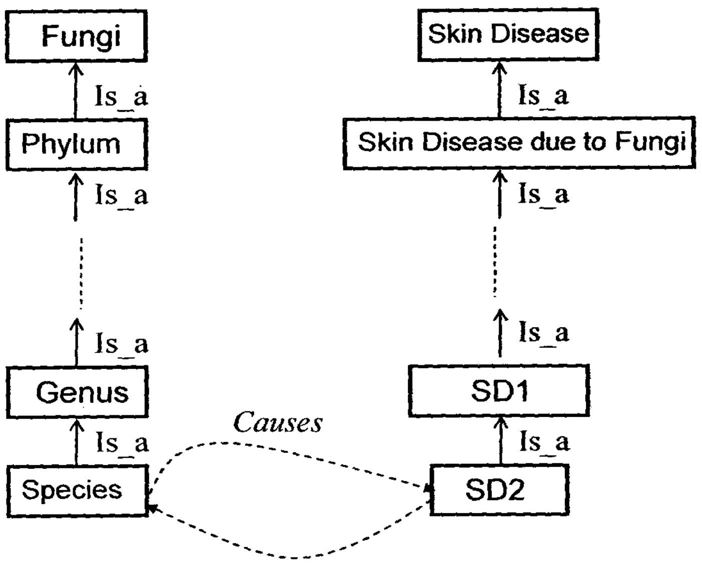


Fig. 4.1. Changing the fungi name can change the related disease name.

Fungal Meningitis is an infectious disease caused by three types[139] of fungi (*Candida albicans*, *Cryptococcus Neoformans*, and *Histoplasma*). Cryptococcal Meningitis is caused by fungus *Cryptococcus Neoformans*[140] and is typically seen in patients with immune deficiency (Immuno-Incompetent) such as AIDS. It usually results from an infection that spreads to patient's brain from another part of her body. This disease has been a subject for study in both dermatology [Leu90] and neurology [ST95] for a long time. The knowledge about this disease (i.e. symptoms, causes, etc.) are scattered in several existing ontologies and knowledge bases, which need to be aligned. As described in Section III 3.5.4.1 and also pointed out in [ZKE+06], and [CH07] we can model the alignment of two taxonomical structures (O_1 and O_2) by means of a pair of mappings from an ontology O (Figure 4.2).

[139] Meningitis Research Foundation of CANADA: http://www.meningitis.ca/en/what_is_meningitis/fungal.shtml

[140] Here is the lineage of *Cryptococcus neoformans* in the FungalWeb Ontology: Fungi; Dikarya; Basidiomycota; Agaricomycotina; Tremellomycetes; Tremellales; Tremellaceae; Filobasidiella; Cryptococcus neoformans (Filobasidiella neoformans).

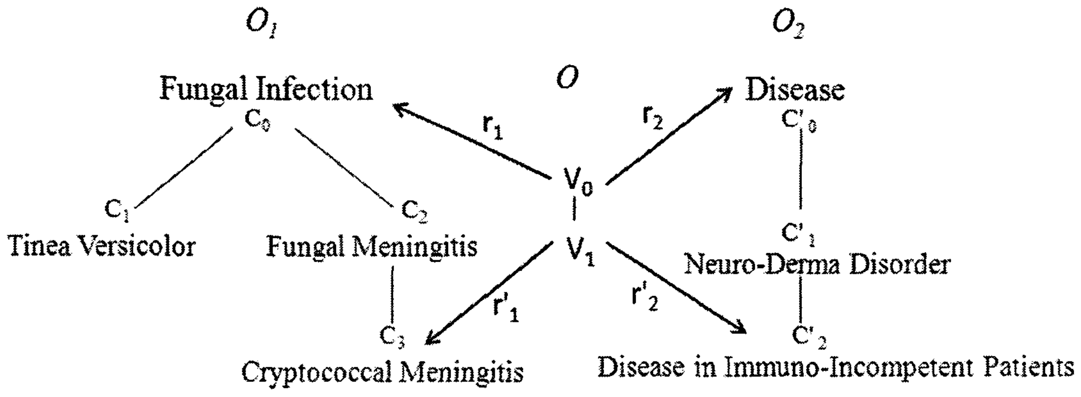


Fig. 4.2. The diagrammatic representation of the alignment between the two taxonomies from O_1 (Fungal disorders), and O_2 (Diseases) using a set of mappings from ontology O (using the format given in [CH07]).

In order to achieve a composite knowledge of the disease's properties we have used the categorical product to represent this integrated view (Figure 4.3). As can be seen in the Figure 4.3 medical specialty is the product arrow of the two branches in medicine, which includes the attributes of both domains.

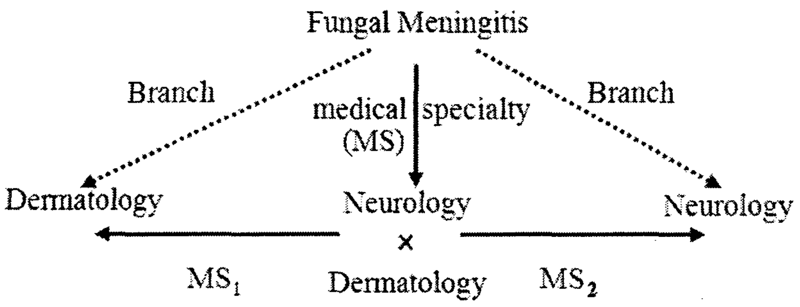


Fig. 4.3. Determining the medical specialty for a particular disease through product.

As mentioned in Section III 3.5.4.2 in order to merge two unrelated ontologies we can simply perform the disjoint union (or co-product). In our domain, we need to update and improve the ontological structure of the FungalWeb and SKDON Ontologies regularly for the annotation of fungal genes and analyzing the role of the fungi species in various diseases. For example, the older version of the FungalWeb Ontology did not have sufficient terminology to annotate genes involved in *Malassezia* infections. To meet this new requirement, the updated version of the ontology has gained 26 additional terms addressing these infections.

As we represented in our research, category theory within the RLR framework has a significant potential to be considered as a supplementary tool to capture and represent the full semantics of ontology driven applications and it can provide a formal basis for analyzing complex evolving biomedical ontologies. Figure 4.4 demonstrates a portion of the structure of the FungalWeb application in a diagrammatic representation.

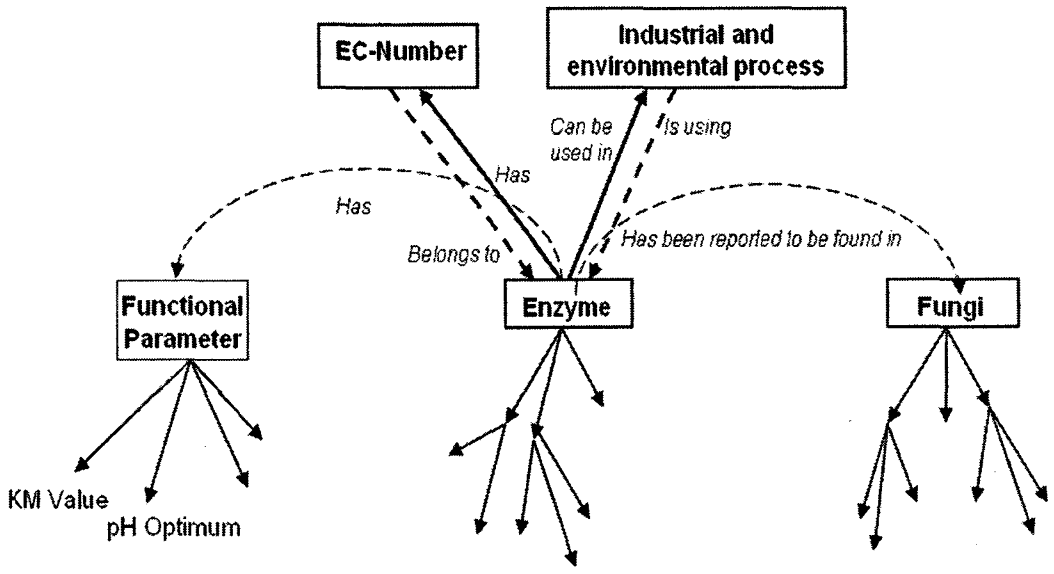


Fig. 4.4. A diagrammatic representation of portion of the FungalWeb application.

As one can see in Figure 4.4 many of the nodes can be considered as one individual graph within the whole ontological structure, with several dependencies to different objects. Figure 4.5 represents this interconnectivity between different ontological components.

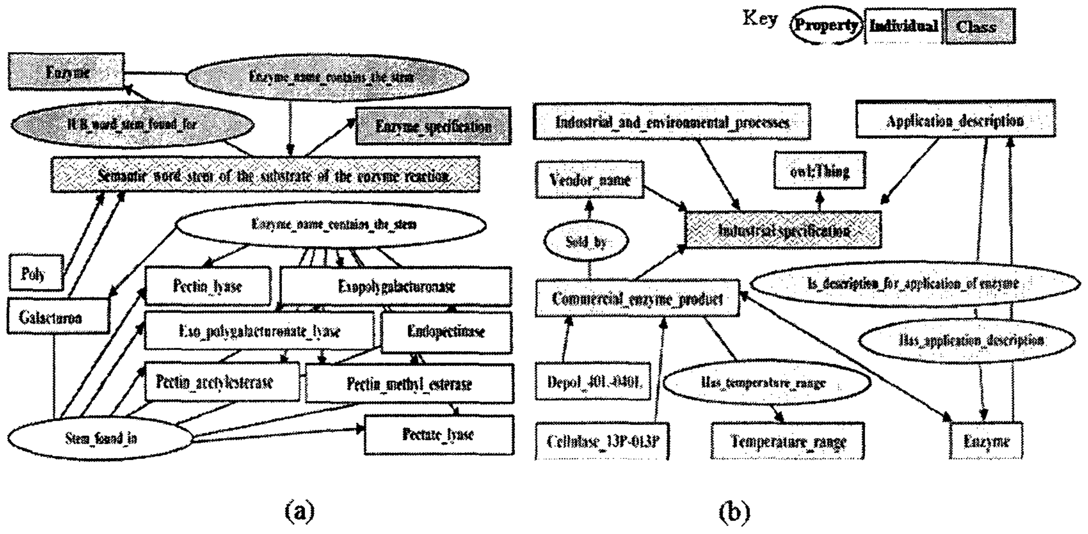


Fig. 4.5. (a) A portion of the FungalWeb Ontology representing the conceptual frame supporting the identification of enzymes acting on polygalacturonic acid. (b) Conceptual frame supporting the identification of enzyme vendors, the characteristics and application domains of their products [BSS+06].

The FungalWeb Ontology as an integrated structure consists of several parts from other knowledge resources, which combined through their aligned components and merged into a consistent framework. Figure 4.6 represents an example of partial merging of two conceptual models (in the left) via a common component Substance/Product.

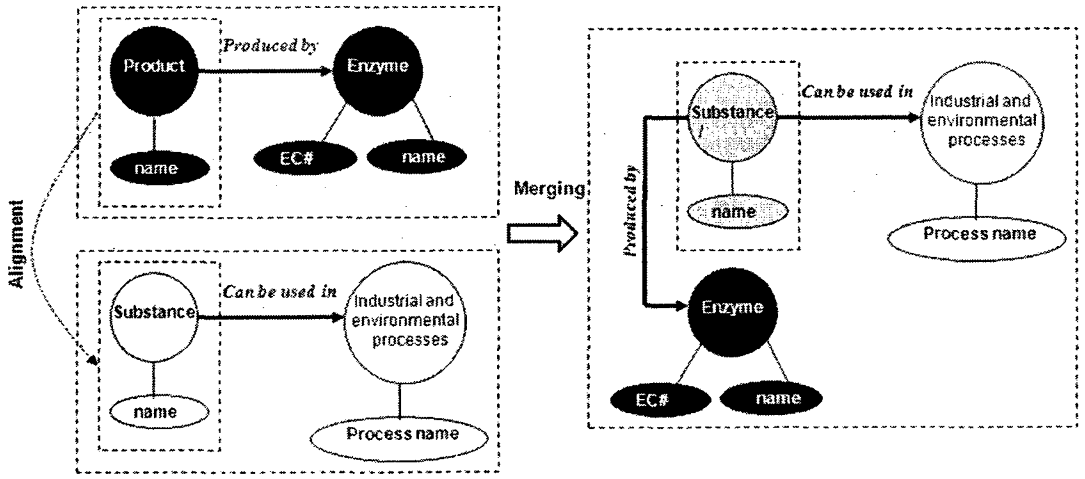


Fig. 4.6. A merging process based on the common elements between two parts of the FungalWeb Ontology.

The representation of a change in a part of the FungalWeb Ontology using graph transformation has been already shown in Section III 4.5.4.3 (see Figure 3.70). Also some examples for defining transformation rules and applying the double-pushout approach have been demonstrated in Table 3.2 and Figure 3.71 respectively. In fact ontologies are not isolated structures, but they tend to be reused as much as possible. The Semantic Web ultimate vision is to bring the existing ontologies, knowledge bases, controlled vocabularies, thesauri, databases and linked data sources under one umbrella, in such a way that they can communicate with each other and with users in a coordinated interactive manner. As mentioned earlier in Section III.1, the FungalWeb ontology is in close contact with other resources such as Gene Ontology, TAMBIS, SwissProt, BRENDA, and etc (Figure 4.7).

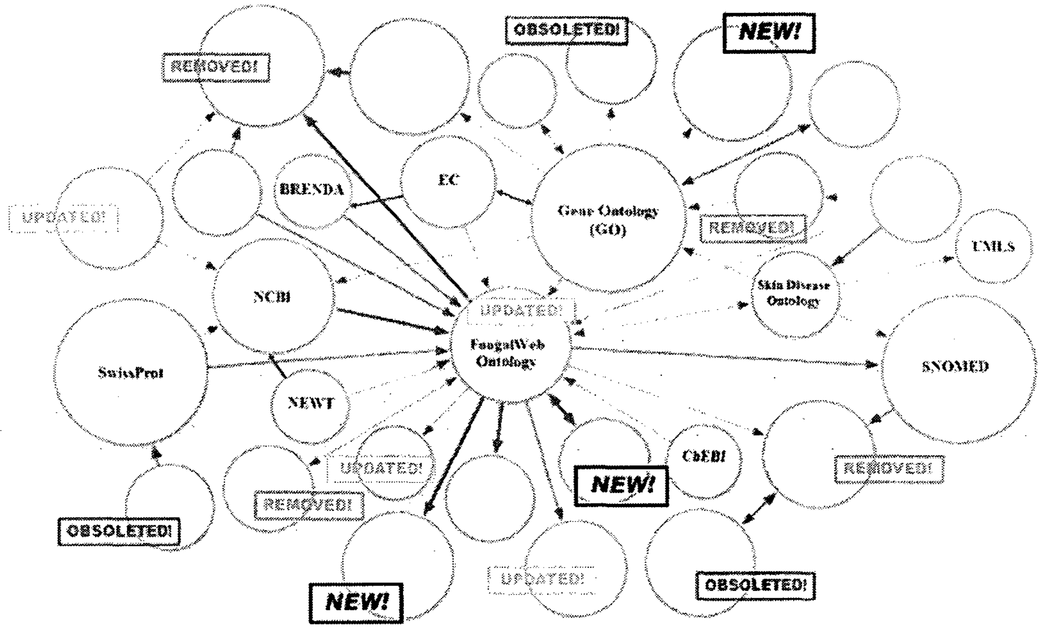


Fig. 4.7[141]. Interrelated distributed ontologies, knowledge bases and data sources in the FungalWeb project.

[141] For the visualization purpose, we used the format presented at W3C "Linking Open Data" project. http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData#dbpedia-lod-cloud

It is highly desirable that all changes within a resource can be tracked and all the impacts of such changes as well as their directions can be recognized and indentified. In our approach changes to each part of the ontology can cause the conceptual design changes its state, which may cause alterations to other dependent artifacts. In order to represent different states of our conceptualization, we use a categorical discrete state-model, which describes the states and events in the ontological structure using a diagrammatical notation. The discrete state-model can be specified by a state space (all potential states), a set of initial states and a next state function. Based on our application we designed our class diagrams following the method described in [Whi97] (Figure 4.8). The Op_i arrows in this figure represent the operations performed on the ontological structure. In this case, the operation or event op_1 causes an object in state St_1 to transition to state St_2. The operation Op_1 has no effect upon the object if it is in any other state, since there is no arrow labeled Op_1 which originates in any other state.

$$\begin{tikzcd}
 & \varnothing \arrow[d, bend left, "\text{Destroy}"] & \\
 & St_1 \arrow[u, bend left, "\text{Create}"] \arrow[dr, "op_1"'] & \\
St \arrow[ur, "\pi_1"] \arrow[rr, "\pi_2"] \arrow[dr, "\pi_3"'] & & St_2 \arrow[ul, bend right, "op_3"'] \arrow[dl, "op_2"'] \\
 & St_3 \arrow[uu, bend right=80, "op_3"'] &
\end{tikzcd}$$

Fig. 4.8. A Class diagram for part of the FungalWeb class structure that represents the transition between states.

The object ∅ in the diagram is the null state. The create arrow represents the creation of the object by assigning an identifier to the object and setting its state to the initial defined state, and the destroy arrow represents its destruction [Whi97].

As we described in Section III 4.5.1 the hierarchical graph transformation can be used to analyze the changes in interrelated biomedical resources in the sense of a sequence of transitions and transformations. These transformations assist for studying changes in the micro level (in the nodes of each internal graph) and the macro level (changes in lattice structure). Defining appropriate transformation rules, such as what is represented in Figure 4.9, is the first step towards performing a transformation. As mentioned earlier (III 4.3.1.1) finding proper pushout complements is one of the key point in categorical graph transformation.

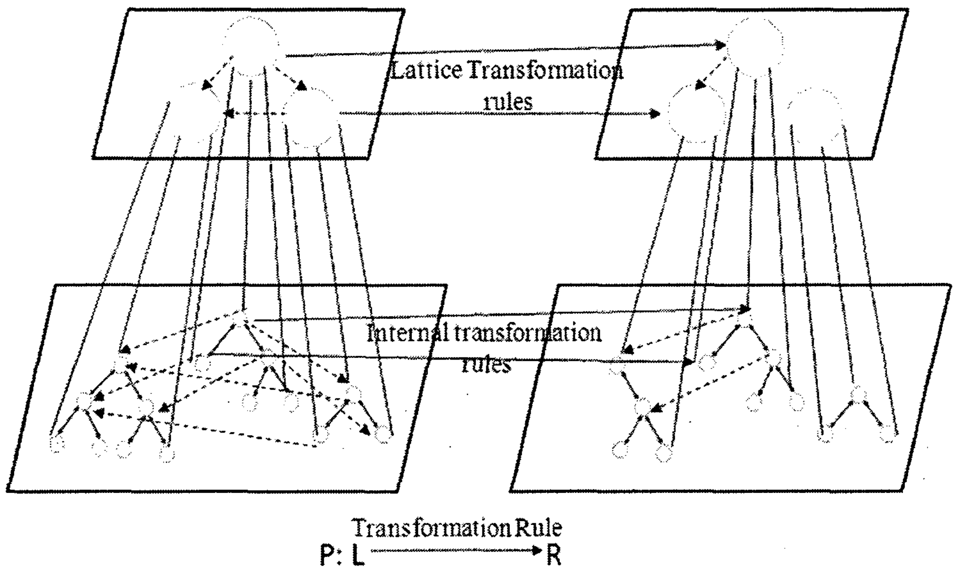


Fig. 4.9. A distributed transformation rule, which regulates the transformation of different interconnected ontologies in two abstraction levels, namely internal and lattice.

Recalling the definition of category DGRAPH in Section 4.5.1 and using the approach proposed in [Tae99] a pushout over distributed graph morphisms with respecting to both lattice (network) and internal (local) morphisms can be constructed, which enables us to apply the defined pushout-based transformation rules (see Section III 4.3) to describe changes in the distributed ontologies.

IV.2 Case Study 2: Managing Requirement Volatility in an Ontology-Driven LIMS

In an ideal situation, the requirements for a biomedical system should be completely and unambiguously determined before design, coding, and testing take place. The complexity of bioinformatics applications and their constant evolution lead to frequent changes in their requirements: often new requirements are added and existing requirements are modified or deleted, causing parts of the software system to be redesigned, deleted, or added. Such changes lead to volatility in the requirements of biomedical applications. In this section, which is partially based on our published journal paper [SOK+09][142] and conference paper [SH07c], we deal with an important problem of requirements volatility in the context of an ontology-driven clinical Laboratory Information Management System (LIMS) [Mcd93, AMF00]. A LIMS is a software application for managing information about laboratory samples, users, instruments, standards, and other laboratory functions and products. It forms an essential part of electronic laboratory reporting (ELR), and electronic Communicable Disease Reporting (CDR). ELR is a key factor in public health surveillance, improving real-time decision making based on messages reporting cases of notifiable conditions from multiple laboratories [OSM01]. Combining these reports with clinical experiments and case studies makes up a CDR system [WC05]. This framework,

[142] The definitions of the requirements' refinement models and the effects of various requirements on each other were contributed by the two co-authors.

along with the active participation of physicians specializing in fungal infectious diseases, infection control professionals, and lab technicians, is aimed at generating automated online reporting from clinical laboratories to improve the quality of lab administration, health surveillance, and disease notification. It provides security, portability, and accessibility over the Web, as well as efficiency and data integrity in clinical, pharmaceutical, industrial, and environmental laboratory processes.

Research Problem: Requirements volatility is: "a measure of how much program requirements change once coding begins" [ED07]. Bioinformatics applications with frequently changing requirements have a high degree of volatility, while projects with relatively stable requirements have a low one [MD99]. Higher requirement volatility will result in higher development and maintenance costs, the risk of schedule slippage, and an overall decrease in the quality of the services provided. Therefore, requirement volatility is considered one of the major obstacles to using a LIMS. In this section, we propose an innovative approach for the automatic tracing of volatile requirement changes based on their formal representation in an ontological framework and using category theory as a solid mathematical foundation.

Approach: Investigating the factors that drive requirement change is an important prerequisite for understanding the nature of requirement volatility. This increased understanding will minimize that volatility, and improve the process of requirement change management. One of the most important volatility factors is the diversity of requirement definitions in the application domain, which may lead to confusing and frustrating communication problems between application users and software engineers [Wie03]. Conceptualization of the requirements using an ontology minimizes the requirement volatility by providing a deep and common understanding of the requirements [DS06], which is essential in order for bioinformatics application developers to manage the changes successfully. In this section we apply our proposed approach to model LIMS requirements with an emphasis on nonfunctional requirements, their dependencies and interdependencies using category theory. The resulting categorical model represents the functional requirements (FRs) and nonfunctional requirements (NFRs) based on an investigation of their dependencies and interdependencies, which is considered critical to success in tracing requirement changes. Requirement traceability, defined as "the ability to describe and follow the life of a requirement in both a forwards and backwards direction" [GF94] is an essential part in performing requirement maintenance and change management processes. Moreover, the extent to which change traceability is exploited is viewed as an indicator of system quality and process maturity, and is mandated by existing standards [ANR+06]. These changes have to be monitored for consistency with the existing categorical framework in the LIMS context. After capturing the LIMS requirements in an ontological framework – to provide a common shared understanding of the requirements – empowered with category theory, we recruit our RLR framework for handling volatile requirement identification, integrated change management and consistency monitoring in a LIMS (Figure 4.10).

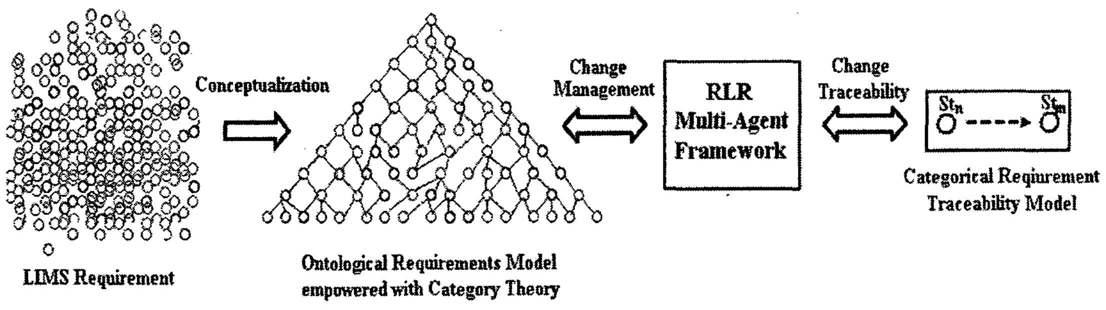

LIMS Requirement
Conceptualization
Ontological Requirements Model empowered with Category Theory
Change Management
RLR Multi-Agent Framework
Change Traceability
St_n
St_m
Categorical Requirement Traceability Model

Fig. 4.10. General view on the proposed approach for managing requirement volatility

The RLR framework then assists and guides the software developer through the change management process.

IV 2.1 MYCO-LIMS Requirements Overview

The Mycology Laboratory Information Management System (MYCO-LIMS) is our modeled experimental application for managing information about laboratory samples, users, instruments, standards, and other laboratory functions and products, and provides security, portability, and accessibility over the Web, efficiency, and data integrity in clinical, pharmaceutical, and industrial laboratory processes. MYCO-LIMS is an ontology-driven object-oriented application for a typical fungal genomics lab performing sequencing and gene expression experiments in the domain of medical mycology. In our context, the conceptual framework for requirement management outlines possible courses of action and patterns for describing a system's specifications and requirements. In complex biomedical systems development, a requirement change typically causes a ripple effect and forces the categorical requirements model to be altered as well. MYCO-LIMS is used in the FungalWeb integrated system to respond to queries regarding the clinical, pharmaceutical, industrial, and environmental processes related to pathogenic fungal enzymes and their related products. It is estimated that laboratory data account for 60-80% of the data generated during the entire clinical trial process [Kra07].

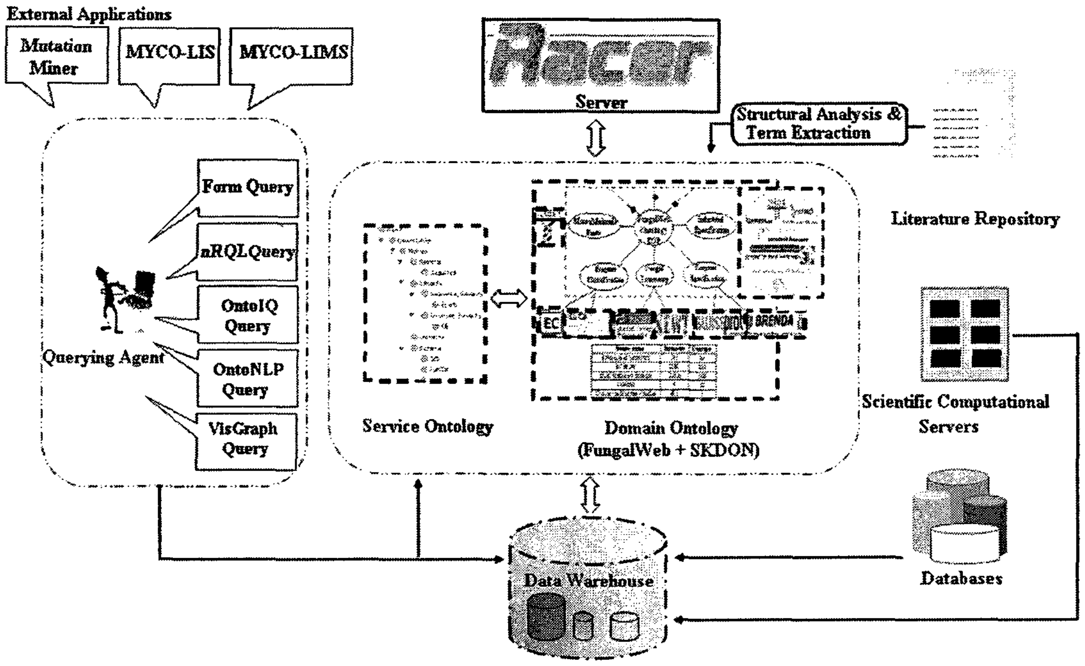


Fig. 4.11. The FungalWeb infrastructure.

The FungalWeb semantic Web infrastructure (Figure 4.11) consists of the FungalWeb Ontology, SKin Disease Ontology (SKDON), a text-mining framework and intelligent agents. In addition several external applications such as MYCO-LIMS, MYCO-LIS, and Mutation Miner [BW06] have been designed for knowledge exchange.

