9-20-2010 12:00 AM

Ontological View-driven Semantic Integration in Open Environments

Yunjiao Xue, *The University of Western Ontario*

Supervisor: Hamada H. Ghenniwa, *The University of Western Ontario*
Joint Supervisor: Weiming Shen, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering
© Yunjiao Xue 2010

Recommended Citation
Xue, Yunjiao, "Ontological View-driven Semantic Integration in Open Environments" (2010). *Electronic Thesis and Dissertation Repository*. 16.
https://ir.lib.uwo.ca/etd/16

Ontological View-driven Semantic Integration in Open Environments

(Spine title: Ontological View-driven Semantic Integration in Open Environments)

(Thesis format: Monograph)

by

Yunjiao Xue

Graduate Program in Engineering Science

Department of Electrical and Computer Engineering

A thesis submitted in partial fulfilment of the

requirements for the degree of

Doctor of Philosophy

The School of Graduate and Postdoctoral Studies

The University of Western Ontario

London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
SCHOOL OF GRADUATE AND POSTDOCTORAL STUDIES

CERTIFICATE OF EXAMINATION

Supervisor	Examiners
Dr. Hamada H. Ghenniwa	Dr. Yong Zeng
Co-Supervisor	Dr. Robert E. Mercer
Dr. Weiming Shen	Dr. Miriam A. M. Capretz
Supervisory Committee	Dr. Quazi M. Rahman

The thesis by

Yunjiao Xue

entitled:

Ontological View-driven Semantic Integration in Open Environments

is accepted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Date September 20, 1010

Chair of the Thesis Examination Board

Abstract

In an open computing environment, such as the World Wide Web or an enterprise Intranet, various information systems are expected to work together to support information exchange, processing, and integration. However, information systems are usually built by different people, at different times, to fulfil different requirements and goals.

Consequently, in the absence of an architectural framework for information integration geared toward semantic integration, there are widely varying viewpoints and assumptions regarding what is essentially the same subject. Therefore, communication among the components supporting various applications is not possible without at least some translation. This problem, however, is much more than a simple agreement on tags or mappings between roughly equivalent sets of tags in related standards. Industry-wide initiatives and academic studies have shown that complex representation issues can arise.

To deal with these issues, a deep understanding and appropriate treatment of semantic integration is needed. Ontology is an important and widely accepted approach for semantic integration. However, usually there are no explicit ontologies with information systems. Rather, the associated semantics are implied within the supporting information model. It reflects a specific view of the conceptualization that is implicitly defining an ontological view.

This research proposes to adopt ontological views to facilitate semantic integration for information systems in open environments. It proposes a theoretical foundation of ontological views, practical assumptions, and related solutions for research issues. The proposed solutions mainly focus on three aspects: the architecture of a semantic integration enabled environment, ontological view modeling and representation, and semantic equivalence relationship discovery.

The solutions are applied to the collaborative intelligence project for the collaborative promotion / advertisement domain. Various quality aspects of the solutions are evaluated and future directions of the research are discussed.

Keywords

Semantic integration, ontology, ontological view, open environment, frame, modeling, representation, semantic relationship discovery, tree similarity-based, instance-based

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Dr. Hamada H. Ghenniwa for guiding my PhD study at The University of Western Ontario (UWO). In the past four years I have had an amazing experience working with you. Your deep insight into the research problems that we are interested in and unique view on the problems always inspire me and drive me into deeper thinking and analysis. Completely understanding the problem and finding solutions to solve it. This is the most important methodology I have learned from you. Also, thank you for providing me so many opportunities and suggestions in terms of my career after graduation.

Second, I want to thank my co-supervisor, Dr. Weiming Shen, for everything you offered to help me grow. My life was totally changed when you and Dr. Ghenniwa decided to accept me as a student in your research group four years ago. That decision created a bridge for me to come to Canada and start my new research at the UWO. In the following years, you helped me build my research, gain experience from the projects conducted at the National Research Council Canada (NRC), and design my career life as a researcher. I will never forget all the valuable suggestions you gave me. What you and Dr. Ghenniwa have done for me shows me what the best supervisor would be.

Third, I must thank NRC and EK3 for providing me many working opportunities, and thank the UWO and Natural Sciences and Engineering Research Council of Canada (NSERC) for providing financial support for my research. I need to thank all the colleagues in the Cooperative Distributed Systems Engineering (CDS-EnG) group, NRC, and EK3 Technologies Inc. (EK3). I had the pleasure of working with these wise people who inspired my thinking through many discussions and collaborative projects. Here is a list of names that I would like to mention: Chun, Daisy, Abdul, Raafat, Wafa, Mohamed, Sherif, Fujun, Adrian, Mohammad, Abdou and Hamil from CDS-EnG; Qi, Shuying, Brian, Henry, Xiaoping, Jie and Mert from NRC; David, Joe, Kevin, Lingling, Ziyard, Weiping, Shawn, Lihua, Adam, Justin, Steven, Zhan, Joseph, Wajid, Ed, Ken and Nick from EK3.

Last but not least, I thank my parents, Mr. Huimin Xue and Mrs. Jie Zhong. You gave me life at the beginning and support in every moment. Without these there would be nothing that I could achieve. I also want to thank a special girl in my life, Jessie Wang. You cheered me up when I was fighting to complete the thesis. It is so wonderful to have your support in this and all the later stages.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

An information system is the entire combination of infrastructures, organizations, personnel, and software components within a specific boundary for collecting, processing, persisting, transferring, presenting, delivering, and exchanging information. In the past several decades, a great number of information systems have been developed and deployed. More systems are under design or development.

The information systems are usually deployed in an open environment. An open environment is a computing environment where various platforms, technologies, protocols, and standards coexist, and decentralized applications collaborate through interoperability.

Information systems need to connect to and interact with each other to perform advanced tasks. With interactions and interoperations among them, the systems are able to achieve common goals collaboratively, avoiding the necessity of building a super-large system with all the required functions (which will be expensive and infeasible for any organization), and serve humanity better. Therefore, people can view them as one whole system (logically) instead of many separate ones, and can access the complete set of services and multiple underlying information sources through a unified portal, with no need to worry about the effort of finding various service entries and handling various conflicts between them. Such a system is an integration of multiple ones, and such integration requires the systems to understand, communicate, and cooperate with each other. Among these three goals, the most fundamental one is to make the systems understand each other and achieve common agreements on domain concepts and relationships managed by different systems.

However, as many information systems are growing larger, more complex, and more distributed, it becomes increasingly difficult for anyone to effectively organize and work with the information and systems. As a case, semantics-based information integration in various organizations has been hindered by differences in the software applications and by the structural and semantic heterogeneity of the different information sources [De Bruijn, et al., 2003]. In an open environment, the information systems, even within the same domain, are often heterogeneous in terms of their (1) supporting infrastructures (hardware platforms, operating systems, communication facilities, etc); (2) syntactic representations of information; (3) schematic designs of information models, and (4) semantics of information. This is a common problem in many areas such as enterprise application integration where numerous ad-hoc programs have typically been created to perform the integration process. These heterogeneities present major practical and research challenges. This problem has made information retrieval and collaboration among information systems extraordinarily difficult. For example, searching and finding resources and information are becoming particularly challenging tasks. As such, there is an emerging requirement to integrate these information sources and applications to provide consistent services to global users.

There has already been a large body of solutions that address the first three challenges [Sheth, 1998]. The fourth challenge, also known as the *semantic integration* problem [Vetere and Lenzerini, 2005 and Noy, 2004], is an important topic of great interest to this research and one that is increasingly attracting attention within research and industrial communities.

Semantic integration intends to resolve *semantic incompatibility / heterogeneity* among various information systems. The major reason for semantic incompatibility / heterogeneity is the lack of specifications on the semantics of information. As a possible scenario: if you want to find out how many types of the fruit *apple* there are in the world from the Internet, using current search engines (with the search key word "*apple*"), you may only be able to get a set of pages containing the word "*apple*" in their texts. You must create a type list by yourself after reading the returned pages. Imagine that you get millions of returned page links, and actually most of them have no relation to the fruit "*apple*" but just contain that word. Whereas, the pages containing specific names of apples like "*spitzenburg*" (it is also a kind of apple) will be omitted since there is no semantic relationship between "*spitzenburg*" and "*apple*" specified in most of the web pages. More seriously, pages written in another language like French and Chinese will be ignored, which is not acceptable. However, things can become even worse. When we conduct a real search case with Google [1], we cannot easily get those pages about fruit "*apple*" — the most highly ranked pages (those we see in the first few screens) are about the computer company "*Apple Inc.*", not the fruit. As a matter of fact, the search engine does not know what "*apple*" is. It just guesses that maybe the users are more interested in these pages (about the company) based on the historical data it collected from previous search cases.

Another example of the problem concerns the high number of online book-sellers today, with each of them having their own database containing the information about the books it sells. You can search through each seller's website to find the information about the books that you like. If you want to find the best price for one book from several sellers you will need to search one by one and compare the results yourself. Currently, an automatic cross-seller search is not feasible since each seller has a database that is different from others in terms of both structure and content. For example, if the price in seller *A*'s database is called "*Price*" but "*Cost*" in *B*'s database, a regular search engine will never know that they are referring to the same thing without the support of a

[1] http://www.google.com.

semantic relationship between the two items. Therefore, the search engine can neither get results correctly from various databases nor combine them into a unified result set [2].

The two scenarios reveal the importance of information semantics and semantic integration, which are very crucial issues for large-scale information sharing, information retrieval, and information integration in the Internet era. In recent years more and more researchers are focusing on this field. However, there is still a long way to go.

As a category of solutions for the semantic integration problem, schema matching [Rahm and Bernstein, 2001 and Wick, et al., 2008] aims at finding correspondences between schema elements such as database tables and columns. Schema matching can be viewed as the pairing of attributes (or groups of attributes) from the source schema and attributes of the target schema such that the pairs are likely to be semantically related. Schema matches can be discovered by analyzing the similarity of schema information, preservation of constraints, domain knowledge, and instance data. The limitation of this solution is in the lack of a concept model.

Ontology-driven semantic integration is another category of solutions for the semantic integration problem [Hakimpour and Timpf, 2001]. Traditionally, it is based on available ontologies. The *ontology integration* can be applied by discovering semantic correspondences among a set of formal ontologies and (sometimes) creating a more complete ontology [Wache, et al., 2001], given that multiple source ontologies are available. However, in many scenarios, this is not the case. Instead, the "ontologies" are implied in a different format, such as the underlying information representations.

[2] Currently some solutions on cross-seller search have been delivered. The premise is that they already knew the database schemas of different sellers and have finished schema integration (based on semantics) to some extent.

The limitations of the two categories of solutions reveal a gap between the traditional solutions and the actual open environments. New research is required to be conducted to bridge such a gap.

This research is dedicated to understanding the nature of ontologies, semantics, and semantic heterogeneities, to analyzing the research issues, and to building solid theoretical foundations and engineering solutions to address the semantic integration problem in open environments.

The rest of the thesis is organized as follows. Chapter 2 analyzes related work on semantic integration, including some view points from cognitive science, schema-based structural approaches, and ontology-based semantic approaches. Several integration systems are introduced briefly in this chapter. Chapter 3 explores fundamental concepts in terms of information semantics and semantic integration, where various views of semantics and semantic integration are discussed. It also presents the research problems, and practical assumptions as well as the fundamental hypothesis for this research. Chapter 4 discusses the research issues and proposed solutions. Chapter 5 provides the implementation and validation of the results. Chapter 6 concludes the work.

Chapter 2 Literature Review

2.1 Semantic Integration from A Cognitive Science Perspective

2.1.1 Cognitive Science

As a kind of intelligent creature, humans are able to acquire knowledge, perceive and memorize information, reason facts and rules based on obtained knowledge and information, collaborate or compete with each other, analyze situations, make decisions and create solutions for problems, and finally react to the world. Humans are said to be behaving intelligently when they choose courses of action that are relevant to achieving their goals, when they reply coherently and appropriately to questions that are put to them, when they solve complex problems, or when they design or create something useful and novel.

Cognitive science is the study of intelligence and intelligent systems, with particular reference to intelligent behavior as computation [Posner, 1989]. Cognitive science is dedicated to discovering how humans build mental models for the external world, conduct intelligent thinking, and interact with the world. From cognitive science's point of view, the activities of the human mind are highly similar to computations used by modern computers. Many similar mechanisms and patterns in terms of acquiring information and processing information can be identified in both human thinking and the workings of computers. Therefore, it can also provide some support in the research of semantic integration.

One of cognitive science's focuses is how the intelligent behavior of a human being, thinking, is conducted. In cognitive science, two approaches, *reasoning* and *searching*, are most often considered as the pattern of thinking.

On the one hand there is an approach that starts with language and logic and that views thinking as a process of inference or reasoning, usually using a language-like representation. On the other hand, another approach views thinking (especially problem solving and concept attainment) as a process of a heuristic search for problem solutions, generally using representations of the world model [Posner, 1989]. The research on semantics and logic, language acquisition, parsing, reading, and discourse mainly use the language-and-reasoning approach, whereas the research on categories, induction and problem solving largely employ the heuristic-search approach.

2.1.2 Architecture of Cognition

In cognitive science the notion of *architecture* is derived from computer science, where the term stands for the hardware structure that produces a system that can be programmed. The concept of architecture for cognitive science is the appropriate generalization and abstraction of the concept of computer architecture applied to human cognition: the fixed system of mechanisms that underlies and produces cognitive behaviour.

The classical view about cognition assumes that both computers and human minds have at least the following three distinct levels of organization [Posner, 1989]:

(1) *The semantic level* (or *knowledge level*). This level explains why people, or appropriately programmed computers, do certain things by saying that they know and what their goals are, and by showing that these are connected in certain

meaningful or rational ways.

(2) *The symbol level*. The semantic content of knowledge and goals is assumed to be encoded by symbolic expressions. Such structured expressions have parts, each of which also encodes some semantic content. The codes and their structures, as well as the regularities by which they are manipulated, are another level of organization of the system.

(3) *The physical (or biological) level*. For the entire system to run, it has to be realized in some physical form. The structure and the principles by which the physical object functions correspond to the physical or the biological level.

The three-level organization defines the classical computational or cognitive architecture.

Act* [Anderson, 1983] is the first theory of cognitive architecture with sufficient detail and completeness. The following Figure 2-1 gives the basic architecture of Act*.

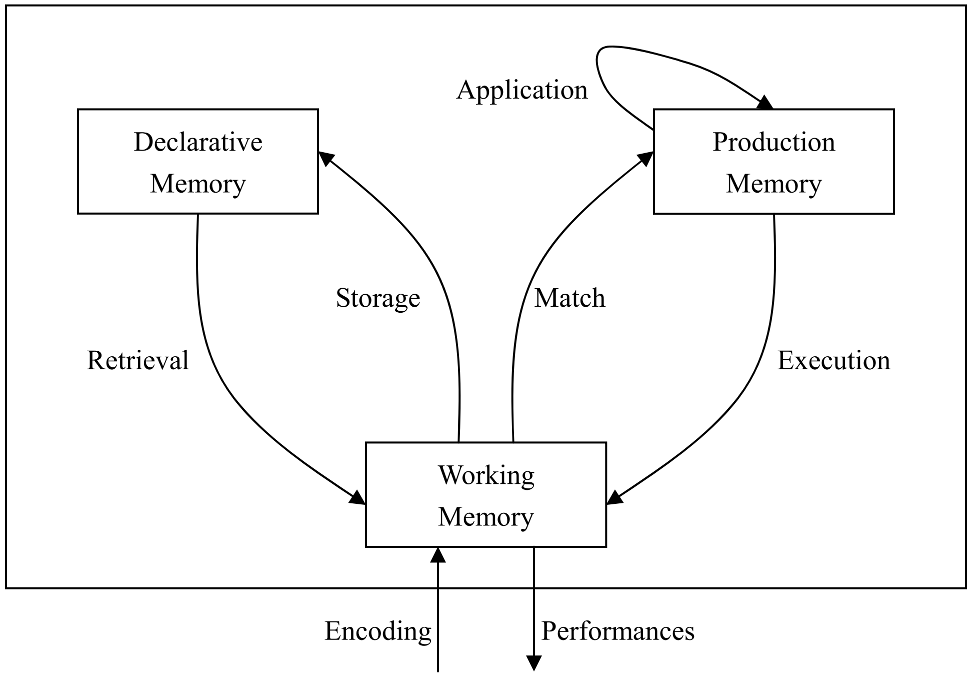


Figure 2-1. Architecture of Act* [Anderson, 1983].

In this architecture there is a long-term declarative memory in the form of a semantic net and a long-term procedural memory in the form of productions. Each production has a set of conditions that test elements of a working memory and a set of actions that create new structures in the working memory. The working memory is activation-based; it contains the activated portion of the declarative memory plus declarative structures generated by production firings and perception. Activation spreads automatically through working memory and from there to other connected nodes in the declarative memory. New productions are created by compiling the effects of a sequence of production firings and retrievals from declarative memory.

2.1.3 From Cognitive Science to Semantic Integration

Generally, semantic integration is intended to discover semantic relationships, such as *equivalent to*, *is-a*, or *part-of*, between some subjects (mainly *concepts*) based on obtained knowledge about the world. A semantic integration system must understand integration requirements and be able to analyze the requirements, develop solutions, and provide reasonable results to the requestor. This is a process very similar to human thinking, which is a significant intelligent behaviour. The cognitive science perspective provides some foundations and inspiration for analyzing, designing, and building a semantic integration system.

According to the classical view of computing and cognition, certain kinds of systems, including both minds and computers, operate on *representations* that take the form of symbolic codes [Posner, 1989].

Similar to the three-level architecture of cognitive, in the semantic integration problem, a three-tier hierarchy in terms of information and knowledge needs to be considered:

- ***Ontology level***. For a domain of discourse, an ontology should be committed to by all information systems to provide a conceptually consistent understanding for any subject in the domain. It is conceptualization-dependent (i.e., different ontologies for different domains' conceptualizations) but technology-independent (not directly manipulated by specific technical method).

- ***Meta-data level***. Meta-data is an explicit specification for information in a specific information system, following definitions about concepts and relationships contained in an ontology to which the system is committed. The semantic integration can be done at this level. That means that the duty of semantic integration is to find semantic relationships between meta-data elements from various information sources. This level is technology-dependent, i.e., specific methods are required to handle different formalisms used to build the meta-data, such as a database, data warehouse, structured documents, or arbitrary files.

- ***Instance-data level***. In some cases specific instance data is required to be compared and analyzed to discover semantic relationships. This level is technology-dependent. For example, the instance data can be represented in a literal, graphical, or analogical format. The semantic integration service must be sensitive to any kind of these representations.

2.1.4 Process Model of Semantic Integration

When two intelligent entities (such as two persons) are having a conversation, semantic integration takes place at every moment during the conversation. The conversation is a process of exchanging conceptualizations of the two intelligent entities and achieving common agreement on the intended meaning of the content exchanged in the conversation.

The conversation must rely on some specific formalisms, such as speeches, written documents, or graphs, which are in fact various representations of (the same) conceptualizations and act as the medium of the conversation. On the other hand, during a conversation, the same representation may be exchanged but the intelligent entities need to identify that they are homonyms and are referring to different concepts.

An elaborated analysis on the process of semantic integration between two intelligent entities E_1 and E_2 is as follows:

(1) *Subject selection*: E_1 determines the subject to express, for example, a concept C in the conceptualization as a part of E_1's mental model).

(2) *Representation schema selection*: E_1 determines the schema of the representation it prefers to use in this conversation, such as verbal speech.

(3) *Representation instance generation*: E_1 generates a representation instance for the subject that is being exchanged following the construction rules of the chosen representation schema, such as a specific word to say.

(4) *Representation instance delivering*: E_1 delivers the representation instance to E_2, e.g., by speaking that word.

(5) *Representation instance perceiving*: E_2 perceives the representation instance delivered by E_1, e.g., by hearing a voice or reading a document, to create some kind of mental reaction in its memory.

(6) *Subject reconstruction*: E_2 converts the perceived representation instance into a subject in its mental model. This is an initial understanding of the representation. For example, if E_2 sees "Car" (which is actually some line shapes) on a piece of paper, first it needs to convert this vision into the word "Car" which then can be identified to be denoting a concept.

(7) *Subject matching*: E_2 tries to match its initial understanding to some existing subject in its mental model to know the actual meaning of the representation instance it perceives.

(8) *Match verifying* (optional): the process from (1) to (7) is repeated (this time E_2 is the initiator) to verify that E_2's subject is equivalent to the one that E_1 wants to deliver.

This process can be illustrated by the following Figure 2-2:

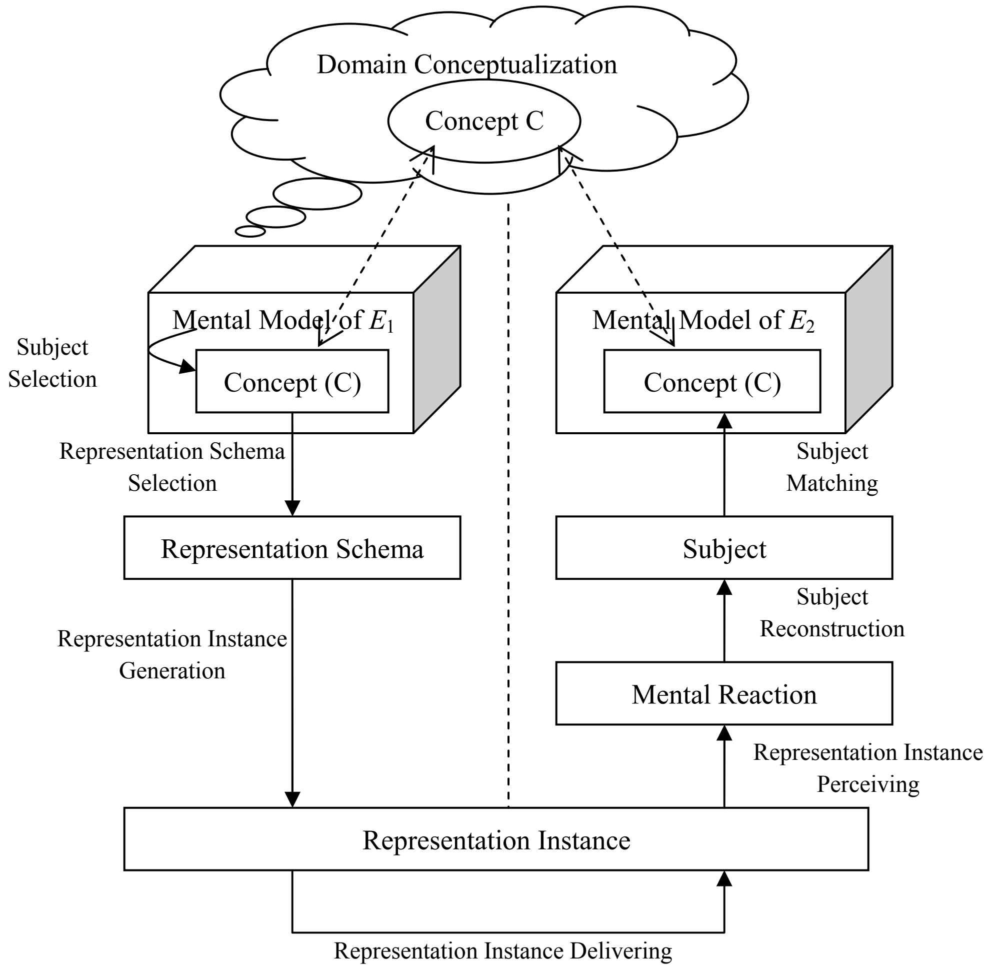


Figure 2-2. Process model of semantic integration.

In this process, there is an important premise that the two intelligent entities share a common concept in a specific domain that makes it possible for them to achieve a successful semantic integration. Without the common concept, there is no possibility of understanding each other unless they have another capability of learning (which will not be touched on in this research), to create new concepts in their mental model.

In step (7), the mechanism that a semantic integration system adopts must combine two ways in which human thinking is executed: searching and then inferring. As an example, when a human perceives some representation, such as a picture, he may search in his mental model to find something that can be exactly mapped to the content of the picture (or highly similar to it). If the searching fails to find any candidate, he will start inferring based on his knowledge. Say, since the content of the picture shows a mechanical object with four wheels, it might be an automobile. The inference is guided by a series of IF … THEN … rules that can lead to possible answers (usually reliable and reasonable, depending on the richness of his knowledge). Neither searching nor inferring can do the integration just by itself.

In step (8), if two sides of the conversation own similar knowledge backgrounds, the semantic integration can be achieved very easily. If their backgrounds are not very similar, or the representation generation and perceiving are not well done, e.g., not clearly hearing the other one's talk, then usually a process similar to the *Three-Way Handshake* in TCP/IP protocol needs to be applied. For example, after one person finishes talking (the first way), another one needs to make sure his understanding is correct by asking "do you mean *A*?" (the second way), and finally the first one answers "Yes, I do mean *A*" to confirm that they have a common understanding (the third way), given that both of them correctly delivered and perceived the representation *A* in the three rounds. If this is not the case, the first one may have to answer that "No, actually I mean *B*" and restart the process of verification.

To support the verification, a set of communication primitives in which both sides have a consistent understanding, e.g., some simple words with precise meanings and which people can clearly say and hear, must be pre-defined. In a semantic integration service that accepts integration requests and responds, even the verification might be missing (since the semantic integration service is the only intelligent entity). As such, a set of primitives that is used to describe the requests and organize responses is also necessary to support establishing a consistent understanding of each other.

In the above process, if E_2 cannot successfully finish the subject matching, e.g., E_2 encounters a word that it never knows, E_2 can interrupt the process. For example, E_2 can answer that “I don’t know what you are talking about”. Another alternative is a passive learning process: if E_2 does not understand the previous representation (r_1), it can ask E_1 to provide another representation, r_2, and repeat the process. Assuming that E_2 can understand r_2, it can create a semantic relationship in its mental model, e.g., “r_1 equals to r_2” or “r_1 is highly similar to r_2”. The third alternative is positive learning: E_2 tries to find a conceptualization that it guesses can match r_1, and starts from (8) to verify its correctness, e.g., by asking E_1 “do you mean r_2?” where r_2 is the representation of the matching subject.

2.2 A General Architecture for Semantic Integration

A general integration architecture for dealing with the heterogeneity of different information sources is described in [Theodoratos, 2002]. This architecture chooses one model as a common data model [Sheth and Larson, 1990] which models global concepts, and converts the modeling languages of the data sources into this model. Underlying information sources are wrapped by software wrappers [Hammer, et al., 1997; Roth and Schwartz, 1997] that translate between the source’s local language, model, concepts and the global concepts. A mediator [Wiederhold, 1992] resolves the query over the global concepts into sub-queries over information sources, sends the sub-queries to wrappers, then integrates the query results returned from the wrappers by resolving conflicts, redundancies, etc. according to application requirements. The architecture is shown in the following Figure 2-3.

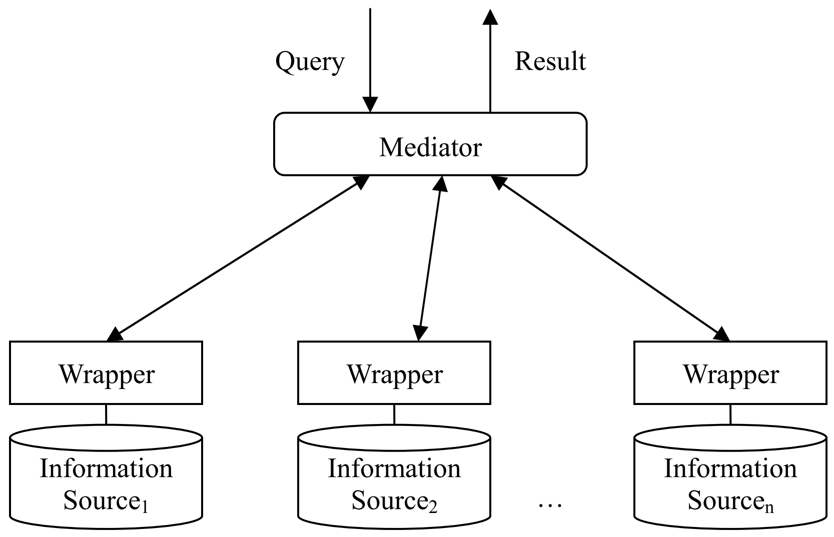


Figure 2-3. A general integration architecture [Theodoratos, 2002].

There are two basic approaches, one is the schema-based structural approach or mediated schema approach, and the other one is the semantic approach or ontology-driven approach. We will discuss them in the following two sections.

2.3 Schema-based Structural Approaches

In structural approaches, the integration is done by providing or generating a globally unified schema that characterizes the underlying information sources. The global schema can be a physically independent one, or a logically produced one (by establishing matching correspondence among source schemas).

2.3.1 Schema Integration Fundamentals

Schema integration is one effective way to achieve data integration. Its target is to develop a unified representation of information structured and stored differently in separate databases. It mainly addresses the problem of syntactic and schematic inconsistencies, e.g., differing structures.

As pointed out in [Mendling, et al., 2005], basically three approaches can be distinguished in this context: *manual*, *semi-automatic*, and *automatic schema integration*. A survey reported in [Batini, et al., 1986] uses the four steps of pre-integration, comparison, conformation, and merging and restructuring to compare different integration methodologies. Manual integration leverages the knowledge of a domain expert. *Semi-automatic schema integration* relies on assertions to state semantic relationships between the concepts of different schemas. These assertions represent integration rules that are used by a so-called integrator to generate the global schema [Spaccapietra, et al., 1992]. Although this approach is less time-consuming, it also depends on a domain expert to state the assertions.

Automatic schema integration uses techniques from information retrieval and artificial intelligence to detect semantic relationships. An overview available in [Rahm and Bernstein, 2001] describes different research prototypes that mainly discover equivalence relationships automatically. Recently, an approach has been presented to automatically discover equivalence, subsumption, intersection, disjointedness, and incompatibility [Rizopoulos, 2004]. In general, a certain trade-off between human effort and the quality of the integrated schema can be expected. In practice, a so-called automated approach still requires validation by the domain expert.

2.3.2 Schema Matching Fundamentals

Schema matching is a basic problem in traditional database-based application domains such as data integration, E-business, data warehousing, and semantic query processing [Rahm and Bernstein, 2001]. *Match* is a fundamental operation in the manipulation of data schemas, which takes two schemas S_1 and S_2 as input and produces a mapping between elements of the two schemas that correspond semantically to each other [Li and Clifton, 1994; Doan, et al., 2000; Mitra, et al., 1999].

[Rahm and Bernstein, 2001] proposes a comprehensive analysis on schema matching. In its analysis, a *schema* is defined as *a set of elements* connected by some *structure*. Representations are required for the schemas. Available and widely accepted representations include the entity-relationship (ER) model, object-oriented (OO) model, XML, or directed graphs. A *mapping* contains a set of *mapping elements*, each of which indicates that certain elements of schema S_1 are mapped to certain elements in S_2. Furthermore, each mapping element can have a *mapping expression* which specifies how the S_1 and S_2 elements are related. The mapping expression may be directional; for example, a certain function from the S_1 elements is referenced by the mapping element to the S_2 elements referenced by the mapping element, or it may be non-directional, that is, a relation between a combination of elements of S_1 and S_2. It may use simple relations over scalars (e.g., =, <), functions (e.g., addition or concatenation), ER-style relationships (e.g., is-a, part-of), set-oriented relationships (e.g., overlaps, contains), or any other terms that are defined in the expression language being used. The *match* operation is defined to be a function that takes two schemas S_1 and S_2 as input and returns a mapping between those two schemas as output, called *match result*. Each mapping element of the match result specifies that certain elements of schema S_1 logically correspond to certain elements of S_2, where the semantics of the correspondence is expressed by the mapping element's mapping expression.

2.3.3 Automatic Schema Matching

Schema matching can be performed manually. However, manually specifying schema matches is tedious, time-consuming, error-prone, and therefore an expensive process [Rahm and Bernstein, 2001], especially when the number of information sources is growing rapidly and the systems are becoming larger and more complex. Therefore, automated support for schema matching is required to provide faster and less labor-intensive integration approaches.

There have been implementations of multiple match algorithms or *matchers* based on different methods. The matchers may consider only schema information, instance data (i.e., data contents), or use hybrid methods.

A. Schema-level approaches

Schema-level matchers only consider schema information, not instance data. The available information includes the usual properties of schema elements [Giunchiglia and Yatskvich, 2004], such as name, description, data type, relationship types (part-of, is-a, etc), constraints, and schema structure (e.g., [Doan, et al., 2001 and Mitra, et al., 1999]). A general implementation compares each S_1 element with each S_2 element and determines a similarity metric in the range (0, 1) for each pair. Only the combinations with a similarity value above a certain threshold are considered as match candidates. The similarity metrics can be used to identify the best match candidates [Castano, et al., 2001 and Doan, et al., 2000]. On the other hand, structural-level matching can discover matching combinations of elements that appear together in a structure.

Linguistic approaches are useful for schema-level matching. Two categories of important approaches, name matching and description matching are discussed in [Rahm and

Bernstein, 2001]. Name matching takes schema elements with equal or similar names into consideration. The similarity of names can be defined and measured in various ways, including:

- Equality of names (the exact same names). An important sub-case is the equality of names from the same XML namespace which ensures that the same names indeed bear the same semantics.

- Equality of canonical name representations after stemming and other preprocessing. This is useful to deal with special prefix/suffix symbols (e.g., CName→customer name and EmpNO→employee number).

- Equality of synonyms. For example, *car* can be matched to *automobile*. General natural language dictionaries and domain-specific dictionaries are useful to deal with synonyms.

- Equality of hypernyms (name of a class's super-class). E.g., *book* **is-a** *publication* and *article* **is-a** *publication* imply that *book* can be matched to *article*.

- Similarity of names based on common substrings; edit distance, pronunciation, soundex (an encoding of names based on how they sound rather than how they are spelled), etc. [Bell and Sethi, 2001]. For example, *representedBy* can be matched to *representative*, *ShipTo* can be matched to *Ship2*, and *Business-to-Business* can be matched to *B2B*.

- User provided name matches, such as *reportsTo* = *manager* and *issue* = *but*.

An exception that is usually misleading is in the case of homonyms which are equal or similar names referring to different concepts. For example, the term "*class*" can have different interpretations in different situations, e.g. a group of students or a lesson of a course. By providing context information such as the domain of discourse, the ambiguity can be distinguished or reduced.

Description matching uses comments and description (usually written in natural language to express the intended semantics of schema structures and elements) provided along with the schemas that can also be evaluated linguistically to determine the similarity between the schema elements. Simple approaches, such as extracting key words from the description and sophisticated technologies, such as natural language understanding, can be applied to look for semantically equivalent elements. For example, the iMAP system pays attention to the description of elements, in addition to other schema information [Dhamankar, et al., 2004].

Another category of the schema matching method adopts constraint information contained in schemas to determine the similarity of schema elements [Larson, et al., 1989]. The constraints include data types, value ranges, uniqueness, optionality, relationship types, cardinalities, repeatability, reference, etc. For example, similarity can be based on the equivalence of data types and domains, of key characteristics (e.g., unique, primary, foreign), or of relationship cardinality (e.g., 1:1 relationships), or of *is-a* relationships.

Rule-based matching techniques constitute another collection of schema matching solutions [Madhavan, et al., 2001 and Melni, et al., 2002]. Rule-based techniques discover similar schema elements by exploiting schema-level information using hand-crafted rules. For example, two elements match if they have the same name and the same number of sub-elements. The rules can exploit all possible information, including element name, data types, structures, number of sub-elements, and integrity constraints.

B. Instance-level approaches

Instance-level data can give important insight into the contents and meaning of schema elements [Rahm and Bernstein, 2001]. When the useful schema information is limited or the schemas are ambiguous, as is often the case for many structured or semi-structured information sources, the analysis on data instances will become very helpful. Even when substantial schema information is available, the use of instance-level matching can also be valuable to uncover incorrect interpretations of schema information. For example, it can help disambiguate between equally plausible schema-level matches by choosing to match the elements whose instances are more similar.

Many approaches in schema-level matching can be applied to instance-level matching. For text elements a *linguistic characterization*, based on information retrieval techniques, is the preferred approach. This approach evaluates the similarity of two schema elements by comparing the relative frequencies of words and combination of words in their data instances. For numerical data type, *statistical characterization*, such as numerical value ranges, averages, or value patterns, can provide insight into the similarity of the corresponding schema elements.

Various approaches have been proposed to perform instance-level matching, such as rules, neural networks, and machine learning techniques [Berlin and Motro, 2001; Doan, et al., 2000; Li and Clifton, 1994; Li, et al., 2000]. Learning-based approaches can exploit data instance-level information. For example, Doan et al. proposed the LSD system, which employs the Naive Bayes learning method over data instances [Doan, et al., 2001]. The Naive Bayes method can easily construct some probabilistic rules based on the analysis of data instances that find similarity between schema elements which names do not reveal enough similarity clues. Note that the learning-based approaches are classified as instance-level approaches, but in fact they can also utilize schema-level information.

C. Hybrid Approaches

Since each matching approach has a specific applicability for a given match task, a matcher that uses just one single approach is unlikely to achieve as many good match candidates as one that combines several approaches [Rahm and Bernstein, 2001]. Therefore, some hybrid approaches are proposed, including two folders: a *hybrid matcher* that integrates multiple matching approaches based on multiple criteria or information sources (e.g., by using name matching with namespaces and thesauri combined with data type compatibility), and *composite matchers* that combine the results of independently executed matchers, including hybrid matchers.

One important issue of note is to the impossibility of determining, fully automatically, all matches between two schemas, primarily because most schemas have some semantics that affect the matching criteria but that are not formally expressed or often even not documented [Rahm and Bernstein, 2001]. Therefore, the result of the match operation is only a set of *match candidates*, which can be accepted, rejected, or modified by the user. Furthermore, the user should be able to specify matches for elements which are meaningful that the system fails to discover.

2.4 Ontology-driven Semantic Approaches

In structural approaches, we also consider the semantics of information schemas, in which the underlying conceptualization is not clearly identified. The focus is that "two (or more) schema elements have the same meaning and they can match". In semantic approaches, semantics is explicitly identified by establishing conceptual models such as ontologies, and the focus is "two (or more) ontology elements refer to the same concept in a common conceptualization (therefore they are semantically identical)", as depicted in the following Figure 2-4.

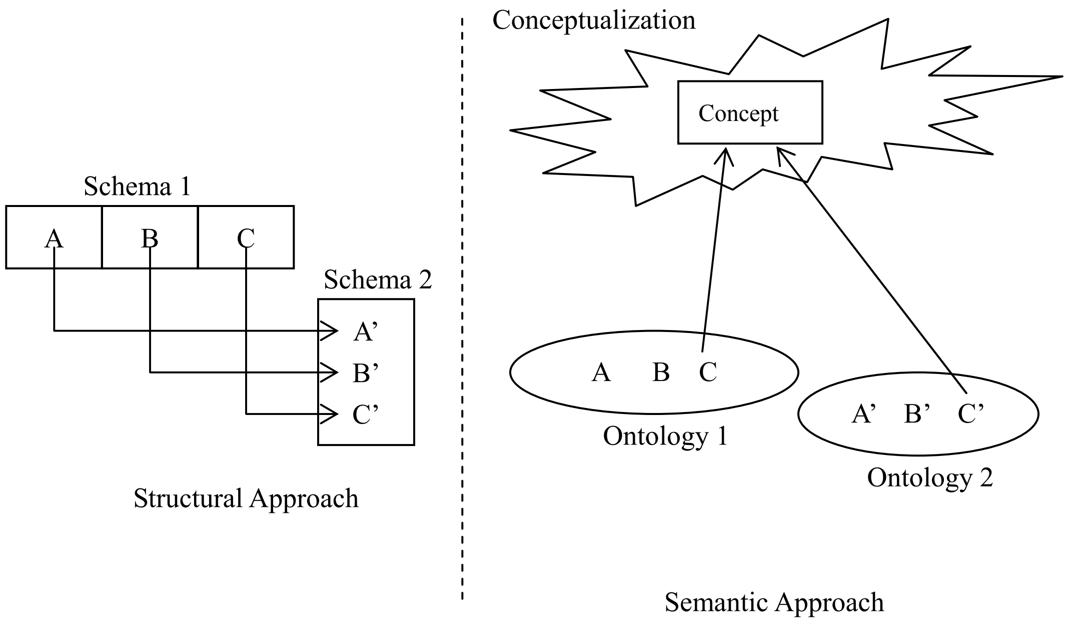


Figure 2-4. Difference of two types of approaches in terms of semantics.

Since it is difficult to integrate the structural aspects of information sources from the semantic perspective due to inherent embedded semantics within local schemas and implicit assumptions, recently ontologies have been introduced to the area of semantic integration as a possible solution to obtain semantic interoperability [Wache, et al., 2001]. In ontology-driven (or ontology-based) approaches, integration is obtained by sharing a common ontology among various information sources, or generating a global ontology that covers the underlying local ontologies of each source. Applying the general integration architecture in this context, the mediator’s job is to integrate ontologies; the wrappers’ job is to translate from the global ontology to local ontologies (if applicable) and then from local ontology to local schema in terms of its conceptual model before the data sources can deal with queries.

2.4.1 Concept of Ontology: An Informal View

Ontologies have been recognized as a fundamental infrastructure for advanced approaches to knowledge management [Arroyo, 2007]. Ontologies are useful for many different applications that can be classified into several areas [Jasper and Ushold, 1999]. The common idea for all of these applications is to use ontologies in order to reach a common understanding of a particular domain [Stuckenschmidt and van Harmelen, 2005], which may be reused and shared across applications and groups [Chandrasekaran, et al., 1999]. The use of ontologies also helps to reach a common understanding of the meaning of terms. In contrast to syntactic standards, the understanding is not restricted to a common representation or a common structure. Therefore, ontologies are a promising candidate that can support semantic interoperability and information retrieval, especially in the Semantic Web [Berners-Lee, et al., 2001].

Many definitions about "ontology" have been proposed. A basic definition about ontology is "the specification of conceptualizations, used to help programs and humans share knowledge" [Gruber, 1993]. An ontology can also be understood as a model that defines the concepts, properties, and relations of a domain of discourse [Crubzy, et al., 2003].

Some people view ontology, in the simplest case, as a hierarchy of concepts related by subsumption relationships, such as things, events, and a set of relations that are specified in some way in order to create an agreed-upon vocabulary for exchanging information. An ontology establishes a joint terminology between members of a community of interest and these members can be human or automated agents. It can be viewed as a semantic substrate for information integration and aggregation processes, providing explicit semantics which may be useful for information exchange between heterogeneous sources.

Ontologies facilitate interoperability between heterogeneous systems involved in a domain of common interest. It is known that any information system uses its own ontology, either implicitly or explicitly [Li, et al., 2005]. As described in [Tan, et al., 2006], ontologies are used for communication between people and organizations by providing a common terminology over a domain. They provide the basis for interoperability between systems. They can be used for making the content in information sources explicit and serve as an index to a repository of information.

Tom Gruber, an AI specialist at Stanford University, proposes a richer definition: "An ontology is a formal, explicit specification of a shared conceptualization" [Gruber, 1995]. Here,

- "explicit" means that "the type of concepts used and the constraints on their use are explicitly defined";
- "formal" refers to the fact that "it should be machine readable";
- "shared" refers to the fact that "the knowledge represented in an ontology is agreed upon and accepted by a group";
- "conceptualization" refers to an abstract model that consists of the relevant concepts and relationships that exist in a certain situation. In another sense, a conceptualization is an abstract, simplified view of the world that we wish to model for some purpose.

The basis of ontology is *Conceptualization*. *Conceptualization* consists of:

- the identified concepts (objects, events, beliefs, etc). e.g. concepts *Professor* and *Course* in education domain;
- the conceptual relationships that are assumed to exist and to be relevant, e.g.

relationship "*Professor* **teach** *Course*".

In the information management and knowledge sharing areas, ontology can be defined as follows [Fisseha, 2003]:

(i) An ontology is a vocabulary of concepts and relations rich enough to enable us to express knowledge and intention without semantic ambiguity.

(ii) An ontology describes domain knowledge and provides an agreed-upon understanding of a domain.

(iii) Ontologies are collections of statements written in a language such as RDF [3] that define the relations between concepts and specify logical rules for reasoning about them.

An ontology can contain not only concepts and relations, but also logical elements that can support reasoning and inferring. A *formal ontology* consists of logical axioms that convey the meaning of terms for a particular community [Bishr, et al., 1999]. A set of logical axioms defining one term is called *intensional definition* and there is only one intensional definition per term for each community [Hakimpour and Geppert, 2002]. Intensional definitions are estimating *intensional relation* (defined in [Guarino, 1998]). For instance, "*Faculty*" is an intensional relation and its estimation by an intensional definition is:

$$\iota[\text{Faculty}(x)] = \text{Employee}(x) \wedge (\exists y: \text{Course}(y) \wedge \text{teaches}(x, y)).$$

Formal ontologies are considered more than schema definitions in databases. Schemas are mainly concerned with organizing data in databases, but formal ontologies are concerned with the understanding of the members of a community and help to reduce ambiguity in communications.

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

The reason that ontologies are becoming so important is that currently we lack standards (shared knowledge) which are rich in semantics and represented in a machine understandable form. Ontologies have been proposed to solve the problems that arise from using different terminology to refer to the same concept or using the same term to refer to different concepts. With the aid of ontologies, semantic queries can exploit conceptual knowledge that is independent of local schemas. By contrast, non-semantic approaches result in queries defined in terms of local structural organization of data, e.g. XQuery [4] and SQL [5]. In this case, the heterogeneity of information sources means that different queries must be written to match multiple schemas.

The ability to exchange information at run time, also known as interoperability, is an important topic. Ontologies are often used as interlinguas for providing interoperability [Uschold and Gruninger, 1996]: they serve as a common format for data interchange. Each system that wants to interoperate with other systems has to transfer its information into this common framework.

Ontologies are expressed in languages that are machine process-able and can be used for reasoning [Noy, 2003]. The expressed artifact is also called an “ontology model”, given that the ontology itself is abstract. In ontology-based approaches, the description of information semantics (local ontologies or conceptual models of information sources) may be represented in ER [6], UML [7], RDF, or other logic models. Many ontology languages have been proposed. Some are based on description logics (DL) [Badder and Sattler, 2001], such as OWL and LOOM [Arens, et al., 1996], and some are frame-based [Brachman and Levesque, 1984], such as F-logic [Kifer, et al., 1995].