Microarrays are produced in different proportions, depending on the specific requirements of the gene expression study being initiated. A typical microarray may include thousands of distinct cDNA probes [JF02]. Preparation of an array begins with the clone set deliverance in the form of plates or tissue samples (with associated data) from a vendor or other source [JF02]. MYCO-LIMS will be able to maintain the taxonomy for each plate or sample in the system, such that a user can easily see the life cycle of the entity. The LIMS is based on MGED-specified [MGE] microarray data exchange standards, such as MIAME [MIM] or MAGE-ML [MAG]. Software in general and MYCO-LIMS in particular are characterized both by their functional behavior (what the system does) and by their non-functional behavior (how the system behaves with respect to some observable attributes like reliability, reusability, maintainability, etc.). Both aspects are relevant to software development and are captured correspondingly as functional requirements (FRs) and non-functional requirements (NFRs).

IV 2.2 LIMS Functional Requirements (FRs)

MYCO-LIMS is a Web-based system capable of providing services such as managing microarray gene expression data and laboratory supplies, managing patients, physicians, laboratories supplies or vendors' information, managing and tracking samples information, and managing orders.

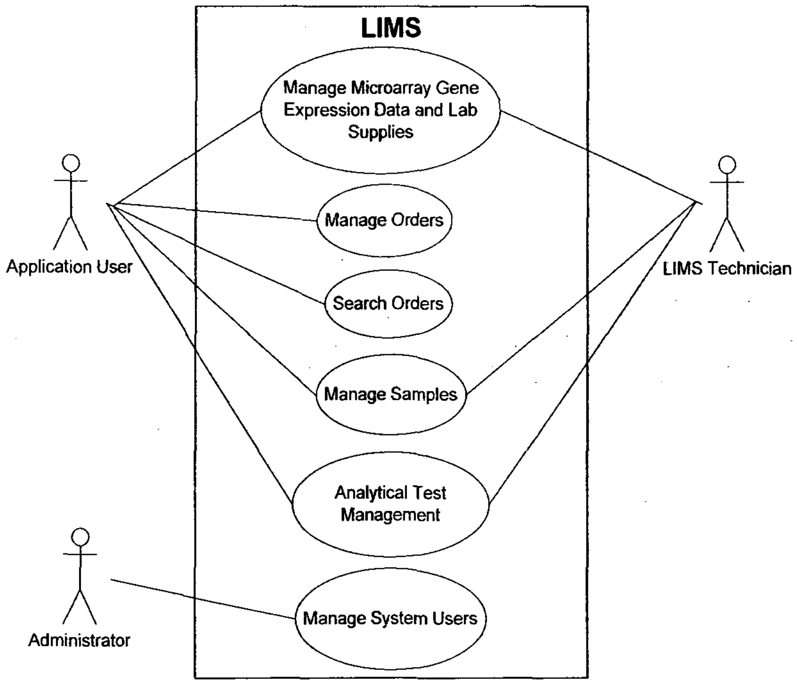


Fig. 4.12. The LIMS use case diagram.

Figure 4.12 summarizes some of the main actors and services of MYCO-LIMS application in a standard Use-Case Diagram. MYCO-LIMS is capable of receiving multiple orders or cancelation requests at the same time. It requires its users to have a certain level of privileges to access any of the functionalities, except when searching for a product. The privileges are granted automatically upon successful authentication.

Here, we choose one functional requirement, "Manage Order", and decompose it into two more specific requirements, "view orders" and "place order", which each of them decompose to more detail requirements. Figure 4.13 presents the functional model, and shows that an FR is realized through the various phases of development by many functional models (e.g. in the object-oriented field, a use-case model is used in the requirements engineering phase, a design model is used in the software design phase, etc.).

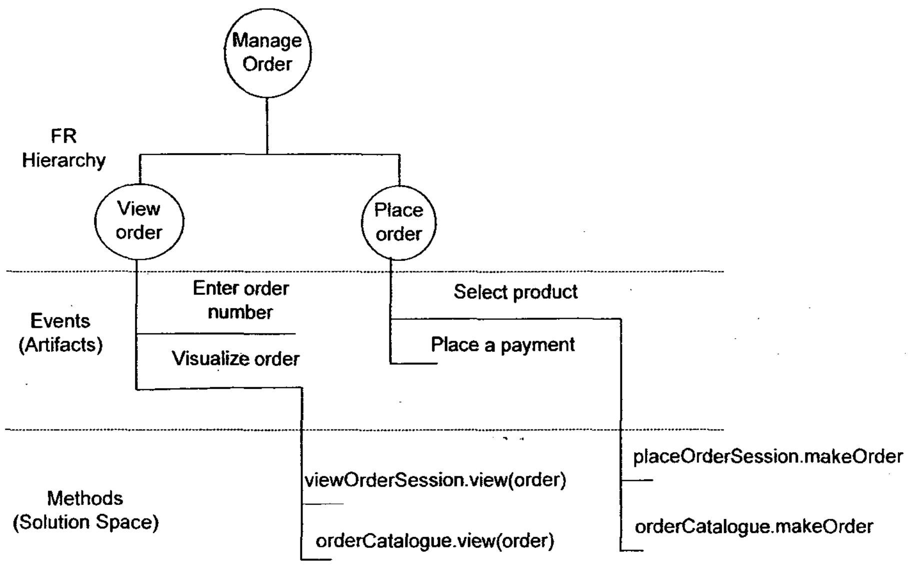


Fig. 4.13. Illustration of MYCO-LIMS FR traceability model.

Each model is an aggregation of one or more artifacts (e.g. use case and sequences of events representing scenarios for the use-case model, classes and methods for the design model). For instance, the View Order use case is refined to a sequence of events <enter order number, visualize order> illustrating an instance of View Order service; each event is refined as a method (viewOrderSession.view and viewCatalogue.view correspondingly) in the design phase. Modeling FRs and their refinements in a hierarchical way gives us the option of decoupling the task of tracing FRs change from a specific development practice or paradigm. Figure 4.13 visualizes the FR hierarchical model for the chosen case study through the hierarchy graph that forms a primary taxonomy for analyzing ontological relationships between requirements.

IV 2.3 LIMS Nonfunctional Requirements (NFRs)

The use-case diagram shown in Figure 4.12 specifies the FRs of MYCO-LIMS services. Dealing with NFRs, such as performance, scalability, accuracy, robustness, accessibility, resilience, and usability, is one of the most important issues in the software engineering field today. NFRs impose restrictions by specifying external constraints on the software design and implementation process [KS98], and therefore need to be considered as an integral part of the process of conceptual modeling of the requirements. Here we propose a formal approach to NFR modeling, and traceability.

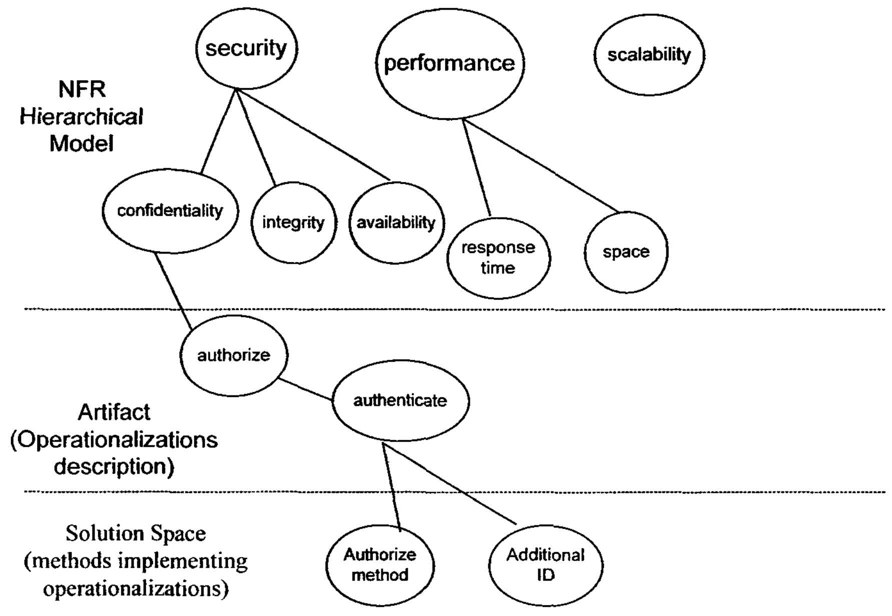


Fig. 4.14. Illustration of MYCO-LIMS NFR traceability model

In this approach as represented in Figure 4.14 a LIMS' NFR is decomposed into more specific NFRs. Let us consider the requirements of "managing orders with good security" and "maintain the users' transactions with good performance". The security as an NFR may refer to a quite general domain and may need to be broken down into smaller specific parts such as integrity, confidentiality, and availability. In the security example, each sub-NFR has to be satisfied for the security NFR to be satisfied. The sub-NFRs are refined (operationalized) into solutions that will satisfy the NFR (e.g. for confidentiality, can be achieved either through implementing authorization or the use of additional ID).

IV 2.4 Integrating FRs and NFRs into an Ontological Framework

Each software requirement usually intracts with other requirements and in this interaction they affect each others in various ways. Understanding FR/NFR relations is necessary for consistent change management of the requirements. When an application is in action, it is somehow clear to check whether a particular FR has been met or not, as it can be explicitly specified in its definition. But, it is not that simple for NFRs since they can be defined based on different quantitative and descriptive statements, which are not always easy to process. The NFRs often have been modeled with correspondence to FRs in the design process.

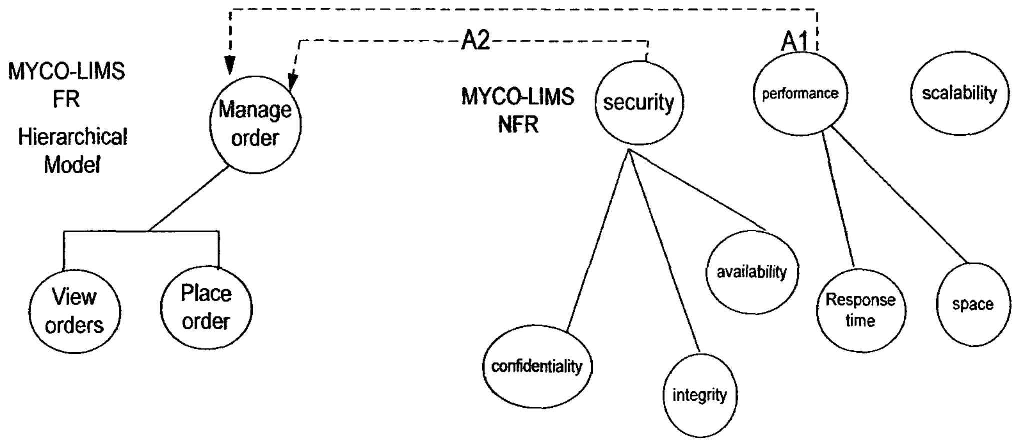


Fig. 4.15. Illustration of MYCO-LIMS NFRs/FRs dependencies hierarchical model.

Despite the importance of the traceability, it has been widely neglected in operational NFRs change models. This area needs a special attention, because NFRs are subjective in nature and have a broad impact on the system as a whole. Here, we illustrate our approach towards finding an effective method for conceptualizing NFRs based on their hierarchy and interrelations with FRs in the MYCO-LIMS invoicing system case study. For example, associating response time NFR to the View Order use case would indicate that the software must execute the functionality within an acceptable duration (see association A_1, Figure 4.15). Another example is associating security NFR to the "Manage order" FR, which would indicate that the interaction between user and the software system in the "Manage order" service must be secured (see association A_2, Figure 4.15), which also precisely implies that the user interface for other interactions is not required to be secured.

If an association exists between a parent NFR and a functionality (e.g. association A_2 between *security* and *manage_order*, or A_1 between *performance* and *manage_order*) (Figure 4.15), there will be an association between operationalizations derived from NFRs and methods derived from the functionality (e.g. *authorize* derived from *security*, and *placeOrderSession.makeOrder* derived from *manage_order*) (Figure 4.16). Figure 4.16 illustrates the refinement of the interactions. The complete change management model would require the refinement of performance and scalability into operationalizations and methods, and the identification of the associated interaction points to which they are mapped.

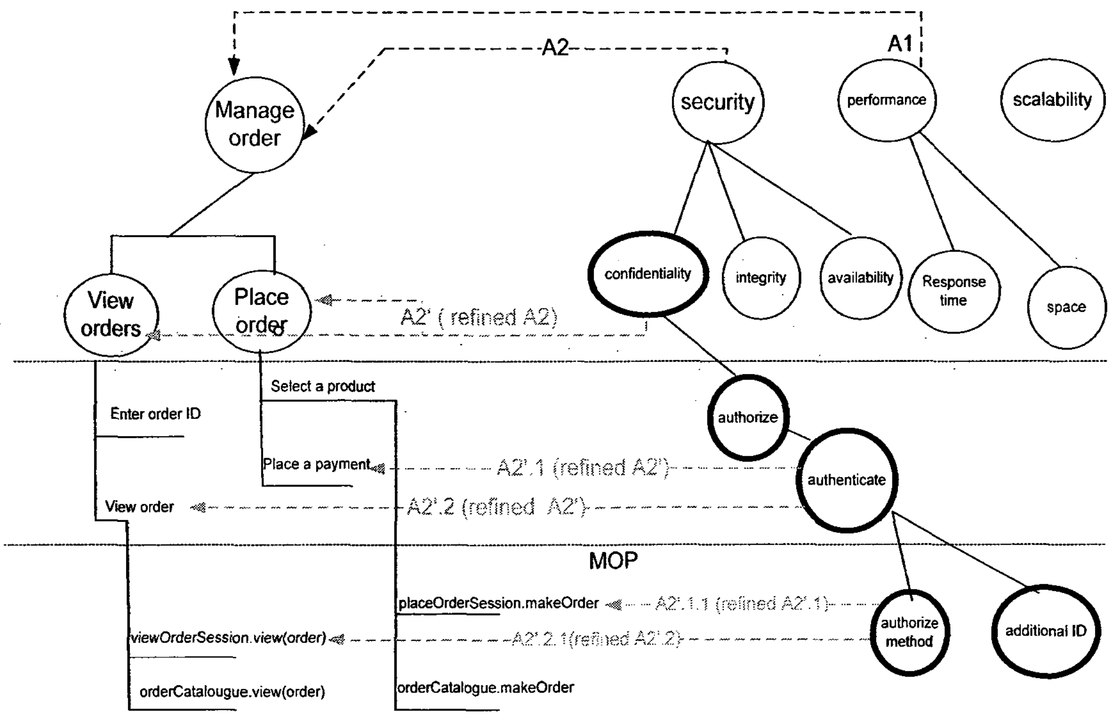


Fig. 4.16. MYCO-LIMS Requirements associations' refinement

A change in FRs or NFRs can be authorized if and only if that change is consistent with the existing requirements model. This process can be improved using the RLR framework by defining a set of consistency rules based on a formal presentation of the FR and NFR hierarchies and their relations, and these rules will be controlled automatically before a change is authorized. The conceptualization of FR and NFR hierarchies and their interconnections form the bases for analyzing ontological relationships between requirements in the Service Ontology (Figure 4.11). The NFR/FR ontological framework introduced in this section can be visualized through a categorical hierarchical graph, which makes it possible to keep track of the required behavior of the system using dynamic views of software behaviors from requirements elicitation to implementation. The following section introduces a generic categorical model of requirements with an emphasis on NFRs and their interdependencies and refinements through using category theory as a mathematical formalism, independent of any programming paradigm.

IV 2.5 Generic Categorical Representation of Requirements and their Traceability

As mentioned in our study (Chapter III), categorical analysis offers a great potential for managing structural changes in evolving hierarchical structures. In order to explicitly reason about the impact of NFRs and their refinements on the project throughout the software development process, we explicitly represent NFRs, FRs, and their dependencies and refinements using category theory. Figure 4.17 captures the generic view on the requirements modeling process where Requirements Group, Hierarchical Model, Artifacts, and Solution Space are categories representing the project requirements, the analysis models, the refined representations of the project requirements, and the requirements implementation respectively. The arrows are morphisms, which capture the refinement processes; namely, decomposition, operationalization, and implementation defined as shown in Figure 4.17.

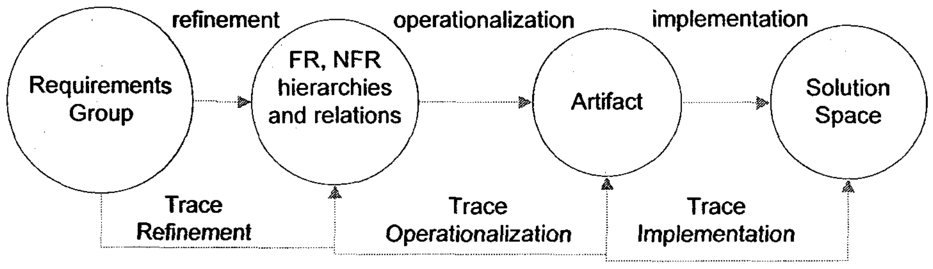


Fig. 4.17. Generic categorical framework for requirement traceability.

Figure 4.17 shows that a requirement is realized through consecutive refinements by hierarchical models, where each model is an aggregation of one or more artifacts. The implementation arrow refines the artifacts into solutions in the target system that will satisfy the requirements. These solutions provide operations, processes, data representations, structuring, and constraints in the target system to meet the requirements represented in the Requirements Group. High-level FRs are refined in the requirements analysis phase into more specific sub-FRs (use cases and their relations, e.g. FR Hierarchy Mode), which are then operationalized as use-case scenarios describing instances of interactions between the actors and the software, and modeled as events (Artifacts), which are implemented as methods (Solution Space). More general NFRs are refined into an NFR hierarchy where the offspring NFRs can contribute fully or partially towards accomplishing a goal for the parent. The sub-NFRs are operationalized into solutions (Artifacts) in the target systems, which will sufficiently satisfy the NFR.

The requirement refinements are then expressed formally in terms of the composition operator °, assigning to each pair of arrows f and g, with cod f = dom g, a composite arrow $g \circ f$: dom $f \rightarrow$ cod g (cod f is a notation for a codomain, and dom f is the notation used to indicate the domain of a function f). In this case, each requirement object belonging to the Requirements Group category will be refined to its implementation belonging to the Solution Space. The resulting solution forces preservation of the requirements and their relations, which are modeled with the *trace* arrows. The consistency between the solution and the original requirements can be guaranteed by the composition of categorical arrows representing morphisms. As a result, each change to a requirement or its refinement belonging to the domain of f will be traced to its refinement belonging to the codomain of g by means of the composition of the corresponding trace arrows.

IV 2.6 Categorical representation of FRs, NFRs hierarchies and their interdependencies

The category *FR, NFR hierarchies,* and *relations* (Figure 4.18) consists of objects representing FRs and NFRs, their decomposition into sub-FR and sub-NFR (which are also FR and NFR correspondingly), and their impact associations; above concepts are treated jointly and in an integrated fashion. Four areas have been defined for impact detection in which NFRs require change management support: (i) impact of changes to FRs on NFRs (inter-model integration); (ii) impact of changes to NFRs on FRs (inter-model integration); (iii) impact of changes to NFRs on sub-NFRs and parent NFRs (intra-model integration); and (iv) impact of changes to NFRs on other interacting NFRs (intra-model integration).

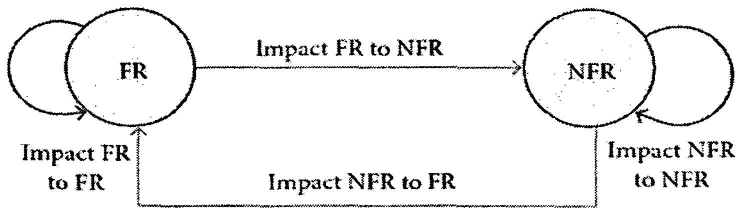


Fig. 4.18. *FR, NFR hierarchies, and relations* in a categorical framework

IV 2.7 Categorical representation of the Solution Space

The Solution Space category contains State Space *SS* (all potential states including initial states), State Transition *ST* (next state function), Class *C* categorical objects, and Methods arrows. The *trace implementation* morphism traces the effect of the changes to Artifact objects on the Solution Space objects. In Figure 4.19, for instance, we illustrate the refinement of an event from the Artifact category to a state transition object *ST*.

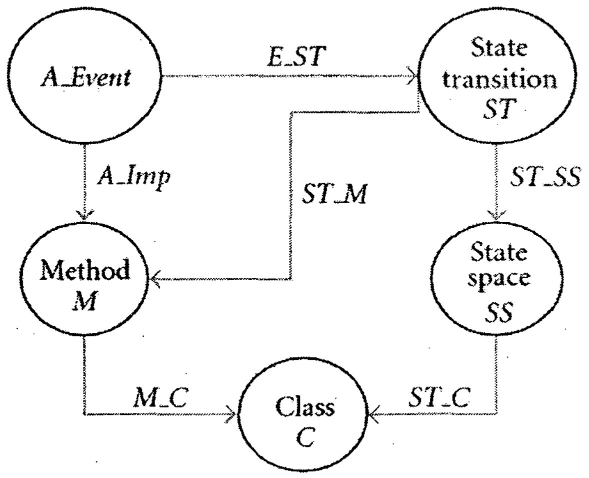


Fig. 4.19. Tracing the changes to the state spaces, classes, and methods

Moreover, each state transition *ST* is defined on the state space *SS* (arrow *ST_SS*) linked by a function *ST_C*: *ST* → C to a class *C*. The state transitions are implemented by methods captured with the function *ST_M*: *ST* → *AP_M*, and belonging to a class *C* (see function *M_C*). The above functions support the tracing mechanism and are captured formally in Figure 4.19. The changes are then represented formally in terms of the composition operator °; for instance, *E_ST* ° *ST_SS* ° *ST_C* will trace a change in dom *E_ST* (which is *A_Event*) to the codomain of *ST_C* (which is Class *C*).

As we mentioned in Section III.2 category theory can be used for the taxonomical representation of requirements to help in the study of the ontological relationship between the various nodes within the hierarchy. Category theory has been used in RLR to integrate time factor, and represent and track changes in ontological structure in time through using the notion of state capturing an instance of a system's FRs, NFRs and associations at certain period of time. For example, a change in the Authorize Method would affect the method "placeOrderSession.makeOrder" in state St_1 of the system, which will be traced to changes in state St_2 (Figure 4.20).

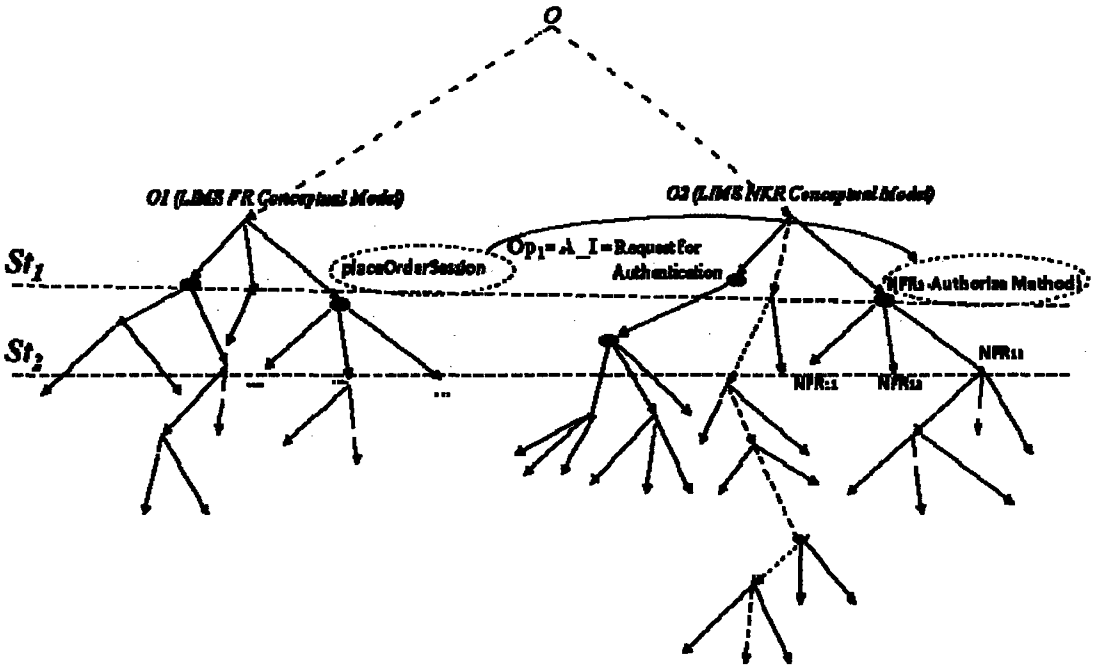

O
O1 (LIMS FR Conceptual Model)
O2 (LIMS NFR Conceptual Model)
St_1
St_2
placeOrderSession
Op_1= A_I = Request for Authentication
NFRs Authorize Method
NFR11
NFR21
NFR12

Fig. 4.20. The representation of evolving MYCO-LIMS functional requirements (FR) and nonfunctional requirements (NFRs).

Generally speaking, changes to each NFR would lead to changes in the conceptual framework. As mentioned in Section IV 2.5, we are monitoring the effect of FR or NFR changes through their refinement relations, that is: (1) identifying the "slice" of the conceptual framework that will be affected by the change; (2) applying the consistency rules to make sure the change does not introduce any inconsistencies in the "slice"; and (3) implement the change, if authorized. Explicitly capturing of the temporal evolution of the requirements can aid MYCO-LIMS developers and maintainers to deal with requirements change management in highly dynamic clinical applications.

The RLR change management framework is modeled as an intelligent control loop, which has one state for each of the above stages (1), (2), and (3), the events modeling the change of state. Considering the requirements to be organized in a lattice-like ontological framework, in order to represent the various states of our conceptualization, we use a categorical discrete state model (explained in Section III 3.5), which describes the states and events in the ontological structure using a diagrammatical notation (Figure 4.21).

$$\begin{tikzcd}
St_1 \arrow[r, "Op_1"] \arrow[rrr, dashed, bend left, "Op_n"] & St_2 \arrow[l, dotted, bend left, "Op_r"] \arrow[r, "Op_2"] & St_3 \arrow[r, dashed, "Op_m"] \arrow[ll, dotted, bend left, "Op_r"] & St_n \arrow[lll, dotted, bend left=50, "Op_r"]
\end{tikzcd}$$

Fig. 4.21. Tracking different operations and their compositions along with their states in an evolving structure, which can be used to generate patterns for the learning agents.

After studying the changes in FRs and NFRs in one conceptual model we can extend our analysis to monitor the changes in requirements of several interrelated applications following our hierarchical distributed graph transformation (explained in section III 4.5.1, and III 4.5.2).

IV.3 Case Study 3: Analyzing the Evolutionary Relationships between Species

In this section, which is partially based on our published papers [SH08b, SH08c], we propose the use of our introduced methodology to provide an underlying formalism for capturing and analyzing the evolutionary behavior of the fungi phylogeny. In an experiment we have employed ontologies rather than cladistics, to reconstruct phylogeny trees and to analyze the evolutionary relationships between species. Also the lexical chaining technique has been used for the incremental population of evolving ontological elements. We also present some of our ideas about using adjoint functors to analyze structural transformation in phylogenetic trees, which can be pursued as a possible direction in our future work.

IV 3.1 Introduction on Taxonomies and Phylogenies

The major efforts to reorganize taxonomies of species over time can be summarized as the dynamic identification of essential classifying properties for a class and the collection of all beings that share values for these properties into that class [PST04]. It is commonly believed that all species are descended from a common ancestral gene pool through gradual divergence [Fut05] and form different kingdoms in the tree of life.