[4] http://www.w3.org/TR/xquery/

[5] http://en.wikipedia.org/wiki/SQL

[6] http://en.wikipedia.org/wiki/Entity-relationship_model

[7] http://www.uml.org/

In the definitions presented above the term “semantics” is frequently involved. Basically semantics refers to the intended meaning of something, usually a symbol. In an ontology related community, the term “semantics” has another explanation but is still similar to “intended meaning”: semantics refers to the relationship between words (data) and the world – the things the words (data) describe [Partridge, 2002]. [Partridge, 2002] defines that ontology is about the existence of a set of objects; it also differentiates the fact that an *ontology model* is a model that directly reflects the ontology.

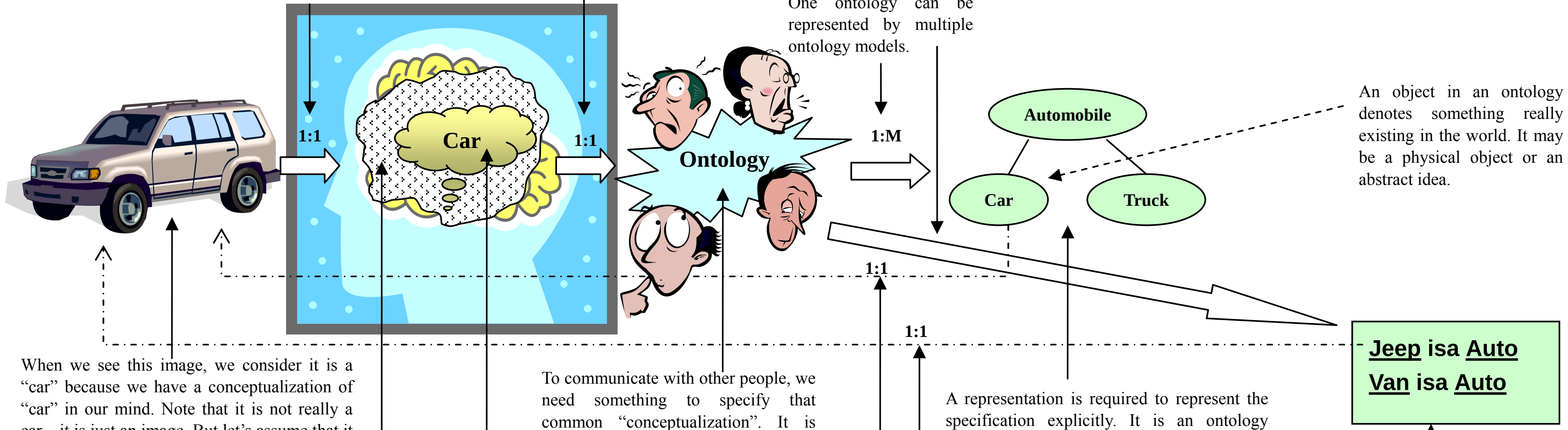
As a summary, the concepts of conceptualization, ontology, model and representation of ontology, semantics, and semantic integration based on ontology are illustrated in the following Figure 2-5.

One real-world object corresponds to one unique conceptualization (in specific domain).

One conceptualization is specified by one unique ontology.

One ontology can be represented by multiple ontology models.

An object in an ontology denotes something really existing in the world. It may be a physical object or an abstract idea.



When we see this image, we consider it is a "car" because we have a conceptualization of "car" in our mind. Note that it is not really a car—it is just an image. But let's assume that it is the real "car". The actual intended meaning of a concept is domain-dependent. For example, sometimes we mention "car" as a specific individual car, and in other cases we may mention it as a category of individual cars. Here, let us assume that it refers to a category of individual cars. As far as the exceptional cases such as someone considering it as a "plane", they exceed the scope of our discussion and will not be touched.

To communicate with other people, we need something to specify that common "conceptualization". It is "ontology". An ontology is a specification of a conceptualization. It is an abstraction among a group of minds.

A representation is required to represent the specification explicitly. It is an ontology model. An ontology model is defined by a kind of formal language to explicitly describe an ontology. Note that when we are talking about "ontology" in practice, it usually refers to an ontology model developed by applying that language.

Conceptualization in one person's mind. Conceptualization is abstract (abstraction of external world in individual mind). We can only understand it but we cannot "see" it. Here, let's assume this graph and this word represent a conceptualization of "car".

If we have another ontology model (in another form of representation), the semantic integration is to discover that the object "Jeep" is also related to the same real world object, thus it is the same as "Car" in the other ontology model.

The semantics relates an object in the conceptualization (represented as an element in an ontology model) to a unique object in the real world. The cardinality of the relationship between the real world object and ontology object is 1:1.

The domain from which the conceptualization derives. That means, the same real world object will lead to different conceptualizations in different domains. Generally speaking, when we are talking about conceptualization, we imply that we are in a specific domain, or a context of the problem we are interested in.

Figure 2-5. Conceptualization, ontology, model and representation of ontology, semantics, and semantic integration based on Ontology.

2.4.2 Ontology-driven Semantic Integration

Regular information retrieval techniques have several shortcomings. First, they rely on the input vocabulary of the user, which might not be completely consistent with the vocabulary used in the information systems. Second, a specific encoding may significantly reduce the recall of a query since related information with a different encoding is not matched. Finally, full-text analysis may reduce precision because the meanings of the words in the texts might be ambiguous.

Traditional integration solutions may result in some significant drawbacks [Hu, et al., 2007]: (1) it is challenging to check the consistency and discover conflicts among domain terminologies; (2) using some traditional ways such as schema matching, the equivalence mappings can be realized but the inheritance mechanism of concepts cannot be implemented; (3) implicit knowledge cannot be discovered without reasoning. Therefore, ontology is often viewed as a key component to realize semantic integration.

The use of ontologies as semantic translators is a viable approach to overcome the problem of semantic heterogeneity [Hakimpour and Timpf, 2001]. Ontologies provide machine-readable semantics of information sources that can be communicated between applications and humans. Using an ontology to explicate the vocabulary can help overcome some of these problems. When used for the description of available information as well as for query formulation, an ontology serves as a common basis for matching queries against potential results on the semantic level. The use of informal ontologies like WordNet [Fellbaum, 1998] increases the recall of a query by including synonyms in the search process. The use of more formal representations like conceptual graphs [Sowa, 1999] further enhances the retrieval process, because a formal representation can be used to increase recall by reasoning about inheritance relationships and precision by matching structures. To summarize, according to

Guarino [Guarino, et al., 1999], ontologies help to decouple the description and query vocabularies and increase precision as well as recall.

In ontology-driven approaches, integration is obtained by sharing common ontologies among the information sources. Mappings are created between the ontologies and local information models. According to the mapping direction, approaches are classified into two categories [Levy, 1999; Levy, 2000; Li and Chang, 2000]: global-as-view [Chawathe, et al., 1994] and local-as-view [Genesereth, et al., 1997].

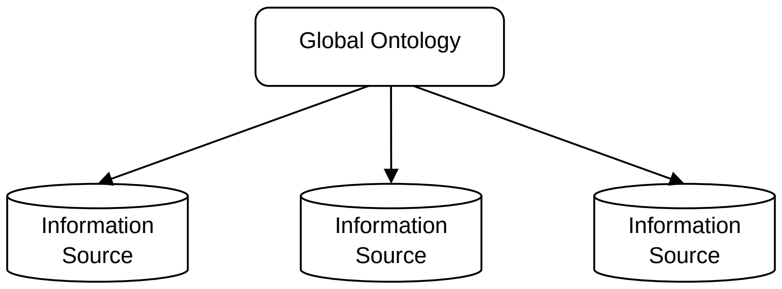
In global-as-view approaches, each item in a global ontology is defined as a view (query) over source schemas/ontologies. It is adopted in most data integration systems. In local-as-view approaches, each item in each source schema/ontology is defined as a view (query) over the global ontology. Many recent research works on data integration follow this approach. The major challenge of this approach is that in order to answer a query expressed over the global schema, one must be able to reformulate the query in terms of queries to the sources. While in the global-as-view approach such a reformulation is guided by the definitions in the mapping; here the problem requires a reasoning step in order to infer how to use the sources for answering the query.

The local-as-view approach better supports a dynamic environment where information sources can be added to the integration system without the need of restructuring the global ontology (given that the new systems are still committed to the global ontology). Hence, the major work on information integration is to develop algorithms for answering queries using these views.

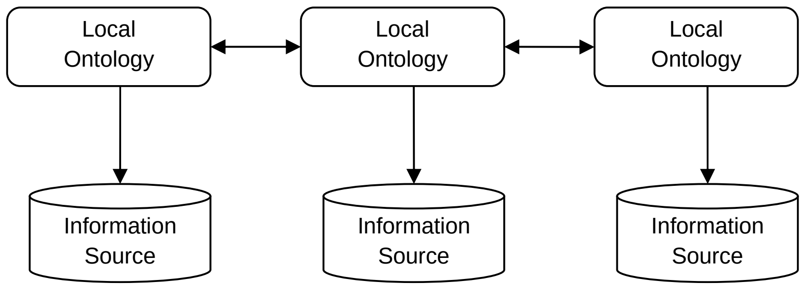
While many systems and approaches use ontologies as an explicit description of the information semantics (i.e., to describe the meaning of information), the role and use of ontologies differs between the approaches. According to the role and use of ontologies, three different categories of approaches can be identified: single-ontology approaches, multiple-ontology approaches, and hybrid approaches [Klein, 2001;

Stuckenschmidt and van Harmelen, 2005; Wache, et al., 2001]. With respect to the role and use of ontology, more than 20 approaches have been developed to support intelligent information integration based on information semantics, including SIMS [Arens, et al., 1993], TSIMMIS [Garcia-Molina, et al., 1995], OBSERVER [Mena, et al., 2000], CARNOT [Collet, et al., 1991], Infosleuth [Nodine, et al., 1999], KRAFT [Preece, et al., 1999], PICSEL [Levy, et al., 1996], DWQ [Calvanese, et al., 1998(2)], Ontobroker [Fensel, et al., 1998], SHOE [Heflin, et al., 1999], MECOTA [Wache, et al., 1999], BUSTER [Visser, 2004], COIN [Goh, 1997]. Some approaches provide a general framework where all three categories can be implemented [Calvanese, et al., 1998(2)].

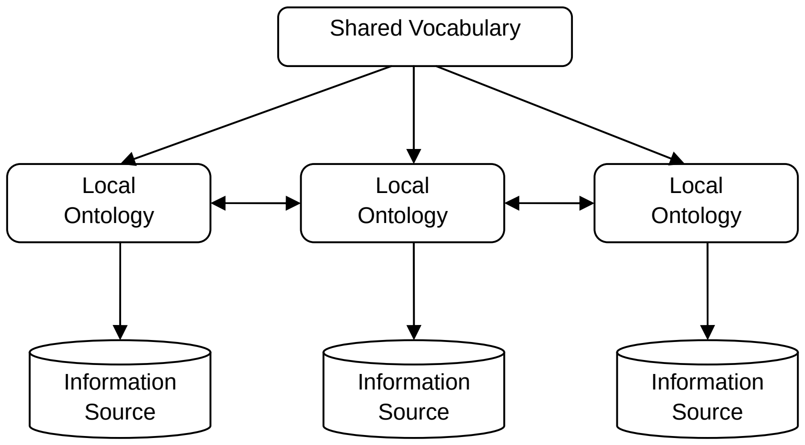
The following Figure 2-6 gives an overview of the three architectures.



(a) Global ontology



(b) Local ontologies



(c) Hybrid approach

Figure 2-6. Different architectures of employing ontologies [Wache, et al., 2001].

A. Single-ontology (Global Ontology) Approaches

Single-ontology approaches use one global ontology that provides a shared vocabulary for the specification of the semantics. All information resources are related to the one global ontology. An independent model of each information source must be described for this system by relating the objects of each source to the global domain model, i.e., elements of the structural information sources are projected onto elements of the ontology. The relationships clarify the semantics of the source objects and help to find semantically corresponding objects.

Single-ontology approaches can be applied to the integration problems where all information sources to be integrated provide nearly the same view of a domain. SIMS and Ontobroker are important representatives of this group. But if one information source has a different view of a domain, e.g., by providing another level of granularity, finding the minimal ontology commitment [Gruber, 1995] becomes a difficult task. Also, single-ontology approaches are susceptible to changes in the information sources, which can affect the conceptualization of the domain represented in the ontology. Depending on the nature of the changes in one information source it can imply changes in the global ontology and in the mappings to the other information sources. These disadvantages lead to the development of multiple-ontology approaches.

B. Multiple-ontology (Local Ontology) Approaches

In multiple-ontology approaches, each information source is described by its own ontology (local ontology). In principle, the local ontology can be a combination of several other ontologies but it cannot be assumed that the different local ontologies share the same vocabulary. OBSERVER is a prominent example of this group, where the semantics of each information source is described by a separate local ontology.

The major advantage of multiple-ontology approaches is that no common and minimal ontology commitment about one global ontology is needed. Each local ontology can be developed without reference to the others. No common ontology with the agreement of all information sources is needed. This ontology architecture can simplify the change, i.e. modifications in one information source or the adding and removing of sources. However, in reality the lack of a common vocabulary makes it extremely difficult to compare different local ontologies. To overcome this problem, an additional representation formalism defining the mapping is provided. The mapping identifies semantically corresponding terms of different local ontologies, e.g. which terms are semantically equal or similar. But the mapping also has to consider different views of a domain, e.g. different aggregation and granularity of the ontology concepts. In practice the mapping is very difficult to define due to the many semantic heterogeneity problems that may occur.

C. Hybrid Approaches

To overcome the drawbacks of the single- or multiple-ontology approaches, hybrid approaches were developed. Similar to multiple-ontology approaches, the semantics of each source is described by its own ontology. But in order to make the source ontologies comparable to each other they are built upon one global shared vocabulary [Goh, 1997 and Wache, et al., 1999]. The shared vocabulary contains the basic terms (the primitives) of a domain. In order to build complex terms of a local ontology the primitives are combined by some operators. Because each term of a local ontology is based on the primitives, the terms become more comparable than in multiple-ontology approaches. Sometimes the shared vocabulary is also an ontology [Stuckenschmidt and Wache, 2000].

In hybrid approaches the major point is how the local ontologies are described. In COIN the local description of information, so called context, is simply an attribute value vector. The terms for the context stems from a global domain ontology and the information itself. In MECOTA, each source concept is annotated by a label which combines the primitive terms from the shared vocabulary. The combination operators are similar to the operators known from the description logics, but are extended, e.g., by an operator which indicates that an information item is an aggregation of several separated information pieces. The BUSTER system uses the shared vocabulary as a (general) ontology, which covers all possible refinements, e.g., the general ontology defines the attribute value ranges of its concepts. A local ontology is one (partial) refinement of the general ontology, e.g., restricts the value range of some attributes.

Because local ontologies only use the vocabulary of the general ontology, they remain comparable.

The use of a shared vocabulary can be viewed as a translation process from a shared vocabulary to each local ontology, therefore one advantage of a hybrid approach is that new sources can easily be added without modification to any other ontology. The use of a shared vocabulary makes the local ontologies comparable and avoids the disadvantages of multiple ontology approaches.

2.4.3 **Ontology Integration**

2.4.3.1 Basic Concept

Ontology integration is an important topic in ontology-based integration approaches. Ontology plays an important role in concept modeling, knowledge representation, and semantics-based information integration. As more and more ontologies are constructed in different domains, the heterogeneity of ontologies becomes another significant issue for information integration. In the following several scenarios ontology integration is required:

(1) Multiple ontologies in one domain are constructed separately but none of them is widely accepted as "standard" ontology for that domain. Each ontology covers different aspects of the domain, although an overlapping portion may exist among them. Therefore, ontology integration is necessary to reuse existing ontologies and build a new ontology which incorporates knowledge (including concepts, properties, individuals, relationships, axioms, functions, etc) dispersing in these ontologies.

(2) Ontologies for different domains exist, and a new ontology for interdisciplinary use is required to be built to incorporate knowledge in these domains.

(3) For some purposes, more than two ontologies are required to be used together. For example, an organization like a company needs to reuse both a public ontology and its own ontology about its business. Moreover, if two companies are merged into one, then their existing ontologies should be merged accordingly to create a new one to eliminate possible semantic conflict. In such cases ontology integration is also necessary.

In these cases, ontology builders may want to use already existing ontologies as the basis for the creation of new ontologies by extending the existing ontologies or by combining knowledge from different ontologies. It is a very complex process as a part of the ontology development lifecycle [Pinto and Martins, 2004]. After ontology integration is done, semantic integration can be supported by the integrated ontology, or by semantic mapping among multiple ontologies.

A thorough review of ontology integration can be found in [Kalfoglou and Schorlemmer, 2003]. Some other research also provides overviews of ontology integration [Calvanese, et al., 2002; Klein and Noy, 2003; Noy, 2004; Wache, et al., 2001].

As there are various definitions on "ontology integration", [Pinto, 1999] proposes three terms to distinguish different meanings: *integration*, *merge*, and *use*. In [Pinto, 1999] *use* means using ontologies in applications, which is not closely related to our topic, therefore we will not discuss it in this research. Another important aspect that is not included in its analysis is *alignment* or *mapping*. The following are descriptions for each term.

A. Alignment/Mapping

Alignment occurs when two or more ontologies are brought into mutual agreement, making them consistent and coherent. That is, to determine semantic relationships between elements from the source ontologies. The semantic relationships may include equivalence, specialization/generalization, or other types of relationships.

Mapping, particularly, is an alignment that relates similar concepts or relations from different source ontologies with overlapping parts to each other by an equivalence relation.

Sowa discussed the concept of alignment in [Sowa, 1997]. According to Sowa, alignment is a mapping of concepts and relations between two ontologies A and B that preserves the partial ordering by subtypes in both A and B. If an alignment maps a concept or relation x in ontology A to a concept or relation y in ontology B, then x and y are said to be equivalent. The mapping may be partial: there could be many concepts in A or B that have no equivalents in the other ontology.

Kalfoglou et al. proposed a formal definition for ontology mapping in [Kalfoglou and Schorlemmer, 2003]. In their definition, an ontology is a pair $O = (S, A)$, where S is the (ontological) signature – describing the vocabulary – and A is a set of (ontological) axioms – specifying the intended interpretation of the vocabulary in some domain of discourse. A total ontology mapping from $O_1 = (S_1, A_1)$ to $O_2 = (S_2, A_2)$ is a morphism f: $S_1 \rightarrow S_2$ of ontological signatures, such that, $A_2 \models f(A_1)$, i.e. all interpretations that satisfy O_2's axioms also satisfy O_1's translated axioms. A partial ontology mapping from $O_1 = (S_1, A_1)$ to $O_2 = (S_2, A_2)$ exists if there exists a sub-ontology $O_1' = (S_1', A_1')$ ($S_1' \subseteq S_1$ and $A_1' \subseteq A_1$) such that there is a total mapping from O_1' to O_2.

B. Merging

The *merging* of ontologies creates a new ontology containing knowledge included in the source ontologies based on the alignment relationships between the ontologies. This operation merges different ontologies about the same subject into a single one that unifies them.

According to Pinto [Pinto, 1999], on one hand in merging we have a set of ontologies (at least two) that are going to be merged (O_1, O_2, …, O_n in Figure 2-7), and on the other hand, the resulting ontology (O in Figure 2-7). The goal is to make a more general ontology about a subject by gathering knowledge from several other ontologies in that same subject into a coherent volume. The subject of both the merged and the resulting ontologies are the same (*S* in Figure 2-7) although some ontologies are more general than others, that is, the level of generality of the merged ontologies may not be the same.

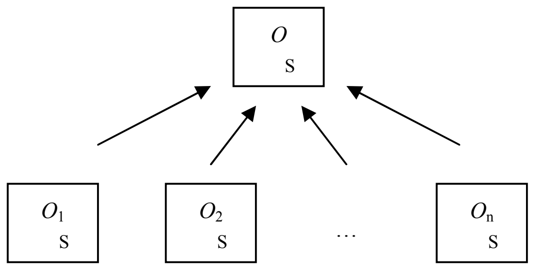


Figure 2-7. Merging of ontologies.

C. Integrating

Integrating ontologies also creates a new ontology by reusing other available ontologies through assembling, extending, or specializing. Different than merging, in integrating the source ontologies and resultant ontology can be in different subjects.

According to Pinto [Pinto, 1999], in integration we have, on one hand, one or more ontologies that are integrated (O_1, O_2, …, O_n in Figure 2-8), and on the other hand, the ontology resulting from the integration process (*O* in Figure 2-8). The domains of the different integrated ontologies are usually different among themselves, that is, each ontology integrated in the resulting ontology is usually about a different domain either from the resulting ontology (*D* in Figure 2-8) or the various ontologies integrated (D_1, D_2, …, D_k, where usually $k = n$, in Figure 2-8). The integrated ontologies are those that are being reused. They are a part of the resulting ontology. The ontology resulting from the integration process is what we want to build and although it is referenced as one ontology it can be composed of several modules. When the integrated ontology is reused by the resulting ontology, the integrated concepts can be (1) used as they are, (2) adapted (or modified), (3) specialized (leading to a more specific ontology on the same domain), (4) augmented by new concepts (either by more general concepts or by concepts at the same level).

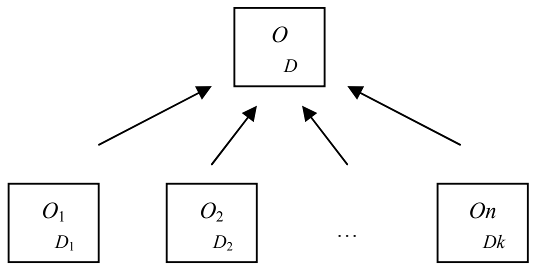


Figure 2-8. Integration of ontologies.

In [Klein, 2001] Klein discusses a fairly complete set of definitions for terms often mentioned in this field. Among these definitions,

- Combining: using two or more different ontologies for a task in which their mutual relation is relevant.
- Merging and integration: creating a new ontology from two or more existing ontologies with overlapping parts, which can be either virtual or physical.
- Articulation: the points of linkage between two aligned ontologies, i.e., the specification of the alignment.
- Translating: changing the representation formalism of an ontology while preserving the semantics.
- Transforming: changing the semantics of an ontology slightly (possibly also changing the representation) to make it suitable for purposes other than the original one.
- Version: the result of a change that may exist next to the original.
- Versioning: a method to keep consistent the relation between newly created ontologies, the existing ones, and the data that conforms to them.

2.4.3.2 Tasks for Ontology Integration

In [Noy, 2003] Noy proposed some specific challenges in ontology integration that must be addressed in the near future:

- Finding similarities and differences between ontologies in an automatic and semi-automatic way;
- Defining mappings between ontologies;
- Developing an ontology integration architecture;
- Composing mappings across different ontologies;
- Representing uncertainty and imprecision in mappings.

They can be viewed as a general architecture of ontology integration tasks. Particularly, in ontology integration, some tasks should be performed to resolve differences and conflicts between ontologies. The tasks lie at two levels.

A. Language Level

1. Syntax

For instance, a concept “Faculty” may be represented as

<rdfs:Class ID=”Faculty”> in RDF schema [8], and

(defconcept Faculty) in LOOM [9].

2. Logical representation

For instance, a rule denoting that two sets have no elements in common can be represented as

[8] http://www.w3.org/TR/rdf-schema/

[9] http://www.isi.edu/isd/LOOM/LOOM-HOME.html

disjoint A B,

or

A subclass-of (not B), B subclass-of (not A)

3. Language expressivity

For instance, some languages can express negation but some others cannot.

B. Ontology Level

1. Conceptualization mismatch

A conceptualization mismatch can cause a difference in the way a domain is interpreted. For example, a difference may exist in scope, meaning that two domains from two ontologies do not contain exactly the same instances.

2. Explication

Explication can cause a difference in the way the conceptualization is specified. For instance, with different modeling paradigms, abstract concepts like time, action, plan, location, etc. may be represented differently. Another case is the difference in modeling intension. For example, in one ontology concept "Circle" is modeled as a sub-concept of "Ellipse" implying that a "round circle" is a special ellipse in which the major axis and minor axis are identical. In another ontology concept "Circle" may be modeled as a super-concept of "Ellipse" implying that an ellipse is a special case of a round shape.

Another case of explication is in terminological mismatch. Terminological mismatch contains two categories:

(1) Synonym terms: different terms specifying the same concept. For example, car vs. automobile, or terms from different languages like English and French with the same meaning.

(2) Homonym terms: the same terms are used for different concepts. The "Circle" example can be viewed as a case of a homonym mismatch. Another example is the term "Conductor" which has a different meaning in music than in the electric engineering domain.

Encoding is another case of explication. For instance, we can have several date formats for a date concept like dd/mm/yy or yyyy-mm-dd, or use a different unit to represent a metric, like miles and kilometres.

2.4.3.3 Ontology Integration Process and Methodology

McGuinness introduces a specification of the integration process in [McGuinness, et al., 2000], where ontology integration consists of (the iteration of) the following steps:

(1) find the places in the ontologies where they overlap;

(2) relate concepts that are semantically close via equivalence and subsumption relationships (aligning);

(3) check the consistency, coherency and non-redundancy of the result.

As pointed out by Noy in [Noy, 2003], it may never be possible to find all alignments / mappings between ontologies completely and automatically since some of the intended semantics can only be discerned by humans. However, ontology integration on a large scale will be possible only if we can make significant progress in identifying mappings automatically or semi-automatically. Methodologies are necessary to guide and support the automatic or semi-automatic ontology integration.

(1) Basic Strategy for Discovering Concept Similarity

The comparison of concept similarity is a fundamental issue for ontology integration. Alignment, mapping, or merging can be possible only if the concepts from different ontologies that have semantic similarity are discovered.

The basic alignment algorithm in ArtGen [Mitra and Wiederhold, 2002] calculates the similarity between concepts based on their names which are seen as lists of words. One method to compute the similarity between a pair of words is based on the similarity between the contexts (1000-character neighbourhoods) of all occurrences of the words in a set of domain-specific Web pages.

In FCA-MERGE [Stumme and Maedche, 2001] the user constructs a merged ontology based on a concept lattice. The concept lattice is derived using a formal concept analysis based on how documents from a given domain-specific corpus are classified to the concepts in the ontologies using natural language processing techniques. OntoMapper [Prasad, et al., 2002] provides an ontology alignment algorithm using Bayesian learning. A set of documents (abstracts of technical papers taken from ACM's digital library and Citeseer) is assigned to each concept in the ontologies. Two raw similarity scores matrices for the ontologies are computed. The similarity between the concepts is calculated based on these two matrices using the Bayesian method.

Some systems implemented alignment algorithms based on the structure of the ontologies. Most of them rely on the existence of previously aligned concepts. For instance, Anchor-PROMPT [Noy and Musen, 2001] determines the similarity of concepts by the frequency of their appearance along the paths between previously aligned concepts. The paths may be composed of any kind of relations. SAMBO [Lambrix and Tan, 2006] provides a component where the similarity between concepts is augmented based on their location in the *is-a* hierarchy relative to already aligned concepts. OntoMapper does not require previously aligned concepts and takes the documents from the sub-concepts into account when computing the similarity between two concepts.

(2) Research on Methodologies

An early methodology for ontology merging in a medical domain is proposed in [Gangemi, et al., 1998]. The methodology to build ontologies presented in [Uschold and King, 1995] includes an integration step. This methodology proposes that integration should be done either during capturing (knowledge acquisition), or coding (implementation) or both. However, the problem is recognized as difficult and no solutions for the problem of how integration is performed are proposed or discussed herein.

The methodology to build ontologies proposed in [Gruninger, 1996] also refers to integration. This methodology mentions two kinds of integration: "combining ontologies that have been designed for the same domain" and "combining ontologies from different domains". According to this methodology, ontologies are built based on ontology building blocks and foundational theories. According to the building blocks and foundational theories of the ontologies being integrated, integration is distinguished as: integration (at the level) of the building blocks - the most simple; integration (at the level) of the foundational theories, which is more difficult and may result in only partial integration; and ontology translation when the ontologies are so different that they share neither the building blocks nor the foundational theories, which makes integration extremely difficult.

METHONTOLOGY [Fernandez, et al., 1997 and Fernandez, et al., 1999] is another methodology to build ontology that also considers integration. It proposes that the development of an ontology should follow an evolving prototyping life cycle and not a waterfall one. This methodology proposes that ontology building, and therefore ontology integration, should be done preferably at the knowledge level (in conceptualization) and not at the symbol level (in formalization, when selecting the representation ontology) or at the implementation level (when the ontology is codified in a target language).

The methodology followed by Skuce to find the ontological distinctions presented in [Skuce, 1997] was by brainstorming, followed by meetings with other researchers interested in the problem. The proposed methodology begins with the creation of a group involving a diverse group of researchers working in different locations. Each member develops a list of primitives, distinctions and categories carefully chosen, defined and carefully documented (choices and definitions). The choices are presented to the group for discussion and approval. Only when they are agreed upon can they get to the formalization stage. The idea is to try to find a standardized upper model that would greatly ease some kinds of integration efforts.

Other methods include: Hovy and colleagues describe a set of heuristics that researchers at ISI/USC used for the semi-automatic alignment of domain ontologies to a large central ontology [Hovy, 1998]. Their techniques are based mainly on the linguistic analysis of concept names and natural-language definitions of concepts. PROMPT uses the structure of ontology definitions and the structure of a graph representing an ontology to suggest to the ontology designer which concepts may be related [Noy and Musen, 2003]. GLUE applies machine-learning techniques to instance data conforming to ontologies to find related concepts [Doan, et al., 2002].

2.4.3.4 Ontology Integration Systems and Tools

Ontology integration is a complicated process. It is difficult to find the terms that need to be aligned, and the consequences of a specific mapping (unforeseen implications) are difficult to see. Semi-automatic tools are required to guide the user through the process and focus this attention on the likely points for action, and enable reusability of alignments in the context of ontology maintenance.

A number of ontology integration systems exists that support users to find inter-ontology relationships. Some of these systems can also perform merging and create a new ontology based on the source ontologies and the alignment relationships. [McGuiness, et al., 2000] provides the first tool to help in the merge process.

(1) A General Framework

Lambrix et al. proposed a general framework for ontology alignment [Lambrix and Tan, 2006], as depicted in the following Figure 2-9. Many ontology alignment systems can be described as instantiations of this framework.

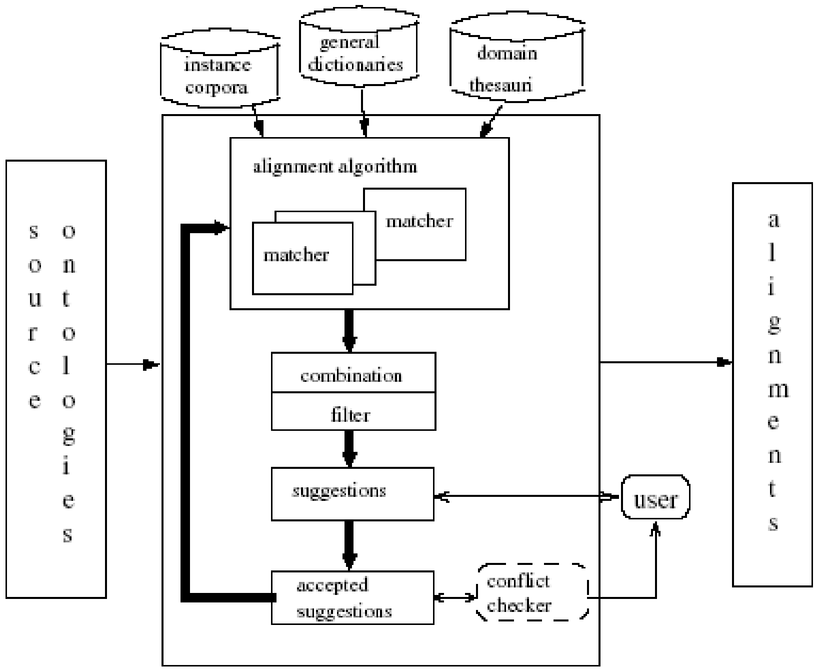


Figure 2-9. A general framework for ontology alignment [Lambrix and Tan, 2006].

In this framework, an alignment algorithm receives two source ontologies as input. The algorithm can include several matchers. These matchers calculate similarities between the terms from the different source ontologies. The matchers can implement strategies based on linguistic matching, structure-based strategies, constraint-based approaches, instance-based strategies, and strategies that use auxiliary information or a combination of these. Alignment suggestions are then determined by combining and filtering the results generated by one or more matchers. The pairs of terms with a similarity value above a certain threshold are retained as alignment suggestions. By using different matchers and combining them and filtering in different ways, different alignment strategies will be obtained. The suggestions are then presented to the user who accepts or rejects them. The acceptance or rejection of a suggestion may influence further suggestions. Further, a conflict checker is used to avoid conflicts introduced by the alignment relationships. The output of the alignment algorithm is a set of alignment relationships between terms from the source ontologies.

In this framework the matchers use different strategies to calculate similarities between the terms from different source ontologies. They use different kinds of knowledge that is exploited during the alignment process to enhance their effectiveness and efficiency. Some of the approaches employed are described as follows:

- Strategies based on linguistic matching. These approaches make use of textual descriptions of the concepts and relations such as names, synonyms and definitions. The similarity measure between concepts is based on comparisons of the textual descriptions.

- Structure-based strategies. These approaches use the structure of the ontologies to provide suggestions. The similarity of concepts is based on their environment. For instance, using the *is-a* relation, an environment can be defined using the parents (or ancestors) and the children (or descendants) of a concept.

- Constraint-based approaches. In this case axioms are used to provide suggestions. For example, knowing that the range and domain of two relations are the same may be an indication that there is a relationship between the relations.

- Instance-based strategies. In some cases instances are available directly or can be obtained. When instances are available, they may be used in defining similarities between concepts.

- Use of auxiliary information. Dictionaries and thesauri representing general or domain knowledge, or intermediate ontologies may be used to enhance the alignment process. They provide external resources to interpret the intended meaning of the concepts and relations in an ontology.

- Combining different approaches. The different approaches use different strategies to compute similarity between concepts. Therefore, a combined approach may give better results.

(2) SAMBO

SAMBO [Lambrix and Tan, 2006] is an ontology alignment and merging tool developed according to the above framework. SAMBO supports ontologies in the OWL [10] format. The system separates the process into two steps: aligning relations and aligning concepts. In the suggestion mode several kinds of matchers can be used and combined. The pairs of terms with a similarity value above a threshold are shown to the user as alignment suggestions. For each of the alignment suggestions the user can decide whether the terms are equivalent, whether there is an *is-a* relation between the terms, or whether the suggestion should be rejected. If the user decides that the terms are equivalent, a new name for the term can be given as well. If the user rejects a suggestion where two different terms have the same name, he is required to rename at least one of the terms. At each point during the alignment process the user can view the ontologies represented in trees with the information on which actions have been performed, and the user can check how many suggestions still need to be processed.

In addition to the suggestion mode, the system also has a manual mode in which the user can view the ontologies and manually align terms. The source ontologies are illustrated using *is-a* and *part-of* hierarchies. The user can choose terms from the ontologies and then specify an alignment operation. After the user accomplishes the alignment process, the system receives the final alignment list and can be asked to

[10] http://www.w3.org/TR/owl-features/

create the new ontology. The system merges the terms in the alignment list, computes the consequences, makes the additional changes that follow from the operations, and finally copies the other terms to the new ontology.

(3) Protege PROMPT

Protege is a tool for creating, editing, browsing, and maintaining ontologies [11]. PROMPT is one of its plug-ins, including several interactive tools for ontology merging and aligning [Noy and Musen, 2003]. iPROMPT is the ontology merging tool in the PROMPT suite [Noy and Musen, 2000]. When merging two ontologies, iPROMPT creates a list of initial suggestions based on the underlying alignment algorithms. The suggestions can, for instance, be to merge two terms, or to copy a term to the new ontology. The user can then perform an operation by accepting one of the suggestions or creating his own suggestions. iPROMPT then performs the operation and additional changes that follow from that operation. The list of suggestions is then updated and a list of conflicts and possible solutions to these conflicts is created. This is repeated until the new ontology is ready.

(4) Ontolingua Server

Ontolingua Server is an ontology development environment for collaborative ontology construction, addressing the problem of ontology integration [Farquhua, et al., 1995 and Farquhua, et al., 1997]. This tool allows collaborative ontology building and also provides an ontology library, where tested ontologies are gathered and made publicly available. To allow reuse of the ontologies available at the Ontolingua Server library, a set of integration operations was identified, specified, defined, and made available to ontology builders. Users are allowed three operations: inclusion, polymorphic refinement and restriction (specialization). Inclusion is used when the ontology is included (from the library of ontologies kept by the tool) and used as it is. Polymorphic refinement extends one operation so that it can be used with several

[11] Protege. http://protege.stanford.edu/index.html

kinds of arguments. Restriction makes simplifying assumptions that restrict the included axioms. The Ontolingua Server also provides facilities for local symbol renaming. This facility enables ontology developers to refer to symbols from other ontologies using names that are more appropriate to a given ontology and to specify how naming conflicts among symbols from multiple ontologies are to be resolved.

(5) FOAM

FOAM [12] is a semi-automatic tool for aligning and merging two or more OWL ontologies. When merging ontologies in semi-automatic mode, FOAM proposes alignment suggestions and the user can accept or reject these suggestions. The output of the system after processing all the suggestions is the accepted list of alignments.

2.5 Introduction to Several Integration Systems

During the 1990s, the emergence of distributed computing, middleware technology, and standards has allowed people to increase focus on the heterogeneity that is intrinsic to data. This has supported particularly syntactic and structural interoperability, and allowed people to address issues at the information level. As the future information system increasingly addresses the information and knowledge level issues, it will require further semantic interoperability. Semantic interoperability requires that the information systems understand the semantics of the information sources as well as the user's information requests, and use mediation or information brokering to satisfy the information request.

During the past two decades, there was an increase in the adoption of ad hoc standards, resulting in significant progress towards achieving system, syntactic, and structural interoperability. Structural and a limited form of semantic interoperability are achieved by adoption of general purpose metadata standards, such as Dublin Core

[12] http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/

[Mudumbai, 1997], as well as metadata standards in various domains such as bibliography [Beard and Smith, 1998], space and astronomy, geographical, environmental [Gunther and Voisard, 1998], and ecological [Reichman, et al., 1999].

Early works focused on data integration based on databases. Data integration is the process which takes as input a set of databases, and produces as output a single unified description of the input schemas (the integrated schema) and the associated mapping information supporting integrated access to existing data through the integrated schema [Parent and Spaccapietra, 1998].

For example, Clio+Garlic [Farquhua, et al., 1995] was developed by IBM, mainly targeted at the transformation of legacy data into a new target schema. It introduced an interactive schema mapping paradigm based on value correspondences: through providing GUI for the users to specify how a value of a target attribute can be created from a set of values of source attributes. According to the user-specified value correspondences, the query/view definition will be automatically discovered using DBMS query optimization techniques. In addition, it has a mechanism for users to verify the mappings.

Early work on the SIMS system [Arens, et al., 1996] included a central domain that is linked to the component databases and an AI-style planner that decompose queries for efficient access. SIMS requires the system designer to build a model of the application domain and to define the contents of each source (database, Web server, etc.) in terms of this model. The SIMS planner provides a single point of access for all the information: the user expresses queries without needing to know anything about the individual sources. SIMS translates the user's high-level request, expressed in a subset of SQL, into a query plan [Ambite and Knoblock, 2000], a series of operations including queries to sources of relevant data and manipulation of the data.

Later works employed ontologies to help integration at the concept level. By using ontology for explication and transformation of context knowledge users can achieve interoperability at the semantic level [Calvanese, et al., 1998(1) and Stuckenschmidt and Wache, 2000].

For example, Information Manifold [Kirk, et al., 1995] employs a local-as-view approach. It has the explicit notion of global schema/ontology. Its general mediator, independent of sources and queries, takes declarative descriptions of the contents and capabilities of a set of sources over the global concepts as input. A new source can be added by providing its descriptions and providing a corresponding wrapper. A dialect of description logics, called CARIN, is used for source description. The Bucket algorithm was developed in this project for rewriting the query over the global schema into queries to suitable sources.

In the BUSTER project, semantic integration is viewed as context integration [Visser, 2004] since information can only be well understood in its context. The context appears in terms of assumptions about the meaning of information but the assumptions are often not explicated. Semantic integration can be achieved through context transformation where context information has been explicated, descriptions of information entities are completed, and entities are interpreted in a new context. In context theory, a context is a collection of linguistic expressions providing an explicit description of the domain. Or, it can be viewed as a set of parameters with each representing one special aspect of the context described and a set of values can be assigned to the parameters describing the current context (e. g. {$\text{parameter}_1 = \text{value}_1$, $\text{parameter}_2 = \text{value}_2$, …, $\text{parameter}_n = \text{value}_n$}).

In later work of SIMS, the EDC project [Hovy, 2003] took this a step further, addressing the problem of the semi-automated construction of the single central model and linking it to a large general purpose term taxonomy or ontology Omega. The system provides dynamically planned access to data about petroleum products' prices and volumes, provided in a variety of forms and on a variety of media, by the Energy Information Administration, the Bureau of Labor Statistics, the Census Bureau, and the California Energy Commission, in the form of over 50, 000 data tables. In order to more rapidly construct the domain models, systems are developed for automatically identifying terminology glossary files from websites, extracting and formalizing the glossary definitions, clustering them appropriately, and automatically embedding them into the existing ontology and domain model.

Chapter 3 Problem Analysis

3.1 A Thorough Discussion on Fundamental Terms

In the literature review, we have touched on a rich set of terms that are used in the semantic integration field in various situations. This section presents deeper analysis of some of the fundamental terms based on our research, and gives further discussion on their natures.

3.1.1 **Information-related Terms**

I. Universe, World, Domain, and Real World Object

The *Universe* is the entire aggregation of everything that exists anywhere. According to the axiomatic theory [Zeng, 2008], everything in the universe is an object and there are relations between objects. The *World* is a subset of the universe that humans can perceive, memorize, understand, analyze, and reason about. A *Domain* is a portion of a world that some people are interested in and concerned about. In information related research, people often use the term *domain* to refer to a set of closely related objects.

An object can be physical or abstract, and can be perceived in some way. For instance, we <u>**measured**</u> that we walked "*2 miles*" <u>**in**</u> "*The University of Western Ontario*"; we <u>**talked to**</u> a professor "*Jack Smith*" when the watch <u>**showed**</u> the time was "*10:00AM, March 1st, 2008*".

II. Data, Information and Information System

In the context of computing, data is computational symbols. Information is data that has been given meaning, or, data with specification. Information is manged by information systems.

Generally speaking, an information system can be seen as the entire infrastructure, organization, personnel, and software components for collection, manipulation, storage, transmission, presentation, dissemination, and disposition of information [INFOSEC, 1999]. In the IT domain, an information system is a computer system composed of hardware and software applications as well as other necessary infrastructures to provide information and services. The hardware includes CPU, memory, disk, etc. that provide capabilities to store and process digital data. The software applications are mainly the ones that gather, manipulate, manage, persist, analyze, and present information. The infrastructures include operating systems, network protocols, software libraries, and network connections, to name a few, that provide system level support for the information services. Information services are functionalities handling information gathering, persistence, management, and retrieval.

From the external perspective, the usages of an information system focus on information persistence and information retrieval through specific service interfaces without concerning their internal design and implementation details.

The capabilities of information systems mainly lie in two categories: information providing and information searching or retrieval [Visser, 2004]. As for the former category, conventional database or formatted file-based systems are good examples of providing rich and dynamic information to any user that has authority to access them. In recent decades the Internet has offered the world a new dimension in terms of providing information for various needs. The major reason is that the HTML language allows people to share their information in a simple but effective way. This language is simple and easy to learn, and almost anybody with a basic knowledge about syntax or simple programming skills could design a web page and put it on the Web. Another reason is that standard network protocols such as HTTP and TCP/IP have been well supported by various computation platforms (including hardware and software platforms) which make the information accessing a simple, fast, and reliable job.

The latter category, searching information which can filter from a large amount of available information and provide the user intended results with desired formats, is just as important. Databases and some file-based systems (e.g. XML) support information searching with a clearly defined query written in a specific syntax. On the Internet, information searching is a little different. As described in [Visser, 2004], early browsers or search engines offered the opportunity to search for specific keywords, mostly searching for strings, and the latest versions of search engines, such as Google, provide a far more advanced search based on statistical evidences or smart context comparisons and rank the results accordingly. In most of the search cases, the users are prompted with results in a rather simple way but they have to manually analyze and choose their intended results from a very large result set where many of the results are partially or totally irrelative.

Although most of the information systems provide services with rich capabilities, we usually view them as information-centred systems instead of functionality-centred ones and view the services as facilities that support information management, sharing, processing, and exchanging. Therefore, from outside, the information systems are usually treated as information containers / repositories or information resource providers which encapsulate the internal functional components and interact with external environments via well defined interfaces.

The reason most responsible for affecting the search quality is the lack of semantics, for both the information itself and the query requirements. In an ideal world where semantics relating to anything and everything are clearly and precisely specified, one can expect that computers will help humans handle the semantics and manage the large amount of information in a perfect manner. This shows that information semantics plays an important role in information integration.

III. User

A user of an information system is either a producer of information or a consumer of the information and services, or possibly both. It does not matter whether the user is a human or another software application as long as it can interact with the system following the pre-defined interfaces and constraints. In our work, we often specifically refer to "human expert" where we emphasize the role of a human being. We also use the term "system" to denote a software application (such as a software agent) that interacts with an information system.

IV. Conceptualization

We differentiate two ways of illustrating the term "Conceptualization".

A. Conceptualization as a result of perceiving the world

Conceptualization refers to an abstract model that consists of the relevant concepts and the relationships that exist in a certain domain. In a sense, a conceptualization (of a certain domain) is an abstract and simplified view in one's mind of the partial world that one cares about for some purpose.

This term is also specially referred to as "shared conceptualization" which emphasizes the common consensus accepted by a community. To make conversations and exchange of information between humans meaningful and reasonable, people need to establish a shared conceptualization for a specific domain such that they have a common understanding of what they talk about. For example, in the education domain, people know the concepts "*Professor*", "*Student*", "*Class*" jointly. Whenever one mentions "*Professor*", the other one will know exactly the correct concept that is being discussed instead of incorrectly thinking of something else.

Conceptualization is domain-dependent. For example, in the education domain "*Class*" means a group of students sharing the same course. However, in the hotel domain "*Class*" may be used to identify the rank and category of the hotels.

Therefore, usually we need to limit our discussion in a domain of discourse to ensure that the expressing and exchanging of information make sense.

B. Conceptualization as a process of perceiving the world

In this sense, conceptualization is the process of abstracting the real world objects and creating abstract notions, i.e., the concepts for them, in human cognition. It refers to a set of mental activities that recognize the world and build a mental reflection of the world in human minds.

In our work we adopt the first way of using this term, i.e., the conceptualization of a world, of a domain, etc. In a conceptualization we can identify *concept*, which is discussed in the following section.