In this process of constant evolution, Fungi were promoted from one subclass in the Plant kingdom to a kingdom of their own based on gene mutation. A gene mutation, whether hereditary or new is a permanent change in the DNA sequence that makes up a gene [MAH08]. These changes, which can be insertions, deletions or rearrangements of genetic information happen in relation to time and alter the evolutionary taxonomies of different species. Thus, through several changes (based on mutations), the fungal classes are promoted, moved, folded, deleted, merged, and renamed as more is discovered about life on Earth. One of the primary goals of taxonomists is to reflect evolutionary history (phylogeny) in the biological classification [Tax99]. Phylogenetic trees demonstrate how a group of species are related to one another. To analyze the evolutionary relationships between groups of organisms for the purpose of constructing family trees, biologists currently use a method called cladistics or "phylogenetic systematics". Through this method, organisms are classified based on their evolutionary relationships; to discover these relationships, primitive and derived attributes should be analyzed [Clo96]. An extensive collection of evidences for the importance of systematics and taxonomy (with emphasis on fungal taxonomy) in biological research recently became available, provided by researchers from the British Mycological Society[143]. In summary, cladistics is based on the following assumptions [Phy]:

1. Any group of organisms is related by their descent from a common ancestor. Thus, there is a meaningful pattern of relationships between all collections of organisms.
2. The taxonomic trees should be binary, which means that new organisms may come into existence when currently existing species divide into two groups.
3. Changes in attributes occur in lineages over time.

[143] http://www.parliament.uk/parliamentary_committees/lords_s_t_select/evidenceselect.cfm

The third statement is the most important rule in cladistics. In fact, only when attributes and characteristics change one can recognize various lineages or groups [Phy]. Cladistic analysis has proved useful for analyzing evolutionary trees, but it does face several issues, mostly addressed in [Clo96], and [Rob86].

In order to overcome some of the issues that affect the cladistic inferencing, we have employed the FungalWeb Ontology, as a conceptual backbone to provide a common formal specification for each species in the fungal evolutionary tree. "Lexicon chaining" as a natural language processing (NLP) technique has been proposed for dynamically populating the ontology. To analyze the temporal fungal phylogeny, we also use category theory. In the following, after discussing the cladistic technique for studying evolutionary trees and the related issues, the relations between ontology, taxonomy and phylogenies will be utilized. Then we explain our categorical method along with an ontology-driven technique, to facilitate semi-automatic phylogeny construction and analyzing evolutionary relations between species.

IV 3.2 Phylogenetic Systematics (Cladistics)

As mentioned in Chapter II the taxonomical classification has a long history in biology; since the time of Darwin (1809–82) and his theory of natural selection [WDB] there have been debates between two groups of taxonomists [Tax99]:

1. Classical taxonomists working on "Linnaean classification" [Bru97], a system based on a hierarchy of formal ranks (family, genus, etc.) and binomial nomenclature.

2. Cladists working on phylogenetic classification or cladonomy [Bru97], which is a clade-based classification system, without any formal ranks, including the genus, and no binomial nomenclature [Bru97], [DG92].

Cladistic approaches are being used to analyze the evolutionary trees based on primitive and derived attributes. Primitive attributes (plesiomorphic) are those attributes of a fungus that are shared by all members of the group. Having "fruiting body" is a primitive attribute for all species of Basidiomycota (a major phyla in the fungi kingdom), which has been inherited from their common ancestor. Primitives are not very helpful for analyzing the relationship between organisms in a specific group [Clo96].

When we try to construct a family tree for all Basidiomycotas, it is not helpful to note that they all have fruiting bodies, and it does not help us in determining the relationships between different species. Derived attributes (apomorphic) are advanced features that only appear in a number of members [Clo96]. In fact, the derived attributes are crucial to construct evolutionary relationships. For example, the shared derived attribute that defines the Ascomycota is the ascus [WK92]. Nuclear fusion and meiosis occur inside the ascus where one round of mitosis follows meiosis to leave 8 nuclei, and 8 ascospores [WK92], [TSB06]. Accordingly, Fungi can be divided into two biological groups: without ascus and with ascus. The intersection of these two groups (a node) can be represented in an evolutionary diagram (cladogram) as a point at which a new species (with ascus fungi) evolved [Clo96]. Having ascus is a synapomorphy (a derived attribute shared by two or more taxa) of the Ascomycetes group. In cladistic method synapomorphies are used to construct phylogenies. A synapomorphy of one group might be primitive for another group. By analyzing sufficient attributes cladistics aims to generate a family tree where either all members are descended from a single, common ancestor (monophyletic) or from several common ancestors (polyphyletic) [Clo96]. If the group includes some, but not all, of the descendants of a single common ancestor, it is called paraphyletic [Nat]. Cladistic analysis is currently performed using various software applications such as PHYLIP (Phylogeny Inference Package) [Fel05], PAUP [Swo] and MacClade [MM].

A data matrix similar to the one demonstrated by Figure 4.22. provides the input for cladistic analysis. This matrix simply summarizes the answers to questions such as: does a fungus have a set of attributes, or not? The answers are short and simple ([yes, no] or [1, 0]). The more species and the more attributes one puts in an analysis, the more likely it gets close to the accurate family tree [Clo96].

1. **Cell walls composed of glucan and chitin: Yes (1), No (0)**
2. **Has non-septate vegetative hyphae: Yes (1), No (0)**
3. **Has ascus: Yes (1), No (0)**
4. **Has fruitting body: Yes (1), No (0)**

Attribute No.	1	2	3	4
Glomeromycota	0	0	0	0
Chytridiomycota	1	0	0	0
Zygomycota	0	1	0	0
Basidiomycota	0	0	0	1
Ascomycota	0	0	1	0

Fig. 4.22. An example of a sample data matrix for analyzing major fungi clades (Ascomycota, Basidiomycota, Zygomycota, Chytridiomycota and Glomeromycota.

IV 3.3 Issues in Cladistic Analysis

There are some known issues in cladistic analysis [Clo96]:

I. **Convergent evolution:** If one defines having a fruiting body as an attribute of fungi basidiomycota, and considering that many plants have also fruiting bodies, should basidiomycota be considered closer relatives of plants than of the ascomycota fungi? The answer is negative. In fact, basidiomycota and ascomycota have a number of shared derived attributes that closely link them. Convergent evolution produces homoplasies. A homoplasy [Sim61, Wak91] can be defined as: "a resemblance between taxa that can be ascribed to processes other than descent from a common ancestor and which implies phylogenetic relationships that conflict with the best estimate of phylogeny for the taxa" [CW01]. By providing and analyzing as many different attributes as possible this problem can be reduced [Clo96].

II. **Reversals can cause problems:** As an example, whales unlike all the mammals do not have fur, because the fur of their mammalian ancestors has been lost in an aquatic environment [Clo96, Mam].

III. **Considering fossils with missing parts:** In this case, the attributes associated with those missing parts are represented by question marks and ignored when generating the cladogram.

IV 3.4 Formal Ontology, Taxonomy and Phylogenetic Analysis

Taxonomy in knowledge representation is considered as a collection of terms or entities organized in a hierarchical structure (implying parent-child relationships). Ontologies in the context of semantic web consist of "taxonomies and a set of inference rule" [BHL01]. There may be more than one taxonomy for an ontology in a domain of interest, based on the granularity and the chosen subsets of ontological characteristics.

Ontologies in the real world evolve over time as we fix errors, reclassify the taxonomy, and add or remove concepts, attributes, relations, and instances. Consistently modifying and adjusting the hierarchical structure of ontologies in response to changing data or requirements can provide new insight for studying evolutionary changes (or mutations in evolutionary phylogenies) in biological taxonomies occuring over time. Ontologies follow the open world assumption, which asserts that the captured knowledge is always incomplete, therefore if something cannot be inferred from what is defined in the knowledgebase, it is not necessarily false. The open world assumption is especially important when we represent knowledge with a dynamic system, which is gradually improved as we discover new facts. In cases such as the real world phylogeny analysis our knowledge is always incomplete and the facts described by the system can never be fully known. Due to the evolutionary nature of cladistics, it is possible to study the way in which attributes change (the direction in which attributes change, and the relative frequency of the change) over time within groups [Zan02] in an ontological framework. In order to study various changes in an ontologically inferred phylogenetic tree one can focus on ontology evolution and change management techniques.

Our ontology change management framework as introduced in Chapter III aims to maintain the dynamic structure of ontologies and controlled vocabularies, to preserve the validity and consistency of ontological knowledge. Analyzing the evolving fungal taxonomy within the FungalWeb framework, as discussed in Case study 1, facilitates ontological inferencing - which provides a valuable source of information for clarifying the explanations of complex evolutionary scenarios for fungi species - rather than cladistics inferencing. The ontology inferencing allows us looking at the diversity of the species within different groups by comparing the descendants of an ancestor to find out the patterns of origin and extinction. It also empowers biologists to examine different hypotheses about adaptation [WDB], [Zan02]. Currently, there is a need for a comprehensive methodology to describe how chronological alterations in ecological and environmental conditions [And95] have formed the adaptive evolution of fungal clades.

IV 3.5 Ontology Learning for Managing Evolving Taxonomies

By changing the knowledge, ontologies need to be incrementally updated to provide valid information for the human/agent learner. In our approach, we have used the Lexical chaining method to (semi-) automatically construct and populate the FungalWeb ontology by extracting relevant terms and relations from a structured or unstructured text corpus or other types of data. The Lexical chaining algorithm [HS98] reads a text corpus and places words in a related chain based on semantic similarity, using a set of reference dictionaries such as WordNet[144] 3.0, Integrated Taxonomic Information System (ITIS)[145] and TreeBase[146] (a database of phylogenetic knowledge). As an example, based on one of our experiments focused on patient information leaflets to populate the medical subset of the FungalWeb Ontology, consider the following patient information:

[144] http://wordnet.princeton.edu/
[145] http://www.itis.gov/
[146] http://www.treebase.org/treebase/

Patient A, a white male, nine years old, has recently found multiple, widespread scaly red patches on his abdomen, chest, face, and arm. The physician diagnosed his disease as "Rosacea" and prescribed antibiotics.

Using the lexical chaining algorithm described at [BE99], one can distinguish several possible chains such as:

{Patient A, nine years old, male, white, his, abdomen, chest, face, arm};

{Multiple, widespread, scaly, red, patches};

{Physician, diagnose, disease, Rosacea, prescribed, antibiotics}.

The chain of words together indicates a topic related to particular concepts in the related ontology. Different algorithms may generate different chains. For the evaluation some criteria such as reiteration, density and length of the chain [MH91] can be considered. Then using the RLR agent-based framework, the related ontologies – which provide the underlying knowledge for the learner agent – can be dynamically populated and validated using a description logics reasoner (e.g. RACER) (Figure 4.23).

If some species have similar properties and genomes, it is very likely that they evolved from a common ancestor. The similarity of genomes is computationally measured based on the number and likelihood of different mutations (insertion, deletion, duplication or substitution of base pairs) [Mat02]. We have used the FungalWeb Ontology to determine the taxonomic provenance [BSS+06] for fungal species, in order to study the evolutionary relationships based on logical and ontological inferencing.

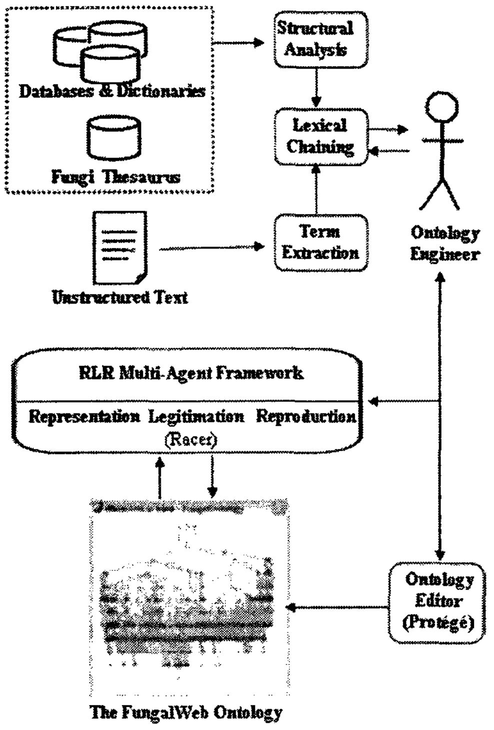


Fig. 4.23. Framework for ontology learning and population

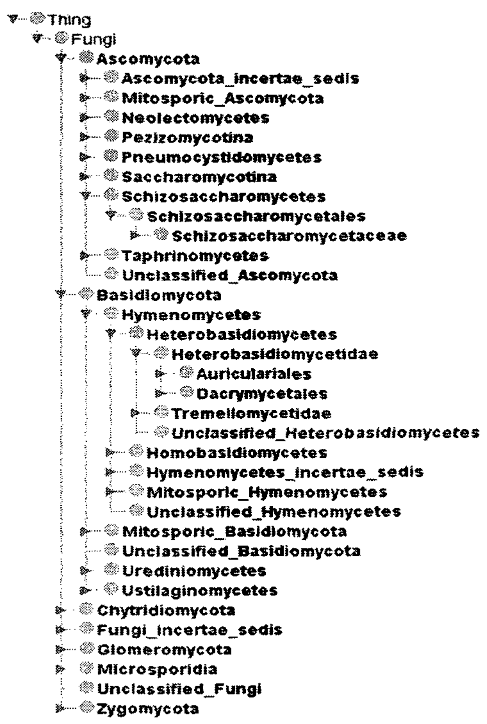

```
Thing
  Fungi
    Ascomycota
      Ascomycota_incertae_sedis
      Mitosporic_Ascomycota
      Neolectomycetes
      Pezizomycotina
      Pneumocystidomycetes
      Saccharomycotina
      Schizosaccharomycetes
        Schizosaccharomycetales
          Schizosaccharomycetaceae
      Taphrinomycetes
      Unclassified_Ascomycota
    Basidiomycota
      Hymenomycetes
        Heterobasidiomycetes
          Heterobasidiomycetidae
            Auriculariales
            Dacrymycetales
          Tremellomycetidae
          Unclassified_Heterobasidiomycetes
        Homobasidiomycetes
        Hymenomycetes_incertae_sedis
        Mitosporic_Hymenomycetes
        Unclassified_Hymenomycetes
      Mitosporic_Basidiomycota
      Unclassified_Basidiomycota
      Urediniomycetes
      Ustilaginomycetes
    Chytridiomycota
    Fungi_incertae_sedis
    Glomeromycota
    Microsporidia
    Unclassified_Fungi
    Zygomycota
```

Fig. 4.24. Domain model of fungal taxonomy

By querying the FungalWeb Ontology the enzymologist can find the related fungal species: *Pichia stipitis* and *Saccharomyces cerevisiae*. Identifying the common lineage between the found organisms requires identifying the highest taxonomic group that unites all species known to produce the enzyme of interest, akin to finding a common ancestor [BSS+06]. Within the FungalWeb Ontology, a fungal taxonomy is represented in a deep hierarchy of taxonomic units/concepts. The defined key properties between "fungi" and "enzyme" allow for the identification of species found to produce 2-deoxyglucose-6-phosphatase. One can identify the common lineage for these fungal species by using the description logic reasoner, the RACER, via the command *instance types,* which retrieves the concepts that instantiate each fungal species individual. A simple example of such queries is shown in Query 1. The common lineage of "2-deoxyglucose-6-phosphatase"-

producing fungi, is a family of yeast in the order *Saccharomycetales* called *Saccharomycetaceae*, known for its reproduction by budding and use to ferment carbohydrates (WordNet definition).

Query 1: This query uses RACER command "Instance types" to retrieve results for all fungi that produce the enzyme 2-deoxyglucose-6-phosphatase (EC# 3.1.3.68) as well as their ancestors. The common subset identifies the common lineage between the species:

```
<<:?X :http://a.com/ontology#Fungi:>
<<:?X :http://a.com/ontology#Ascomycota:>
<<:?X :http://a.com/ontology#Saccharomycotina:>
<<:?X :http://a.com/ontology#Saccharomycetes:>
<<:?X :http://a.com/ontology#Saccharomycetales:>
<<:?X :http://a.com/ontology#Saccharomycetaceae:>
```

Analyzing and managing both syntactic and semantic changes in the fungal taxonomy can be used to derive a a meaningful pattern of relationships between the species, which assists automating the phylogeny tree reconstruction.

IV 3.6 Categorical Phylogenetic Analysis

After constructing the ontological structure one can also employ category theory and graph transformation to represent, analyze, and track the changes in the evolutionary trees in the same way that we used it for analyzing evolving biomedical ontologies. In an ontology-driven phylogenetic tree changes, actions (or mutations), and transitions can be formally modeled through our introduced framework as described in Section III 3.5 to capture the full semantics of evolving hierarchies.

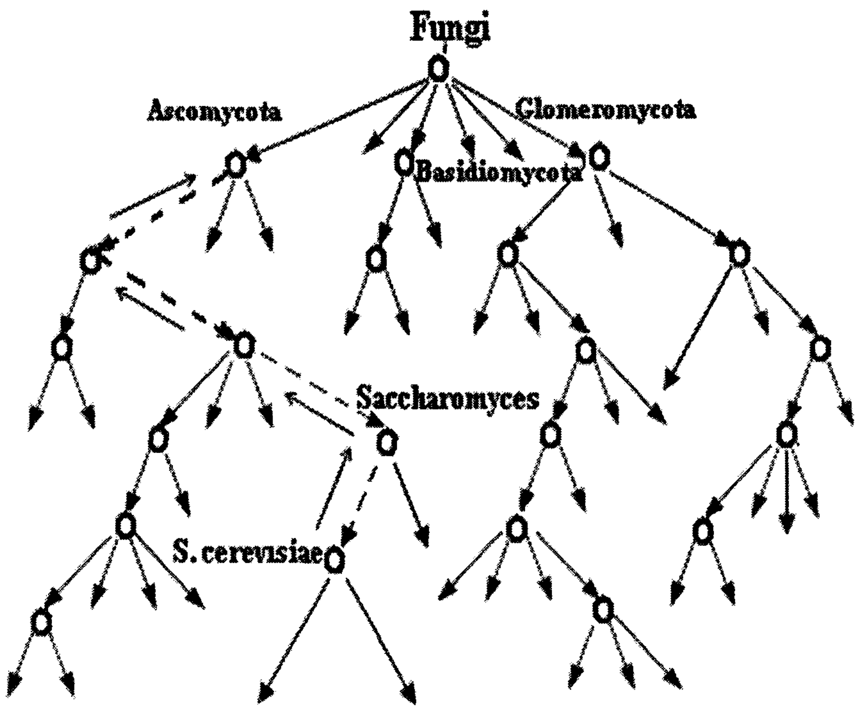


Fig. 4.25. The categorical representation of ontology inferred phylogeny for yeast *Saccharomyces cerevisiae* which depicts the transition between various evolutionary states.

Category theory is also capable of solving problems related to reverse analysis (mentioned in cladistics method) through *recursive domain equations* [SP82]. Categorical constructors also may be used for analyzing the bifurcating pattern of cladogenesis [Phy], through pushouts and pullbacks. Placing an organism in a phylogeny tree and associating a set of roles based on its evolutionary characteristics may sometimes lead to redundancy in the taxonomy. One of the major issues in phylogeny analysis is finding and identifying equivalent classes and relationships. Category theory enables us to deal with the problem of logical equality [Maz07] by using isomorphism, which has been introduced in Section III 3.5.4.1 and Section III 3.5.5.1.[147]

[147] Bijections in the category of sets are examples of isomorphism

IV 3.7 Structural Transformations and Functors

One of the interesting subjects in phylogenetic analysis is comparing two morphological structures and finding their similarities and differences, in order to study their transformations and find a common origin or to place them into their appropriate ranks (e.g. finding a common lineage between humans and birds, which leads to Amniota[148]).

The transformations of evolving structures can be studied in terms of functors, or more accurately adjoint, simple adjoint, and weak adjoint functors [BS73], where the adjointness relation between two structures embodies a link and similarity between them. As an example from life science taken from [BS73], the scientific findings explain the similarities between the nuclei of cells of some of the derived species in different stages of their life cycles. This similarity can be represented in an abstract way using "an isomorphism between the sets of temporal events in the two similar nuclei, together with an isomorphism between the sets of possible transformations (differentiations) of the equivalent totipotent nuclei" [BS73]. These isomorphisms are examples of an adjointness between equivalently similar nuclei of different cells, which can be considered dynamic living structures. If the isomorphisms have been restricted only to specific subsets of temporal events, or subsets of possible transformations (differentiations), we can talk about simple adjointness; otherwise, if in an adjointness we substitute epimorphisms[149] for isomorphisms, the weak adjointness will be obtained [BS73].

[148] For more information see : http://tolweb.org/amniota

[149] Epimorphism is any morphism in a concrete category whose underlying function is surjective [Rei70].

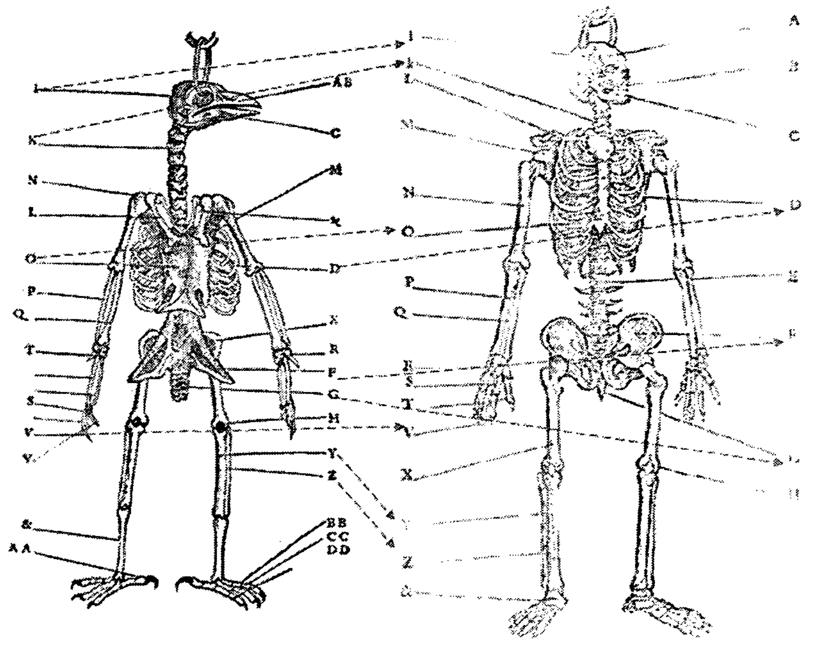


Fig. 4.26. The comparison between the skeleton of Bird (lest) and Human (right) based on the Belon's book[150] of birds (1555).

An evolving hierarchical structure [BS73] can be analyzed within a commutative categorical diagram consisting of a set of objects within this structure; the state space, which varies (unlike traditional definitions of evolving systems) according to the transformation rules, along with the collection of of all temporal events that produce the changes from one given stage to the next. Categorically the changes in this evolving structure can be studied [BS73] as a series of functors from the state space to a category of numbers indicating the states. Following this model, starting from an initial state, we can determine the number of possible states, necessary for performing a specific change to an evolving ontological structure, by transformation rules, and for analytical simplicity, we consider it fixed for a given system. The abstract categorical framework at each state, along with the transformation rules, which provide the appropriate links from

[150] L'histoire de la nature des oyseaux, avec leurs descriptions, & naifs portraicts. (The history and nature of birds) par Pierre Belon du Mans published in 1555.

one state to another one, diagrammatically demonstrate the dependency of different elements at given states of an evolving structure. We consider further research on this part as our future work.

IV 3.8 Challenges and Limitations in Phylogenetic Analysis

Some of the challenges that we faced in applying our approach are as following: In the task of employing lexical chaining algorithm we had the problem of non-cohesive [BE99] text corpuses which dramatically reduce the efficiency of our approach. Therefore we decided to start with the assumption that the target text is cohesive. Another problem is due to ontological incompleteness. Although the use of ontology inferred phylogeny is a very useful way forward, its success highly depends on taxonomic expertise and the availability of rich consistent collections of defined concepts for accurate and precise inferencing.

V. Discussion, Challenges and Future Works

This chapter concludes our research, highlights our contribution to the field, and discusses some of the limitations of the proposed approach along with suggestions for the direction of future research.

V.1 Summary of the Thesis

"First comes thought; then organization of that thought, into ideas and plans; then transformation of those plans into reality. The beginning, as you will observe, is in your imagination."

Napoleon Hill (1883-1970)

Biomedical knowledge is constantly expanding in volume, scope, and granularity to cover different aspects of the domain and all advances in the field. This growth creates new opportunities and new challenges for researchers, physicians, nurses, lab technicians, patients, health policy makers, and agencies. Ontologies, which provide the conceptual backbone for many of the existing knowledge-based systems, generally must change to update their ontological 'truth'. The heterogeneity of biomedical ontologies and the volatility of their knowledge sources increase the odds of different structural alterations. Our research aims to assist a biomedical ontology engineer in capturing, tracking, and analyzing the changes in ontologies within the distributed semantic web environment.

One issue in the domain of ontology evolution is the lack of formal change models with clear, comprehensible semantics. Due to the limitations of set theoretic based knowledge representation languages (including the popular web ontology languages RDFS and OWL) for dynamic conceptual modeling, we examined the applicability of categorical representation for ontology change management and agile application modeling. The semantic web can be conceptualized as an interconnected collection of categorically described ontologies and the progressive modification of their descriptions. Categorical logic [Law63] offers valuable insights for modeling the declarative semantics of ontologies, which are stratified structures distributed in the heterogeneous semantic web environment. The functorial semantics can be employed as the categorical generalization of operational semantics for studying various ontological states and analyzing pre/post conditions for ontological transitions, independent of any specific choice of knowledge representation language. It also provides agile access to the magnifying function (zoom-in/zoom-out) over interconnected ontologies in the distributed and heterogeneous semantic web environment.

Another issue in this area is overreliance on the human factor in different stages of decision making to perform a change. To remedy this issue, we have introduced a novel multi-agent framework to handle changes in bio-ontologies with minimum human intervention, while still benefitting from human rationality where necessary. Using category theory with its dynamic nature as a complementary knowledge representation tool facilitates the capture of the full semantics of evolving bio-ontologies and provides a formal basis to represent agent interactions.

The third issue, a crucial one, is how to ensure consistency of evolving ontologies. This issue itself can give rise to several other problems related to security, trust, provenance, and so forth. It has been partially addressed using a rule-based hierarchical distributed graph transformation approach to define consistent transitions between the states with the ability to reveal conflicts and inconsistencies.

Besides demonstrating the usability of our method in managing alterations in biomedical ontologies, we have also explored the potential of our proposed approach to solve other computational problems, such as managing requirement volatilities (with emphasis on non-functional requirements) and reconstructing evolutionary phylogenies in bioinformatics. Using our category-based framework, we defined a change management strategy to monitor and maintain non-functional requirements (NFRs) in a software development life cycle. Ontologies represented in categorical depiction can describe abstract NFRs, which are difficult to model with object-oriented languages. The NFR's hierarchy volatility can be managed using our RLR framework. In addition, we have used our method to handle formal ontological inferencing, rather than cladistics, to reconstruct phylogeny trees and analyze the evolutionary relationships between species. The major efforts for the reorganization of taxonomy over time can be summarized as the dynamic identification of essential classifying properties for a class and the collection of all beings that share values for these properties into said class. For our experiments, we focused on the FungalWeb Ontology and phylogeny of fungi, but the method can be generalized for all other species and domains.

Although the problems discussed in this thesis are sometimes of a more philosophical and linguistic nature, our focus on the "formalization" and "operationalization" aspects as two distinct features of a scientific approach [Hey90], along with the use of a mathematically sound theory (category theory) and graph transformation method, helped us to deal with the computational side. In fact our introduced approach, based on the insights from category theory, can be employed to develop algorithms and tools to assist ontology change management. In the end, we hope our attempt will be seen as a process towards providing a workflow for the implementation of a generic all-in-one algorithm and model for biomedical Ontology change management.

Throughout this thesis we have accomplished our research objectives in different extents and managed to answer many of our motivating questions.

- As mentioned in Section I 1.2, the first objective was to identify the effects of changes in bio-ontologies with emphasis on the FungalWeb ontology. We have addressed this by studying different biomedical ontologies and their editorial procedure in Section II.6. We have also classified different types of changes in the FungalWeb Ontlogy, with their origins and their effects on the ontology (Section III.1) as well as the impact on the related disease ontology in sections (IV.1).
- The second objective has been partially accomplished by studying the factors affecting the consistency of evolving ontologies (Section II 3.3, Section II.4, and Section II.6), and proposing a method to deal with this issue using RLR (Section III 2.3.3) and employing category theory (Section III 3.5.5.2), along with graph transformation method (Section III 4.5 (specifically III 4.5.4.2)).
- To analyze changes in distributed biomedical ontologies, which was the third objective, we employed hierarchical distributed graph (HD graph) transformation, and utilize our approach in several examples including the case study in (Section IV.1);
- To deal with the overreliance on human factor in current practices in ontology evolution (Objective 4), we designed RLR (Section III.2) an agent-based framework to capture, represent and analyze changes in bio-ontologies with minimum human intervention, which formalized using category theory and graph transformation.

- To achieve the fifth objective, which was examining category theory as a formalism for ontological change management, we have used categories extensively from studying changes in bio-ontologies within RLR (See Section III 3.5) to model agent interactions and protocols (Section III 3.5.6). We have also employed the categorical approaches for graph transformation to consistently manage changes in a rule-based manner in distributed environments.
- In order to address the sixth objective, our proposed approach has been used for modeling agent communications (Section III 3.5.6, Section III 4.5.4, and Section III 4.5.5) and analysis of ontology evolution by means of distributed graph transformation (Section III.4). The potential of our approach has been shown through several scenarios in Chapter IV.

The sections, which address the research questions, can be found in detail in Section (I 1.3).

V.2 Highlights of Major Contributions

The healthcare industry deals with large-scale integrated projects, including a variety of information services, resource allocation modules, planning, education, and production lines. From the ontological perspective, biomedical knowledge bases are highly heterogeneous and dynamic. In this thesis, we have presented an approach to incorporate categorical representations and graph transformations into an agent-based configuration, yielding an integrated framework to analyze and manage changes in biomedical ontologies. In particular, we have presented the following contributions[151]:

i. A semantics for evolving ontologies within a distributed semantic web environment, in terms of the semantics of transformation of nested graphs (Section III 4.5) ;

ii. A study of change management in some of the popular biomedical ontologies, the existing challenges, as well as the available tools and algorithms (Section II 6.1, and Section II 6.2);

iii. The modeling of a collaborative multi-agent framework (RLR) for managing changes in biomedical ontologies with minimum human intervention, and with the ability to generate reproducible results, through an argumentive structure, whenever necessary (Section III.2);

iv. Formalizing the agents' interactions and communications using category theory and graph transformation within the RLR framework. (Section III 3.5.6, Section III 4.5.4, and Section III 4.5.5);

v. The introduction of a categorical syntax to analyze changes in evolving biomedical ontologies and to incorporate change in terms of temporal states into our proposed agent-based framework (Section III 3.4, and Section III 3.5);

vi. A sketch of an ontological model transformation through a rule-based graph transformation approach (Section III 4.5.3, and Section III 4.5.4);

vii. An extension of hierarchically distributed graph transformation rules to coherently manage changes in distributed evolving ontologies at different levels of abstraction (III 4.5, and IV 4.1);

[151] The details of contributions can be found at the end of the related sections.

viii. An analysis of the practical usage of our framework in three different domains: knowledge representation (biomedical ontologies), software engineering (requirement management), and bioinformatics (phylogenetic analysis) (Chapter IV).

As mentioned in Chapter I, the goal of the RLR framework is to assist an ontology engineer in performing change management in a more effective manner, including reproducing the results of a change and ensuring the consistency of the affected ontology. In summary, in this research we addressed the management of changes in temporal biomedical ontologies, both as an individual standalone unit and as a unit interacting with other existing elements within the distributed semantic web environment, by studying human behavior and modeling an adaptive agent-based framework to minimize human intervention, as well as by introducing a representation formalism to support this framework using category theory and hierarchical distributed graph transformation. We have also used categorical formalisms to specify and represent changes in a declarative fashion, which can be used to define the transformation rules. Moreover, understanding the nature of human behaviour and agents' communications in a typical MAS can save time and effort in the design process. From our experience so far, some of the concrete advantages of our introduced model are:

- The representation of events, time, actions, and operations employed in different scenarios of a dynamic ontological framework is an effective way to trace model changes;

- The independency of the framework from any particular domain, algorithm, protocol, or implementation language and its abstractness makes it more flexible for reuse in many application domains that use different formalisms and platforms;
- Employing transformation rules to perform changes ensures the consistency of the evolving ontologies in different states;
- Following the double-pushout approach for defining model transformation, which isolates the parts that remain unchanged, enables concurrent changes within an integrated knowledge-based system with minimum interruption to the system's operation.
- The abstract categorical notions and their ability to specify objects and their relations in different levels of granularities, together with graph oriented semantics, enable us to describe the complex evolving structure in a consistent manner, which is beyond the capability offered by OWL's single semantic structure.

V.3 Challenges and Limitations

One of the characteristics that distinguishes our research is the focus on breadth of coverage. In order to model a comprehensive change management mechanism, we had to deal with several concepts, issues, and challenges from different domains (cf. Chapter II) in this thesis. Thus, extra efforts have been made to grasp the key concepts from different areas. However, this is the nature of multi-disciplinary research such as computational biology and health informatics.

In the process of employing category theory as the core formalism for the RLR framework, we had to deal with a variety of challenges, including the reasoning issues and management of conceptualization changes.

However, we are able to provide basic reasoning and inferencing for categories, though we still must improve the reasoning capability to cover more advanced services. The representation of conceptualization changes is another challenge, especially for abstract concepts and notions. To overcome this, we plan to work on grammatical change algorithms in linguistics and language evolution. In the same way, one can see that in general the formal representation still faces bottlenecks in several domains, including agent negotiation processes, cost/benefit estimations, and prediction of all effects of a change. Minimizing human intervention is another issue in the "Reproduction" phase, although improvement of the learning and negotiation algorithms for the agents may reduce the problem.

In order to manage complex situations in ontology change management, we still need to add more expressivity to the underlying formalism. For example, we need to define more constraints and induce several conditions to enrich the RLR semantics. Using sketches [Wel93, BW05], which are categorical constructors, is a potential solution that can be used as graphs with some commutative diagrams (conditions) to specify a set of conditions and constraints on a structure, along with specifying the objects that are limits/colimits with some conditions. In this way, one can precisely determine the expected outcome for the category of agents.

Another challenge is related to the implementation of the framework. Since the tools (and GUI) supporting automatic ontology change management are not yet fully available, we hope to continue our research towards the tool development. Althought, despite its advantages, the abstractness and minimalism of categorical formalism decreases the degree of expressivity, which necessitates more efforts for implementation of complex applications. Last but not least, there is the challenge of choosing a standard hierarchical graph model. Despite the existence of vast amount of researches and literature in this domain, still no common standard model exists [BKK05]. Different researchers have defined different concepts and models based on their application scenarios. We also tried to adapt some of the available models into our framework to reflect the hierarchical nature of ontologies and their compositions in a semantic web environment.

V.4 Potential Improvements and Future Work

Our proposed approach has still room for improvement in several areas, some of which have been considered for future work. As far as future work is concerned, further effort is necessary to incorporate this framework into an implemented operational ontology development tool and explore its implications when confronting rigorous changes in the real world.

Incorporating new knowledge in an ontology, must be in a way that it should not contradict the existing 'truth'. Therefore as a vital part of ontology maintenance one should always watch for the consistency and coherency of the evolving ontologies. During the agents' collaboration and negotiation in RLR, each action is evaluated for its potential consequences on the detected and identified inconsistencies in each context. Then, either the action should be banned or the inconsistencies must be resolved. Ideally these processes should be examined every time the state transition has occurred to ensure that the ontological consistency still holds. The consistency management in our model includes several options including:

- Enforcing the actions for prohibiting the alterations that may lead to inconsistencies that often inherit to different versions and endure over the substantial part of the ontology's life cycle. This has been done by defining a set of conditions on transformations. Checking consistency of the graph transformation and whether a sound graph structure exists or not, along with controlling the consistency conditions have been broadly addressed by Heckel & Wagner [HW95].
- Employing tools such as AGG [Tae04] for automatically checking the consistency of a transformation.
- Isomorphic Reasoning and Commutative Inference: In order to validate the categorical diagrams the partial isomorphism in the semantic web environment can be defined based on the similarity in structural relationships between syntax, semantics, and the resources of the knowledge in ontological frameworks. From a categorical point of view, the simplest type of isomorphic reasoning involves an explicit and continuous mapping of the correspondences and similarities at the syntactic level while ignoring the semantics. This method enables us to perform reasoning about the dynamic structure of ontologies. For example, in the case of context change in ontology evolution, since the applicability of specific knowledge in one context does not automatically indicate the validity of the reasoning in the new context, thus the isomorphism between different states of the ontological structures and the knowledge they implied needs to be carefully analyzed. A common sense approach to get insight into a categorical diagrammatic structure and trace its various states, is to follow and

chase the diagrams depicting the objects and morphisms, to check whether the diagram is commutative or not and ensure the equality of the compositions. A diagram is commutative "iff whenever p and p' are paths with the same source and target, then the compositions of morphisms along these two paths are equal" [Gog91]. Putting two commutative diagrams together yields another commutative diagram. The diagram chasing along with commutative inference allow us the state space analysis to examine all the *potential state transitions* based on a *derived transformational pattern*. Therefore, one of the fundamental functionalities in ontology engineering that is the traceability of isomorphic reasoning processes through time from an initial ontology version to its current operational version can be performed.

- Using the semi-automated reasoning system introduced in [KKR06] for basic category-theoretic reasoning, which captures the basic categorical constructors, functors, and natural transformations, and provides services to check consistency, semantic coherency, and inferencing, is another option

In order to fully utilize the potential of reasoning and consistency checking in our framework, we are still working on this part as our ongoing research.

Categorical logic provides a reasoning service for changing ontologies, although for better analysis of changes within the states, it needs to be extended. Such an extension might be achieved, through our future work, by imposing some constraints, as proposed in [May83], on the occurrence of events and then deriving the appropriate state description. In addition, we plan to generalize our usage of category theory along with other formalisms such as colored Petri nets and Named graphs to improve the visualization of the changes. Also, to address some of the issues related to changes in conceptualizations and to improve the negotiation and learning processes, we want to extend the RLR framework towards inclusion of an NLP engine to deal with changes from a linguistic point of view. Based on our experience in dealing with category theory, we feel that this formalism still has plenty of potential left to be used for ontology change management; thus, the categorical constructors such as *sketches*, *n-categories,* and *enriched categories* are due for examination in future work.

In the employed graph transformation approach, we restricted ourselves to using typed labeled graphs; however, in order to increase the expressivity of the graph representation, one may want to employ hypergraphs instead. Although using hypergraphs increases the expressivity of our formalism, it also induces a tremendous amount of complexity on the reasoning process (comparable with using OWL Full as the representation language). In addition, extending the types of interactions between different change actions at the internal and external levels of our introduced HD graphs could be another possible enhancement. Moreover, modeling a rule-based query engine that enables us to pose complex queries to changing knowledge bases is another possible task to be pursued.

References

[AAG05] Ato, M., Ato, E., and Gómez, J. (2005) Analyzing change among developmental stages with categorical models. Quality and Quantity 39(1): 87–108.

[AAR+03] Al-Aboud, K., Al-Hawsawi, K., Ramesh, V., Al-Aboud, D. and AL-Githami, A. (2003) An Appraisal of Terms Used in Dermatology. SKINmed 2(3): 151–153.

[AB96] Arnold, R., and Bohner, S. (1996) Software Change Impact Analysis. Wiley-IEEE Computer Society Press; 1st edition.

[AB09] Arkoudas, K., Bringsjord, S. (2009) Vivid: An AI framework for heterogeneous problem solving. Artificial Intelligence, 173(15): 1367–1405.

[ABB+06] Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D. et al. (2000) Gene Ontology: tool for the unification of biology. Nat Genet., 25: 25–29.

[ACC01] Antoniol, G., Canfora, G., and Casazza, G. (2001) Andrea De Lucia: Maintaining traceability links during object-oriented software evolution. Softw., Pract. Exper. 31(4): 331–355.

[ADM+05] Aumueller, D., Do, H.H., Massmann, S., and Rahm, E. (2005) Schema and ontology matching with COMA++. In Proc. of the ACM SIGMOD int'l conference on Management of data, Baltimore, Maryland, pp. 906–908.

[ADM+07] d'Aquin, M., Doran, P., Motta, E., and Tamma, V.A..M. (2007) Towards a Parametric Ontology Modularization Framework Based on Graph Transformation. In Proc. of WoMO'07, CEUR 315.

[AE01] Artale, A., and Franconi, E. (2001) A survey of temporal extensions of description logics. Annals of Mathematics and Artificial Intelligence, 30(1-4): 17–210.

[AGM85] Alchourron, C.E., Gärdenfors, P., and Makinson, D. (1985) On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. J. of Symbolic Logic, 50: 510–530.

[AGM+90] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990) Basic local alignment search tool.J. Mol. Biol., 215: 403–410.

[AHS90] Adamek, J., Herrlich, H., and Strecker, G.E. (1990) Abstract and Concrete Categories: The Joy of Cats. J. Wiley & Sons.

[AL91] Asperti. A., and Longo, G. (1991) Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist. The MIT Press.

[Alp07] Alpheccar's blog (2007) Category Theory and the category of Haskell programs: Part 1, (Accessed 10 Dec 2009) http://www.alpheccar.org/en/posts/show/74

[ALT07] Artale, A., Lutz, C., and Toman, D. (2007) A Description Logic of Change. In Proc. of the 20th Int'l Joint Conference on Artificial Intelligence (IJCAI'07), Hyderabad, India, Jan 6-12, pp. 218–223.

[Amb96] Ambler, S. (1996) A Categorial Approach to the Semantics of Argumentation. Mathematical Structures in Computer Science 6(2): 167–188.

[AMF00] Avery, G., McGee, C., and Falk, S. (2000) Implementing LIMS: A 'How-To' Guide. Analytical Chemistry, 72(1): 57–62.

[AN09] Al-Mubaid, H., and Nguyen, H.A. (2009) Measuring Semantic Similarity Between Biomedical Concepts Within Multiple Ontologies. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 39(4): 389–398.

[And95] Andersen, N.M. (1995) Cladistic Inference and Evolutioanry Scenarios: Locomotory Structure, Function, and Performance in Water Striders. Cladistics, 11(3): 279–295.

[And81] Anderson, J. R. (1981) Concepts, propositions, and schemata: What are the cognitive units? In J. Flowers (edi.) Nebraska Symposium on Motivation. Lincoln, Nebraska: Uni. of Nebraska.

[ANR+06] Aizenbud-Reshef, A., Nolan, B.Y., Rubin, J., Shaham-Gafni, Y. (2006) Model Traceability. IBM System Journal, 45(3): 515–526.

[APR98] Agusti, J., Puigsegur, J. and Robertson, D.S. (1998) A visual syntax for logic and logic programming. Journal of Visual Languages and Computing, 9(4): 399–427.

[ARL07] Ashri, R., Rahwan, I., Luck, M. (2007) Architectures for Negotiating Agents. In Proc. of the 3rd Intl. Central & Eastern European Conference on Multi-Agent Systems and Applications (CEEMAS'03), Prague, Czech Republic, LNCS 2691, Springer, pp. 136–146.

[Art04] Artale, A. (2004) Reasoning on temporal conceptual schemas with dynamic constraints. In Proc. of 11th *Int.* Sympo. on *Temporal Representation and Reasoning* (TIME'04), Tatihou Island, Normandie, France, IEEE Comp. Soc., pp. 79–86.

[Aub90] Aubin, J.P. (1990) Fuzzy differential inclusions, Problems Control Inform. Theory 19: 55–67.

[Ave09] Aven, T. (2009) Identification of safety and security critical systems and activities. Reliability Engineering & System Safety, 94(2): 404–411.

[Awo06] Awodey, S. (2006) Category Theory. Oxford University Press.

[Awo09] Awodey, S. (2009) Categorical Logic. Lecture notes. http://www.andrew.cmu.edu/user/awodey/catlog/

[AY03] Avery, J., Yearwood, J. (2003) DOWL: A Dynamic Ontology Language. In Proc. of IADIS-ICWI'03, pp. 985–988.

[AY05] Avery, J., and Yearwood, J. (2005) A formal description of ontology change in OWL. In Proc. of the 3rd Intl. Conf. on Information Technology and App. (ICITA'05), Vol. 2, IEEE Comp. Society, pp. 238–243.

[Bai00] Bairoch, A. (2000) The ENZYME database in 2000. Nucleic Acids Res, 28: 304–305.

[BAP+02] Bernabé, M., Ahrazem, O., Prieto, A., and Leal, J.A. (2002) Evolution of Fungal Polysaccharides F1SS and Proposal of Their Utilisation as Antigenes for Rapid Detection of Fungal Contami nants. E. Journal of Env., Agr. & Food Chem. 1(1): 30–45.

[Bar87] Bartnicki-Garcia, S. (1987) The cell wall in fungal evolution. In Evolutionary biology of the fungi. Cambridge University Press, New York, N.Y., pp. 389–403.

[Bar93] Barfield, L. (1993). The User Interface Concepts and Design. New York: Addison Wesley, pp. 108–112.

[Bau95] Bauderon, M. (1995) A uniform approach to graph rewriting: the pullback approach. In Proc. of the 21st Int'l Workshop on Graph-Theoretic Concepts in Comp. Scie. (WG'95), Aachen, Germany, LNCS 1017, Springer, pp. 101–115.

[BBB+98] Baker, P.G., Brass, A., Bechhofer, S., Goble, C., Paton, N., Stevens, R. (1998) TAMBIS-Transparent Access to Multiple Bioinformatics Information Sources. Proc Int'l Conf Intell Syst Mol Biol, 6: 25–34.

[BBE91] Bjerknes, G., Bratteteig, T., and Espeseth, T. (1991): Evolution of Finished Computer Systems: The Dilemma of enhancement, Scandinavian Journal of Information Systems, 3: 25–46.

[BBF+01] B´erard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A. et al. (2001) Systems and Software Verification. Model-Checking Techniques and Tools, Springer.

[BCC+02] Buttler, D., Coleman, M., Critchlow, T., Fileto, R. et al. (2002) Querying Multiple Bioinformatics Data Sources: Can Semantic Web Research Help? SIGMOD Record 31(4): 59–64.

[BCM+03] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., and Patel-Schneider, P.F. (2003) The Description Logic Handbook: Theory, Implementation, and App. Cambridge University Press.

[BDG06] Bobillo, F., Delgado, M., and Gómez-Romero, J. (2006) A Crisp Representation for Fuzzy SHOIN with Fuzzy Nominals and General Concept Inclusions. In Proc. of the 2nd Workshop on Uncertainty Reasoning for the Semantic Web (URSW'06).

[BE99] Barzilay, R. and Elhadad, M. (1999) Using lexical chains for text summarization. In: I. Mani, M. T. Maybury (eds.) Advances in automatic text summarization, Cambridge, MA, The MIT Press, pp. 111–121.

[Bec] Bechhofer S. GALEN Documentation, frequently asked questions. (Accessed 16 Dec 2009) http://www.opengalen.org/faq/faq1.html

[Bec06] Beckett, D. (2006) SPARQL RDF Query Language Reference v1.8. (Accessed 10 Jan 2009) http://www.dajobe.org/2005/04-sparql/SPARQLreference-1.8.pdf.

[BEE+04] Bouquet, P., Ehrig, M., Euzenat, J., Franconi, E., Hitzler, P., Krötzsch, M., Serafini, L. et al. (2004) Specification of a common framework for characterizing alignment. Knowledge Web Deliverable 2.2.1v2, University of Karlsruhe. http://www.aifb.uni-karlsruhe.de/WBS/phi/pub/kweb-221.pdf

[BEF+06] de Bruijn, J., Ehrig, M., Feier, C., Martíns-Recuerda, F. et al. (2006) Ontology Mediation, Merging, and Aligning. In: Davies, J., Studer, R., and Warren, P. (eds.) Semantic Web Technologies. J. Wiley & Sons.

[Bel01] Bell, J.L. (2001) Observations on Category Theory. Axiomathes, 12(1-2): 151–155.

[Bel06] Bell, J.L. (2006) Abstract and Variable Sets in Category Theory. In Giandomenico Sica (ed.) What is Category Theory? Polimetrica Publisher, Italy, pp. 9–16.

[Ben98] Bench-Capon, T.J.M. (1998) Specification and implementation of Toulmin dialogue game. In Proc. of Legal Knowledge-Based Systems. JURIX: The 11th Conference, pp. 5–19.

[Bés97] Bés, M.O. (1997). Analysis of a human error in a dynamic environment: The case of air traffic control. In Proc. of the Workshop on Human Error and Systems Development, Glasgow.

[Bés99] Bés, M.O. (1999). A case study of a human error in a dynamic environment. Interacting with Computers, 11(5): 525–543.

[Bev03] Bevir, M. (2003) Notes toward an Analysis of Conceptual Change. Social Epistemology, 17(1): 55–63.

[BFK+95] Beitler, M., Foulds, R., Kazi, Z., Chester, D., Chen, S., and Salganicoff, M. (1995). A Simulated Environment of a Multimodal User Interface for a Robot. In Proc. of the RESNA 95 Annual Conference, Vancouver: RESNA95, 490–492.

[Bri04] Brickley, D. (editor) (2004) RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February. http://www.w3.org/TR/rdf-schema/

[BGL00] Bench-Capon, T.J.M., Geldard, T., and Leng, P.H. (2000) A method for the computational modelling of dialectical argument with dialogue games. Artificial Intelligence and Law, 8(2-3): 233–254.

[BH90] Bridgeland, D.M., and Huhns, M.N. (1990) Distributed Truth Maintenance. In Proc. of 8th National Conf. on Artificial Intelligence (AAAI'90), Boston, MA, pp. 72–77.

[BH04] Baresi, L., Heckel, R. (2004) Tutorial Introduction to Graph Transformation: A Software Engineering Perspective. In Proc. of the 2nd Int'l Conference on Graph Transformations (ICGT'04), Rome, Italy, LNCS 3256, Springer, pp. 431–433.

[BHB09] Bizer, C., Heath, T., and Berners-Lee, T. (2009) Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems, 5(3): 1–22.

[BHL01] Berners-Lee, T., Hendler, J., and Lassila, O. (2001) The semantic web. Scientific American, pp. 30–37.

[Bir98] Bird, R. (1998) Introduction to Functional Programming using Haskell, 2nd edi. Prentice Hall.

[BJ01a] Bauderon, M., and Jacquet, H. (2001) Pullback as a Generic Graph Rewriting Mechanism. Applied Categorical Structures 9(1): 65–82.

[BK08] Bryant, D., and Krause, P (2008) A review of current defeasible reasoning implementations. Knowledge Eng. Review 23(3): 227–260.

[BKK+87] Banerjee, J., Kim, W., Kim, H.J., and Korth, H.F. (1987) Semantics and implementation of schema evolution in object-oriented databases. ACM SIGMOD Record, 16(3): 311–322.

[BKK05] Busatto, G., Kreowski, H.J., and Kuske, S. (2005) Abstract hierarchical graph transformation. Mathematical Structures in Computer Science 15(4): 773–819.

[Bla84] Blass, A. (1984) The Interaction Between Category Theory and Set Theory. Mathematical Applications of Category Theory, 30, Providence: AMS, 5–29.

[BM99] Bench-Capon, T.J.M., and Malcolm, G. (1999) Formalising Ontologies and Their Relations. In Proc. of the 10th int'l conference on Database and Expert Systems Applications (DEXA'99), Florence, Italy, LNCS 1677, Springer, pp. 250–259.

[BMO01] Bauer, B., Müller, J.P., and Odell, J. (2001) Agent UML: A Formalism for Specifying Multiagent Software Systems. International Journal of Software Engineering and Knowledge Engineering 11(3): 207–230.

[Boh95] Bohner, S.A. (1995) A graph traceability approach for software change impact analysis. Ph.D. thesis, George Mason University.

[BÖS+05] Beyer, K.S., Özcan, F., Saiprasad, S., and Van der Linden, B. (2005) DB2/XML: designing for evolution. In Proc. of the ACM SIGMOD Conference on Management of Data, Baltimore, Maryland, USA, ACM press. pp. 948–952.

[BR99] Bailey, P.S., and Read, J. (1999) Software implementation of clinical terminologies: The use of component technology (tutorial), AMIA '99 Annual Symposium. Washington, DC.

[BRG+06] Bradbury, J.S., Rutherford, I., Graves, M., Tweedle, J., and Rosebrugh, R. (2006) User Guide for Graphical Database for Category Theory 3.0. Mount Allison Uni. 30 pp. (Accessed 25 Feb 2010) http://mathcs.mta.ca/research/rosebrugh/gdct/pdf/userguide/userguide.pdf

[Bru97] Brummitt, R.K. Taxonomy versus cladonomy, a fundamental controversy in biological systematic. Taxon, vol. 46, 1997, pp. 723–734.

[BS73] Bâianu, I., and Scripcariu, D. (1973) On adjoint dynamical systems. Bulletin of Mathematical Biology, 35(4): 475-486.

[BSF02] Boger, M., Sturm, and Fragemann, P. (2002) Refactoring Browser for UML. In Proc. of NetObjectDays'02, LNCS 2591, pp. 366–377.

[BSH+06] Bossung, S., Sehring, H.W., Hupe, P. and Schmidt, J.W. (2006) Open and Dynamic Schema Evolution in Content-intensive Web Applications. In Proc. of the 2nd Intl. Conf. on Web Information Systems and Technologies (WEBIST06), pp. 109–116.

[BSK+07] Bodenreider, O., Smith, B., Kumar, A., and Burgun, A. (2007) Investigating subsumption in SNOMED CT: An exploration into large description logic-based biomedical terminologies. Artificial Intelligence in Medicine, 39(3):183–195.

[BSS+06] Baker, C.J.O., Shaban-Nejad, A., Su, X., Haarslev. V., Butler G. (2006) Semantic web infrastructure for fungal enzyme biotechnologists. Journal of Web Semantics 4(3), 168–180.

[BT94] Brazier, F.M.T., and Treur, J. (1994). User Centered Knowledge-Based System Design: a Formal Modelling Approach. EKAW'94, LNCS 867, Springer, 282–302.

[BVH+04] Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks et al. (2004) OWL Web Ontology Language Reference. Feb 2004 http://www.w3.org/TR/owl-ref/

[BW05] Barr, M., and Wells, C. (2005) Toposes, Triples and Theories. Originally published on 1985 by Springer, Reprints in Theory and Applications of Categories, No. 1, 2005, pp. 1–289.

[BW06] Baker, C.J.O., and Witte, R. (2006) Mutation Mining – A Prospector's Tale. Information Systems Frontiers, 8 (1): 47–57.

[BZ06] Breu, S., and Zimmermann, T. (2006) Mining Aspects from Version History. In Proc. of the 21st IEEE/ACM Int'l Conference on Automated Software Engineering (ASE'06), 18–22 Sep, Tokyo, Japan, pp. 221–230.

[CBB+00] Discala, C., Benigni, X., Barillot, E., and Vaysseix, G. (2000) DBcat: A Catalog of 500 Biological Databases. Nucleic Acids Research, 28(1):8–9.

[CCS05] Capobianco, M., Chesñevar, C.I., and Simari, G.R. (2005) Argumentation and the Dynamics of Warranted Beliefs in Changing Environments. Autonomous Agents and Multi-Agent Systems 11(2): 127–151.

[CCS07] Capobianco, M. R., Chesñevar, C. I., and Simari, G. R. (2007) On the construction of Dialectical Databases. Inteligencia artificial, 35: 89–100.

[CCV+04] Caldwell, B., Chisholm, W., Vanderheiden, G., and White, J. (2004) Web Content Accessibility Guidelines 2.0. W3C Working Draft 11 March 2004. http://www.w3.org/TR/2004/ WD-WCAG20-20040311/

[CD02] Cabral, C.H., and Duarte, C. (2002) A Logico-Categorical Semantics of XML/DOM. In Proc. of the 2nd Web dynamic Workshop at WWW'02, Honolulu, Hawaii, USA.

[CDJ01] Colomb, R.M., Dampney, C.N.G., and Johnson, M. (2001) Category-theoretic fibration as an abstraction mechanism in information systems. Acta Inf. 38(1): 1–44.

[Ced] Cederqvist P, Version Management With CVS Copyright 1993–2005 Free Software Found Inc. http:// ftp.gnu.org/non-gnu/cvs/source/stable/1.11.22/cederqvist-1.11.22.pdf. (Accessed 15 Nov 2009).

[CG00] Corcho, O. and Gómez-Pérez, A. (2000) A roadmap for ontology specification languages., 12th Intl' Conference on Knowledge Engineering and Knowledge Management (EKAW-2000), France, Springer.