V. Concept in Conceptualization

A *concept* is anything that objectively exists in the real world and is rationally identified as existing in a conceptualization in terms of a domain of discourse. The concepts may be referring to physical objects such as persons and animals, or abstract ideas such as actions, times, distances and numbers. By "rationally" we focus on the shared conceptualization. For instance, in the education domain, normal people usually agree that "a *University* has many *Professors* and *Students*", but do not care about some others like "*Car*" or "*Tax*" (even though they are important in other domains and they may also be concerned in the education domain in some special situations).

A concept is defined by a set of *properties*. Each set of properties characterizes a specific aspect of a concept. For example, from the academic perspective, a *Professor* has properties *Name*, *Degree*, *Department*, *Title*, and *Publication*, whereas from the administrative perspective, another set of properties *Name*, *Year of Start*, *Salary*, *Address*, and *Contact* would apply. Even though there is very little commonality between these two sets, they are still depicting the same concept.

When we are concerned with the categorization of a set of real world objects, we differentiate *instances* and *concepts*. A concept can be instantiated as a set of instances, i.e., a concept conceptually stands for a set of instances that share some remarkable characteristics. Similar to classes and objects in the Objected-Oriented paradigm, a concept is instantiated to an instance by assigning *values* to its properties (each property may get one or more values, e.g. a professor's *Publication* has multiple paper tiles).

Note that the division of concepts and instances is depending on people's interest. As an example, a concept "*Human*" in the education domain is identified to have instances "*Professor*", "*Staff*", and "*Student*" if we just want to know what roles we have in a university. However, if we also care about individual persons under each role, we need to regard "*Professor*" as another concept which can be instantiated to multiple instances. We will specifically use *concept* or *instance* where we need to clarify the level that we are working on.

A concept can also be described by other concepts and *relationships*. That is, the semantics of a concept is defined through a set of *semantic relationships* that associates the concept to other concepts semantically. In the text we use underlined words or phrases to stress the semantic relationships. For example, we can define a *Professor* as "a *Person* who <u>works at</u> a *University*, <u>teaches</u> *Courses*, and <u>conducts</u> *Research*". In this definition, we characterize the concept *Professor* with other concepts - *Person*, *University*, *Course*, and *Research*, as well as the semantic relationships <u>work at</u>, <u>teach</u>, and <u>conduct</u>.

A property of a concept is possibly another concept. As an example, *Publication* of a *Professor* is a concept which has properties *Title*, *Abstract*, *Co-Authors*, *Publisher*, etc. Obviously *Co-Authors* and *Publisher* are sometimes regarded as concepts with other properties and semantic relationships. Therefore, the statement "a concept has some properties" is actually a concretization of the semantic relationship, i.e., a specific type of semantic relationship "<u>has-property</u>". To give more exact meaning to the semantic relationships, in some cases such relationships are elaborated upon, such as "(*Professor*) deliver-publication (*Publications*)" or "(*Publication*) published-by (*Publisher*)". We will simply state that a concept has some properties where no confusion will arise. If other semantic relationships are to be considered, we will explicitly mention them.

VI. Model and Modeling

Human cognition can be aware of a concept in a domain. In some way, this kind of awareness is reflected in human minds, and analysis and reasoning of the concepts can be done in human cognition. To better explicate, present, analyze, process, and communicate the concepts people need to extract the abstract concepts from their cognition and specify them. A model of a concept is a theoretical construct that provides formal or informal specification to the concept. A model theory defines various constructs, rules of applying the constructs, and meanings of the rules to specify the concepts. The combination of constructs, rules, and meanings is also referred to as a model language.

In the context of information systems, an information model is an explicit, formal, and structured specification of the concepts and relationships managed by the system. An information system is usually built upon an information model such as a relational schema.

Modeling refers to the process, activities, and regulations of creating a model of a concept (or a group of concepts) following the adopted model theories and methodologies.

There are two levels of concept models: the conceptual level and the representation level. The conceptual level model can be viewed as a kind of internal representation in human cognition (even how it is represented in human cognition remains unknown). For example, when you think about "Car" you build a model in your mind because it is never the case that you have a real car in your mind.

The representation level model is an explicit artifact created to conceptually represent the concept. For instance, in an E-R (Entity-Relationship) model each concept is modeled as an *entity* and represented by a rectangle, and semantic relationships are modeled as *relations* between entities and represented by a diamond, where both entities and relations have properties that are represented by ellipses. Other paradigms also developed their ways for modeling and representing. The Concept-Graph model defines that a conceptual graph is a structure representing *concepts* and *conceptual relations*. Concepts are linked to each other through the conceptual relations. It constrains that there are no links between a concept and another concept, and no links between a relation and another relation. Semantic Network is another way to model concepts as well as semantic relationships. A semantic network is a graphical specification of knowledge that shows *objects* and their *relationships*. In a semantic network, objects (or concepts) are modeled as nodes, and links between the nodes describe the relationship between the objects.

VII. Context

There are several interpretations for the term "context". In the most natural sense, in literature environment, the context of a word or a phrase is a body of words or phrases surrounding it that helps to determine its interpretation. In a broader sense, the context of something under consideration is the set of facts or circumstances that surrounds it and is relevant to it from specific points of view.

We view the context of a concept as a set of concepts other than the concept itself in a domain that semantically relates to it and helps to interpret its semantics. For instance, given two concepts "*Professor*" and "*Faculty*" in the education domain, their semantics cannot be ultimately determined due to the lack of context. Suppose that we describe "*Professor*" as a *Person* who works in a *University* and does *Research*, and describe "*Faculty*" as a type of *Employee* of *Universities* whose major responsibility is *Research*, with these given contexts it is sound to infer that they are actually the same concept. Contrarily, if a different context is provided by describing "*Faculty*" as an *Organizational Division* set up by a *University*, the contexts help us distinguish it from the concept "*Professor*".

Similarly, the context of an instance is a set of instances and concepts other than the instance itself in a domain that semantically relates to it and helps to identify its semantics and to which concept it belongs. For instance, simply given two instances *Jack Smith* and *Adam* in a university, there are not enough clues to identify what concepts they instantiate. We can reasonably infer that both are instances of the concept "*Person*" but this is not sufficient if we need to know their individual roles. Secondly, the inference may be wrong as it is possible that *Adam* is the name of a robot instead of a real person. Providing that *Jack Smith* teaches the course *Programming* and *Adam* takes the course *Programming*, based on these contexts it is known that *Jack Smith* is actually a *Professor* and *Adam* is a *Student*.

VIII. Representation

A representation of a concept or an instance following some model theory is an artifact created to conceptually represent the concept or the instance. The representation can be visual (i.e., can be perceived by human vision, such as a graph), hearable (i.e., can be perceived by human hearing, such as spoken words), touchable (i.e., can be handled manually, such as a model of a building), or formal (i.e., readable for human and process-able for computers, such as numbers), to name a few.

Strictly speaking, anything we are using to denote a concept, such as a written word, a spoken word (a voice), a figure, a graph, an expression, a statement, is a specific representation. To better illustrate the research issues, we will use an English word with the italic font and a capital first letter to denote a concept at the conceptualization level, such as *Professor*. It is not viewed as a representation. Other cases will be referred to as representations, i.e., "professor", "prof.", etc. At the model level, we may use some words, symbols, and expressions to describe the model. They are also distinguished from the representations. It is worth mentioning also, strictly speaking, that the model level can be viewed as a representation of the conceptualization. For instance, if we use an E-R paradigm to model the domain, we "represent" a concept as an *entity*, i.e., the entity is a representation of the concept. However, in our work we focus on the representation of the models and instances given than the modeling paradigm is provided.

At the representation level, we use the term *attribute* other than *property* to describe a representation of a concept model. In some cases we may use model-specific terms, such as *column* in terms of a relational table schema as a representation of a concept model following the relational model theory.

To sum up, a category of real world objects can be conceptualized as a concept, a concept can be modeled in different ways, coming up with different models (therefore a concept is represented as a specific element in a model), and each model can be illustrated by different representations. As an example, the concept *Professor* can be modeled as an entity in an E-R model, and this entity can be represented as a rectangle attached with a set of ellipses representing its properties; it can be modeled as a frame with slots for name, degree, title, etc, and the frame can be represented as a tabular form.

Figure. 3-1 depicts various levels in terms of the concepts discussed above. Note that we use a specific representation (two cartoons, in this example) to stand for the real world objects (two persons *Jack Smith* and *Peter Ken*).

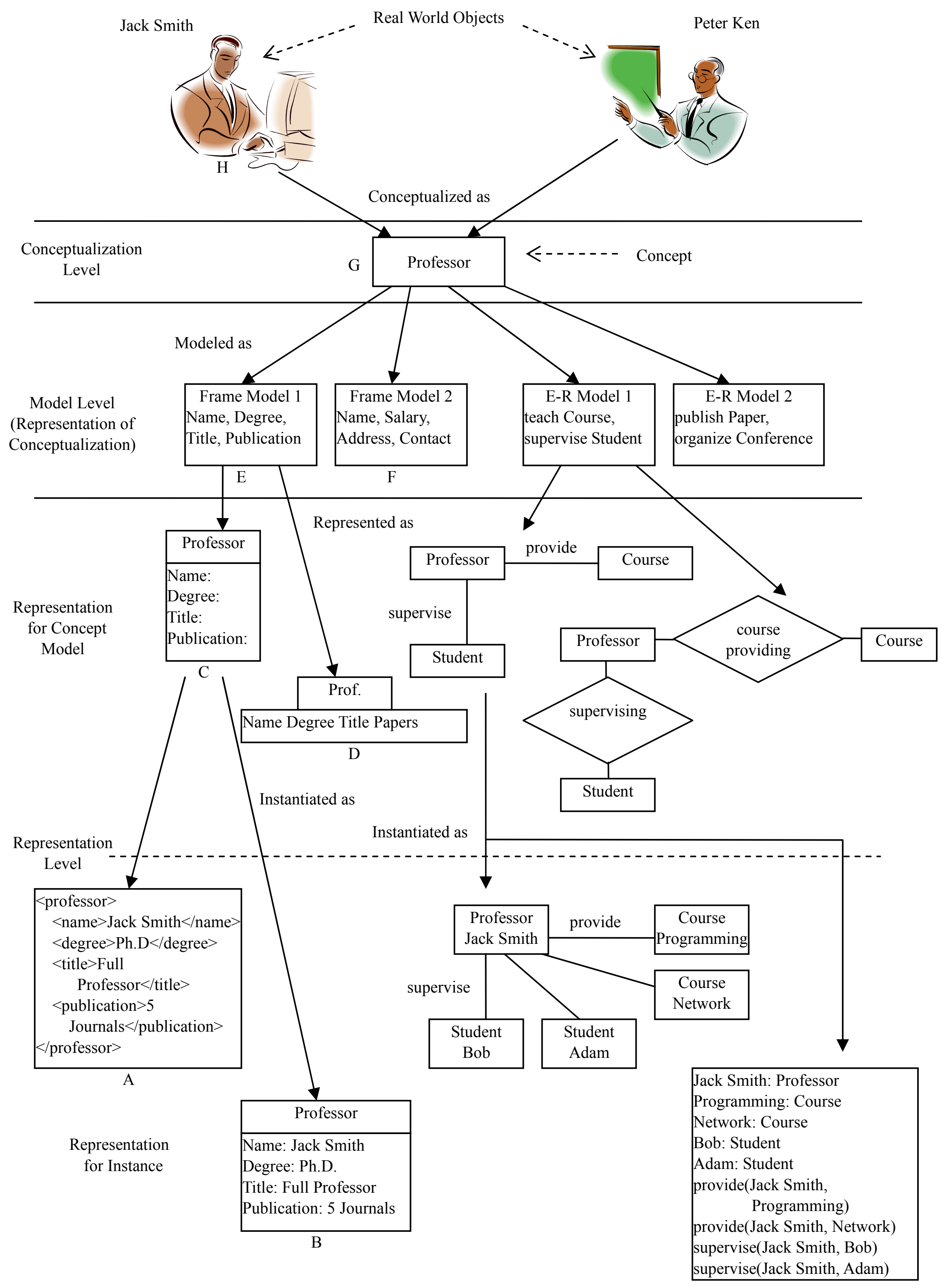

Real World Objects
Jack Smith
Peter Ken
H
Conceptualized as
Conceptualization Level
G
Professor
Concept
Modeled as
Model Level (Representation of Conceptualization)
Frame Model 1 Name, Degree, Title, Publication
Frame Model 2 Name, Salary, Address, Contact
E-R Model 1 teach Course, supervise Student
E-R Model 2 publish Paper, organize Conference
E
F
Represented as
Representation for Concept Model
Professor
Name: Degree: Title: Publication:
C
Prof.
Name Degree Title Papers
D
Professor
provide
Course
supervise
Student
Professor
course providing
Course
supervising
Student
Instantiated as
Instantiated as
Representation Level
<professor> <name>Jack Smith</name> <degree>Ph.D</degree> <title>Full Professor</title> <publication>5 Journals</publication> </professor>
A
Professor Jack Smith
provide
Course Programming
Course Network
supervise
Student Bob
Student Adam
Representation for Instance
Professor
Name: Jack Smith
Degree: Ph.D.
Title: Full Professor
Publication: 5 Journals
B
Jack Smith: Professor
Programming: Course
Network: Course
Bob: Student
Adam: Student
provide(Jack Smith, Programming)
provide(Jack Smith, Network)
supervise(Jack Smith, Bob)
supervise(Jack Smith, Adam)

Figure 3-10. From real world objects to representations.

In reverse, one single representation may mean different things. For instance, in Figure 3-2, an English world **Apple** is used as a representation of an entity in an E-R model. This model actually comes from two conceptualizations, which are conceptualizing two totally different real world objects respectively.

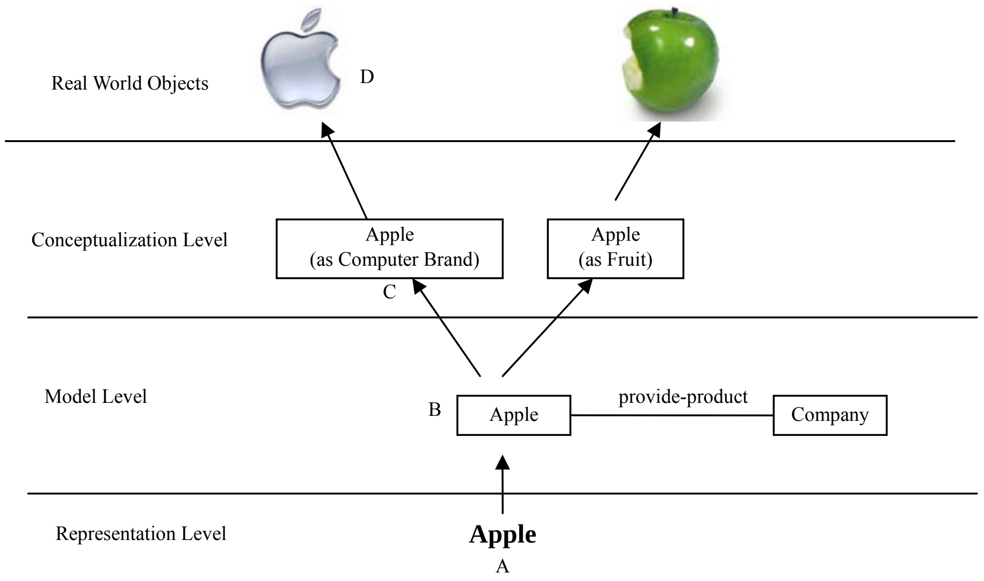


Figure 3-11. From representation to real world objects.

According to the analysis above, the ideal semantic integration may include four aspects:

A. Given different instance representations (e.g., A and B in Figure 3-1), discover whether they are instantiating the same concept model representation (e.g., C in Figure 3-1). If so, they refer to the same real world object (e.g., H in Figure 3-1).

B. Given different concept model representations (e.g., C and D in Figure 3-1), discover whether they are representing the same model (e.g., E in Figure 3-1) of some concept. If so, they refer to the same concept (e.g., G in Figure 3-1).

C. Given different concept models (e.g., E and F in Figure 3-1), discover whether they are modeling the same concept (e.g., G in Figure 3-1). If so, the two models are conceptually equivalent.

D. Given a specific representation (e.g., A in Figure 3-2), identify which model it is representing (e.g., B in Figure 3-2), then identify which concept (e.g., C in Figure 3-2) the model is capturing, and finally identify which real world object (e.g., D in Figure 3-2) the concept is conceptualizing.

IX. Schema

Information in computer systems can be viewed as a digital representation of domain concepts, instances, and relationships. The usages of information systems require that they have well-defined schemas for information storage and manipulation. A schema is a representation of a concept model following specific model theory. Schema refers to the organization of elements in a model theory. For instances, a relational table schema named "professor" organizes a set of columns in a tabular form representing a model of the concept *Professor*, and a XML schema named "professor" organizes a set of XML tags as another form of representing the same model.

The meaning of a schema is implied in its design and structure through its name, element names, element features, etc. The meaning of a schema can be determined if we establish a semantic mapping from itself to a known concept. The mapping should be explicitly and formally represented to support the processing of semantics. Since the schemas are one of the major sources of information semantics, this research will pay special attention to the schemas.

3.1.2 Information Semantics

3.1.2.1 Semantics Fundamentals

Information semantics and semantic integration have become active topics in several disciplines, such as databases, information integration, and ontologies. Researchers and practitioners have conducted great number of works on semantic integration to facilitate interoperability between different information systems [Noy, 2004].

According to [Meersman, 1995], semantics refers to a user's interpretation of the computer representation of the world – i.e., the way users relate computer representation to the real world. The ability to incorporate detailed semantics of data in computers will provide greater consistency in its use, understanding, and application [Magnini, et al., 2003]. One of the principal benefits of introducing the semantics is the reduction of human involvement in the process of information understanding and information integration.

Vetere [Vetere and Lenzerini, 2005] thinks that semantics is a mapping (also known as "interpretation function") which involves:

- Expressions: a system of manifested symbols (e.g. a formal language).
- Contents: a system of something else which is not necessarily apparent (e.g. sets of objects or events in (some abstraction of) the "real world").

Roughly speaking, semantics refers to "the intended meaning of something". This simple definition involves two aspects: what "something" is and what "meaning" is. "Something" is the abstraction of the external world in human minds, and is expressed in specific forms such as symbols, formulas, texts, voices, or graphs. Put simply, it may be concepts abstracted from some concrete objects like trees, animals, cars, rocks, and persons, or from some logical ideas like time, space, weights, and volumes, or from some actions like eating, looking, walking, etc. In more complex cases, "something" can refer to a comprehensive fact composed by concepts and relationships, such as a statement "*Dr. Jackson introduces us to many interesting topics in ES 250*", as depicted in Figure 3-3.

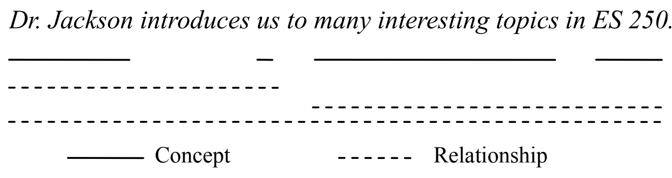

Dr. Jackson introduces us to many interesting topics in ES 250.
Concept
Relationship

Figure 3-12. A comprehensive fact.

It is difficult to define "meaning". As an alternative, it can be interpreted as the intension of specific concepts, relationships, or comprehensive facts. Their intension is expressed by some kinds of formalisms that are used to represent the meaning visually, and is meaningful only after the expressions are understood correctly by those who read them. In some cases, the reader will be a non-human object like a computer or a software agent, which is an important research issue in semantic integration. For example, given an expression (in a specific formalism) that represents a fact:

I DB 100

It is certain that very few people are familiar with this expression. Therefore, most people cannot understand it without any explanation of its semantics. In the computer programming domain, we can illustrate it with the following expression in another form, or, we can say that its meaning is:

int I = 100;

It is reasonable to claim that more people will be able to understand this form. Its meaning or semantics is trivial for people who are familiar with C, C++, or Java programming.

We can extend the "path" to interpret the meaning of "*int I = 100*". That is to say, we can explain that its meaning is identical to:

Dim I as Integer = 100

It is a variable declaration statement in Visual Basic (VB) language. People who are familiar with VB other than C or C++ can now understand it. For people who are only familiar with Perl language, another interpretation can be provided further:

my $I = 100;

The semantics implied by these expressions can be further interpreted in a natural language sentence: define a variable which name is *I*, type is *integer*, and initial value is *100*. Note that here we use natural language (which is also a formalism to express the semantics of something) to explain the meaning of the previous formalisms.

For people who are not familiar with programming but have fundamental knowledge in computer science, a variable is a storage unit in memory space which is referenced by its name. For someone unfamiliar with computer science, more details may be required to explain the semantics of the expressions.

Note that, from the beginning, we are limiting the domain of discourse to computer programming. In other domains, "*I DB 100*" may have completely different meanings.

Another key issue to mention is that we suppose that people who have a similar background and normal intelligence will achieve a common understanding of the same expression (at least in one specific domain). However, we must be aware that exceptions exist. For instance, a programmer may consider the expression "*int I = 100*" in another way, unconsciously or purposely. That becomes more complicated. We will not consider this exceptional case because it is really not a problem we can solve and it is very rare. In fact, some research did touch on the topic of discovering malicious semantics interpretation [Doan and McCann, 2003 and McCann, et al., 2003], but more work remains to be done.

From the above interpreting process, we can see that semantics in a specific domain can be represented in some kind of language (or formalisms), and interpreted by other kinds of languages (or formalisms). Natural language such as English is the ultimate formalism we use to interpret the intension of something. The continuous interpretations at different levels form an interpretation chain, as shown in Figure 3-4, where the same semantics can be interpreted by multiple formalisms, and at each level some specific formalism is employed to interpret its upper levels and can be most readable for a specific group of readers.

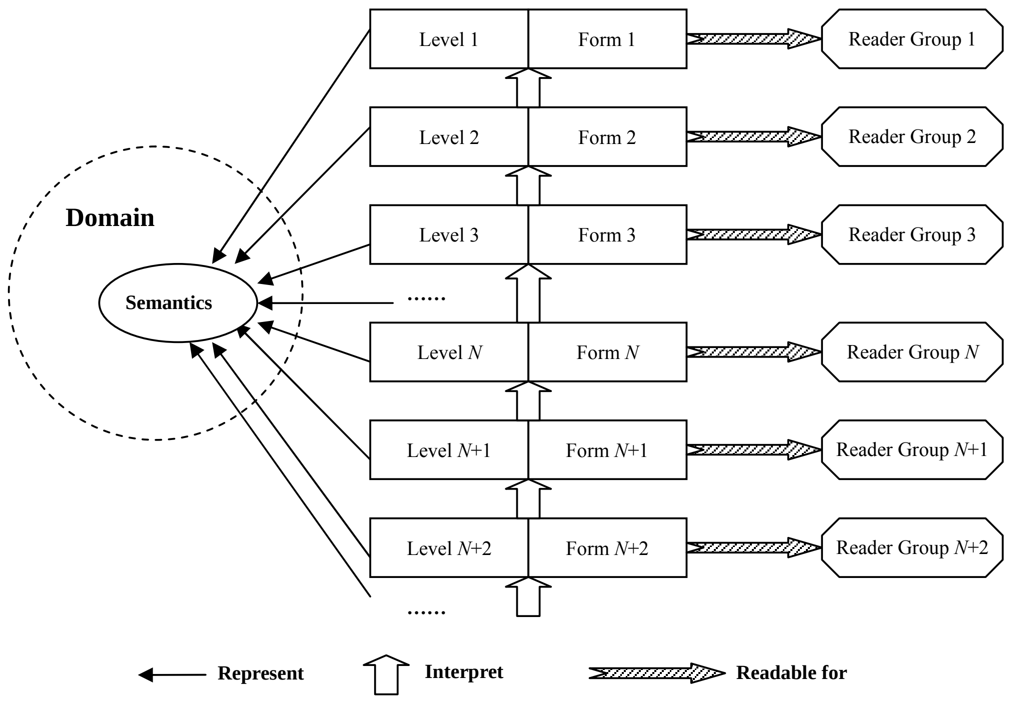


Figure 3-13. A semantics interpretation chain.