[CG05] Crous, P.W., and Groenewald, J.Z. (2005) Hosts, species and genotypes: opinions versus data. Australas Plant Path 34(4):463–470.

[CG07] Corcho, O., and Gómez-Pérez, A. (2007) ODEDialect: a Set of Declarative Languages for Implementing Ontology Translation Systems. J. UCS, 13(12): 1805-1834.

[CGC+07] Carrigan, N., Gardner, P.H., Conner, M., and Maule, J. (2007) The impact of structuring the interface as a decision tree in a treatment decision support tool. In Proc. of the 3rd Symp. of the HCI and Usability for Medicine and Health Care, USAB'07, Graz: Springer, pp. 273–288.

[CGG03] Crous, P.W., Groenewald, J.Z., Gams, W. (2003) Eyespot of cereals revisited: ITS phylogeny reveals new species relationships, European J. Plant Pathol. 109: 841–50.

[CGL01] Calvanese, D., de Giacomo, G., and Lenzerini, M. (2001) A Framework for Ontology Integration. In the The Proc. of the 1st Semantic Web Working Symposium (SWWS'01), Stanford University, California, USA, pp. 303–316.

[CGM+04] Crous, P.W., Groenewald, J.Z., Mansilla, J.P., Hunter, G.C., Wingfield, M.J. (2004) Phylogenetic reassessment of Mycosphaerella spp. and their anamorphs occurring on Eucalyptus. Studies in Mycology, 50: 195–214.

[CGS+04] Crous, P.W., Gams, W., Stalpers, J.A., Robert, V., and Stegehuis, G. (2004) MycoBank: an online initiative to launch mycology into the 21st century. Studies in Mycology 50: 19–22.

[CH07] Cafezeiro, I., and Haeusler, E.H. (2007) Semantic Interoperability via Category Theory. In Proc. Challenges in Conceptual modeling in the 26th intl. conference on Conceptual Modeling (ER'07), Auckland, New Zealand, Nov 5-9, CRPIT 83 Australian Comp. Soc., pp. 197–202.

[Cha90] Chang, S.K. (ed.): 1990, Principles of Visual Programming Systems, Prentice Hall, New York.

[Cha00] Chalupsky, H (2000) OntoMorph: A Translation System for Symbolic Knowledge. In Proc. of 7th int'l conf. on Principles of Knowledge Representation & Reasoning (KR'00), Breckenridge, Colorado, USA, Morgan Kaufmann Pub., pp. 471–482.

[Cha-1] Change Management for RDFS/OWL Ontologies, part 1: http://isegserv.itd.rl.ac.uk/cvs-public/~checkout~/swbp/vm/change-management/part1.html

[Cha-2] Change Management for RDFS/OWL Ontologies, part 2: http://isegserv.itd.rl.ac.uk/cvs-public/~checkout~/swbp/vm/change-anagement/part2.html

[Che04] Cheney, J. (2004) Category Theory for Dummies (I). (Accessed 15 Nov 2009) http://homepages.inf.ed.ac.uk/jcheney/presentations/ct4d1.pdf

[CHR08] Cafezeiro, I., Haeusler, E.H., and Rademaker, A. (2008) Ontology and Context. In Proc. of 6th IEEE Intl. Conf. on Pervasive Computing and Communications (PerCom'08), 17–21 March, Hong Kong, pp. 417–422.

[CHS+04] de Coronado, S., Haber, M.W., Sioutos, N., Tuttle, M.S., and Wright, L.W. (2004) NCI Thesaurus: using science-based terminology to integrate cancer research results. In Proc. of Medinfo. 2004; 11(Pt 1): 33–7.

[Cim96a] Cimino J.J. (1996) Formal descriptions and adaptive mechanisms for changes in controlled medical vocabularies. Methods of Information in Medicine, 35(3):202–210.

[Clo96] Clos, L.M. (1996) What is cladistics? Fossil News, Journal of Avocational Paleontology. http://www.fossilnews.com/1996/cladistics.html

[CLR04] Chen, S., Liu, B., and Rundensteiner, E.A., (2004) Multiversion-based view maint-enance over distributed data sources. ACM Trans. Database Syst. 29(4): 675–709.

[CM05] Chung, S., and McLeod, D. (2005) Dynamic Pattern Mining: An Incremental Data Clustering Approach. J. Data Semantics, 2: 85–112.

[CMR96] Corradini, A., Montanari, U., and Rossi, F., (1996) Graph processes, Fund. Inform. 26(3,4): 241–266.

[CMR+97] Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., and Löwe, M. (1997) Algebraic Approaches to Graph Transformation - Part I: Basic Concepts and Double Pushout Approach. Handbook of Graph Grammars, Vol 1: Foundations. World Scientific, pp. 163–246

[Coh04] Cohen SM (2004) Identity, Persistence, and the Ship of Theseus. Department of philosophy, University of Washington. http://faculty.washington.edu/smcohen/320/theseus.html

[Com01] Committee on Quality of Healthcare in America Institute of Medicine. (2001) Crossing the quality chasm: a new health system for the 21st century. Washington, DC: National Academy Press.

[Cor05] Corcho, O. (2005) A Layered Declarative Approach to Ontology Translation with Knowledge Preservation. Frontiers in AI & Applications116, IOS Press, pp. 20.

[COS+98] Campbell, K.E., Oliver, D.E., Spackman, K.A., and Shortliffe, E.H. (1998) Representing Thoughts, Words, and Things in the UMLS. J Am Med Inform Assoc, 5(5):421–431.

[CRG+96] Chawathe, S.S., Rajaraman, A., Garcia-Molina, H., and Widom, J. (1996) Change Detection in Hierarchically Structured Information. In Proc. of the ACM SIGMOD int'l conference on Management of data, Montreal, Quebec, Canad, pp. 493–504.

[Cro05] Crous, P.W (2005) Plant pathology is lost without taxonomy. Outlooks on Pest Management16:119–123.

[Cry97] Crystal, D. (1997) The Cambridge Encyclopedia of Language. 2nd edi. Cambridge Uni. Press.

[CS01] Chechik, M., and Easterbrook, S. (2001) Reasoning about Compositions of Concerns. In Proc. of the Workshop on Advanced Separation of Concerns in Software Eng. at ICSE'01.

[CS05b] Cushion, M.T., Stringer, J.R. (2005) Has the Name Really Been Changed? It Has for Most Researchers. Clinical Infectious Diseases, 41, 1756--1758 (2005)

[CS06] Ceusters, W., and Smith, B. (2006) A Realism-Based Approach to the Evolution of Biomedical Ontologies. in Proceedings of AMIA Annual Symposium.

[CSC07] Couto, F.M., Silva, M.J., and Coutinho, P.M. (2007) Measuring semantic similarity between Gene Ontology terms. Data & Knowledge Engineering, 61(1):137–152.

[CSG5a] Ceusters, W., Smith, B., and Goldberg, L. (2005) A Terminological and Ontological Analysis of the NCI Thesaurus. Method Inform Med 44:498–507.

[CSK+04] Ceusters W, Smith B, Kumar A, Dhaen C (2004) Ontology-Based Error Detection in SNOMED-CT. In Proc. of the 11th World Congress on Medical Informatics; MEDINFO04, IOS; 482–486.

[CW01] Collard, M., and Wood, B. (2001) Homoplasy and the early hominid masticatory system: inferences from analyses of extant hominoids and papionins. J. of Human Evolution, 41(3): 167–194.

[CZ06] Cimino ,J.J., and Zhu, X. (2006) The practical impact of ontologies on biomedical informatics. Yearb Med Inform. 2006:124-35.

[DAB+04] Dolin, R.H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F.M., Biron, P.V., Shabo, A. HL7 Clinical Document Architecture, Release 2.0 (Last Published: Sun 12/12/2004. http://www.e-ms.ca/documents/pdf_v3ballotCDA_ 2005Jan.pdf

[DC94] Diskin, Z., Cadish, B. (1994) Algebraic Graph-Oriented = Category Theory Based. Manifesto of categorizing database theory. Tech. Report # 9406, Frame Info. Sys., Latvia. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.5787

[DCP05] Dunn-Davies, H.R., Cunningham, J., and Paurobally, S. (2005) Propositional Statecharts for Agent Interaction Protocols. Electr. Notes Theor. Comput. Sci. 134: 55-75.

[DD04] Deridder, D., and D'Hondt, T.A. (2004) Concept-Centric Approach to Software Evolution. In Proc. of ACM OOPSALA'04 workshop, Vancouver, Canada.

[Dec93] Decortis, F. (1993). Operator strategies in a dynamic environment in relation to an operator model. Ergonomics Special issue: Cognitive processes in complex tasks, 36(11), 1291–1305.

[DHK02] Depke, R., Heckel, R., and Küster, J.M. (2002) Formal agent-oriented modeling with UML and graph transformation. Sci. Comput. Program. 44(2): 229-252.

[DHP02] Drewes, F., Hoffmann, B., and Plump, D. (2002) Hierarchical Graph Transformation. J. Comput. Syst. Sci. 64(2): 249–283.

[Dek95] De Keyser, V. (1995). Time in Ergonomics. Ergonomics, 38(8), 1639–1661.

[Dev01] Devedžic, V. (2001) Knowledge Modeling - State of the Art. Integrated Computer-Aided Engineering, 8(3): 257–281.

[DG92] De Queiroz, K., and Gauthier, J. (1992) Phylogenetic taxonomy. Ann. Rev. Ecol. Syst., 23: 449–480.

[DGL08] Ontology Evolution in the Life Sciences Project (2008) Database Group Leipzig. http://dbs.uni-leipzig.de/research/projects/bioinformatik/ontology_evolution/gosubontologies

[DH96] Duribreux-Cocquebert, M., and Houriez, B. (1996). A user-centered methodology for knowledge-based systems development: MODESTI. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2, 1208–1213.

[DH05] Doan, A., and Halevy, A.Y. (2005) Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine 26(1):83–94.

[DHH+01] Degen, W., Heller, B., Herre, H., and Smith, B. (2001) GOL: toward an axiomatized upper-level ontology. In Proc. of 2nd Int'l Conf. on Formal Ontology in Information Systems (FOIS'01), Ogunquit, Maine, USA, ACM press, pp. 34-46.

[DHK00] Depke, R., Heckel, R., and Küster, J.M. (2000) Agent-Oriented Modeling with Graph Transformation. In Proc. of AOSE'00, LNCS 1957, Springer, pp. 105-120.

[Din08] Dini, P. (2008) Notes on Relational Biology and Elementary Category Theory. In Proc. of the 2nd Int'l OPAALS Conference on Digital Ecosystems, Tampere, Finland, 7 - 8 October. http://matriisi.ee.tut.fi/hypermedia/events/opaals2008/article/opaals2008-article25.pdf

[DKK+99] Drewes, F., Knirsch, P., Kreowski, H.J. et al. (1999) Graph Transformation Modules and Their Composition. In Proc. of AGTIVEè99, LNCS 1779, Springer, pp. 15-30.

[DLR09] D'Ambros, M., Lanza, M., and Robbes, R. (2009) On the Relationship Between Change Coupling and Software Defects. In Proc. of 16th Working Conference on Reverse Engineering (WCRE'09), IEEE Comp. Soc. pp. 135-144.

[DM07] De Leenheer, P., and Mens, T. (2007) Using Graph Transformation to Support Collaborative Ontology Evolution. In Proc. of the 3rd Intl. Sympo. on App. of Graph Transformations with Industrial Relevance (AGTIVE'07), Kassel, Germany, LNCS 5088 Springer, pp. 44-58.

[DM08] De Leenheer, P., and Mens, T. (2008) Ontology Evolution. in Hepp, M., De Leenheer, P., de Moor, A., and Sure, Y (Eds.) Ontology Management, Semantic Web, Semantic Web Services, and Business Applications. Semantic Web And Beyond Computing for Human Experience Vol. 7 Springer, pp. 131–176.

[DMM07] De Leenheer,P., de Moor, A., and Meersman, R. (2007) Context Dependency Management in Ontology Engineering: A Formal Approach. J. Data Semantics, 8: 26–56.

[DMQ05] Dou, D., McDermott, D.V., and Qi, P. (2005) Ontology Translation on the Semantic Web. J. Data Semantics, 2: 35–57.

[Don05] Donnelly, M. (2005) Containment relations in anatomical ontologies. 2005 AMIA Fall Symposium, pp. 206–210.

[DP97] Dubois, D., Prade, H. (1997) The three semantics of fuzzy sets. Fuzzy Sets and Systems 90 (1997) 141–150.

[DP04] Ding, Z., Peng, Y. (2004) A probabilistic extension to ontology language OWL. In Proc. of the 37th Hawaii Inl. Conference on System Sciences, (HICSS-37), 10 pp.

[DR04] Dutta, P.K., and Radner, R. (2004) Self-enforcing climate-change treaties. PNAS, 101(14): 5174-5179.

[Dra97] Dragoni, A.F. (1997) Belief revision: from theory to practice. The Knowledge Engineering Review, 122(2):147–179.

[DS06] Dobson, G., and Sawyer, P. (2006) Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web. In Proc. of the Intl. Seminar on Dependable Requirements Eng. of Computerised Systems at NPPs, Norway.

[DSS93] Davis, R., Shrobe, H., Szolovits, P. (1993) What is a Knowledge Representation? AI Magazine, 14(1):17–33.

[Dub] The Dublin Core Metadata Initiative (DCMI) http://dublincore.org

[DW08] Diskin, Z., and Wolter, U. (2008) A Diagrammatic Logic for Object-Oriented Visual Modeling. Electronic Notes in Theoretical Comp. Sci., 203(6, 21):19–41.

[Eas98] Easterbrook, S. Category theory for beginners, Tutorial given at ASE'98, Oct 1998. http://www.cs.toronto.edu/~sme/presentations/cat101.pdf

[EBL+03] Elkin, P.L., Brown, S.H., Lincoln, M.J., Hogarth, M., and, Rector, A. (2003) A formal representation for messages containing compositional expressions. Int'l J. of Medical Infor, 71(2-3):89–102.

[ECP+02] Estrin, D., Culler, D., Pister, K., and Sukhatme, G. (2002) Connecting the Physical World with Pervasive Networks. IEEE Pervasive Computing, 1(1): 59–69.

[ED07] Eisenbarth, M., and Dörr, J. (2007) Facilitating Project Management by Capturing Requirements Quality and Volatility Information. In Proc. of Workshop on Measuring Requirements for Project and Product Success (MeReP'07), Palma de Mallorca, Spain,

[EEP+06] Ehrig, H., Ehrig, K., Prange, U., and Taentzer, G. (2006) Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series, Springer.

[EHK+90] Ehrig, H., Habel, A., Kreowski, H.J., and Parisi-Presicce, F. (1990) From Graph Grammars to High Level Replacement Systems. In Proc. of the 4th int'l workshop on Graph-Grammars and Their App. to Comp. Sci. Bremen, Germany, LNCS 532, Springer, pp. 269–291.

[EHK+97] Ehrig, H., Heckel, R., Korff, M., Löwe, M. et al. (1997) Algebraic Approaches to Graph Transformation - Part II: Single Pushout Approach and Comparison with Double Pushout Approach. Handbook of Graph Grammars, World scientific, pp. 247–312.

[Ehr79] Ehrig, H. (1979) Introduction to the algebraic theory of graph grammars (a survey). in proc. of int'l workshop on Graph-Grammars and their Application to Com. Sci. and Biology, Bad Honnef, Germany, LNCS 73, Springer, pp. 1-69.

[EJ03] Van Eetvelde, N., and Janssens, D. (2003) A Hierarchical Program Representation for Refactoring. Electr. Notes Theor. Comput. Sci. 82(7):91–104.

[EKL90] Ehrig, H., Korff, M., and Löwe, M. (1990) Tutorial Introduction to the Algebraic Approach of Graph Grammars Based on Double and Single Pushouts. In Proc. of 4th Int'l Workshop Graph-Grammars and Their App. to Comp. Sci., Bremen, Germany, LNCS 532, Springer, pp. 24-37.

[EM45] Eilenberg S, Mac Lane S (1945) General Theory of Natural Equivalences. T Am Math Soc 58:231–294.

[Enc04] Encyclopedia Everything, unique up to isomorphism, Jan 16 2004 http://everything2.com/title/unique+up+to+isomorphism

[Enz84] Enzyme nomenclature Recommendations (1984) of the nomenclature committee of the int'l union of biochemistry. Academic Press, Orlando/NY 1984. pp. 646.

[EN07] Elmasri, R., and Navathe, S.B. (2007) Fundamentals of Database Systems. 5th Ed.. Addison Wesley.

[EOP06] Ehrig, H., Orejas, F., and Prange, U. (2006) Categorical Foundations of Distributed Graph Transformation. In Proc. of 3rd int'l Conference on Graph Transformations (ICGT'06), Natal, Rio Grande do Norte, Brazil, LNCS 4178, Springer, pp. 215-229.

[EP05] Ehrig, H., Prange, U. (2005) Modeling with Graph Transformation. In Proc. of InterSymp'05. IIAS. Baden-Baden, Germany. http://tfs.cs.tu-berlin.de/publikationen/Papers05/EP05.pdf

[EPS73] Ehrig, H., Pfender, M. and Schneider, H. J. (1973) Graph grammars: An algebraic approach. In Proc. of 14th Symposium on Foundations of Comp. Science, Iowa City, Iowa, IEEE Comp Society, pp. 167–180.

[EPT04] Ehrig,H., Prange, U., and Taentzer, G. (2004) Fundamental Theory for Typed Attributed Graph Transformation: Long Version. at: http://iv.tu-berlin.de/TechnBerichte/2004/2004-09.pdf

[ER97] Engelfriet, J., and Rozenberg, G. (1997) Node Replacement Graph Grammars. In: Handbook of graph grammars and computing by graph transformation: volume I. foundations, World Scientic Publishing Co., pp.1–94.

[ER00] Engels, G., and Heckel, R. (2000) Graph Transformation as a Conceptual and Formal Framework for System Modeling and Model Evolution. In Proc. of ICALP'00, Geneva, Switzerland, LNCS 1853, Springer, pp. 127–150.

[ES95] Engels, G., and Schürr, A. (1995) Encapsulated hierarchical graphs, graph types, and meta types. Electr. Notes Theor. Comput. Sci. 2: 101–109.

[ES04] Ehrig, M., and Sure, Y. (2004) Ontology mapping - an integrated approach. in: 1st European Semantic Web Symposium (ESWS'04), Heraklion, Greece, LNCS 3053, Springer, pp. 76–91.

[ES07] Euzenat, J., and Shvaiko, P. (2007) Ontology Matching. Springer-Verlag, Berlin.

[EV06] Ehresmann AEC, Vanbremeersch JP (2006) The Memory Evolutive Systems as a Model of Rosen's Organism-(Metabolic, Replication) Systems. Axiomathes, 16: 137–154.

[FBA98] Frisvad, J.C., Bridge, P.D., Arora, D.K. (1998) Fungal chemical taxonomy. Marcel Dekker, Inc., New York-Basel-Hong Kong.

[FC03] Fornara, N. and Colombetti, M. (2003) Defining Interaction Protocols using a Commitment BasedAgent Communication Language. In Proc. of AAMAS'03, ACM Press, pp. 520–527.

[Fel05] Felsenstein, J. PHYLIP (Phylogeny Inference Package) ver. 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle, 2005.

[FFG+95] Fortnow, L., Freivalds, R., Gasarch, W.I., Kummer, M. et al. (1995) Measure, Category and Learning Theory. Proc. of the 22nd Colloq. on Automata, Languages and Programming (ICALP'95), Szeged, Hungary, LNCS 944 Springer, pp. 558–569.

[FFM+94] Finin, T.W., Fritzson, R., McKay, D.P., and McEntire, R. (1994) KQML As An Agent Communication Language. In Proc. of the 3rd Int'l Conf. on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, pp. 456–463.

[FGG08] Fluri, B., Giger, E., Gall, H. (2008) Discovering Patterns of Change Types. In Proc. of the 23rd Intl. Conf. on Automated Soft. Eng. (ASE'08), 15-19 Sep 2008, L'Aquila, Italy, pp. 463–466.

[FHP+06] Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., and Wache, H. (2006) Inconsistencies, Negations and Changes in Ontologies. In Proc. of 21st National Conf. on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, July 16–20, Boston, Massachusetts, AAAI 2006.

[FKS02] Falappa, M., Kern-Isberner, G., and Simari, G.R. (2002) Explanations, belief revision and defeasible reasoning, Artificial Intelligence, 141(1-2): 1–28.

[Fia04] Fiadeiro, J.L. (2004) Categories for Software Engineering," Springer, 1st edition

[Flo06] Flouris G (2006) On Belief Change and Ontology Evolution. Ph.D. thesis in the Department of Computer Science, University of Crete.

[FMK+08] Flouris G, Manakanatas D, Kondylakis H, Plexousakis D, Antoniou G (2008) Ontology change: classification and survey. Knowl Eng Rev 23(2):117–152.

[FMU82] Fagin, R., Mendeizon, A., and Ullman, J. (1982) A simplified universal relation assumption and its properties, ACM Trans. on Database Systems 7(3):343–360.

[For98] Forsythe, D.E. (1998). Using ethnography to investigate life scientists' information needs. Bull Med Libr Assoc., 86(3), 402–9.

[FPA06] Flouris, G., Plexousakis,D., and Antoniou, G. (2006) Evolving Ontology Evolution. In Proceedings of the SOFSEM 2006. Merín, Czech Republic, Springer, 14-29.

[FR98] Fromkin, V., and Rodman, R. (1998) An introduction to language, 6th edition. Fort Worth: Harcourt Brace.

[Fra08] Frank, A.U. (2008). Shortest Path in a Multi-Modal Transportation Network, KI Künstliche Intelligenz, 3:14-18.

[FSU06] Fact Sheet of Unified Medical Language System (UMLS) Semantic Network (2006) last updated on 28 Mar 2006. http://www.nlm.nih.gov/pubs/factsheets/umlssemn.html. Accessed 10 Jan 2009.

[FT05] Franconi, E., and Tessaris, S. (2005) A unified logical framework for rules (and queries) with ontologies. In proc. of W3C Workshop on Rule Lang. for Interoperability, Washington, D.C., USA.

[Fut05] Futuyma, D.J. (2005) Evolution. Sunderland, Massachusetts: Sinauer Associates, Inc.

[FVK+00] Fensel, D., Van Harmelen, F., Klein, M., Akkermans, H.,Broekstra, J., Fluit, C. etal. (2000) Ontoknowledge: Ontology-based Tools for Knowledge Management, eBusiness and eWork, 2000, Madrid, October.

[FWP+07] Fluri, B., Würsch, M., Pinzger, M., and Gall, H.C. (2007) Change distilling: Tree differencing for fine-grained source code change extraction. IEEE Transactions on Software Eng. (TSE), 33(11): 725-743.

[Gam00] Gambetta. D. (2000). Can We Trust Trust? In D. Gambetta (ed.), Trust: Making and Breaking Cooperative Relations. Oxford: University of Oxford, Chapter 13.

[Gär90] Gärdenfors, P. (1990) Knowledge in Flux: Modeling the Dynamics of Epistemic States. MIT Press, Cambridge, MA, 1990.

[GBB+94] Greuter, W., Barrie, F.R., Burdet, H.M., Chaloner, W.G. et al. (1994) International code of botanical nomenclature (Tokyo Code) adopted by the 15th Intl. Botanical Congress, Yokohama, Aug-Sep 1993. Regnum Veg. 131. Koeltz Scientific Books, Königstein, Germany.

[GBP+96] Glenn, A.E., Bacon, C.W., Price, R., and Hanlin, R.T. (1996) Molecular phylogeny of Acremonium and its taxonomic implications. Mycologia 88:369–383.

[GF94] Gotel, O.C.Z., and Finklestein, A.C.W. (1994) An analysis of the requirements traceability problem. In Proc. of the 1st Int'l Conference on Requirements Engineering (ICRE'94), Colorado Springs, CO, USA, pp.94–101.

[GFP09] Gall, H.C. , Fluri, B., Pinzger, M., (2009) Change Analysis with Evolizer and ChangeDistiller. IEEE Software, 26(1): 26–33.

[GGS99] Guarro, J., Gene', J., and Stchigel, A.M. (1999) Developments in fungal taxonomy. Clinical Microbiology Reviews, 12(3): 454–500.

[GGW03] Ganesan, P., Garcia-Molina, H., and Widom, J. (2003) Exploiting hierarchical domain structure to compute similarity. ACM Trans. Inf. Syst. 21(1): 64-93.

[GHJ98] Gall, H., Hajek, K., and Jazayeri, M. (1998) Detection of logicalcoupling based on product release history. In Proc. of Int'l IEEE Conference on Software Maintenance (ICSM'98), pp. 190–198.

[GHM04] Gutiérrez, C., Hurtado, C.A., and Mendelzon, A.O. (2004) Foundations of Semantic Web Databases. In Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Sympo. on Principles of Database Systems (PODS'04), Paris, France, pp. 95-106.

[GVH03] Gyapay, S., Varró, D., and Heckel, R. (2003) Graph Transformation with Time. Fundam. Inform. 58(1): 1–22.

[GHV04] Gamma, R.J.E., Helm, R., and Vlissides, J. (2004) Design patterns: Elements of reusable object-oriented software, Addison-Wesley.

[GHV07] Gutierrez, C., Hurtado, C.A., and Vaisman, A.A. (2007) Introducing Time into RDF. IEEE Trans. Knowl. Data Eng. 19(2): 207–218.

[Gil06] Gilbert, M.C. (2006) The Dialectics of Knowledge Management. http://news.gilbert.org/DialecticsKM

[GL02] Giugno, R., Lukasiewicz, T. (2002) P-SHOQ(D): A Probabilistic Extension of SHOQ(D) for Probabilistic Ontologies in the Semantic Web. pp. 86–97.

[GL05] Gao, M., and Liu, C. (2005) Extending OWL by Fuzzy Description Logic. In Proc. of ICTAI'05, pp. 562–567.

[GLT89] Girard, J.Y., Lafont, Y., Taylor, P.: Proofs and Types. Cambridge University Press (1989)

[GMZ99] Gupta, A., Masthoff, J., and Zwart, P. (1999). Improving the User Interface to Increase Patient Throughput. In Proceedings of the First Workshop on Human Error and Clinical Systems (HECS'99), Scotland: Glasgow, 15-17 April.

[GOB] GO Browser representing blood pressure terminologies. (Accessed 10 Jan 2009) (http://www.informatics.jax.org/searches/GO.cgi?id=GO:0008217)

[Gog91] Goguen, J. (1991) A Categorical Manifesto. Mathematical Structures in Comp. Sci., 1(1): 49–67.

[Gol06] Goldblatt, R. (2006) Topoi; The Categorial Analysis of Logic. Mineola, NY, Dover Publications.

[GOM01] GO Meeting collected notes, July 14-15,2001, Hosted by Judy Blake and the Jackson Lab in Bar Harbor, ME. compiled by L. Reiser. http://www.geneontology.org/minutes/collected_minutes.txt

[GON06] GO Newsletter, Issue No. 1 May 2006. (http://www.geneontology.org/newsletter/archive/200605.shtml#bp)

[GON07a] GO Newsletter, No. 4 February 2007. http://www.geneontology.org/newsletter/archive/200702.pdf

[GON07b] The Gene Ontology Newsletter, Issue No. 5 May 2007. http://www.geneontology.org/newsletter/archive/200705.pdf

[GON07c] GO Newsletter, Issue No. 6 Aug. 2007. (http://www.geneontology.org/newsletter/current-Newsletter.shtml)

[Got07] Gottwald, S. (2007) Many-Valued Logics. In: Handbook of the Philosophy of Sciences. Vol. 5: Philosophy of Logic (D. Jacquette ed.), North-Holland: Amsterdam, pp. 545-592. available at: http://www.uni-leipzig.de/~logik/gottwald/SGforDJ.pdf

[GPS98] Große-Rhode, M.., Parisi-Presicce, F., and Simeoni, M. (1998) Spatial and Temporal Refinement of Typed Graph Transformation Systems. In Proc. of MFCS'98, LNCS 1450, Springer, pp. 553–561.

[Gra84] Gray, J.W. (1984) Mathematical Applications of Category Theory. American Mathematical Society, p. 11.

[Gru93] Gruber, T.R. (1993) A translation approach to portable ontologies. Knowledge Acquisition 5(2): 199–220.

[Gru95] Gruber, T. R. (1995) Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human and Computer Studies, 43(5/6):907–928.

[GSG04] Grenon P, Smith B, Goldberg L (2004) Biodynamic ontology: Applying BFO in the Biomedical Domain, in Pisanelli DM (ed). Ontologies in Medicine. Proceedings of the Workshop on Medical Ontologies, Rome, IOS Press, Studies in Health Technology and Informatics, vol 102:20–38.

[GSV04] Gabel, T., Sure, Y., and Voelker, J. (2004). KAON – ontology management infrastructure. SEKT informal deliverable 3.1.1.a, Inst. AIFB, Uni. of Karlsruhe. http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/SEKT-D3.1.1.a.pdf

[Gua95] Guarino N (1995) Formal Ontology, Conceptual Analysis and Knowledge Representation. Int J Hum-Comput St 43(5/6):625–640.

[Gua98] Guarino, N. (1998) Formal Ontology and Information Systems. In Proceedings of FOIS'98, Trento, Italy, IOS Press, pp 3–15.

[Guo02] Guo, J. (2002) Using Category Theory to Model Software Component Dependencies. In Proc. of the 9th IEEE Intl. Conf. on Engineering of Computer-Based Systems (ECBS'02), 8-11 April, Lund, Sweden, pp. 185-194.

[Güt04] Güttner, J. (2004) Object Databases and the Semantic Web. Ph.D. Thesis, Brno University of Technology.

[Hal01] Halpin, T. (2001) Information Modeling and Relational Databases. Morgan-Kaufmann.

[Han03] Hansson, S.O. (2003) Ten Philosophical Problems in Belief Revision. J. of Logic and Computation, 13(1): 37–49.

[Har88] Harel, D. (1988) On visual formalisms., Communication ACM, 31(5):514–530.

[Har05] Harris, M.A. (2005) Why GO there? Ensuring that the Gene Ontology Meets Biologists' Needs. The Gene Ontology Consortium and EMBL-EBI, Wellcome Trust Genome Campus, Hinxton, UK.

[Har05b] Harnad, S. (2005) To Cognize is to Categorize: Cognition is Categorization. in Cohen, H., and Lefebvre, C. (eds.) Handbook of categorization in Cognitive Science. Elsevier, pp. 19–43.

[Haw93] Hawksworth, D.L. (1993) Name changes for purely nomenclatural reasons are now avoidable. Systema Ascomycetum 12:1–6.

[Haw04] Hawksworth, D.L. (2004) Fungal diversity and its implications for genetic resource collections. Stud Mycol 50:9–17.

[HB93] Hartson, H., and Boehm-Davis, D. (1993). User interface development processes and methodologies. Behavior & Information Technology, 12(2), 98–114.

[HBM02] Hendler, J., Berners-Lee, T., and Miller, E. (2002) Integrating Applications on the Semantic Web. Journal of the Institute of Electrical Engineers of Japan, 122(10): 676–680.

[HBE+04] Haase, P., Broekstra, J., Eberhart, A., Volz, R. (2004) A Comparison of RDF Query Languages. In Proc. of the 3rd Int'l Semantic Web Conference (ISWC'04), LNCS 3298, Springer, pp. 502–517.

[HC04] Healy, M.J., Caudell, T.P. (2004) Neural Networks, Knowledge and Cognition: A Mathematical Semantic Model Based upon Category Theory. Tech Report EECE-TR-04-020, Uni. of New Mexico. https://repository.unm.edu/dspace/bitstream/1928/33/2/EECE-TR-04-020.pdf

[HC04b] Heckel, R., and Cherchago, A. (2004) Application of Graph Transformation for Automating Web Service Discovery. In Proc. of Language Engineering for Model-Driven Software Development 2004, Dagstuhl Seminar Proceedings 04101.

[HC06] Healy, M.J., and Caudell, T.P. (2006) Ontologies and Worlds in Category Theory: Implications for Neural Systems. Axiomathes 16:165–214.

[HCE+96] Heckel, R., Corradini, A., Ehrig, H., and Löwe, M. (1996) Horizontal and Vertical Structuring of Typed Graph Transformation Systems. Mathematical Structures in Comp. Science 6(6): 613–648.

[Hea86] Heath, I.B. Nuclear division: a marker for protist phylogeny. Prog. Protis.1988, 1:115-162.

[Hea00] Healy, M.J. (2000) Category Theory Applied to Neural Modeling and Graphical Representations. in Proc of the Proceedings of the IEEE Intl. Joint Conference on Neural Networks, (IJCNN'00), Como, Italy, July 24-27, 2000, Volume 3, pp. 35-40.

[Hea07] Healy, M.J. (2007) Category Theory as a Mathematics for Formalizing Ontologies. http://johnsymons.files.wordpress.com/2007/10/healy-tao-r3.pdf

[Hec06] Heckel, R. (2006) Graph Transformation in a Nutshell. Electr. Notes Theor. Comput. Sci. 148(1): 187–198.

[Hed08] Hedden, H. (2008) Controlled Vocabularies, Thesauri, and Taxonomies. The Indexer, 26(1): 33–34.

[HEK+06] Hitzler, P., Euzenat, J., Krötzsch, M., Serafini, L., Stuckenschmidt, H. et al. (2006) Integrated view and comparison of alignment semantics. Tech Rep. D2.2.5, AIFB, Uni. of Karlsruhe. http://www.aifb.uni-karlsruhe.de/Publikationen/showPublikation_english?publ_id=1125

[HES+05] Hayes, P.J., Eskridge, T.C., Saavedra, R., Reichherzer, T., Mehrotra, M., and Bobrovnikoff, D. (2005) Collaborative knowledge capture in ontologies. In Proc. of the 3rd Int'l Conference on Knowledge Capture (K-CAP'05), Banff, Alberta, Canada. ACM, pp. 99–106.

[Hey90] Heylighen, F. (1990) Representation and Change. A Metarepresentational Framework for the Foundations of Physical and Cognitive Science, (Communication & Cognition, Gent), 200 p.

[Hey95] Heywood VH (ed.) (1995) Global Biodiversity Assessment. Cambridge Uni. Press.

[HFO+03] Hartel, F.W., Fragoso, G., Ong, K., and Dionne, R. (2003) Enhancing quality of retrieval through concept edit history. 2003 AMIA Annu Symp Proc, pp. 279–83.

[HG04] Hardie RP, Gaye RK (translators) (2004) Physics by Aristotle. eBooks@Adelaide, Adelaide University: http://etext.library.adelaide.edu.au/a/aristotle/a8ph/index.html

[HH00] Heflin, J., Hendler, J.A. (2000) Dynamic Ontologies on the Web. In proceedings of AAAI/IAAI 2000, pp. 443–449.

[HHJ+07] Hudak, P., Hughes, J., Jones, S.P., and Wadler, P. (2007) A history of Haskell: being lazy with class. In Proc. of the 3rd ACM SIGPLAN conf. on History of programming language. San Diego, CA, pp.12-1 – 12-55.

[HHL99] Heflin, J., Hendler, J., and Luke, S. (1999) Coping with changing ontologies in a distributed environment. In Proc. of the Workshop on Ontology Management at the 16th National Conf. on Artificial Intelligence (AAAI'99), Berlin, AAAI Tech. Report WS-99–13, pp.74-79.

[HHT96] Habel, A., Heckel, R., and Taentzer, G. (1996) Graph Grammars with Negative Application Conditions. Fundam. Inform. 26(3/4): 287–313.

[HIA92] Hilton, E., Isenberg, H.D., and Alperstein, P. (1992) Ingestion of yogurt containing Lactobacillus acidophilus as prophylaxis for candidal vaginitis. Ann Intern Med, 116:353–7.

[Hin08] Hines, P. (2008) Machine semantics. Theoretical Computer Science, 409(1):1–23.

[HJK+95] Heimann, P., Joeris, G., Krapp, C.A., Westfechtel, B. (1995) A programmed graph rewriting system for software process management. Electr. Notes Theor. Comput. Sci., 2: 127–136.

[HKE+05] Hitzler, P., Krötzsch, M., Ehrig, M., and Sure, Y. What is ontology merging? - a category theoretic perspective using pushouts. (2005) In Proc. of the 1st Intl. Workshop on Contexts & Ontologies: Theory, Practice and Applications (C&O), AAAI Press, pp. 104–107.

[HKL05] Harman, M., Korel, B., Linos, P.K. (2005) Special issue on software maintenance and evolution. IEEE Transactions on Software Engineering, 31(10): 801–803.

[HKR08] Hartung, M., Kirsten, T., Rahm, E. (2008) Analyzing the Evolution of Life Science Ontologies and Mappings. In Proc. of the 5th Int'l Workshop in Data Integration in the Life Sciences (DILS'08), Evry, France, LNCS 5109, Springer, pp. 11–27.

[HKS+95] Hawksworth, D.L., Kirk, P.M., Sutton, B.C., and Pegler, D.N. (1995) Ainsworth and Bisby's dictionary of the fungi, 8th ed. Intern. Myco. Institute, Egham, UK.

[HL02] Heifetz, R.A., Linsky, M. (2002). Leadership on the Line: Staying Alive Through the Dangers of Leading. Boston: Harvard Business School Press, 1st edition.

[HLR] Health Level 7 Reference Information Model http://healthinfo.med.dal.ca/hl7intro/CDA_R2_normativewebedition/infrastructure/rim/rim.htm

[HLS+98] Humphreys BL, Lindberg DAB, Schoolman HM, and Barnett GO (1998) The Unified Medical Language System An Informatics Research Collaboration. J Am Med Inform Assoc 5(1):1–11.

[HLW97] ter Hofstede, A.H.M., Lippe, E., and van der Weide, T.P. (1997) Applications of a categorical framework for conceptual data modeling. Acta Informatica, 34(12): 937–963.

[HM01] Haarslev V, Möller R. RACER System Description. In Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR01), Siena, Italy, June 18–23, 2001, p.701–706.

[HM03] Haarslev, V., Möller, R.: Description Logics for the Semantic Web: Racer as a Basis for Building Agent Systems. KI 17(3) 10–15 (2003)

[HMW04] Haarslev, V., Möller, R., and Wessel , M. (2004) Querying the Semantic Web with Racer + nRQL. In Proc. of KI'04 Int'l Workshop on Applications of DLs (ADL'04), Ulm, Germany, Sep. 24.

[Hof99] Hoffmann, B. (1999) From Graph Transformation to Rule-Based Programming with Diagrams. In Proc. of AGTIVE'99, LNCS 1779, Springer, pp. 165–180.

[Hol98] Holsti, K.J. (1998) The Problem of Change in International Relations Theory. Paper No. 26 from CIR Working Paper Series.

[Höp03] Höppner, F.: Knowledge discovery from sequential data. PhD thesis, Technical University Braunschweig, Germany, (2003).

[Hor07] Horrocks, I. (2007) Ontology Engineering: Tools and Methodologies. Tutorial in SemanticDays07. (www.comlab.ox.ac.uk/people/ian.horrocks/Seminars/download/ SemanticDays07-tutorial.ppt)

[HOY+09] Healy, M.J., Olinger, R.D., Young, R.J., Taylor, S.E., Caudell, T., and Larson, K.W. (2009) Applying category theory to improve the performance of a neural architecture. Neurocomputing, (Article in Press)

[HP04] Heflin, J., Pan, Z. (2004) A model theoretic semantics for ontology versioning. In Proc. of the 3rd Int'l Semantic Web Conf. (ISWC'04), Hiroshima, Japan, LNCS 3298, Springer, pp. 62–76.

[HP04b] Horrocks, I., Patel-Schneider, P.F. (2004) Reducing OWL entailment to description logic satisfiability. J. Web Sem. 1(4): 345–357.

[HR97] Hawksworth, D.L., Rossman, A.Y. (1997) Where are all the undescribed fungi? Phytopathology. 87: 888–891.

[HR01] Hardiker, N.R., Rector, A.L. (2001) Structural validation of nursing terminologies. J Am Med Inform Assoc, 8(3):212–221.

[HR04] Holt, C.A., and Roth, A.E. (2004) The Nash equilibrium: A perspective. PNAS, 101(12): 3999–4002.

[HS98] Hirst, G., and St-Onge, D. Lexical chains as representations of context for the detection and correction of malapropisms," In: Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA, 1998.

[HS04] Haase, P., and Sure, Y. (2004) D3.1.1.b State-of-the-Art on Ontology Evolution. Institute AIFB, University of Karlsruhe, Deliverable D3.1.1.b, EU-IST Project IST-SEKT. http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/SEKT-D3.1.1.b.pdf

[HS05] Haase, P., and Stojanovic, L. (2005) Consistent evolution of OWL ontologies. In Proc. 2nd European Semantic Web Conference (ESWC'05), LNCS 3532, Springer, pp. 398–412.

[HTI90] Hirakawa, M., Tanaka, M. and Ichikawa, T. (1990) An iconic programming system, HI–VISUAL, IEEE Transactions on Software Engineering, 16(10):1178–1184.

[HVD02] Heflin, J., Volz, R., and Dale, J. (2002) Requirements for a Web Ontology Language. http://www.w3.org/TR/2002/WD-webont-req-20020307/

[HW95] Heckel, R., and Wagner, A. (1995) Ensuring Consistency of Conditional Graph Grammars - A Constructive Approach. Electronic Notes in Theoretical Computer Science, 2:118–126.

[HW80] Halford, G.S., and Wilson, W.H. (1980) A category theory approach to cognitive development. Cognitive Psychology, 12(3):356–411.

[IB08] Ingenerf, J., and Beisiegel, T. (2008) A version management system for SNOMED CT. Stud Health Technol Inform., 136:827–32.

[IBM] What is ontology? IBM: http://www.alphaworks.ibm.com/contentnr/semanticsfaqs

[IBMH] Healthcare 2015: Win-win or lose-lose? A portrait and a path to successful transformation, IBM. http://www-03.ibm.com/industries/healthcare/us/detail/landing/G883986O04888I88.html

[IEEE98] IEEE 1998, IEEE STD 1219. IEEE standard for software maintenance. IEEE Standard collection: http://standards.ieee.org/reading/ieee/std_public/description/se/1219–1998_desc.html

[Jac99] Jacobs, B. (1999) Categorical Logic and Type Theory. North-Holland, Elsevier, Amsterdam.

[Jao06] Jao (2006) Programmers go bananas, March 17. Available at: http://programming-musings.org/2006/03/17/programmers-go-bananas/

[Jar05] Jarrar, M. (2005) Towards Methodological Principles for Ontology Engineering. Ph.D. theis, Faculty of science, Vrije Universiteit Brussel.

[JBA96] Johnson, S.D., Barwise, J., and Allwein, G. (1996) Towards the rigorous use of diagrams in reasoning about hardware, in Allwein, G., and Barwise, J. (editors), Logical Reasoning with Diagrams, Oxford University Press, pp. 201–223.

[JIB07] Jøsang, A., Ismail, R., and Boyd, C. (2007). A survey of trust and reputation systems for online service provision. Decision Support Systems, 43(2):618–644.

[JMY04] Jurisica I, Mylopoulos J, Yu ESK (2004) Ontologies for Knowledge Management: An Information Systems Perspective. Knowl Inf Syst 6(4): 380–401.

[JPN+98] Jennings, N.R., Parsons, S., Noriega, P., and Sierra, C. (1998) On argumentation-based negotiation. In Proc. of Intl. Workshop on Multi-Agent Systems (IWMAS98), Boston, USA.

[JNP09] Johnson, M., Naumann, D., Power, J. (2009) Category Theoretic Models of Data Refinement. Electronic Notes in Theoretical Computer Science, 225:21–38.

[JPV+98] Jannink, J. and Pichai, S. and Verheijen, D. and Wiederhold, G. (1998) Encapsulation and Composition of Ontologies. In: AAAI Workshop on AI and Information Integration, July 27, Madison, WI. http://ilpubs.stanford.edu:8090/309/

[JR01] Johnson, M., and Rosebrugh, R.D. (2001) View Updatability Based on the Models of a Formal Specification. In Proc. of Int'l Sympo. of Formal Methods Europe (FME'01) pp. 534-549.

[JR08] Johnson, M., and Rosebrugh, R. (2008) Ontology engineering, universal algebra, and category theory. (book chapter) http://www.mta.ca/~rrosebru/articles/oeua.pdf

[JS95] Jensen, C.S., and Snodgrass, R.T. (1995) Semantics of Time-Varying Attributes and their Use for Temporal Database Design. In Proc. of OOER'95, Gold Coast, Australia, LNCS 1021, Springer, pp. 366–377.

[JSW98] Jennings, N.R., Sycara, K.P., and Wooldridge, M. (1998) A Roadmap of Agent Research and Development. Autonomous Agents & Multi-Agent Systems 1(1):7–38

[KAF92] Krause, P., Ambler, S., and Fox, J. (1992) The Development of a "Logic of Argumentation". In Proc. of IPMU'92, Palma de Mallorca, Spain, LNCS 682, Springer, pp. 109–118.

[Kai05] Kainen, P.C. (2005) Category Theory and Living Systems. In Proc. of Int'l Conference "Charles Ehresmann : 100 ans", Université de Picardie Jules Verne à Amiens, 7-9 Oct. http://pagesperso-orange.fr/vbm-ehr/ChEh/articles/Kainen.pdf

[Kal06] Kalyanpur, A.(2006) Debugging and Repair of OWL Ontologies. Ph.D. thesis in University of Maryland, USA.

[Kar01] Karlsson, D. (2001) A design and prototype for a decision support system in the field of urinary tractinfections - application of OpenGALEN techniques for indexing medical information. Medinfo., pp. 479–83.

[KBK01] Kreowski, H.J., Busatto, G., and Kuske, S. (2001) GRACE as a unifying approach to graph-transformation-based specification. Electr. Notes Theor. Comput. Sci. 44(4):1–15.

[KC98] Kini, A., and Choobineh, J. (1998). Trust in Electronic Commerce: Definition and Theoretical Consideration. In Proc. of the 31st Intl. Conf. on System Sciences, IEEE, pp. 51–61.

[Kem06] Kemerling G (2006) The Origins of Western Thought, Kemerling philosophy page, last update on Dec 20, 2006. (http://www.philosophypages.com/hy/2b.htm#hera)

[Ken04] Kent, R.E. (2004) The IFF Foundation for Ontological Knowledge Organization. Cataloging & Classification, 37(1): 187 – 203.

[KF00] Kam, P.S., Fu, A.W. Discovering temporal patterns for interval-based events. In: Kambayashi, Y., Mohania, M.K.,Tjoa, A.M. (eds.) DaWaK 2000. LNCS 1874, Springer, pp. 317–326.

[KF01] Klein, M.C.A., Fensel, D. (2001) Ontology Versioning for Semantic Web. In Proc. of 13th Intl' Semantic Web Working Workshop (SWWS'01), Stanford.

[KFK+02] Klein MCA, Fensel D, Kiryakov A, Ognyanov D (2002) Ontology Versioning and Change Detection on the Web. In proceedings of EKAW 2002, LNCS: 197–212.

[KHE+05] Krötzsch, M., Hitzler, P., Ehrig, M., and Sure, Y. (2005) Category Theory in Ontology Research: Concrete Gain from an Abstract Approach. Tech. Report, AIFB, Uni. of Karlsruhe.

[KK99] Knirsch, P., and Kreowski, H.J. (1999) A Note on Modeling Agent Systems by Graph Transformation. In Proc. of AGTIVE'99, LNCS 1779, Springer, pp. 79–86.

[KKK06] Kreowski, H.J., Klempien-Hinrichs, R., and Kuske, S. (2006) Some Essentials of Graph Transformation. Ésik, Z., Martín-Vide, C., Mitrana, V. (Eds.) Recent Advances in Formal Languages & App. Studies in Computational Intelligence Vol. 25 Springer, pp. 229–254.

[KKR06] Kozen, D., Kreitz, C., Richter, E. (2006) Automating Proofs in Category Theory. In: Furbach, U., Shankar, N. (eds.) IJCAR'06. LNCS, vol. 4130, Springer, pp. 392–407.

[KL07] Kang, S.H., and Lau, S.K. (2007) Ontology Revision, An Application of Belief Revision Approach. in Sharman, R., Kishore, R., and Ramesh, R. (eds.) Ontologies, A Handbook of Principles, Concepts and Applications in Information Systems. Springer, pp. 297–318.

[Kle01] Klein, M. (2001) Combining and relating ontologies: an analysis of problems and solutions. In: Gomez-Perez, A. et al. (eds.), Workshop on Ontologies & Info. Sharing, IJCAI'01, Seattle, USA.

[Kle04] Klein M (2004) Change Management for Distributed Ontologies. Ph.D thesis, Vrije University.

[KLG+07] Keberle, N., Litvinenko, Y., Gordeyev, Y. and Ermolayev, V. (2007) Ontology evolution analysis with OWLMeT. In Proc. of Int'l Workshop on Ontology Dynamics (IWOD'07), Innsbruck, Austria, pp. 1–12

[KM90] Kramer, J. and Magee, J. (1990) The Evolving Philosophers Problem: Dynamic Change Management. IEEE Transactions on Software Engineering 16(11):1293–1306.

[KM98] Kramer, J., and Magee, J. (1998) Analysing dynamic change in software architectures: a case study. In Proc. of the 4th int'l Conference on Configurable Distributed Systems, pp.91–100.

[KN03] Klein MCA, Noy NF (2003) A Component-Based Framework for Ontology Evolution. In Proceedings of the IJCAI-03, CEUR-WS, vol. 71.

[KNG07] Kim, M., Notkin, D., and Grossman, D. (2007) Automatic inference of structural changes for matching across program versions. In Proc. of Int'l 29th IEEE Conf. Software Eng. (ICSE'07), Minneapolis, MN, USA, May 20–26, pp. 333–343.

[KOT+06] Knublauch, H., Oberle, D., Tetlow, P., and Wallace, E. (2006) A Semantic Web Primer for Object-Oriented Software Developers, W3C Working Group Note 9.

[Kos09] Kostakos, V. (2009) Temporal graphs. Physica A 388: 1007–1023.

[KP02] Kreowski, H.J., and Plump, D. (2002) Appligraph: Applications of Graph Transformation. (Final report) available at: http://www.informatik.uni-bremen.de/theorie/appligraph/

[KPS+06] Kalyanpur, A., Parsia, B., Sirin, E., and Grau, B.C. (2006) Repairing Unsatisfiable Concepts in OWL Ontologies. In Proc. of the 3rd European Semantic Web Conf. (ESWC'06), Budva, Montenegro, June 11-14, LNCS 4011 Springer, pp. 170–184.

[KPS+06b] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C., and Hendler, J.A. (2006) Swoop: A Web Ontology Editing Browser. J. Web Sem. 4(2):144-153.

[Kra07] Krasovec, E.D. Limber Labs, 2007. http://www.starlims.com/ICT_Autumn_2007.pdf

[KS98] Kotonya, G., Sommerville, I. (1998) Requirements Engineering: Processes and Techniques. J. Wiley & Sons.

[KS03] Kalfoglou, Y., and Schorlemmer, M. (2003) Ontology mapping: the state of the art. Knowl. Eng. Rev., 18(1):1–31.

[KS03a] Kumar A, Smith B (2003) The Unified Medical Language System and the Gene Ontology, KI2003: Advances in Artificial Intelligence, LNCS 2821:135–48.

[KSS04] Kumar A, Schulze-Kremer S, Smith B (2004) Revising the UMLS Semantic Network. In Proc. of the 11th World Cong. on Medical Informatics MEDINFO'04. IOS Press, 1700–4.

[Lab00] Labov, W. (2000) *Principles of Linguistic change.* Volume II: Social Factors. Oxford: Blackwll.

[LAH07] Leitner, G., Ahlström, D., and Hitz, M. (2007). Usability of Mobile Computing in Emergency Response Systems - Lessons Learned and Future Directions. In Proceedings of the 3rd Symposium of the HCI and Usability for Medicine and Health Care, USAB2007, Graz: Springer, 241–254.

[Lam89] Lambek, J. (1989) On some connections between logic and category theory. Studia Logica, 48(3): 269–278.

[LAS05] Liang Y, Alani H, Shadbolt N (2005) Ontology Change Management in Protege. In Proceedings of the 1st AKT Doctoral Symposium.

[Lee98] Lee, M.L. (1998) Change Impact Analysis of Object-Oriented Software. PhD Thesis, George Mason University.

[Leh96] Lehman, M.M. (1996) Laws of Software Evolution Revisited. In Proc. of the 5th European Workshop on Software Process Technology, LNCS: 1149, pp. 108–124.

[LéJ74] LéJohn, H. B. 1974. Biochemical parameters of fungal phylogenetics. Evol. Biol. 7:79–125.

[LEO06] Lambers, L., Ehrig, H., and Orejas, F. (2006) Efficient Detection of Conflicts in Graph-based Model Transformation. Electr. Notes Theor. Comput. Sci. 152:97–109.

[Leu90] Leung, C.Y. (1990) Antifungal Therapy in Dermatology. Journal of the Hong Kong Medical

Association, 42(4): 203–205.

[Lew47] Lewin, K. (1947). Frontiers in Group Dynamics 1. Human Relations 1, 5–41.

[LG06] Lorence, D.P., and Greenberg, L. (2006) The Zeitgeist of Online Health Search: Implications for a Consumer-Centric Health System. J Gen Intern Med. 21(2): 134–139.

[Lim09] Limit (category theory) (2009, June 8). In Wikipedia, The Free Encyclopedia. Retrieved, Nov 17, 2009, from http://en.wikipedia.org/w/index.php?title=Limit_(category_theory)&oldid=295110702

[Lin01] Lind, J. (2001) Specifying Agent Interaction Protocols with Standard UML. In Proc. of AOSE'01, LNCS 2222, Springer, pp.136–147.

[LLM+06] Liu, H., Lutz, C., Milicic, M., and Wolter, F. (2006) Updating Description Logic ABoxes. In Proc. of the 10th Int'l Conference on Principles of Knowledge Representation and Reasoning (KR'06), Lake District of the United Kingdom, June 2-5, AAAI Press, pp. 46–56.

[LM04] Lammari, N., Métais, E. (2004) Building and maintaining ontologies: a set of algorithms. Data Knowl. Eng. 48(2): 155-176.

[Löw93] Löwe, M. (1993) Algebraic approach to single-pushout graph transformation. Theoretical Computer Science, 109(1–2): 181–224.

[LR94] Lorenzi, N. M., and Riley, R. T. (1994). Organizational Aspects of Health Informatics: Managing Technological Change. New York: Springer-Verlag.

[LR00] Lorenzi, N. M., and Riley, R. T. (2000). Managing Change: An Overview. J Am Med Inform Assoc., 7(2), 116–124.

[LR03.a] Lorenzi, N.M., and Riley, R.T. (2003). Public Health Informatics and Organizational Change. In O'Carroll, P.W., Yasnoff, W.A. et al. (Eds.) Public Health Informatics and Information Systems. London: Springer, 179–198.

[LRW+97] Lehman, M.M., Ramil, J.F., Wernick, P.D., Perry, D.E., and Turski, W.M. (1997) Metrics and Laws of Software Evolution - The Nineties View. In Proc. of the 4th International Symposium on Software Metrics. IEEE Computer Society, pp. 20–32.

[LS81] Lambek, J., and Scott, P.J. (1981) Intuitionist type theory and foundations. Journal of Philosophical Logic, 10(1):101–115.

[LS86] Lambek, J., and Scott, P.J. (1986) Introduction to Higher Order Categorical Logic. Cambridge University Press, Cambridge, UK.

[LS06] Lukasiewicz, T., and Straccia, U. (2006) An overview of uncertainty and vagueness in description logics for the Semantic Web. INFSYS Research report 1843–06–07.

[LS09] Lawvere, F.W., Schanuel, S.H. (2009) Conceptual Mathematics: A First Introduction to Categories. 2nd edition, Cambridge University Press. (The 1st ed. published on 1997)

[LSA+06] Letelier, J.C., Soto-Andrade, J., Abarzúa, F.G., Cornish-Bowden, A., and Cárdenas, M.L. (2006) Organizational invariance and metabolic closure: analysis in terms of (M,R) systems. J Theor Biol. 238(4):949–61.

[LSB+03] Lord, P.W, Stevens, R.D., Brass, A., Goble, C.A. (2003) Semantic similarity measures as tools for exploring the Gene Ontology. Pacific Symposium on Biocomputing, 8:601 -612.

[LSG+04] Lennox, C.L., Serdani, M., Groenewald, J.Z., Crous, P.W. (2004) Prosopidicola mexicana gen. et. sp. nov., causing a new pod disease of Prosopis species. Studies in Mycology 50: 187–94.