Since, ultimately, all semantics must be interpreted by a specific natural language and interpretation expressed by natural language can be interpreted in more detailed ways with the same language, we can assume that there is a level number *N* in the above figure meaning that starting from the *N*th level, all the lower levels of interpretation formalisms are natural languages. Note that the reader groups may overlap, i.e., there are readers (people or machines) who can read and understand multiple levels of formalisms.

According to the nature of human thinking, we have several conclusions about semantics:

Conclusion 1: Any level in the interpretation chain is readable for a human. Any formalism is a kind of explicit representation of semantics in human thoughts. People create various forms to express the semantics for different goals; therefore people can understand any of them, although some of them are understandable only by very few people.

Conclusion 2: The interpretation chain is infinite. It can grow in both upper direction and lower direction along with the creation of new representation forms.

Conclusion 3: Reader groups are not totally disjoint. Some individuals may be familiar with different formalisms.

Conclusion 4: Machines (computers) can be members of some high level groups, i.e., the corresponding forms are readable for machines. The levels readable for machines are limited, although they may extend to lower levels along with the advancing of machine design.

Our work will focus on machine readable formalisms and semantics.

3.1.2.2 Structural Semantics and Intensional Semantics

In information systems, the semantics of information is implied by data. A data item has a specific representation formalism, such as number, text, graph, etc. Two aspects should be considered for any data: structure and content.

A specific representation formalism has a set of structural rules that define what elements can be contained in this formalism system, how the elements will be combined to form valid and complex elements, and how to interpret the meaning of an element or a combined result with both their structures and contents. From this view, we divide the semantics of data into two categories: structural semantics and intensional semantics.

A. Structural Semantics

Let’s take a look at a XML document example:

```
<AutomobileCompany name = “AutoLondon”>
    <Car ID=“001”>
        <Name>Audi</Name>
        <Price>500.00</Price>
    </Car>
    <Car ID="002">
        <Name>Benz</Name>
        <Price>800.00</Price>
    </Car>
</AutomobileCompany>
```

There are several structural rules constraining the format of a valid XML document, for example,

(1) It contains tags defined by users. A tag is a string composed of letters and numbers.

(2) Tags are included in "<" and ">" brackets.

(3) Tags should appear in pairs. The latter one of a pair must have one symbol "/" before it.

(4) Under <**AutomobileCompany**> tag, it is allowed to have one or more <**Car**> tag pairs. Under <**Car**> tag, only one <**Name**> and one <**Price**> tag are allowed to appear.

(5) <**AutomobileCompany**> tag has an attribute called "**name**" which value is a string.

(6) …

According to the nature of XML documents and the goal of this document, many other rules can be derived. These rules define what elements are contained in this document and how they can be validly combined. This is a kind of structural semantics that is used to describe its structure or its "looking". In many research contexts, these rules are called "syntax". Here we didn't touch any real-world related meaning of the elements such as "*AutomobileCompnay*", "*Car*", or "*Price*".

Structural semantics can be easily understood by machines. Actually, people can construct compiles or parsers for the machines to handle structural semantics. The machines can run compiles or parsers and then are able to analyze and validate the documents by checking whether there is something in the documents violating the rules. In the past several decades, mature theories, methods, and tools have been developed to support the manipulation of syntax.

B. Intensional Semantics

Intensional semantics is implied by both the data structures and contents. It is much more complex than structural semantics, even for humans. Lots of ideas can be implicated by a target object, say, a document. For example, in the XML document mentioned above, someone who is familiar with XML may read it and get some ideas like:

(1) It describes a company that deals with cars.

(2) The name of the company is "*AutoLondon*".

(3) The company has a car with *ID* 001. Its name is "*Ford*" and its price is 500.00.

(4) …

There are several interesting things in "the meanings of something". First of all, these meanings can be guessed and understood by humans, but they are unreachable for machines. Today's computers still cannot really understand anything. They can only deal with binary data according to the rules designed by developers. They really have no idea about what they are doing.

Second, "the meaning of something" has two levels: one is the schema level and the other one is the instance level. We may discover the schema level first (here we go beyond the "rules" of combining these elements to try to discover the "meaning" of these elements), such as a company that has a name deals with cars and each car in that company has ID, name, and price. In much of the literature, such information is also called “meta-data”. Meta-data is a type of data where something being described is data. Or, as it is often put, meta-data is data about data. A strict definition of meta-data is: meta-data is data associated with objects which relieves their potential users of having full advance knowledge of their existence or characteristics [Dempsey and Heery, 1997]. Meta-data is used to facilitate the understanding, use, and management of data.

Based on the understanding of the schema or meta-data, we can refer to concrete data to get more knowledge, for example, about the fact that a company named “*AutoLondon*” has one car with ID *001*, named “*Ford*”, and priced at *500.00*. Another direction is possible and that is to observe the concrete instances, then extract the schema to gain an overall knowledge about a concerned topic.

Finally, human understandings are not always correct, and in many cases rather vague due to the incomplete information. For instance, some details may be omitted in the document, leading to difficulty in understanding it. Significant ambiguity will affect our knowledge resulting from our guessing. For example, does the “001” car represents one individual car or a type of cars? Is the price for purchasing or renting the car? Is the price in USD or CAD? There is no way to confirm these questions unless richer information is provided to reveal the semantics of that data.

3.1.2.3 Source of Semantics

The implicit intensional semantics of information can be elicited from three major sources: the observations of readers, the designer’s knowledge, and the applications.

(1) Observation of readers

Some experienced professionals who are trying to guess the information semantics are able to analyze the underlying data based on their domain knowledge and experiences. For example, when someone sees the element "*Price*" he can infer that, as a usual case, it is a price before tax applied. Unfortunately, such semantics is just a kind of "guessing result" and is not definitely correct. However, we often adopt it as a major source since it is the most available and least costly way.

(2) Designer's knowledge

The designer who creates the above XML document knows exactly the meaning of each of the data elements. For example, the tag "*Price*" is for sale, not for renting; the tag "*Name*" is for a type, not for an individual car, etc. Documents describing the designer's ideas can act as another form of a "designer's knowledge" when the designer is not available. Problems of this type of source are that the designer may be unavailable, or may forget the knowledge after a long time, and the documents may be incorrect, incomplete, or outdated. All of these situations have a negative impact on semantics elicitation.

(3) Applications

Applications, or simply, software programs, are designed to manipulate the data in meaningful ways. People can get knowledge on semantics by reading the programs and observing their execution (e.g., what input they accept, how they act after that, and what output they generate). For instance, the following pseudo code (which can be translated into a real program) handles the XML document with clear goals:

```
if there is a tag <Price> then
  output "The selling price is " + string between tags <Price> and </Price>
  return the string to somebody who is asking for the selling price
end if
if there is somebody inquiring price for renting then
  reply "There is no required information."
end if
```

It is an example of a common practice: data semantics is revealed by the business logic. Since the program can handle the tags and strings correctly, we say that the machine is able to understand the semantics of the data when it is running the program. However, strictly speaking, the machine is still unaware of what "*Price*" is and what "*500.00*" is. At least it is true for all computers in the contemporary era. Computers cannot understand anything—they just do binary computations according to pre-designed principles. They cannot think. Therefore, we need a definition for "machine understanding semantics": if a machine can manipulate some data correctly (according to the human's criteria) with the support of some software application systems, we say that the machine can understand the semantics of the data.

Since data semantics is in fact handled by software applications, theoretically we can construct more new applications to deal with any possible semantics, but it is certainly a very costly way. If an application is designed and constructed to be flexible, it will be able to handle various cases if new semantics descriptions are provided (that implies, it "knows" the meaning of different data) without modifying itself or requiring new ones and hence save investment.

Since any program is written by humans, and the processing logics in the programs are derived from human thinking, the most original source of any semantics is still the designer's knowledge. However, in most occasions, we just interact with computers and applications, and we don't have the opportunity to interact with their original designers. Therefore, we still hold applications in high regard as a major source of semantics.

3.1.2.4 Semantics Discovery

As mentioned in section 3.1.2.1, specific formalisms are required to represent semantics. Obviously, semantics that a kind of formalism can represent is limited, and that expressed by a specific representation (e.g., a concrete XML document) is also limited. The semantics a reader group can understand from that representation is limited, too. One way to enrich semantics and make implicit semantics more explicit is adding new elements to a formalism (or data structure), or adding new elements to a specific representation (or concrete data). As to the former case, research in this area has called the combination of elements that are used to specify various aspects of the information a "context" which can serve to describe the concerned information [Sciore, et al., 1994 and Stuckenschmidt and Wache, 2000]. At a high level, the term "context" is defined as any information that is useful for characterizing the state or the activity of an entity or the world in which this entity operates [Dey, et al., 2001]. Any information must reside in some context and only after the context is clearly declared can we understand the information correctly and exactly.

Of course, the reader groups need a learning process to understand the new structures and new instances. To give an example, if we modify the previous XML document to a new version:

```
<AutomobileCompany name = “AutoLondon”>
    <Car ID=”001”>
        <CarName>Audi 001</CarName>
        <CarType>Audi A6</CarType>
        <Price>
            <Selling>20000.00</Selling>
            <Renting>500.00<Renting>
    </Price>
    </Car>
</AutomobileCompany>
```

With more elements contained in the schema of this XML document and richer texts embedded in the document itself, people now can get a more exact understanding of its semantics. For example, the price includes two categories: selling and renting. The relevant applications are required to be rewritten to involve more logic to express their "understanding" and utilize the new semantics. It can be regarded as a learning process by machines. Only after people understand its semantics and enable applications to handle the semantics correctly can we say that the applications "understand" the new semantics.

But there is more. What if we want to know more about the company, such as, is the rental price for one year, or one month? Does the selling price contain tax? The current version of the document does not provide enough clues for these questions. More tags and contents need to be added to it to express these new semantics.

Let's have a look at another example where the original data is kept unchanged (in the XML example new data is added to the original one to express more semantics), but only semantics related information is appended. Suppose we have a sentence stating one fact:

The first topic of Wireless Sensor Networks is a general introduction about this field.

Based merely on this sentence we have no idea about whether the term "*Wireless Sensor Networks*" is about a speech or a course. If the course option is what the author means, adding some description information (in XML-tag style) will be helpful (this method is also called "annotation" [Ovsiannikov, et al., 1998]):

```
The first topic of
    <course ID="ES 695" department="ECE" level="Graduate">
        Wireless Sensor Networks
    </course>
is a general introduction about this field.
```

An application designed for the purpose of course management knows the semantics of the tags, the extra information, and the term itself, therefore it can handle the course "*Wireless Sensor Networks*" perfectly. What if we want to know more about "*topic*", or "*general introduction*" in the original sentence? What if we want to know more about the "*ECE Department*", or the "*Graduate Level*"? There is no doubt that more information is required to be added to help discover the new semantics. In short, semantics discovery is an infinite process of digging meanings from the raw data.

3.1.3 Semantic Heterogeneity

Semantic heterogeneity occurs when the same real world entity, modeled by two or more people, does not have the same modeling or representation [Hess and Iochpe, 2004]. Since the models or representations are independently developed, they often have different structures, terminologies, or even interpretations, presenting an obvious obstacle for interoperation of the models in a semantically reasonable way.

Some attempts have been made to characterize information heterogeneity in terms of conflicts that can occur on the structural and the semantic level. Research to date has identified a number of factors contributing to information heterogeneity, irrespective of the subject domain. One of the latest and most complete classifications of different kinds of conflicts can be found in [Wache, 2003].

According to the classification proposed in [Goh, 1997], there are three types, each with further subdivisions, which are schematic, semantic and intensional heterogeneities (that can result in data conflicts). A detailed list of the three types is shown below:

- Schematic
    - Data type, the most obvious one being numbers as integers or as strings.
    - Labeling, only the strings of the concept names differ but not the definition. This also includes labeling of attributes and their values.
    - Aggregation, e.g. organizing companies by locations or type of industries.
    - Generalization, e.g. an entity type *Employee* in one model and in another, there are *Faculty* and *Staff*.

- Semantic
  - Naming, includes problems with synonyms (same concept with different terms, e.g. *maize* and *corn*) and homonyms (same term with different semantics, e.g. *worm* as animal, as muscle under tongue and as infection in computers) of concepts and their properties.
  - Scaling and unit. Scaling: one system with possible values *white*, *pink*, *red* and the other uses the full range of RGB; units: metric and imperial system.
  - Confounding, a concept that is the same, but in reality different; primarily has an effect on the attribute values, like latestMeasuredTemperature, which does not refer to one and the same over time.
- Intensional
  - Domain: when two systems represent different knowledge. For example, one can model a flower being composed of a petal, leaves and so forth from a biology perspective, but also from a utilitarian perspective (sellable, the related logistics system).
  - Integrity constraint: the identifier in one model may not suffice for another, for example one animal taxonomic model uses an (automatically generated and assigned) ID number to identify each instance, whereas another system assumes each animal has a distinct name.

Heterogeneity is also referred to as *mismatch* in some literature. The mismatches can be distinguished at two levels: the language level and the model level [Klein, 2001]. The language level is related to the representation of the ontologies, i.e., different constructs, syntax, and semantics of the languages. Mismatches at the language level are those between the mechanisms to define concepts, relations, and so on. The model level, also called ontology level, is a difference in the way the domain is modeled. The distinction between these two levels of differences is often made. In [Kitakami, et al., 1996] and [Visser, et al., 1997] they are called *non-semantic* and *semantic* differences, respectively.

The following is a framework of different types of mismatches that appear at each of the two levels.

- Language level mismatches. Mismatches at the language level occur when ontologies written in different ontology languages are being integrated. Chalupsky defines mismatches in *syntax* and *expressivity* [Chalupsky, 2000]. They can be further distinguished into four types:

  - Syntax. Different ontology languages often use different syntaxes in terms of how the language constructs can be validly connected. For example, in RDF Schema the concept "Human" is defined as <rdfs:Class ID="Human"> and in LOOM the expression (defconcept Human) is used to define the same class.

  - Logical representation. Different logics can be used to represent the same semantics in the ontologies. For example, in some languages it is possible to state explicitly that two classes are disjointed (e.g., disjoint A B), whereas it is necessary to use negation in subclass statements (e.g., A subclass-of (NOT B), B subclass-of (NOT A)) in another language. The point here is not whether something can be expressed—the statements are logically equivalent—but which language constructs should be used to express something.

  - Semantics of primitives (language constructs). Sometimes the same name is used for a language construct in two languages, but the semantics may differ. For example, the OIL RDF Schema syntax [Broekstra, et al., 2001] interprets multiple <rdfs:domain> statements as the interaction of the arguments, whereas RDF Schema interprets it as a union.

  - Language expressivity. This difference implies that some languages are able to express things that are not expressible in other languages. For example, some languages have constructs to express negation, sets, or defaults, but others do not.

- Ontology level mismatches. Mismatches at the ontology or model level happen when two or more ontologies that describe (partly) overlapping domains are combined. These mismatches may occur when the ontologies are written in the same language, as well as when they use different languages.

  - Conceptualization. Visser et al. [Visser, et al., 1997] defines the conceptualization mismatch as a difference in the way a domain is interpreted (conceptualized), which results in different ontological concepts or different relationships between those concepts due to different interests. For instance, in the education domain one may model from the university's perspective and another one concerns the professor's perspective, thus different concepts sets will be derived. [Visser, et al., 1997] makes a distinction between mismatches in the *conceptualization* and *explication* of the ontologies. An explication mismatch is a difference in the way the (same) conceptualization is specified. The following ontology level mismatches are categorized as explication mismatches by Visser et al.

    - Modeling paradigm. This mismatch refers to the fact that different paradigms can be used to represent concepts such as time, action, plans, etc. For example, one model might use temporal representations based on interval logic while another might use a representation based on a point [Chalupsky, 2000].

    - Concept description. This type of differences is called modeling conventions in [Chalupsky, 2000]. Several choices can be made for the modeling of concepts in the ontology. For example, the way in which a hierarchy is built may be different. Considering the modeling of scientific and non-scientific publications, a dissertation can be modeled as publication→scientific publication→book→dissertation, or as publication→book→scientific book→dissertation, or even as a sub-concept of both book and scientific publication.

    - ◆ Synonym terms. Concepts may be represented by different names. A trivial example is the use of the term "car" in one ontology and the term "automobile" in another. This type of problem is also called a term mismatch [Visser, et al., 1997].

    - ◆ Homonym terms. The meaning of a term is different in another context. For example, the term "conductor" has a different meaning in a music domain than in an electric engineering domain. Visser et al. also call this a concept mismatch.

    - ◆ Encoding. Values in the ontologies may be encoded in different formats. For example, a date may be represented as "dd/mm/yyyy" or as "mm-dd-yy", distance may be described in miles or kilometers, etc. To solve these mismatches, a transformation step or wrapper is usually required to eliminate the difference.

- ■ Scope. Wiederhold [Wiederhold, 1994] describes possible differences in the scope of concepts, which is a type of conceptual mismatch. It refers to the fact that two concepts seem to be identical but do not have exactly the same instances, although these intersect. An example is the class "employee"; several administrations use slightly different concepts of employee.

- ■ Model coverage and granularity. This is a mismatch in the part of the domain that is covered by the ontology, or the level of detail to which that domain is modeled. An example presented in [Chalupsky, 2000] is about cars: one ontology might model cars but not trucks, and another one might represent trucks but only classify them into a few categories, while a third one might make very fine-grained distinctions between types of trucks based on their general physical structure, weight, purpose, etc.

The ontology level mismatches cannot be solved easily. For instance, it is difficult to find the terms that need to be aligned. This task is mostly done by hand [Noy and Musen, 2000], which requires knowledge and the decisions of a domain expert. Therefore, it is unrealistic to hope that mapping at the ontology level could be performed completely automatically.

Information heterogeneity has a direct impact on the interoperability of multiple information systems. Researchers and developers have been working on interoperability issues for many years. The following Figure 3-5 shows one perspective on resolving heterogeneity to achieve interoperability [Sheth, 1998]. Focus on the crucial dimension of heterogeneity and corresponding solutions leads to different levels of interoperability: system (mainly due to technological differences, e.g. differences in hardware, operating systems, and communication systems), syntax, structure, and semantics [Hamill, et al., 1997].

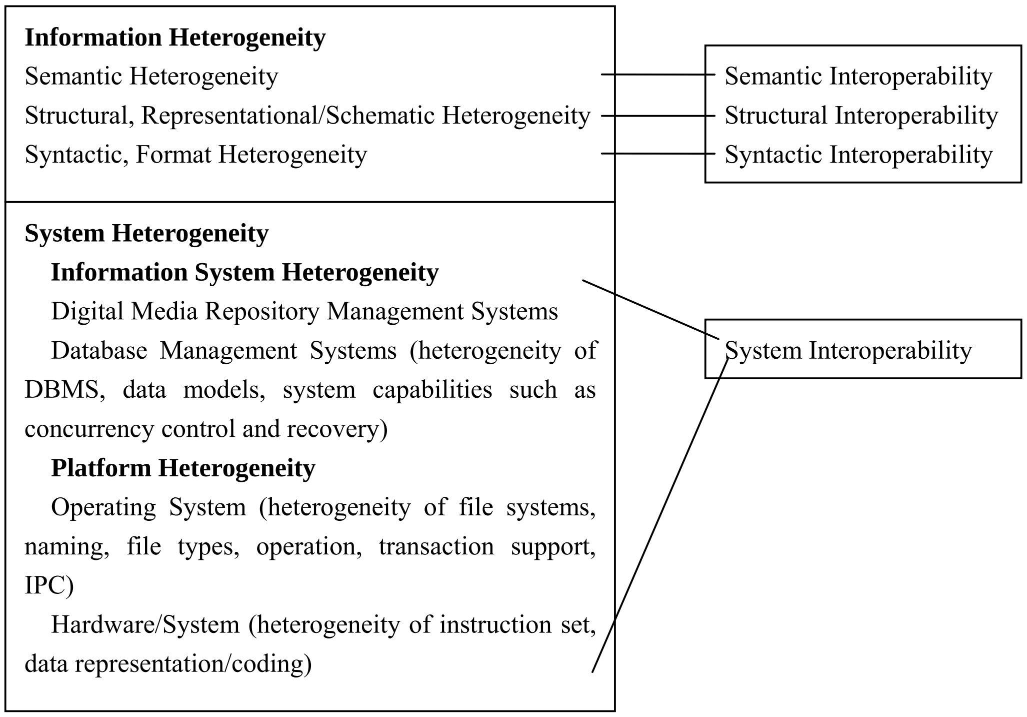

Information Heterogeneity
Semantic Heterogeneity
Structural, Representational/Schematic Heterogeneity
Syntactic, Format Heterogeneity
Semantic Interoperability
Structural Interoperability
Syntactic Interoperability
System Heterogeneity
Information System Heterogeneity
Digital Media Repository Management Systems
Database Management Systems (heterogeneity of DBMS, data models, system capabilities such as concurrency control and recovery)
Platform Heterogeneity
Operating System (heterogeneity of file systems, naming, file types, operation, transaction support, IPC)
Hardware/System (heterogeneity of instruction set, data representation/coding)
System Interoperability

Figure 3-14. A perspective on resolving heterogeneity to achieve interoperability [Hamill, et al., 1997].

3.1.4 **Semantic Integration**

3.1.4.1 Semantic Integration Fundamentals

Semantic integration has been a hot research topic for many years. One of its goals is to support interoperability among information systems. Multiple descriptions about the term "semantic integration" have been developed.

Taking human conversation as an example, the heart of the semantic integration problem is how to tell when two statements are about the same subject [Newcomb, 2003]. In some communities, this is known as the *co-referencing* problem. [Newcomb, 2003] proposes a methodology for semantic integration. The problem that the methodology addresses is the combining of multiple independently conceived representations of networks of subjects and relationships, with their separate, partially redundant *proxies* for the same subjects, in such a way that for each subject there is only one proxy, but no information has been lost. In this statement a *proxy* can be understood as a concrete representation of a subject. The methodology's definition of *semantic integration* is *subject proxy uniqueness*.

According to Vetere et al. [Vetere and Lenzerini, 2005], semantic integration has to resort to *conceptual mappings* that make different data/process descriptions equivalent, either pair-wise or with respect to some (partial) unifying ontology.

The conceptual mappings can be [Vetere and Lenzerini, 2005]:

- Any kind of XML transformation rule (e.g. XSLT [13]);
- Specific assertions of ontology languages (e.g. OWL's *sameClassOf*);
- Named views in database federations.

[13] http://www.w3.org/TR/xslt

In general, schemas of various information sources are heterogeneous, i.e. semantically related concepts are captured by different local schemas in different ways, e.g. using different names or different structures. Mendling et al. believed that discovering semantic relationships such as equivalence, subsumption, intersection, disjointedness, and incompatibility between concepts of local schemas plays a central role for semantic integration [Mendling, et al., 2005].