[LSM+98] Lucking, R., Serusiaux, E., Maia, L.C., Pereira, E.C.G. (1998) A Revision of the Names of Foliicolous Lichenized Fungi Published by Batista and Co-workers Between 1960 and 1975. The Lichenologist, March 1998, 30(2):121–191.

[LWS+00] Lukoit, S., Wilde, N., Stowell, S., and Hennessey, T. (2000) TraceGraph: Immediate Visual Location of Software Features. In Proc. of International Conference on Software Maintenance (ICSM'00), San Jose, California, USA, pp. 33–39.

[LWY05] Li, L., Wu, B., Yang, Y. (2005) Agent-Based Approach for Dynamic Ontology Management. KES (3), Springer, pp.1–7.

[LX93] Lieberherr, K.J., and Xiao, C. (1993) Object-Oriented Software Evolution. IEEE Transactions on Software Engineering, 19(4): 313 – 343.

[LZ05] Livshits, V.B., and Zimmermann, T. (2005) DynaMine: finding common error patterns by mining software revision histories. In Proc. of ESEC/SIGSOFT FSE 2005, Lisbon, Portugal, Sep. 5-9, 2005. ACM Press, pp. 296–305.

[MA01] McDonald, J., and Anton, J. (2001) SPECWARE - Producing Software Correct by Construction. Kestrel Institute Tech. Rep. KES.U.01.3., March 2001. ftp://ftp.kestrel.edu/pub/papers/specware/specware-jm.pdf

[Mac71] MacLane S (1971) Categories for the Working Mathematician (corrected 1994), Springer.

[Mac79] MacDonald, G.F. (ed.) (1979) I do not Exist, in Perception and Identity, London: Macmillan.

[Mac01] Mack, G. (2001) Universal Dynamics, a Unified Theory of Complex Systems. Emergence, Life and Death. Commun. Math. Phys. 219, 141 – 178.

[Maz07] Mazur, B. (2007) When is one thing equal to some other thing? Available at: (http://www.math.harvard.edu/~mazur/preprints/when_is_one.pdf)

[MAG] MAGE-ML: MicroArray Gene Expression Markup Language: http://xml.coverpages.org/MAGEdescription2.pdf

[Mag99] Magee J (1999) The Problem of change. In the website of Thomistic Philosophy at the Center for Thomistic Studies at the University of St. Thomas, Houston, Texas, last update on 1999. http://www.aquinasonline.com/Topics/change.html. Accessed 10 Jan 2009.

[MAH08] Mutations and Health from Genetics Handbook, Genetic Home Reference. Available: http://ghr.nlm.nih.gov/. June 20, 2008.

[Mäk00] Mäkäräinen, M. (2000). Software change management processes in the development of embedded software. PhD thesis, Espoo: VTT Publications. http://www.vtt.fi/inf/pdf/publications/2000/P416.pdf

[Mam] Mammal Encyclopaedia Article: http://www.naturalresearch.org/Mammal/encyclopedia.htm

[Mat02] University of Virginia, CS201J Course material, Fall 2002, Available: http://www.cs.virginia.edu/cs201j-fall2002/ problem-sets/ps4/

[May83] Mays, E. (1983) A Modal Temporal Logic for Reasoning about Change. In Proc. of 21st Annual Meeting of the Association for Computational Linguistics (ACL) Cambridge, MA, US, pp. 38–43.

[Mcd93] Mcdowall, R.D. (1993) A Matrix for the Development of a Strategic Laboratory Information Management System. Analytical Chemistry, 69(20): 896A–901A.

[MCF81] MacFarlane, A.I.: Dynamic structure theory: A structural approach to social and biological systems. Bulletin of Mathematical Biology. 43(5), 579--591 (1981).

[MD99] Malaiya, Y.K., and Denton, J. (1999) Requirements volatility and defect density. In Proc. of the 10th Int'l Symp. on Software Reliability Engineering, pp. 285–294, Boca Raton, FL, USA.

[MDS00] Meta Data Services Programming (SQL Server 2000). Available at: http://msdn.microsoft.com/en-us/library/aa179133(SQL.80).aspx

[MED+05] Mens, T., van Eetvelde, N., Demeyer, S., and Janssens, D. (2005) Formalizing refactorings with graph transformations. Journal of Software Maintenance 17(4): 247–276.

[Men99] Mens, T. (1999) A Formal Foundation for Object-Oriented Software Evolution. Ph.D. Thesis, Vrije University Brussel.

[Men01] Mens, T. A Formal Foundation for Object-Oriented Software Evolution. in proceedings of the IEEE Intl. Conf. on Software Maintenance (ICSM'01), Florence, Italy, 2001, pp. 549–552.

[Men04] Menendez, D. (2004) category-extras. http://hackage.haskell.org/cgi-bin/hackage-scripts/package/category-extras-0.1

[Men05] Mens, T. (2005) On the Use of Graph Transformations for Model Refactoring. In: Proc. of GTTSE'05, LNCS 4143, Springer, pp.219-257.

[MG06] Mens, T., Gorp, P.V. (2006) A Taxonomy of Model Transformation. Electr. Notes Theor. Comput. Sci. 152: 125–142.

[MGE] Microarray and Gene Expression Data – MGED (http://www.mged.org/index.html)

[MGH+09] Motik, B., Grau, B.C., Horrocks, I., and Sattler, U. (2009) Representing Ontologies using, Description Logics, Desctiption Graphs, and Rules. Artif. Intell. 173(14): 1275-1309.

[MH91] Morris, J., and Hirst, G. (1991) Lexical cohesion computed by thesaural relations as an indicator of the structure of text. J. Computational Linguistics. 17(1) (March 1991), 21–45.

[MIM] Minimum Information about a Microarray Experiment: http://www.mged.org/Workgroups/MIAME/miame.html

[Mit90] Mitchell, T. M. (1990). The need for biases in learning generalizations. In: Shavlik, J.W., and Dietterich, T.G. (eds.) Readings in machine learning. Morgan Kaufmann, pp. 184-191.

[Miz04] Mizoguchi, R. (2004) Tutorial on Ontological Engineering: Part 3: Advanced Course of Ontological Engineering. New Generation Comput. 22(2): 193–220.

[MM] Maddison, D.R., and Maddison, W.P. MacClade: a computer program for phylogenetic analysis. published by Sinauer Associates, Avilable at: http://macclade.org/index.html

[MME+06] Maguitman, A.G., Menczer, F., Erdinc, F., Roinestad, H., and Vespignani, A. (2006) Algorithmic Computation and Approximation of Semantic Similarity. World Wide Web 9(4): 431–456.

[MN95] McCray, A.T., and Nelson, S.J. (1995) The representation of meaning in the UMLS. Method Inform Med 34 (1/2):193–201.

[Mos90] Moser, L.E. (1990) Data Dependency Graphs for Ada Programs. IEEE Transactions on Software Engineering, 16(5): 498–509.

[MRC06] McLaughlin, D., Rinard, P., and Cassutt, M. (2006) Discovery about evolution of fungi has implications for humans. University of Minnesota, 20 Oct, 2006.

[MRM+03] Martin, R.F., Rickard, K., Mejino, J.L.V., Agoncillo, A.V., Brinkley, J.F., Rosse, C. The Evolving Neuroanatomical Component of the Foundational Model of Anatomy. In Proc. of American Med. Info. Assoc. Fall Symp. 2003, p. 927.

[MS92] Monk, S.R., and Sommerville, I. (1992) A Model for Versioning of Classes in Object-Oriented Databases. In Proc. of the 10th British National Conference on Databases (BNCOD'92), Aberdeen, Scotland, LNCS 618, Springer, pp. 42–58.

[MS94] Mullet, K., and Sano, D. (1994) Designing Visual Interfaces: Communication Oriented Techniques, Prentice Hall.

[MS03] Maedche A, Staab S (2003) KAON-The Karlsruhe Ontology and Semantic Web Meta Project. Künstliche Intelligenz (KI) 17(3): 27–30.

[MV01] Maedche A, Volz R (2001) The ontology extraction and maintenance framework text-to-onto. In Proc. of the ICDM'01 Workshop on Integrating Data Mining and Knowledge Management, San Jose, CA, USA.

[MV08] Mason, O., and Verwoerd, M. (2008) Graph Theory and Networks in Biology. arXiv:q-bio/0604006v1. avilable at : http://arxiv.org/abs/q-bio/0604006v1

[MVM10] Miller, F.P., Vandome, A.F., and McBrewster, J. (2010) Interface (computer science): Interface, Abstraction (computer science), Polymorphism in Object- oriented Programming, Indirection, User Interface. Alphascript Publishing.

[MWD+05] Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hirschfeld, R., and Jazayeri, M. (2005) Challenges in Software Evolution. in Proc of 8th IEEE Intl. Workshop on Principles of Software Evolution (IWPSE'05), Lisbon, Portugal, pp. 13–22.

[Nat] The Nature Journal Glossary: http://www.nature.com/nrg/journal/v3/n11/glossary/nrg929_glossary.html

[NCI06] NCI Metathesaurus User Guide 2.2 (2006) ftp://ftp1.nci.nih.gov/pub/cacore/EVS/NCI_Metathesaurus/NCIMetaphraseUserGuide.pdf

[NCL+06] Noy, N.F., Chugh, A., Liu, W., and Musen, M. (2006) A Framework for Ontology Evolution in Collaborative Environments. In Proc. of the 5th International Semantic Web Conference (ISWC'06), Athens, GA, USA, LNCS, Springer, pp. 544-558.

[NDW+88] Notermans, S., Dufrenne, J., Wijnands, L.M., and Engel, H.W. (1998) Human serum antibodies to extracellular polysaccharides (EPS) of moulds, Journal of Medical Veterinary Mycolology 26: 41–48.

[NHI94] Nikoh, N., Hayase, N., Iwabe, N., Kuma, K., and Miyata, T. (1994) Phylogenetic relationships of the kingdoms Animalia, Plantae and Fungi, inferred from 23 different protein species. Mol. Biol. Evol. 11:762–768.

[NHS00a] NHS Information Authority (2000) The Clinical Terms Version 3 (The Read Codes): Introduction and Overview. Ref# 1999-IA-166 v1.0. (the Accessed to the following url on 10 Jan 2009). http://www.connectingforhealth.nhs.uk/systemsandservices/data/readcodes/docs/chap1.pdf.

[NHS00b] NHS Information Authority (2000) The Clinical Terms Version 3 (The Read Codes): Managing Change: Description Change File. Ref# 1999-IA-173 v1.0.

[NFM00] Noy, N.F., Fergerson, R.W., Musen, M.A. (2000) The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In Proc. of 12th Intl. Conf. on Knowledge Eng. and Management (EKAW00), French Riviera, pp. 17–32.

[Nie93] Nielsen, J. (1993). Iterative user interface design, Computer, 26(11), 32–41.

[NJH01] Nelson, S. J., Johnston, D., and Humphreys, B. L. (2001) Relationships in Medical Subject Headings. In: Bean, Carol A.; Green, Rebecca, editors. Relationships in the organization of knowledge. New York: Kluwer Academic Publishers,171–184.

[NK04] Noy, N.F., and Klein, M.C.A. (2004) Ontology Evolution: Not the Same as Schema Evolution, In: Knowledge and Information Systems, 6(4):428–440.

[NLH+08] Nováček, V., Laera, L., Handschuh, S., and Davis, B. (2008) Infrastructure for dynamic knowledge integration—Automated biomedical ontology extension using textual resources. Journal of Biomedical Informatics, 41(5):816–828.

[NLM94] National Library of Medicine, Medical Subject Headings, Bethesda, MD, 1994.

[NM02] Noy, N.F., and Musen, M.A. (2002) PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions. In Proc. of AAAI/IAAI 2002, Edmonton, Alberta, pp. 744-750.

[NM03] Noy, N.F., and Musen, M.A. (2003) The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. Int J Hum-Comput St 59(6): 983–1024.

[NM04] Noy, N.F., and Musen, M.A. (2004) Ontology versioning in an ontology management framework. IEEE Intelligent Systems 19(4) 6–13.

[Nor88] Norman, D.A. (1988). The Psychology of Everyday Things. London: Basic Books.

[Nov07b] Novacek, V. (2007) KWTR: ontology maintenance. Available at: http://semanticweb.org/wiki/KWTR:_ontology_maintenance

[NR08] Noy, N.F., and Rubin, D.L. (2008) Translating the Foundational Model of Anatomy into OWL. Web Semantics, 6(2):133–136.

[NS08] Nyström, M., and Sundvall, E. (2008) Statistics for SNOMED CT January 2008 International Core. Dept. of Biomedical Engineering, Linköping University. http://www.imt.liu.se/~erisu/2008/04-lund/Snomed-jan2008-size-v3.pdf

[OAD+92] Odds, F.C., Arai, T., Di Salvo, A.C., Evans, E.G.V., Hay, R.J., Randhawa, H.S., Rinaldi, M.G., Walsh, T.J. Nomenclature of fungal diseases, A report from a Sub-Committee of the Intl' Society for Human and Animal Mycology (ISHAM). 1992.

[OET+96] Olson, N.E., Erlbaum, M.S., Tuttle, M.S. et al. (1996) Exploiting the metathesaurus update model. In Proc. of 18th Symp. on Computer App. in Medical Care. Philadelphia: Hanley & Belfus, 902.

[OHE96] Orfali, R., Harkey, D., and Edwards, J. (1996) The Essential Distributed Objects Survival Guide. New York: John Wiley & Sons.

[Oli00] Oliver DE (2000) Change management and synchronization of local and shared versions of a controlled vocabulary, Ph.D. thesis, Stanford University.

[Ols93] Olsen, N.C. (1993) The Software Rush Hour. IEEE Software, 10(5): 29–37.

[Oos02] Van Oosten, J. (2002) Basic Category Theory. Lecture Notes (83 pp). BRICS Lecture Series LS-95-01, last update 2002. http://www.math.uu.nl/people/jvoosten/syllabi/catsmoeder.ps.gz

[OPR95] O'Neil, M.J., Payne, C., Read, J.D. (1995) Read Codes Version 3: A User Led Terminology. Meth Inform Med; (34): 187–921.

[OR95] Odds, F.C., and Rinaldi, M.G. (1995) Nomenclature of fungal diseases. Curr. Top. Med. Mycol. 6:33–46.

[OS00] Oliver, D.E., and Shahar, Y. (2000) Change management of shared and local versions of health-care terminologies. Method Inf Med 39(4/5):278–290.

[Osb03] Osborne. M.J. (2003) Nash Equilibrium: Theory. A chapter in "An Introduction to Game Theory", Oxford University Press. http://www.economics.utoronto.ca/osborne/igt/nash.pdf

[OSM01] Overhage, J.M., Suico, J., and McDonald, C.J. (2001) Electronic laboratory reporting: barriers, solutions and findings. Journal of Public Health Management Practice, 7: 60–66.

[OSS+99] Oliver D, Shahar Y, Shortliffe EH, Musen MA (1999) Representation of change in controlled medical Terminologies. Artif Intell Med 15:53–76.

[OT00] Ogawa, T. and Tanaka, J. (2000) CafePie: A Visual Programming System for CafeOBJ, Cafe: An Approach to Industrial Strength Algebraic Formal Methods, Elsevier, pp. 145–160.

[OT09] Okuno, K., and Takahashi, K. (2009) Argumentation System with Changes of an Agent's Knowledge Base. In Proc. of IJCAI'09, Pasadena, California, USA. available online at: http://ijcai.org/papers09/Papers/IJCAI09-047.pdf

[OWL04] OWL Web Ontology Language Overview. 10 Feb 2004. http://www.w3.org/TR/owl-features/

[Pad08] Padberg, J. (2008) Integration of Categorical Frameworks: Rule-Based Refinement and Hierarchical Composition for Components. Applied Categorical Structures, 16(3): 333–364.

[Pal04] Palacz, W. (2004) Algebraic hierarchical graph transformation. Journal of Computer and System Sciences, 68(3): 497–520.

[Pal08] Palacz, W. (2008) Hierarchical graph transformations with meta-rules. Annales UMCS Informatica AI VIII, 2: 89–96.

[Pav96] Pavlovic, D. (1996) Maps II: Chasing Diagrams in Categorical Proof Theory Logic Jnl IGPL, March 1996; 4: 159 - 194.

[PC04] Paglieri, F., and Castelfranchi, C. (2004) Revising beliefs through arguments:bridging the gap between argumentation and beliefrevision in MAS. In Proc. of ArgMAS'04, LNCS 3366, Springer, pp.78–94.

[PCB+04] Papier, A., Chalmers, R.J.G., Byrnes, J.A. and Goldsmith, L.A. (2004) Framework for improved communication: the Dermatology Lexicon Project. J. of the American Academy of Dermatology, 50 (4): 630–634.

[PCC+93] Paulk, M.C., Curtis, B., Chrissis, Chrissis, M.B., and Weber, C. (1993) Capability Maturity Model for Software, Version 1.1. IEEE Software, 10(4): 18–27.

{PCN+04] Purvis, M., Cranefield, S., Nowostawski, M., and Purvis, M. (2004) Multi-Agent System Interaction Protocols in a Dynamically Changing Environment. An Application Science for Multi-Agent Systems, Springer, pp. 95–111.

[Pel91] Peltason, C. (1991) The BACK System - An Overview. SIGART Bulletin 2(3):114-119.

[Per06] Peruzzi, A. (2006) The Meaning of Category Theory for 21st Century Philosophy. Axiomathes, 16(4): 426–459.

[Pfa04b] Pfalzgraf, J. (2004) On Logical Fiberings and Automated Deduction in Many-valued Logics Using Gröbner Bases. RACSAM, Rev. Real. Acad. Ciencias, Ser. A. Mat., 98(1): 213–227.

[Pfa07a] Pfalzgraf, J. (2007) ACCAT and Theoretical Neurobiology: On a Network Structure Modeling Approach. Talk given at the 2nd ACCAT'07 workshop, at ETAPS-2007, March 24-April 1, Braga, Portugal

[Pfa07b] Pfalzgraf, J. (2007) The Base Diagram of a Multiagent System: A Categorical Model of the General Communication Structure. Talk given at Symposium on Multiagent Systems, Robotics and Cybernetics: Theory and Practice.

[PFF+09] Pesquita, C., Faria, D., Falcão, A.O., Lord, P., and Couto, F.M. (2009) Semantic similarity in biomedical ontologies. PLoS Comput Biol. 2009 Jul;5(7):e1000443. Epub 2009 Jul 31.

[PGM99] Pinto, H.S., Gomez-Perez, A., and Martins, J.P. (1999) Some issues on ontology integration. In Proc. of the Workshop on Ontologies and Problem-Solving Methods at 16th Int'l Joint Conf. on Artificial Intelligence (IJCAI-99).

[PH04] Patel-Schneider, P.F., and Horrocks,I. (eds.) (2004) OWL Web Ontology Language Semantics and Abstract Syntax Section 4, Mapping to RDF Graphs. http://www.w3.org/TR/owl-semantics/mapping.html#transformation

[PR69] Pfaltz, J.L., and Rosenfeld, A. (1969) Web Grammars. In Proc. of the 1st Int'l Joint Conference on AI (IJCAI'69), Washington, DC, W. Kaufmann, pp. 609–620.

[Phy] Phylogenetic systematics, a.k.a. evolutionary trees. The centre for understanding evolution, Berkeley University. http://www.ucmp.berkeley.edu/clad/clad1.html

[Pie91] Pierce, P. (1991) Basic Category Theory for Computer Scientists, MIT Press.

[Pit00] Pitts, A. M. (2000) Categorical Logic. Chapter 2 of Abramsky, S., Gabbay, D.M., and Maibaum, T.S.E. (eds.) Handbook of Logic in Computer Science, Vol 5. Algebraic and Logical Structures, Oxford Uni. Press

[Ple06] Plessers, P. (2006) An Approach to Web-based Ontology Evolution. Ph.D. Thesis, Vrije Universiteit Brussel.

[PM01] Pinto, H.S., and Martins, J.P. (2001) Ontology Integration: How to perform the Process. In Proc. of IJCAI2001's Workshop on Ontology and Information Sharing.

[Poi86] Poigné, A.(1986) Elements of Categorical Reasoning: Products and Coproducts and some other (Co-) Limits. In: Pitt, D.H., Abramsky, S., Poigné, A., Rydeheard, D.E. (eds.) CTCS'85. LNCS, vol. 240, pp. 16-42.

[Poi86b] Poigné, A. (1986) Category theory and logic. In Proc. of CTCS'85, Guildford, UK, LNCS 240. Springer, pp. 103–142.

[PPF+03] Phan, I.Q.H., Pilbout, S.F., Fleischmann, W., and Bairoch, A. (2003) NEWT, a new taxonomy portal, Nucleic Acids Res 31(13):3822–3823.

[Pre01] Pressman, R. (2001) Software Engineering- A Practitioner's Approach. 5th ed. McGraw-Hill.

[PRW07] Pohl, M., Rester, M., and Wiltner, S. (2007). Usability and Transferability of a Visualization Methodology for Medical Data. In Proc. of the 3rd Sympo. of the HCI and Usability for Medicine and Health Care, USAB07, Graz: Springer, 171–184.

[PST04] Pinto, H.S., Staab, S., Tempich, C. (2004) DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolving Engineering of oNTologies. In Proc. 16th Eureopean Conference on Artificial Intelligence, (ECAI'04), Valencia, pp. 393–397.

[PT05] Plessers, P., and de Troyer, O. (2005) Ontology change detection using a version log. In Proc. of the 4th Int'l Semantic Web Conference (ISWC'05), Galway, Ireland, LNCS 3729, Springer, pp. 578–592.

[PT07] Pan, J.F., and Thomas, E.D. (2007) Approximating OWL-DL Ontologies. AIn Proc. of AAAI'07, Vancouver, British Columbia, Canada, pp. 1434-1439.

[QY08] Qi, G., and Yang, F. (2008) A Survey of Revision Approaches in Description Logics. In Proc. of 21st Int'l Workshop on Description Logics (DL2008), Dresden, Germany, 353 CEUR-WS.org.

[Rao84] Raoult, J.C. (1984) On Graph Rewriting. Theoretical Computer Science, 32:1–24.

[RBG+97] Rector, A., Bechhofer, S., Goble, C., Horrocks, I., Nowlan., W, and Solomon, W. (1997) The GRAIL concept modeling language for medical terminology. Artificial Intelligence in Medicine, 9(2):139–171.

[RCS+97] Robinson D, Comp D, Schulz E, Brown P. et al. (1997) Updating the Read Codes: User-interactive Maintenance of a Dynamic Clinical Vocabulary. J Am Med Inform Assoc 4(6): 465–472.

[RefS06] SNOMED Clinical Terms Reference Sets, July 2006. http://www.ihtsdo.org/fileadmin/user_upload/Docs_01/Technical_Docs/reference_sets.pdf

[Rei70] Reid, G.A. (1970) Epimorphisms and surjectivity. Inventiones Mathematicae, 9(4): 295–307.

[Rei05] Reinhard, D. (2005) Graph Theory. Third edition, Springer.

[RG04] Rector, A.L., and Rogers, J. (2004) Patterns, Properties and Minimizing Commitment: Reconstruction of the GALEN Upper Ontology in OWL. In Proc. of the EKAW04 Workshop on Core Ontologies in Ontology Engineering, Northamptonshire (UK).

[RH96] Resconi, G., Hill, G. (1996) The Language of General Systems Logical Theory: A Categorical View. In Proc. of the Third European Congress on Systems Science, Rome, pages 1091–1096.

[RL04] Resconi, G., and Jain, L.C. Intelligent Agents: theory and applications. Vol. 155 of Studies in Fuzziness and Soft Computing, Springer-Ver. Berlin, 2004.

[RM03] Rosse, C., and Mejino Jr, J.L.V. (2003) A reference ontology for bioinformatics: the Foundational Model of Anatomy. J Biomed Inform 36:478–500.

[RM07] Rahwan, I., McBurney, P.(2007) Guest Editors' Introduction: Argumentation Technology. IEEE Intelligent Systems 22(6): 21–23.

[RN94] Rector, A.L., and Nowlan, W.A. (1994) The GALEN project.Comput Methods Programs Biomed, 45(1-2): 75–8.

[RN02] Russell, S. and Norvig, P. (2002) Artificial Intelligence: A Modern Approach, Prentice Hall, Upper Saddle River, NJ.

[RNK91] Rector, A.L., Nowlan, W.A., and Kay S. (1991) Foundations for an electronic medical record. Methods Inf Med, 30(3):179–86.

[Rob86] Robinson, H. A Key to the Common Errors of Cladistics. Taxon, 1986, 35(2):309–311.

[Rod95] Roddick, J.F. (1995) A Survey of Schema Versioning Issues for Database Systems. Information and Software Technology, 37(7):383–393, 1995.

[Rom99] Romerales, E. (1999) Amounts of Vagueness, Degrees of Truth. Sorites, 11:41–65.

[Ros58] Rosen, R. (1958) The Representation of Biological Systems from the Standpoint of the Theory of Categories, Bulletin of Mathematical Biophysics 20:245–260.

[Ros00] Rosse, C. (2000) Terminologia Anatomica; Considered from the Perspective of Next-Generation Knowledge Sources. Clinical Anatomy 14:120–133.

[Roz97] Rozenberg G. (ed.) (1997) Handbook of Graph Grammars and Computing by Graph Transformations, Vol. 1: Foundations. World Scientific.

[RR05] Rector, A.L., and Rogers, J.E. (2005) Ontological & Practical Issues in using a Description Logic to Represent Medical Concepts: Experience from GALEN. University of Manchester School of Computer Science Preprint CSPP-35.

[RRJ+03] Rahwan, I., Ramchurn, S.D., Jennings, N.R., McBurney, P., Parsons, S., Sonenberg, L. (2003) Argumentation Based Negotiation. Knowledge Engineering Review 18(4): 343–375.

[RRZ+03] Rector, A.L., Rogers, J., Zanstra, P.E., and Van Der Haring, E. (2003) OpenGALEN. OpenGALEN: open source medical terminology and tools. AMIA Annual Symp. Proc. p. 982.

[RS03] Ram, S., and Shankaranarayanan, G. (2003) Research issues in database schema evolution: the road not taken. University of Arizona, Working Paper #2003-15.

[RSS02] Roger, M., Simonet, A., and Simonet, M. (2002) Toward updates in description logics. In Proc. of the 15th Int'l Workshop on Description Logics (DL'02), Toulouse, France, CEUR-WS Vol. 53.

[RW08] Ribeiro, M., and Wassermann, R. (2008) The Ontology Reviser Plug-In for Protégé. In Proc. of 3rd Workshop on ontologies and their applications (WONTO'08), October 26th, Salvador, Bahia, Brazil. http://www.cin.ufpe.br/~wonto2008/wonto2008_arquivos/Wonto08.pdf

[Ryd85] Rydeheard, D.E. (1985) Functors and Natural Transformations. In Proc. of CTCS'85, LNCS 240, Springer, pp. 43–50.

[SA07] Shahaf, D., and Amir, E. (2007) Towards a theory of AI completeness. In Proc. of 8th Int'l Sympo. on Logical Formalizations of Commonsense Reasoning (Commonsense'07), in AAAI Spring Sympo., California, USA.

[Sam91] Samson, R.A. (1991) Problems caused by new approaches in fungal taxonomy. Mycopathologia, 116: 149–150.

[SAS03] Sure Y, Angele J, Staab S (2003) OntoEdit: Multifaceted Inferencing for Ontology Engineering. Journal on Data Semantics (LNCS) (1):128–152.

[Sat] Sattler, U. Description Logic Reasoners: http://www.cs.man.ac.uk/~sattler/reasoners.html

[SBF98] Studer, R., Benjamins, V.R., and Fensel, D. (1998) Knowledge engineering: Principles and methods. Data and Knowledge Engineering, 25(1-2):161–197.

[SBH+05] Shaban-Nejad A., Baker C.J.O., Haarslev V., and Butler G. (2005). The FungalWeb Ontology: Semantic Web Challenges in Bioinformatics and Genomics. In Proc. of the 4th Intl' Semantic Web Conf. (ISWC'05) Nov. 6-10, Galway, Ireland, LNCS Springer, Vol. 3729, pp. 1063–1066.