Semantic integration is highly domain-dependent. It is widely agreed that domain knowledge is extremely crucial for solving the heterogeneities. The domain knowledge is also very application-specific. For a complex integration system, it is difficult to acquire and use all relevant knowledge. Therefore, usually application-specific domain knowledge will be captured and modeled to support the integration task.

3.1.4.2 Different Views on Semantic Integration

In the following we propose a classification for semantic integration.

A. Structural View

In the structural view, we focus on the structural semantics of data. This perspective is not included in the conventional theory system about semantic integration, but we still include it to make the discussion complete.

(1) Elemental data level

Semantic integration may take place at various levels. The lower level is the *elemental data level*. As a case, taking the number system into account, the binary data "*00001110*", decimal number *14*, string *"14"*, and English word "*fourteen*" are different in representation and internal storage, but they refer to the same value regardless of what this value refers to. Therefore, they have the same semantics in terms of the value of an elemental data item. Semantic integration at this level requires data handlers to identify and maintain such equivalence. Data heterogeneity at this level has already been well managed by operation systems, network protocols, applications, etc. Therefore, these values can be handled consistently and correctly in most occasions.

(2) Structure level

The higher level is the *structure level*. A structure has multiple members possessing rich semantic information. It may be an object, a class, a database table, a document, etc., in terms of the representation format. Let's have a look at the previous XML document (in section 3.1.2.2), and another relational table:

CompanyName	**CarID**	**CarName**	**Price**
AutoLondon	001	Audi	500.00

People who are familiar with both formats will understand their identical meaning, although necessary transformation for the formats is needed when these two representations are manipulated by specific applications. In this example, the XML schema is by nature equivalent to the relation schema (or the table), i.e., one tag of the XML document is equivalent to one column of the table, and one instance block of the XML document is structurally equivalent to one row of the table (as shown in Figure 3-6), no matter what the elements really mean. Semantic integration at this level requires structure handlers to identify and maintain the relationships among the representations (both schema and data instance) in an appropriate way.

B. Semantic View

The structural view described above is helpful for understanding semantic integration. As far as solving this problem in the computation field, another view, the semantic view, is preferable. The semantic view can be further divided into data level, concept level, and knowledge level.

(1) Data level

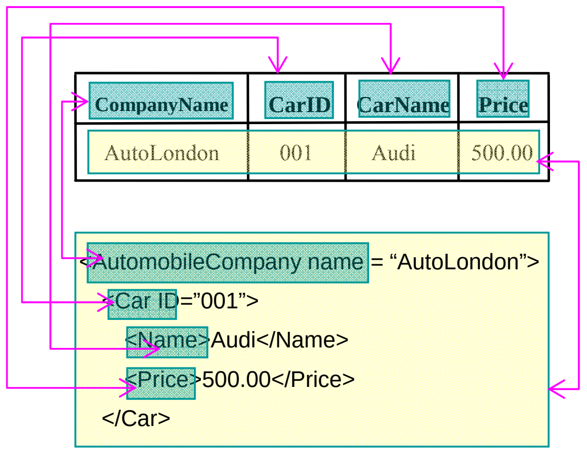


Figure 3-15. Structural mapping between XML document and relational table.

At the data level, we are concerned with the equivalence of data from different concepts. Note that it is totally different from the elemental data level described earlier because here we take the meaning of the data into consideration.

At the elemental data level, the information systems may maintain the equivalence of "five hundreds" and "500" from different data sources in terms of their value without considering their meaning. Things are much more complicated at the data level. As an example, if we find two prices from both the XML document and the database table with the same number: 500.00, the data level has to determine whether they refer to the same money. The answer is not definite. If one is in USD and the other one is in CAD, apparently they are not referring to the same money. Note that here we are considering the application domains and semantics in the domains.

Another interesting example is that in some classifying systems, if we have rank 1, 2, 3, 4, and 5, then 1 is the best one and 5 is the least one. However, in some other systems where rank 1, 2, 3, 4, and 5 are also employed, 5 is the best one and 1 is the least one. Therefore, when we get the same number from two classifying systems (from the elementary data view the two numbers are identical), things will go wrong if we regard them as the same rank. Even if we use the same order for the numbers, what if one system has 5 ranking numbers but another one has only 3 ranking numbers? The same number from two systems may imply the similar ranking position but not exactly the same position, therefore inconsistency appears.

The most complicated case may be the following one: two systems use the same way to describe data in the same domain, but the same data is still referring to different entities. For example, it is possible that we have a company "*AutoLondon*" from the XML document and a company "*AutoLondon*" from the database table. Even the mapping is one to one and the company name is totally identical, it is still possible that they represent different companies, e.g., one company from London in Canada and one from London in the UK. Therefore, we cannot integrate these two items simply. In distributed information retrieving, if we retrieve names of companies that sell cars from multiple information sources and get the two results, we should not merge them into one item, otherwise inconsistency will occur. This example shows that there is almost no way to distinguish them by the computers themselves without any human intervention if there is no sufficient context knowledge.

(2) Concept level

The concept level focuses on the mappings between different information representation formalisms according to their meaning in terms of concept references. For instance, we described the case of car-selling companies earlier, where an XML document and a database table are used to describe the same facts for different applications. A concept mapping determines that the tag <*AutomobileCompany*> and its property "*name*" in XML are mapped to the column "*CompanyName*", i.e., they refer to the same unique subject. Therefore, if we find one company with a specific name, say, *A*, in the XML document and one row in the table which value in the column "*CompanyName*" is also *A*, then we can infer that they are implying the same company. Note that here we have an assumption about the uniqueness of the company names. If this assumption does not hold, things will go wrong if we regard two companies with the same name as the same company.

The concept level looks similar to the structure level, but there are differences. As for the car-selling example, the structure level just defines that one column in the table can be matched to one tag in the XML. Only the concept level can determine which column is matching to which tag and why, based on the domain semantics, as depicted in Figure 3-6.

As regards the XML document mentioned in 3.1.2.4, since there is no way to map the tags <*Price*>, <*Selling*>, and <*Renting*> to anything in the relational table, the integration of price related information is impossible unless new columns are added to this table.

Two facts are important since they can cause confusion: the same name is used for different concepts and different names are used for the same concept. For instance, people often use “Address” and “Location” in different applications, but in most instances they are usually the same thing. Besides, “Category” may be used differently to describe whether a course is for undergraduate students, graduate students, or both of them, whereas in other applications, it may be used to describe whether a course is project-based, thesis-based, or exam-based.

(3) Knowledge level

The highest level is the knowledge level. At this level, people do not care about the formal representation or data structure of knowledge, but only the knowledge itself. Since any knowledge outside of human thought needs some kind of representing formalisms (in the human brain, knowledge may be stored in a specific structure which is still unknown today. It is not taken into account in the computation field), so let’s suppose that we use natural language to specify knowledge. For example, application *A* generates “today’s weather conditions”, and if application *B* can understand *A*’s knowledge, *B* will go to fetch “today’s weather conditions” from *A* and display it to the public in some visual way. Here “today’s weather conditions” is high level knowledge with rich semantics. It is easy for people to imagine and reason.

However, it is hard for computers to understand unless very definite formal specifications are provided, e.g., the weather can be specified by wind, temperature, and rain conditions; the wind condition can be specified by wind speed and wind direction, and the wind speed can be specified by how many miles per hour, etc. An ideal semantic integration should provide such a knowledge-level view to humans. However, this is really very hard to achieve. Note that undoubtedly, computer readable data representations or data structures are definitely required if we intend to develop applications to achieve this objective to some degree.

3.1.4.3 Conceptual Difference of Several Terms

Three terms about integration are used in various situations: *data integration*, *information integration*, and *semantic integration*. In the most general sense, they can be regarded as referring to the same subject. However, they can be further distinguished in different communities.

Data integration and information integration are basically the same thing. The term “Data Integration” is most often used in database and data-warehouse applications, focusing on merging multiple data sources (databases) into an integrated one, including database schemas and data contents, e.g., tables and rows in tables in relational databases [Hai, 2005]. It concerns the data itself, and the integration result is usually one physically independent object, such as one database. In some research, no final integrated data is created but only mappings between schemas are created and maintained [Rahm and Bernstein, 2001].

One of the major data manipulating mechanisms used in data integration is the calculation-based comparison. For example, in the following tables among which *A*, *B* are inputs of integration and *C* is the result, if we have mappings *A.Name* = *B.Type*, *A.Price* = *B.Selling-Price*, (*A*. *Name*, *B.Type*) = *C.Car-Type*, and (*A.Price*, *B.Selling-Price*) = *C.Car-Price*, then based on an equivalence comparison (more complex calculations may be necessary in other cases) we will know that *Ford* is redundant in *A* and *B*, therefore only one of them is kept in the result *C*. Moreover, *Audi* and *BMW* should be added to the result *C*.

Table A

Name	**Price**
Audi	10000.00
Ford	15000.00

Table B

Type	**Selling-Price**
BMW	20000.00
Ford	15000.00

Table C

Car-Type	**Car-Price**
BMW	20000.00
Ford	15000.00
Audi	10000.00

Sometimes the term "Information Integration" is used separately if one tries to emphasize the intended meaning of the data [Doan, et al., 2003] (a commonly used definition says information is data with meaning), so we shift to the concept of semantic integration. Today, semantic integration is mainly focusing on integrating multiple information sources and presenting users with a logically unique and unified "information source", while keeping the source still separate and no physically integrated schema/ontology is created. One of the often used manipulation mechanisms in semantic integration is logical-based reasoning. For example, if we have a knowledge item "*Apple* is-a-kind-of *Fruit*" in source *A* and another item "*Fruit* is-a-kind-of *Plant*" in source *B*, then the integrated result may contain an item "*Apple* is-a-kind-of *Plant*" that is derived by logical reasoning. This example also shows that logical reasoning is one of the important mechanisms used in semantic integration.

What we need to clarify is that we cannot entirely separate these various terms. Actually, they are closely related to each other. Since data (information) integration also needs the support of data semantics, we can regard it as a special case of semantic integration, especially in the database community. On the other hand, many methods and systems that have been explored over the past many years on data integration are also helpful for the research of semantic integration. For instance, schema matching developed in data integration now plays an important role in semantic integration.

3.1.4.4 Semantic Integration at the Application Level

When information systems (computer applications) need to collaborate and exchange information, semantic integration at the application level should be considered to support the task.

The key concern in semantic integration is how to make different applications understand, communicate with, and cooperate with each other. From the architectural perspective, three kinds of methods can be employed to achieve this goal.

(1) Pre-designed interface and information flow

This is fairly common in traditional software development, where a complete concept system (may be implicit) is established first, which provides different components of the architecture a common understanding for the domain of discourse. Based on the shared concepts, the interfaces and information flows for the components are thoroughly determined, therefore each component knows exactly what information it will receive, who will send it information, what the received information means, what information as a result should be sent out by itself after it does some operations on the received information according to its internal business logic, and whom to send.

The following figure shows an architecture example, where each component is an executable unit (with the necessary supporting environment) such as class, sub-procedure, program package, Web Service, or even independent application.

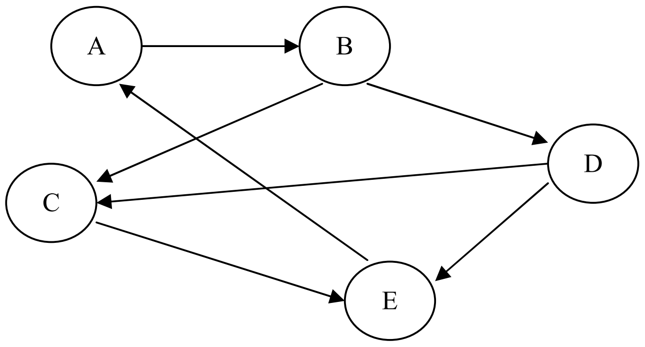


Figure 3-16. Pre-designed interface and information flow.

In such architecture, remarkable human intervention is required when knowledge and business are subject to change. Data structures may be re-defined, interfaces and information flows may be modified, programs may be rewritten or new programs need to be added.

(2) Interact with standard interfaces

This is a popular method in today’s software development. A typical example is Web Service. In such architecture, a “central” component provides specific services through standardized interfaces. The service is designed based on pre-defined rules and requirements. It does not care who will use the service and how they will use it. Other components know exactly what the services mean, the semantics of the exchanged information, and the definitions of the interfaces, so they can access the services via standard calls, and get information returned that they need. In some cases, other components may need to access a registration center to discover the characteristics of the services (like looking up telephone numbers from the yellow pages). The following is an example of this architecture:

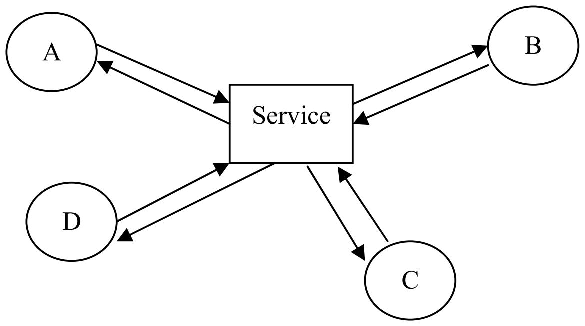


Figure 3-17. Interact with standard interfaces.

In such architecture, components can join or exit freely, which will never affect the functionality of the entire system as long as the services keep working. Human intervention can be reduced significantly. Only configuration specifications for the service side (server) and parameter settings on the accessing side (client) are required (here we do not consider the workload of developing the client components themselves). Any change in one client component will never have any impact on others. Flexibility and extensibility of the whole system are well supported.

(3) Establish interaction between anonymous components

This is an ideal status. In this architecture, no predefined interfaces and information flows are required. The system works based on its member components automatically finding other services, understanding them, and making use of them. In the following sample architecture, there is no central role and the curved arrows represent automatic interactions among components without human intervention. For example, an application needs to find the lowest price for a specific type of car for a customer through the Internet, and it will try to contact websites that offer the price information (the websites are changing, e.g. new ones coming and old ones stopping running), gather information, sort them, then determine the result and return it to the customer. This scenario depends heavily on semantic descriptions provided for each system's information. The interactions occur in an arbitrary manner.

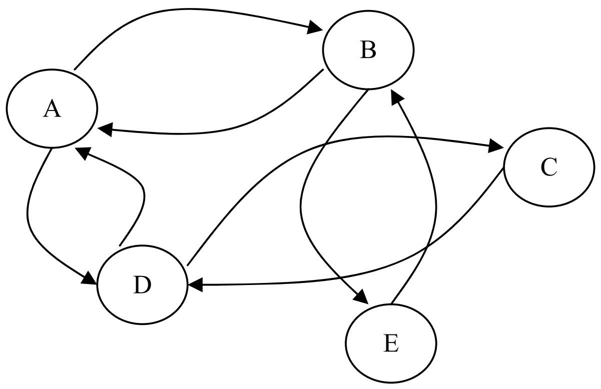


Figure 3-18. Establish interaction between anonymous components.

It looks like a kind of peer-to-peer system but it's not the same. In a typical peer-to-peer system, the interfaces and semantics of information exchanged among peers are strictly determined before the system starts working. What makes such systems flexible is that they allow any peer to join freely to provide service or exit freely at any time without crashing the systems. However, what we emphasize in a semantic integration problem is that there is no pre-defined interface and information semantics.

Actually, to make such systems work, initial human interventions are still required, but it can be minimized. For example, if *A* needs to interact with *B*, only very basic information like the IP address and port number of *B* should be provided by developers or users. Then, *A* will intelligently discover the semantics of the services provided by *B*, learn the manner to communicate with *B*, and cooperate with *B* to carry out some tasks. Note that in this case some common agreements are still necessary for the components to understand each other, such as some basic definitions for the concepts and business logics in a specific domain.

The mechanism discussed above looks like UDDI [14]. However, there are still differences. Traditional UDDI technology focuses on a standard interface definition. From the definition the applications can only get to know how to invoke a service. The semantics of the service itself, the invoking parameters, and the returned values

[14] http://www.uddi.org/pubs/uddi_v3.htm

remain unaware for the applications. Human interventions are required to interpret the service and develop applications that really "understand" the semantics.

The interactions between applications require a supporting environment, which tries to eliminate semantic conflicts, facilitate converting the information with semantics outside of the applications and minimize the possible modification to them. From the viewpoint of implementation, we have to develop a semantic integration mechanism that is accessible for all applications, as shown in the following figure:

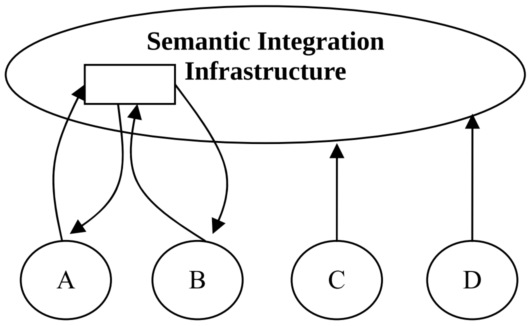


Figure 3-19. Infrastructure for semantic integration.

The rectangle between *A* and *B* acts as a translator to execute the necessary conversion for the input and output of *A* and *B* based on their semantics. The simplest case is, if *A* output speed data in *Miles*/*Hour*, and *B* can only receive a speed data in *Kilometers*/*Hour*, then the translator will do the calculation on the exchanged data to integrate semantics of *A* and *B*. Both *A* and *B* don't need any modification to themselves.

3.1.4.5 Information Context and Semantic Integration

Context plays an important role in information exchange and semantic integration. According to the American Heritage Dictionary, context is (1) the part of a written or spoken statement in which a word or passage at issue occurs and that often specifies its meaning; (2) the circumstances or situations in which a particular event occurs.

Context information [Goh, et al., 1994] of a subject contains information concerning its meaning made by the person or organization owning this subject, and provides the basis for determining the relationships between the subject and the real world aspects it describes. In most cases, the context information is given only implicitly, i.e., it is in the minds of the responsible designer, is specified in textual documentations not available externally, or is reflected in the local applications operating on the corresponding information [Bornhövd, 1998]. The context information is usually lost when information is exchanged across organizational boundaries, and thus, should be made available explicitly as some kind of meta-data.

Therefore, when processing specific information, the statements we make are usually imprecise and they can become correct and meaningful only if they are understood with reference to an underlying context which embodies a number of hidden assumptions. This anomaly is amplified in databases due to the gross simplifications that were made in creating a database schema. For example, a database may contain the schema

```
Employee
    Name: string
    Salary: decimal
```

and a record (*Tom*, *2000*). Without explaining what "*2000*" means (the attribute name "*Salary*" provides some semantics but not enough), e.g., what currency and scale-factor is used, what is the periodicity (daily, weekly, or monthly wage?), or what constitutes the person's salary (does it include year-end bonuses? What about the overtime pay?), we cannot get the accurate and correct understanding about this number.

In information systems, the context of information can be:

- Broad sense: anything other than the concept itself can be its context. For example, in a semantic network, all elements other than the concept consist of its context. In this sense, if we have two representations r_1 and r_2 but don't know their semantic relationship, context may provide some help. For instance, assuming that we know their context c_1 and c_2, and we can understand c_1 and c_2, it is possible to derive some semantic relationships between r_1 and r_2. For example, using a rule "if the contexts of two representations are equivalent, then the two concepts are possibly equivalent", we can infer that r_1 and r_2 are equivalent.

- Narrow sense: a specific structure that provides an environment to enrich the semantics of a concept. For example, two money amounts: 500 and 500 cannot be determined equivalent to each other with merely the number. Given that we established context for them:
  A. 500 (context: currency = USD scale = dollar)
  B. 500 (context: currency = CAD scale = cent)
  we know that A and B are not equivalent. Differently, given that we have the following context:
  C. 500 (context: currency = USD scale = dollar)
  D. 550 (context: currency = CAD scale = dollar)
  we know that C and D refer to the same money (assuming that the exchange rate between USD and CAD is 1:1.1).

3.1.4.6 Ontology-driven Semantic Integration

In chapter 2 we presented some descriptions for ontology-driven semantic integration appearing in literature. In this section we further clarify this term.

The term "ontology-driven semantic integration" is often mentioned together with another term "ontology integration". In the ontology-related research, the term "ontology integration" means anything ranging from combining, merging, using, mapping, matching, aligning, extending, approximating, unifying, and more.

Sometimes these terms are used in an interchangeable manner as if all are synonyms. Actually there are minor differences between these terms if we dig deeply into their meanings. One common point for these terms is that all of them specify some kind of actions or operations on a set of available ontologies. In other words, they focus on the ontologies themselves. As a simplified understanding, ontology integration can be viewed as a process of building a new ontology reusing other available ontologies. To achieve this goal, the relationships such as equivalence and specialization between concepts within different ontologies should be identified by mapping, matching, or aligning.

Ontology-driven semantic integration focuses on semantic integration but uses ontologies as a vehicle of information semantics. An ontology can work as a vehicle since it specifies the semantics through certain structures under a given ontological commitment in a formal and explicit manner. Ontology-driven semantic integration is a mechanism to integrate information at the semantic level using the semantics carried by ontologies. Its purpose is to integrate information instead of integrating the ontologies. It is true that to achieve the integration some concepts and methodologies applied in ontology integration should be adopted, such as mapping, matching, or aligning the concepts.

There is one example that can show their difference well. Assuming that there are two ladders, one can find various ways to connect them into a higher one. This is like ontology integration. It is assumed that one needs to climb to the roof of a house and there are two ladders which heights are just half of the height of the house. Now the purpose is to climb to the roof, not connecting the ladders. But one needs to connect the ladders before the purpose can be achieved. Here the ladders are the vehicle for the purpose. The purpose is not connecting ladders but one still needs to use some methods to connect them. This is like ontology-driven semantic integration.

3.2 A Framework for Semantic Relationships

In the research of ontology integration, the term "semantic relationship" has two meanings, one is the relationships between concepts within an ontology that specify the semantics of concepts, e.g., *Teacher* instructs *Course*, and the other is the relationship between elements from different ontologies, e.g., *Faculty* in *Ontology 1* is equivalent to *Professor* in *Ontology 2*. In this research we take the second meaning.

A framework about what types of semantic relationships there are between different ontologies is necessary to design the semantic integration mechanism. In the following we examine two proposals.

[Li, et al., 2005] establishes a framework for semantic relationships based on concepts and their properties. In the ontology context, a concept has a set of properties that describe its characteristics, and usually has an identifier property that distinguishes each instance from others. It is feasible to compare two concepts by looking at the identifiers as well as other properties.

[Li, et al., 2005] establishes three types of mutually exclusive semantic relationships between existing concepts from different ontologies. We assume that ontology O_i and O_j are in the same domain ($i, j \in N$, where N is the set of natural numbers). $C_i(O_i)$ denotes the set of all concepts within O_i. c_i and c_j are two concepts from the two ontologies, $c_i \in C_i(O_i)$ and c_j where $c_j \in C_j(O_j)$.

Equivalent: two concepts are semantically equivalent, if $\exists$ c_i, c_j, s.t. $c_i \sim c_j$. Namely, these two concepts: (1) have the same denotation names which have the same meaning; (2) are synonyms (two different words that can be interchanged in a context); or (3) their properties are the same or largely overlap.

Inclusive: two concepts are semantically inclusive, if $\exists$ c_i, c_j, s.t. $c_i \leq c_j$ (e.g., c_i is a kind of c_j, or, c_i is a specialization of c_j) or $c_i \geq c_j$ (e.g., c_j is a kind of c_i, or c_i is a generalization of c_j). Namely, the properties of one concept are also the properties of the other. The specialization relationship is also referred to as a hyponym, which is a word that is more specific than a given word. The generalization relationship is referred to as a hypernym, which is a word that is more generic than a given word.