[SBL+05] Sacchi, L., Bellazzi, R., Larizza, C., Porreca, R., Magni, P.: Learning Rules with Complex Temporal Patterns in Biomedical Domains. In: Miksch, S., Hunter, J., Keravnou, E.T. (eds.) AIME'05. LNCS 3581, pp. 23–32. Springer (2005)

[SC06] Smith B, Ceusters W (2006) HL7 RIM: An Incoherent Standard. Studies in Health Technology and Informatics 124:133–138.

[SCC97] Spackman, K.A., Campbell, K.E., Cote, R.A. (1997) SNOMED RT: A reference terminology for healthcare. in Proc of 1997 AMIA Annual Fall Symposium, pp. 640–644.

[SCE+04] Schomburg I, Chang A, Ebeling C, Gremse M, Heldt C, et al. (2004) BRENDA, the enzyme database: updates and major new developments. Nucleic Acids Res 32(DB issue):431–433.

[Sch89] Schneider, H.J. (1989) Describing distributed systems by categorical graph grammars. in proc of 15th int'l workshop on Graph-Theoretic Concepts in Comp. Sci. (WG'89) Castle Rolduc, The Netherlands, LNCS 411, Springer 121–135.

[Sch90] Schmiedel, A. (1990) Temporal Terminological Logic. AAAI, pp. 640–645.

[Sch91] Schmiedel, A. (1991) Integrating Time into Terminological Logics. Description Logics, pp. 105–108.

[SCH+07] Sioutos, N, de Coronado, S, Haber, M.W., et al. (2007) NCI Thesaurus: a semantic model integrating cancer-related clinical and molecular information. J Biomed Inform. 40(1):30–43.

[Sch08a] Schneider, H.J. (2008) Graph Transformations: An Introduction to the Categorical Approach. (Online bok draft) http://www2.informatik.uni-erlangen.de/EN/staff/schneider/gtbook/index.html

[Sch08c] Schneider, H.J. (2008) Implementing the Categorical Approach to Graph Transformations with Haskell. Book Chapter draft: http://www2.informatik.uni-erlangen.de/Personen/schneide/gtbook/appendix-a.pdf?language=en

[SCN+03] Sobel, J.D., Chaim, W., Nagappan, V., and Leaman, D. (2003) Treatment of vaginitis caused by Candida glabrata: use of topical boric acid and flucytosine. American Journal of Obstetrics and Gynecology, 189(5): 1297–1300.

[Scr99] Scribner, P. (1999) Introduction to Ontological Philosophy. http://www.twow.net/MclOtaI.htm

[SDK+03] Smith, M.J., Dewar, R.G., Kowalczykiewicz, K., and Weiss, D. (2003) Towards Automated Change Propagation; the value of traceability. Technical Report, Heriot Watt University. http://www.macs.hw.ac.uk:8080/techreps/docs/files/HW-MACS-TR-0002.pdf

[Sea72] Searle, J. (1972). Chomsky's Revolution in Linguistics". Harman, Gilbert, (editor), 1974. On Noam Chomsky, Anchor Books, New York.

[Sel05] Selinger, P. (2005) Course notes for MATH 4135/5135: Introduction to Category Theory, FALL 2005. Dalhousie University. (Accessed on 12 February 2010) http://www.mscs.dal.ca/~selinger/4135/handouts/notes-2up.pdf

[SFM99] Swann, E.C., Frieders, E.M., and McLaughlin, D. J. (1999) Microbotryum, Kriegeria, and the changing paradigm in basidiomycete classification. Mycologia 91: 51–66.

[SGB00] Stevens, R., Goble, C.A., Bechhofer, S. (2000) Ontology-based Knowledge Representation for bioinformatics. Briefings in Bioinformatics 1(4): 398–414.

[SH06a] Shaban-Nejad, A., Haarslev, V. (2006) Representation of Changes in Ontology Driven Object Oriented Software using Categories. In the proc. of 5th Int'l Semantic Web Conf., ISWC'06 Workshop on Semantic Web Enabled Software Engineering (SWESE'06), Athens, GA, USA.

[SH06b] Shaban-Nejad, A., and Haarslev, V. (2006) Some Issues in Ontology Change Management. Position paper for Canadian Semantic Web Working Symposium (CSWWS 2006), June 6, 2006, Quebec City, QC, Canada.

[SH07a] Shaban-Nejad A, Haarslev V (2007) Managing Conceptual Revisions in a Temporal Fungi Taxonomy, In Proc. of the 20th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2007), Maribor, Slovenia, pp. 624-632.

[SH07b] Shaban-Nejad A, Haarslev V (2007) Categorical Representation of Evolving Structure of an Ontology for Clinical Fungus. In: Bellazzi, R., Abu-Hanna, A., Hunter, J. (eds.) AIME 2007. LNCS, vol. 4594, Springer, Heidelberg, pp. 277–286.

[SH07c] Shaban-Nejad, A., and Haarslev, V. (2007) Towards a Framework for Requirement Change Management in HealthCare Software Applications. In Proc. of the 22nd Annual ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'07), Montreal, QC, Canada, pp. 807–808.

[SH07d] Shaban-Nejad, A., Haarslev, V. (2007) Simulation of Conceptual Representation of Evolving Medical Vocabularies, Presented at COMPMED (Computer Simulation In Medicine), MAY 16-18, 2007, Montreal, Canada. Published in Simulation in Healthcare: The Journal of the Society for Simulation in Healthcare, ISSN 1559-2332, Volume 2, Issue 2, Summer 2007.

[SH08a] Shaban-Nejad, A., and Haarslev, V. Incremental Biomedical Ontology Change Management through Learning Agents. In Proc. of the 2nd KES Intl. Symposium on Agent and Multi-Agent Systems: Technologies and Applications (KES-AMSTA 08), March 27–28, Incheon, Korea, Springer Volume 4953 / 2008: 526–535.

[SH08b] Shaban-Nejad, A., and Haarslev, V. (2008) Ontology-Inferred Phylogeny Reconstruction for Analyzing the Evolutionary Relationships between Species: Ontological Inference versus Cladistics". In Proc. of 8th IEEE Int'l Conf. on BioInformatics and BioEngineering (BIBE'08), 8-10 Oct, Athens, Greece, pp. 1–7.

[SH08c] Shaban-Nejad, A., and Haarslev, V. (2008) Web-based Dynamic Learning through Lexical Chaining: A Step Forward towards Knowledge-Driven Education". In Proc. of 13 Annual SIGCSE Conf. on Innovation and Technology in Comp Sci. Education, ITiCSE'08, Madrid, Spain, June 30- July 2, pp.375.

[SOK+09] Shaban-Nejad, a., Ormandjieva, O., Kassab, M., and Haarslev, V. (2009) Managing Requirements Volatility in an Ontology-Driven Clinical LIMS Using Category Theory, International Journal of Telemedicine and Applications, vol. 2009, Article ID917826, 14 pages, 2009. (PubMed ID: 19343191).

[SH09] Shaban-Nejad, A., and Haarslev, V. (2009) Bio-medical Ontologies Maintenance and Change Management. in Sidhu, A.S., and Dillon, T.S. (eds.) Biomedical Data and Applications. Studies in Computational Intelligence, vol. 224, Springer, pp.143–168.

[SH10a] Shaban-Nejad, A., and Haarslev, V. (2010) Human Factors in Dynamic E-Health Systems and Digital Libraries. To appear in Pease, W., Cooper, M., Gururajan. R. (eds.) Biomedical Knowledge Management: Infrastructures and Processes for E-Health Systems. IGI Global. Information Science Reference – ISR series.

[SH10b] Shaban-Nejad, A., and Haarslev, V. (2010) Towards Autonomous Management of Changes in Distributed Ontologies, Controlled Vocabularies and Linked Data in Biomedical Domain. The 7th Annual Conference of the MidSouth Computational Biology and Bioinformatics Society (MCBIOS'10), Jonesboro, Arkansas, Feb 19–20.

[Sim61] Simpson, G.G. Principles of Animal Taxonomy. New York: Columbia University Press, 1961.

[Skl00] Sklyar, N. (2001) Survey of existing Bio-ontologies, Technical Report 5/2001, Department of Computer Science, University of Leipzig.

[SK03] Stuckenschmidt, H., and Klein, M.C.A. (2003) Integrity and Change in Modular Ontologies. In Proc. of Proceedings of the 18th Int'l Joint Conference on Artificial Intelligence (IJCAI'03), Acapulco, Mexico, August 9-15, pp. 900–908.

[SLC+07] Sacchi, L., Larizza, C., Combi, C., Bellazzi, R. (2007) Data mining with Temporal Abstractions: learning rules from time series. Data Mining Knowledge Discovery 15(2): 217–247.

[SM01] Stumme, G., and Maedche, A. (2001) Ontology Merging for Federated Ontologies on the Semantic Web. In Proc. of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, USA, August 4-5, CEUR-WS/Vol-47, pp. 91–99.

[SM01a] Stumme, G., and Maedche, A. (2001) FCA-MERGE: Bottom-Up Merging of Ontologies. In Proc. of IJCAI 2001, Seattle, Washington, USA, pp. 225–234.

[SM02] Stojanovic L, Motik B (2002) Ontology Evolution within Ontology Editors. In Proceedings of the International Workshop on Evaluation of Ontology-based Tools (EON'02), CEUR-WS-62.

[Smi03] Smith B (2003) Realism, Concepts and Categories or: how realism can be pragmatically useful for information. Talk in OntoQuery at Copenhagen Business School, May 18-22, 2003.

[Smi03.b] Smith, B. (2003) Ontology. in L. Floridi (ed.), Blackwell Guide to the Philosophy of Computing and Information, Oxford: Blackwell, 2003, 155–166.

[Smi05] Smith, B. (2005) Ontologies in Biomedicine: The Good, The Bad and The Ugly. Talk at "Knowledge based bioinformatics Workshop 2005", Montreal, Canada.

[Smi06] Smith B (2006) From concepts to clinical reality: An essay on the benchmarking of biomedical terminologies. J Biomed Inform 39(3):288–298.

[SMM+02] Stojanovic L, Maedche A, Motik B, Stojanovic N (2002) User-Driven Ontology Evolution Management. In Proceedings of the 13th Intl. Conf. on Knowledge Eng. and Knowledge Management (EKAW02), Siguenza, Spain, LNCS: 285–300.

[SMS+03] Stojanovic L, Maedche A, Stojanovic N, Studer R (2003) Ontology evolution as reconfiguration-design problem solving. In Proceedings of the 2nd Intl. Conf. on Knowledge Capture (K-CAP'03), Sanibel Island, FL, USA, ACM, pp.162–171.

[SN01] Scott, J.A., and Nisse, D. (2001). Software configuration management. In: Guide to Software Engineering Body of Knowledge, Chapter 7. Retrieved March 10, 2009 http://www.swebok.org/stoneman/version_1.00/SWEBOK_w_correct_copyright_web_site_version.pd

[SNS+07] Sabetzadeh, M., Nejati, S., Easterbrook, S. and Chechik, M. (2007) A Relationship-Driven Framework for Model Merging. in Proceedings of the Int'l Workshop on Modeling in Software Engineering. ACM press.

[Sol06] Solomon, A. (2006) Pushout: A Mathematical Model of Architectural Merger. in proc of 6th Int'l Perspectives of Systems Informatics (PSI'06), LNCS 4378, Springer, pp. 389–399.

[Sow00] Sowa, J.F. (2000) Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, CA.

[SP82] Smyth, M.B., and Plotkin, G.D. The Category-Theoretic Solution of Recursive Domain Equations. SIAM J. on Computing (SICOMP), 1982, 11(4):761–783.

[Spa05] Spackman, K.A. (2005) Rates of Change in a Large Clinical Terminology: Three Years Experience with SNOMED Clinical Terms. 2005 AMIA Annu Symp Proc. pp. 714–718.

[SPG+07] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., and Katz, Y. (2007) Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2): 51–53.

[SPL+01] Sunyé, G., Pollet, D., Le Traon, Y. and Jézéquel, J.M. (2001) Refactoring UML Models. in: «UML» 2001 — The Unified Modeling Language. Modeling Languages, Concepts, and Tools. LNCS 2185, Springer, pp. 134-148.

[SR04] Smith, B., and Rosse, C. (2004) The role of foundational relations in the alignment of biomedical ontologies. In Proc. of 11th World Congress on Medical Informatics; MEDINFO'04, IOS Press, 444-448.

[SR06] Seidenberg, J., and Rector, A.L. (2006) Web ontology segmentation: analysis, classification and use. In Proc. of WWW'06, pp. 13–22.

[SRP02] Schorlemmer, M., Robertson, D., and Potter, S. (2002) Automated Support for Composition of Transformational Components in Knowledge Engineering. Technical Report EDI-INF-RR-0137, School of Informatics, The University of Edinburgh. http://eprints.aktors.org/138/

[SSB07] Schulz, S., Suntisrivaraporn, B., and Baader, F. (2007) SNOMED CT's Problem List: Ontologists' and Logicians' Therapy Suggestions. in Proc. of Medinfo 2007 Congress, Studies in Health Technology and Informatics (SHTI-series). IOS Press.

[SSG+03] Stojanovic L, Stojanovic N, Gonzalez J, Studer R (2003) OntoManager - A System for the Usage-Based Ontology Management. In Proceedings of CoopIS/DOA/ ODBASE 2003, Catania, Sicily, Italy, LNCS: 858–875.

[SSW01] Schüßler, A., Schwarzott, D., Walker, C. (2001) A new fungal phylum, the Glomeromycota: phylogeny and evolution, Mycol. Res. 105 (12):1413–1421.

[ST95] Slavoski, L.A., and Tunkel, A.R. (1995) Therapy of fungal meningitis. Clin Neuropharmacol. 18(2): 95–112.

[Sto04] Stojanovic, L. (2004) Methods and Tools for Ontology Evolution. PhD. thesis in University of Karlsruhe

[Str] Straccia, U. fuzzyDL: A DL Reasoner supporting Fuzzy Logic reasoning. http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html

[Str01] Straccia, U. (2001) Reasoning within Fuzzy Description Logics. Artif.Intell.Res, 14:137–166.

[Str05] Straccia, U. (2005) A fuzzy description logic for the Semantic Web. In Sanchez, E., ed.: Capturing Intelligence: Fuzzy Logic and the Semantic Web. Elsevier, pp. 73–90

[SVS04] Santos, G., Villela, K., Schnaider, L., Rocha, A., and Travassos, G. (2004) Building Ontology Based Tools for a Software Development Environment. In Proc. of 6th Intl. Workshop on Advances in Learning Soft. Organizations (LSO'04), LNCS 3096, Banff, Canada, pp. 19–30.

[SWK+02] Sycara, K.P., Widoff, S., Klusch, M., Lu, J. (2002) Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. Autonomous Agents and Multi-Agent Systems 5(2): 173–203.

[SWL+03] Stevens, R., Wroe, C., Lord, P., and Goble, C. (2003) Ontologies in bioinformatics. in Staab, S., and Studer, R. (eds). Handbook on Ontologies in Information Sys., Springer, pp. 635-657.

[Swo] Swofford, D. PAUP (V. 4.0) A tool for inferering and interprtting phylogenetic trees. http://paup.csit.fsu.edu/

[SWS03] Smith B, Williams J, Schulze-Kremer S (2003) The ontology of the gene ontology. In Proc. of AMIA2003 Annual Symposium; 609–13.

[Sym08] Symons, J. (2008) Review of Giandomenico Sica (ed.) What is Category Theory? (DRAFT) Studia Logica 89 (2): 285–289.

[Tae94] Taentzer, G. (1994) Hierarchically Distributed Graph Transformation. In Proc. of the 5th Int'l Workshop on Graph Grammars and Their App. to Comp. Sci. (TAGT'94), Williamsburg, VA, USA, LNCS 1073, Springer, pp. 304-320.

[Tae99] Taentzer, G. (1999) Distributed Graphs and Graph Transformation. Applied Categorical Structures 7(4): 431–462.

[Tae04] Taentzer, G. (2004) AGG: A graph transformation environment for modeling andvalidation of software. In Proc. of Applications of Graph Transformations with Industrial Relevance (AGTIVE'04), LNCS 3062, Springer, pp. 446–453.

[Tax99] Taxonomy, Classification, and the Debate about Cladistics, From an appendix in Shinners & Mahler's Illustrated Flora of North Central Texas; 1999, BRIT & Austin College. http://artemis.austincollege.edu/acad/bio/gdiggs/taxonomy.html

[TF05] Thagard, P., and Toombs, E. (2005) Atoms, Categorization and Conceptual Change. in Cohen, H., and Lefebvre, C. (editors) Handbook of categorization in Cognitive Science. Elsevier, pp. 243–254.

[TFH03] Telea, A., and Frasincar, F., and Houben, G.J. (2003) Visualisation of RDF(S)-based Information. In Proc. of 7th Int'l Conf. on Information Visualization (IV'03), London, UK. IEEE, pp. 294-299.

[TKF+99] Taentzer, G., Fischer, I., Koch, M., and Volle, V. (1999) Distributed Graph Transformation with Application to Visual Design of Distributed Systems, In: Ehrig, H., Kreowski, H.-J. et al. (eds.) Handbook of Graph Grammars and Computing by Graph Transformation, Vol 3: Concurrency and Distribution, World Scientific.

[TGM98] Taentzer, G., Goedicke, M., and Meyer, T. (1998) Dynamic Change Management by Distributed Graph Transformation: Towards Configurable Distributed Systems. In Proc. of 6th Int'l Workshop on Theory and App. of Graph Transformations (TAGT'98), Paderborn, Germany, LNCS 1764, Springer, pp. 179–193.

[TGM99] Taentzer, G., Goedicke, M., and Meyer, T. (1999) Dynamic Accommodation of Change: Automated Architecture Configuration of Distributed Systems. In Proc. of ASE'99, pp. 287–290.

[TM05] Trask, R.L, and Mayblin, B. (2005) Introducing Linguistics. Totem Books, USA.

[TMM+96] Taboada, M., Marín, R., Mira, J., Otero, R. P. (1996). Integrating Medical Expert Systems, Patient Data-Bases and User Interfaces. J. Intell. Inf. Syst., 7(3), 261–285.

[Tom99] Tomassi, P. (1999) Logic, London: Routledge.

[Top07] TopBraid Composer, Getting Started Guide Version 2.0. TopQuadrant, Inc. (2007) July 27th. http://www.topbraidcomposer.com/docs/TBC-Getting-Started-Guide.pdf. Accessed 10 Jan 2009.

[Tot08] Toth, D. (2008) Database Engineering from the Category Theory Viewpoint. In Proc. of Intl. Workshop on DAtabases, TExts, Specifications and Objects, Desna, Czech Republic, April 16-18, 2008. CEUR Workshop Proceedings 330.

[Tou58] Toulmin, S. (1958) The Uses of Argument, Cambridge University Press.

[TRR+00] Trombert-Paviot, B., Rodrigues, J.M., Rogers, J.E., Baud, R., et al. (2000) GALEN: a third generation terminology tool tosupport a multipurpose national coding system for surgical procedures. Int J Med Inform, (58-59):71–85.

[TSB06] Taylor, J.W., Spatafora, J., and Berbee, M. Ascomycota. Sac Fungi. Version 09 Oct. 2006. Avialable: http://tolweb.org/Ascomycota/20521 /2006.10.09) in The Tree of Life Web: Project: http://tolweb.org/

[Tuf90] Tufte, E. R.: 1990, Envisioning Information, Graphics Press.

[Tve77] Tversky, A (1977) Features of similarity. Psychological Review, 84(4):327–352.

[UGM07] Udrea, O., Getoor, L., Miller, R.J. (2007) Leveraging data and structure in ontology integration. ACM SIGMOD Conf. Int'l Conference on Management of Data, Beijing, China, pp. 449–460

[UML2] UML 2 Object Diagrams http://www.agilemodeling.com/artifacts/objectDiagram.htm

[UML3] UML basics: The sequence diagram: http://www.ibm.com/developerworks/rational/library/3101.html

[UML08] UMLS documentation (2008) Accessed 10 Jan 2009. http://www.nlm.nih.gov/research/umls/umlsdoc_intro.html#s1_0.

[Van06] Van Polanen Petel, H.P. (2006) Universal Grammar as a Theory of Notation. Axiomathes, 16(4): 460–485.

[Var05] Varzi, A.C. (2005) Change, Temporal Parts and the Argument from Vagueness. Dialectica 59(4): 485–498.

[VEK+05] Volkel, M., Enguix, C.F., Kruk, S.R., Zhdanova, A.V., Stevens, R., and Sure, Y. (2005) SemVersion–Versioning RDF & Ontologies. EU-IST Network of Excellence (NoE) KWEB Deliverable D2.3.3.v1 (WP2.3)

[Ver08] Verhagen, F.C. (2008) Worldviews and Metaphors in the human-nature relationships: An Ecolinguistic Exploration through the Ages. Language & Ecology, 2(3) http://www.ecoling.net/worldviews_and_metaphors_-_final.pdf

[VG06] Völkel, M., and Groza, T. (2006) SemVersion: An RDF-based ontology versioning system. in Proceedings of IADIS Intl. Conf. on WWW/Internet, vol(1): 195-202.

[VGH96] Van Eemeren, F.H., Grootendorst, R.F., Henkemans, F.S.: Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Applications, L. Erlbaum Associates, NJ, USA (1996).

[VH91] Ventrone, V., and Heiler, S. (1991) Semantic Heterogeneity as a Result of Domain Evolution. SIGMOD Rec (ACM Special Interest Group on Management of Data) 20(4):16–20.

[Viz04] Vizenor, L. (2004) Actions in Health Care Organizations: An Ontological Analysis. In Proc. of Medinfo'04. http://ontology.buffalo.edu/medo/HL7_Vizenor.pdf

[VSC04] Vizenor, L., Smith, B., and Ceusters, W. (2004) Foundation for the Electronic Health Record: An Ontological Analysis of the HL7's Reference Information Model. http://ontology.buffalo.edu/medo/HL7_2004.pdf

[Wak91] Wake, D.B. (1991) Homoplasy: the result of natural selection, or evidence of design limitations? Am Nat, 138: 543–567.

[Wan89] Wand, Y.A. (1989) A Proposal for a Formal Model of Objects in Object-Oriented Concepts, Databases, and Applications. In Kim, W., and Lochovsky, F.(eds.), ACM Press Frontier Serie:537–559.

[Wan06] Wang, J. (2006) Computational Approaches to Linguidtic consensus. Diddertation at University of Illinois at Urbana-Champaign.

[War04] Ware, C. (2004) Information Visualization: Perception for Design, 2nd ed., Morgan Kaufmann.

[Was06] Wasserman, R.: The Problem of Change. Philosophy Compass 1 (2006): 48–57.

[WB05] Warren, W., and Brinkley, J.F. (2005) Knowledge-Based, Interactive, Custom Anatomical Scene Creation for Medical Education: The Biolucida System. AMIA Annu Symp Proc., pp. 789–793.

[WC05] Wurtz, R., Cameron, B.J. (2005) Electronic Laboratory Reporting for the Infectious Diseases Physician and Clinical Microbiologist. Clinical Infectious Diseases, 40(11): 1638–1643.

[WCL+00] Wheeler, D.L., Chappey, C., Lash, A.E., Leipe, D.D., Madden ,T.L. et al. (2000) Database resources of the National Center for Biotechnology Information. Nucleic Acids Res 28(1):10–4.

[WDB] Why Do Biologists Need Cladistics? http://www.Ucmp.berkeley.edu/clad/clad5.html

[WE98] Wiels, V., and Easterbrook, S. (1998) Management of evolving specifications using category theory. In Proc. of 13th IEEE Int'l Conf. on Automated Soft. Eng., Honolulu, USA, pp. 12–21.

[Wei06] Weil, B. (2006) Building an Effective eCRM Strategy in Healthcare. Sep. 21, 2006. http://www.envision-ebusiness.com/piicm.asp?itemid=23&recordid=2&submit=getrecord

[Wel93] Wells, C. (1993) Sketches: Outline with References. http://www.cwru.edu/artsci/math/wells/pub/pdf/sketch.pdf

[WH92] Wilde, N., and Huitt, R. (1992) Maintenance Support for Object-Oriented Programs. IEEE Trans. Software Eng. 18(12): 1038–1044.

[WH99] Williamson, K., and Healy, M. (1999) Industrial applications of software synthesis via category theory. In Proc. of the 14th IEEE Intl. Conf. on Automated Soft. Eng., Oct 1999, Cocoa Beach, FL, USA, pp. 35–43.

[WH00] Williamson, K., and Healy, M. (2000) Deriving engineering software from requirements. Journal of Intelligent Manufacturing, 11(1):3–28.

[WHB07] Wang, Y., Haase, P., and Bao, J. (2007) A Survey of Formalisms for Modular Ontologies. In Int'l Joint Conf. on Artificial Intelligence (IJCAI'07) Workshop SWeCKa. Hyderabad, India. http://www.aifb.uni-karlsruhe.de/WBS/ywa/publications/wang07IJCAIWS.pdf

[Whi97] Whitmire SA (1997) Object Oriented Design Measurement, John Wiley & Sons.

[Whi99] Whitmore I (1999) Terminologia Anatomica: new terminology for the new anatomist. Anat Rec (New Anat.) 257:50–53.

[Wie03] Wiegers, K.E. (2003) Software Requirements. 2nd Edition, Microsoft Press.

[Wil94] Williamson, T. (1994) Vagueness. London: Routledge. (dedicated to Sorits Reasoning).

[WJK00] Wooldridge, M., Jennings, N.R., and Kinny, D. (2000) The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3(3): 285–312.

[WK92] Wu, C.G., and Kimbrough, J.W. Ultrastructural studies of ascosporogenesis in Ascobolus immerses. Mycologia, vol. 84, 1992, pp. 459–466.

[WMC+03] Wong, W.K., Moore, A.W., Cooper, G.F., Wagner, M. (2003) Bayesian Network Anomaly Pattern Detection for Disease Outbreaks. In proc of 20th Int'l Conf. on Machine Learning (ICML'03), Washington, DC, USA, AAAI Press, pp. 808–815.

[Woo09] Wooldridge, M. (2009) An Introduction to MultiAgent Systems. 2nd edition, J. Wiley & Sons.

[Wor99] Wordsworth, J.B. (1999) Getting the best from formal methods. Information and Software Technology, 41(14):1027–1032.

[WWF74] Watzlawick, P., Weakland, J. H., and Fisch, R. (1974). Change: Principles of Problem

Formation and Problem Resolution. New York: Norton.

[XS04] Xing, Z., and Stroulia, E. (2004) Understanding Class Evolution in Object-Oriented Software. In proceedings of IWPC'04, pp. 34-45.

[XS06] Xing, Z., and Stroulia, E. (2006) Understanding the Evolution and Co-evolution of Classes in Object-oriented Systems. Int'l Journal of Software Engineering and Knowledge Engineering 16(1): 23–52.

[Yen91] Yen, J. (1991) Generalizing Term Subsumption Languages to Fuzzy Logic. In Proc. of IJCAI'91, pp. 472–477.

[YTT+05] Yamamoto, M., Tanabe, Y., Takahashi, K., and Hagiya, M. (2005) Abstraction of Graph Transformation Systems by Temporal Logic and Its Verification. In Proc. of VSTTE'05, LNCS 4171, Springer, pp. 518–527.

[Zad65] Zadeh, L.A. (1965) Fuzzy sets, Information and Control, 8: 338–353.

[Zal05] Zalta EN (ed.) (2005) Sorites Paradox. Stanford encyclopedia of philosophy. First published on Jan 17, 1997; substantive revision on Aug 15, 2005.

[Zan02] Zander, R.H. On the Present Revolution. Buffalo Museum of Science Website, June 2002, http://www.mobot.org/plantscience/resbot/Phil/Revolution.htm

[ZGC00] Zare, R., Gams, W., Culham, A. (2000). A revision of Verticillium sect. Prostrata I. Phylogenetic studies using ITS sequences. Nova Hedwigia 71: 465–80.

[ZK05] Zhdanova, A.V., and Keller, U. (2005) Choosing an Ontology Language. In Proc. of WEC'05, Enformatika, pp. 47–50.

[ZKE+06] Zimmermann, A., Krötzsch M, Euzenat, J., and Hitzler, P. (2006) Formalizing Ontology Alignment and its Operations with Category Theory. In Proc. of the 4th Intl. Conf. on Formal Ontology in Info. Sys. (FOIS'06), vol. 150 of Frontiers in AI & App., IOS, pp. 277–288.