Disjoint: two concepts are disjoint, if $\exists$ c_i, c_j, s.t. $c_i \cap c_j = \varnothing$. Namely, there is no common property between them.

Bouquet et al. [Bouquet, et al., 2003] identified five types of semantic relationships: equivalent to, less general than, more general than, compatible with, and incompatible with.

We adopt a framework proposed in [Rizopoulos, 2004] which includes five types of relationships to describe how two concepts from different sources are related to each other. The framework takes instances of concepts into consideration. We use Dom(C) to denote the domain of a concept C, i.e., the set of all possible valid instances of C. The types are:

(1) Equivalence: Two concepts C_1 and C_2 are equivalent, denoted as $C_1 \equiv C_2$, if and only if

$$\text{Dom}(C_1) = \text{Dom}(C_2).$$

(2) Subsumption: Concept C_1 is a child concept of C_2, i.e. C_2 subsumes C_1, denoted as $C_1 \subset C_2$, if and only if $\text{Dom}(C_1) \subset \text{Dom}(C_2)$.

(3) Intersection: Two concepts C_1 and C_2 are intersecting, denoted as $C_1 \wedge C_2$, if and only if

$\text{Dom}(C_1) \cap \text{Dom}(C_2) \neq \varnothing$, $\text{Dom}(C_1) \not\subset \text{Dom}(C_2)$, $\text{Dom}(C_2) \not\subset \text{Dom}(C_1)$, and $\exists C$: $\text{Dom}(C_1) \cap \text{Dom}(C_2) = \text{Dom}(C)$.

(4) Disjointness: Two concepts C_1 and C_2 are disjointed, denoted as $C_1 \vee C_2$, if and only if

$$\text{Dom}(C_1) \cap \text{Dom}(C_2) = \varnothing, \text{ and } \exists C: \text{Dom}(C_1) \cup \text{Dom}(C_2) \subseteq \text{Dom}(C).$$

(5) Incompatibility: Two concepts C_1 and C_2 are incompatible, denoted as $C_1 \perp C_2$, if and only if

$$\text{Dom}(C_1) \cap \text{Dom}(C_2) = \varnothing, \neg\exists C: \text{Dom}(C_1) \cup \text{Dom}(C_2) \subseteq \text{Dom}(C).$$

The framework is defined based on concepts instances. By instance we mean two aspects: the first one is the actual entities existing in the world, either physically (e.g., a person, a car, or a dog) or abstractly (e.g., weight, height, or time), and the second one is the digital representations of the actual entities in information systems. In the information system context, what we manipulate is just information represented digitally but not the actual entities, therefore we merely focus on the digital representations. Furthermore, it is impossible to enumerate all instances of a concept (even the digital representations) and compare them with instances of another concept. Therefore, we mainly work on the analysis of the representations of the concept models that abstract and specify the concepts themselves and try to discover relationships among these model representations.

We focus on the equivalence relationship. At the concept model level, one of the challenges to solve is: given different representations of concept models from multiple sources (information systems), discover whether they are referring to the same concept model. For example, a relational table schema in a relational database is a representation of a model, which is modeling a specific concept following the relational model theory. In a distributed environment, given some table schemas from various sources, they may have different table names, different column numbers and different column names, but it is possible that they are representations of the same model for a specific concept. This idea is illustrated in Figure 3-1 (section 3.1.1), i.e., given C and D, answer the question that whether they both represent the same model E (Frame Model 1).

In Figure 3-1 we illustrate a concept *Professor* as well as other concepts *Student* and *Course* associated through some semantic relationships. From a more general point of view, in this Figure C, D, and E (each one is an *object* that we deal with) also have semantic relationships, e.g., C represents E and D represents E, therefore C is equivalent to D, or, if we use a name to identify the model representation, **Professor** is equivalent to **Prof.**. This relationship can be extended to the property level, i.e., **Professor.Name** (denotes the **Name** property of **Professor**) is identical to **Prof..Name**, and **Professor.Publication** is identical to **Prof..Papers**. Such relationships are useful for exchanging information between systems. For instance, after identifying that **Professor** is equivalent to **Prof.**, it is possible to convert an instance representation of **Professor** to the one of **Prof.** while preserving the information semantics.

Besides the equivalence relationship which is defined at the concept level, another type of relationship which is defined at the property level, namely *functional relationships*, is also important for information exchanging while preserving semantics. Given two concept model representations R_1 and R_2 that are representing the same concept model, P_1 and P_2 are property sets from R_1 and R_2, a functional relationship f between P_1 and P_2 is a function that matches P_1's instance values to P_2's instance values through some functional operations, such as mathematical computations or string processing. A common example is the person name, for example, in a table $\mathbf{T_1}$, a column **Name** represents the full name of a person, but in another table $\mathbf{T_2}$, two columns **First_Name** and **Last_Name** are used to represent the full name jointly, therefore a functional relationship is defined for $\mathbf{T_1}$ and $\mathbf{T_2}$ in the form of f: $\mathbf{T_1}$.**Name** = *concatenate*($\mathbf{T_2}$.**First_Name**, *B*.**LastName**) , where *concatenate* represents a string operation.

3.3 Ontology and Ontological View

As discussed in chapter 2, ontology-driven semantic integration is one of the solutions for the semantic integration problem. The traditional solutions are based on available ontologies. Ontology integration can be applied by discovering semantic correspondences among a set of formal ontologies and, sometimes, creating a more complete ontology, given that multiple original ontologies are available. However, in many domains, especially where lots of traditional information systems have been deployed, this prerequisite cannot be met. Instead, the "ontologies" are implied in a different format, such as the underlying information models. For example, a database-centralized information system may work based on a relational database schema. The schema is not a formal ontology but to some extent it specifies the semantics of information that it manages. The schema contains multiple tables and each table can represent a concept. Accordingly, data rows in a table represent instances of the concept. Furthermore, there is no widely-accepted and explicit "domain ontology". The information systems were not built based on the domain ontology, even though they are committed to the same domain.

In these domains, each information model actually reflects a specific conceptual view of the domain conceptualization and is implicitly defining an *ontological view*. In the following sections we will provide the formal definition for *ontological view*. The definition is based on the work of [Guarino, 1998] that is necessary for formally defining *ontology*.

(1) World, Concept, Domain and Possible World

The *World* is the entire aggregation of everything that exists anywhere. The existing things in the world are perceived as *Concepts*. A *Domain* is a portion of the world that is related to a problem to be solved. Formally, a domain D is defined as a set of *concepts* that exist in the domain, i.e., D = $\{C_1, C_2, \ldots, C_n\}$ where each C_i is a concept, $1 \leq i \leq n$.

A *state of affairs* describes a possible situation about how concepts are related to each other. A state of affairs is a certain type of proposition. It is said to *obtain or not* where the proposition is said to be true or false [Menzel, 2008]. A state of affairs is said to *include* a second state of affairs if it is impossible for the former to obtain and the latter to fail to obtain. A state of affairs is said to *preclude* a second state of affairs if it is impossible for them both to obtain. A state of affairs is called *maximal* if, for every other state of affairs, it either includes or precludes that other state of affairs [Plantinga and Davidson, 2003 and Tomberlin and van Inwagen, 1985]. A *maximal state of affairs* is also called a *possible world*. The set of maximal states of affairs of a domain is denoted as W, W = $\{w_1, w_2, \ldots, w_m\}$ where each $w_i \in$W is a maximal state of affairs (possible world).

For example, we consider two concepts *University* and *Student*. One state of affairs is *Student part-time-study-in University*, and another one is *Student full-time-study-in University*. Since each of them precludes another one, i.e., if a student is part-time studying in a university, he is not a full-time student; on another hand, if a student is full-time studying in a university, he is not a part-time student; they compose two possible worlds.

(2) Domain Space and Conceptual Relation

A *domain space* is a structure <D, W>, where D is a domain and W is a set of maximal states of affairs of the domain. Given a domain space <D, W>, a *conceptual relation* ρ^n of arity *n* is a function from a set W of possible worlds to the set of all *n*-ary relations on D, 2^{D^n}, ρ^n : W$\rightarrow 2^{D^n}$.

(3) Conceptualization

A *conceptualization* of domain D is defined as an ordered triple **C** = <D, W, $\Re$>, where $\Re$ is a set of conceptual relations on the domain space <D, W>.

(4) Intended Structure

For each possible world $w\in$W, the *intended structure* of *w* according to a conceptualization **C** = <D, W, $\Re$> is the structure $\mathbf{S}_{w\mathbf{C}}$ = <D, $\mathbf{R}_{w\mathbf{C}}$>, where $\mathbf{R}_{w\mathbf{C}} = \{\rho(w) \mid \rho\in\Re\}$ is the set of extensions (relative to *w*) of the elements of $\Re$. We use $\mathbf{S}_\mathbf{C}$

= $\{\mathbf{S}_{w\mathbf{C}} \mid w \in \mathrm{W}\}$ to denote all the intended structures (or intended world structures) of **C**.

(5) Logical Language

A *logical language* **L** is a composition of a vocabulary V and a set of models of the language. V contains constant symbols and predicate symbols. Given a logical language **L** with a vocabulary V, a *model* of **L** is a structure <**S**, I>, where **S** = <D, **R**> is a world structure and I: $\mathrm{V} \rightarrow \mathrm{D} \cup \mathbf{R}$ is an interpretation function assigning elements of D to constant symbols of V, and elements of **R** to predicate symbols of V. A model fixes a particular extensional interpretation of the language.

Further discussion about logical languages can be found in [Shapiro, 2006].

(6) Intensional Interpretation

An *intensional interpretation* of a language **L** with a vocabulary V is a structure <**C**, $\Im$>, where **C** = <D, W, $\Re$> is a conceptualization and $\Im$: $\mathrm{V} \rightarrow \mathrm{D} \cup \Re$ is a function assigning elements of D to constant symbols of V, and elements of $\Re$ to predicate symbols of V. This intensional interpretation is called *ontological commitment* for **L**, denoted as **K** = <**C**, $\Im$>. If **K** = <**C**, $\Im$> is an ontological commitment for **L**, we say that **L** commits to **C** by means of **K**, where **C** is the underlying conceptualization of **K**. **K** constrains the intensional interpretation of **L**, i.e., the language is used in an intended way for a domain instead of an arbitrary way.

In definitions (5) and (6), both I and $\Im$ assign elements of D to constant symbols of V. The difference is that I assigns elements of **R** to predicate symbols of V while $\Im$ assigns elements of $\Re$ to predicate symbols of V. As an example, we assume that in a domain we have concepts *Student* and *Professor* and professors can teach students. We use *S*, *P* to represent the concepts and *t* to represent the relationship. Here we need to view *S*, *P*, and *t* as pure formal symbols to illustrate the conceptualization, independent of any specific language. Therefore, we have D = {*S*, *P*}, a possible world $w = (P\ t\ S)$, a conceptual relation t such that $t(w) = (P, S)$. A world structure $S_{wC} = \langle D, R_{wC} \rangle$ where $R_{wC} = \{(P, S)\}$.

Assuming that we select English as the language **L** to model the conceptualization, and select a vocabulary V containing words {*Student*, *Profess*, *teach*}, then an interpretation function of **L** maps the predicate symbol "*teach*" to (P, S) (since (P, S) is an extension in terms of the specific world w), while the interpretation function of $\Im$ will map the predicate symbol "*teach*" to t instead of (P, S). In this simple sample it seems that t is equivalent to (P, S), but they actually are not. This can be seen from the following sample:

Assuming that we have concepts *Professor*, *Graduate Student* and *Undergraduate Student* in the domain and they are shortly denoted as *P*, *GS*, and *US*. The fact is, professors can teach both graduate students and undergraduate students, and graduate students (as teaching assistances) can teach undergraduate students. Therefore, we have one possible world w and $R_{wC} = \{(P, GS), (P, US), (GS, US)\}$. In this example we see that R_{wC} can be more complex but t remains the same. A similar example from the mathematical domain is the interpreting of "square computation". A model interprets it as an extensional relation {(1, 1), (2, 4), (3, 9), …} while an intensional interpretation is a formula $y = x^2$. In summary, I maps a predicate symbol to the extension of the conceptual relation, and $\Im$ maps it to the intended meaning of the conceptual relation. The difference between I and $\Im$ can be illustrated with the following Figure 3-11:

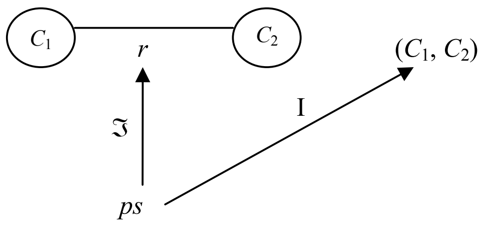


Figure 3-20. Difference between I and $\Im$.

In this figure C_1 and C_2 are concepts and r is a conceptual relation. r maps a possible world to a set of 2-ary relations $\{(C_1, C_2)\}$. $\Im$ maps a predicate symbol *ps* to r and I maps *ps* to an extensional relation of r, (C_1, C_2). In the following discussion, we also simply represent I as an arrow from a predicate symbol to a conceptual relation if no confusion will arise.

(7) Compatible

Given a language **L** with a vocabulary V and an ontological commitment $\mathbf{K} = <\mathbf{C}, \Im>$ for **L**, a model <**S**, I> is *compatible* with **K** if: i) $\mathbf{S} \in \mathbf{S_C}$; ii) for each constant symbol $c \in V$, $I(c) = \Im(c)$; iii) there exists a world w such that for each predicate symbol $p \in V$, I maps such predicate into an admittable extension of $\Im(p)$, i.e. there exists a conceptual relation ρ such that $\Im(p) = \rho \wedge \rho(w) = I(p)$.

(8) Intended Model

Given a language **L** and an ontological commitment **K**, the set $\mathbf{I_K(L)}$ of all models of **L** that are compatible with **K** is called the *set of intended models* of **L** according to **K.**

To illustrate this definition, we assume that multiple concepts are related to each other in a domain. If some concepts can be used as properties of other concepts, they are related through the "*hasProperty*" relationship. Here, just view "*hasProperty*" as a representation for the fact that a concept has a property and does not take it as a phrase from a specific language (English).

To model things, we need to use *language*. A language is not necessarily a natural language that humans use daily such as English; instead, it can be any form, such as text, voice, image, gesture, etc. Given a language **L**, **L** should be complete, i.e., it can model anything for a conceptualization. Since we use language to model things and the language is complete, it can be concluded that in the intentional interpretation <**C**, $\Im$>, $\Im$ is complete. That is, the vocabulary V of **L** is complete and the interpretation is complete, i.e., for any concept in **C**, $\Im$ assigns a constant symbol in V to it and for any conceptual relation in **C**, $\mathfrak{I}$ assigns a predicate symbol to it. On the contrary, for any constant symbol in V, $\mathfrak{I}$ assigns it to a concept and for any predicate symbol in V, $\mathfrak{I}$ assigns it to a conceptual relation.

Differently, a model of a language does not guarantee the completeness, which means that it may just interpret a portion of the domain with a portion of the language. In other words, a model of a language assigns some concepts in the domain to some constant symbols in V and assigns some conceptual relations to some predicate symbols in V.

We illustrate the discussion above with the following Figure 3-12:

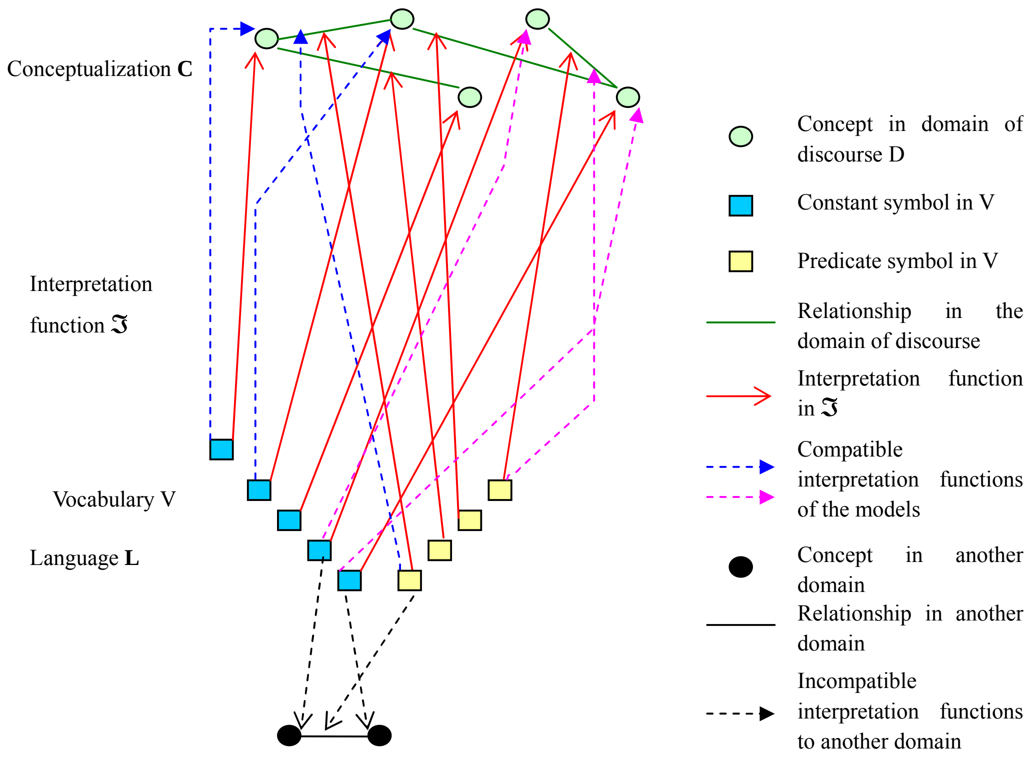

Conceptualization **C**
Interpretation function $\mathfrak{I}$
Vocabulary V
Language **L**
Concept in domain of discourse D
Constant symbol in V
Predicate symbol in V
Relationship in the domain of discourse
Interpretation function in $\mathfrak{I}$
Compatible interpretation functions of the models
Concept in another domain
Relationship in another domain
Incompatible interpretation functions to another domain

Figure 3-21. Difference between interpretation functions from I and $\mathfrak{I}$.

In the above figure, the blue and purple-dashed arrows represent the interpretation functions of two models. According to the definition, these two models are compatible with **K**. The black-dashed arrows represent an interpretation function which is not compatible with **K** since it interprets the symbols to concepts and relationships in another domain. For example, we assume that there is a domain with only two concepts *Professor* and *Student* (*P* and *S*) and a complete language with a vocabulary {*Stu.*, *Pro.*}. An interpretation function may map *Stu.* to *Studio* and *Pro.* To *Professional*, which are two concepts in another domain, therefore this interpretation function is incompatible with **K**.

Following we prove that given one conceptualization, one language, and one ontological commitment, there should be only one set of intended models.

Lemma 1: Given one conceptualization **C** = <D, W, $\mathfrak{R}$>, one language **L** with vocabulary V, and one ontological commitment **K** = <**C**, $\mathfrak{I}$>, there is only one set of intended models of **L** according to **K**.

Proof: Assuming that we have two sets of intended models $\mathbf{I_K(L)}_1$ and $\mathbf{I_K(L)}_2$, $\mathbf{I_K(L)}_1$ and $\mathbf{I_K(L)}_2$ are different. Then, there should be at least one model M which is compatible with **K**, $M \in \mathbf{I_K(L)}_1$ but $M \notin \mathbf{I_K(L)}_2$. Since $M \in \mathbf{I_K(L)}_1$, according to the definition, M is compatible with **K**. According to the definition again, M should be an element of $\mathbf{I_K(L)}_2$ because $\mathbf{I_K(L)}_2$ is composed of all models of **L** that are compatible with **K**. Therefore, such M cannot exist, which means $\mathbf{I_K(L)}_1 = \mathbf{I_K(L)}_2$. □

Following, we prove that for two conceptualizations, if their intended models overlap, the overlapped part is the shared concepts and shared properties. To simplify the problem, here we only consider concepts and one type of relationships associating them with each other: *has-property* (a concept can be a property of another concept).

Lemma 2: Given two conceptualizations $\mathbf{C}_1$ = <D_1, W_1, $\mathfrak{R}$> and $\mathbf{C}_2$ = <D_2, W_2, $\mathfrak{R}$>, one language **L** with vocabulary V, and two ontological commitments $\mathbf{K}_1$ = <$\mathbf{C}_1$, $\mathfrak{I}_1$> and $\mathbf{K}_2$ = <$\mathbf{C}_2$, $\mathfrak{I}_2$>, $\mathfrak{R}$ contains only one conceptual relation ρ meaning a concept has another concept has a property, if the two sets of intended models for $\mathbf{C}_1$ and $\mathbf{C}_2$ overlap, then the overlapped part consists of the shared concepts and shared properties.

Proof: Let $D_1 = \{d_{1i}\}$, $1 \le i \le n$; $D_2 = \{d_{2j}\}$, $1 \le j \le m$; $\mathbf{I}_{K1}(\mathbf{L}) = \{M_{1i}\}$, $1 \le i \le k$; $\mathbf{I}_{K2}(\mathbf{L}) = \{M_{2j}\}$, $1 \le j \le l$. For each $1 \le i \le k$, $M_{1i} = <\mathbf{S}_{1i}, I_{1i}>$, $\mathbf{S}_{1i} = <D_1, \mathbf{R}_{1i}>$; for each $1 \le j \le l$, $M_{2j} = <\mathbf{S}_{2j}, I_{2j}>$, $\mathbf{S}_{2j} = <D_2, \mathbf{R}_{2j}>$.

If $\mathbf{I}_{K1}(\mathbf{L}) \cap \mathbf{I}_{K2}(\mathbf{L}) \neq \emptyset$, then

$\mathbf{I}_{K1}(\mathbf{L}) \cap \mathbf{I}_{K2}(\mathbf{L}) = \{M_{1i}\} \cap \{M_{2j}\} = \{<\mathbf{S}_{1i}, I_{1i}>\} \cap \{<\mathbf{S}_{2j}, I_{2j}>\}$, $1 \le i \le k$ and $1 \le j \le l$.

$<\mathbf{S}_{1i}> \cap <\mathbf{S}_{2j}> = <D_1, \mathbf{R}_{1i}> \cap <D_2, \mathbf{R}_{2j}> = <D_1 \cap D_2, \mathbf{R}_{1i} \cap \mathbf{R}_{2j}>$.

Since $\mathbf{I}_{K1}(\mathbf{L}) \cap \mathbf{I}_{K2}(\mathbf{L}) \neq \emptyset$, $<\mathbf{S}_{1i}> \cap <\mathbf{S}_{2j}>$ is not empty, i.e., $D_1 \cap D_2 \neq \emptyset$ and $\mathbf{R}_{1i} \cap \mathbf{R}_{2j} \neq \emptyset$. $D_1 \cap D_2 \neq \emptyset$ means that there are common concepts in the two conceptualizations. Because $\mathbf{R}_{1i} = \{\rho(w) \mid w \in W_1\}$, $\mathbf{R}_{2j} = \{\rho(w) \mid w \in W_2\}$ and here ρ is the *has-property* conceptual relation, $\mathbf{R}_{1i} \cap \mathbf{R}_{2j} \neq \emptyset$ means that there exist relations $\{(d_a, d_b) \mid d_a \in D_1 \wedge d_a \in D_2 \wedge d_b \in D_1 \wedge d_b \in D_2\}$. That is, each d_b is a shared property of the shared concept d_a. □

For example, we consider a domain where we have concepts *P*, *N*, *D*, *T*, *A*, *S* and relationship *h*, meaning that in this domain *Professor* can have property *Name*, *Degree*, *Title*, *Address*, and *Salary*. Now we select English as the language to model the domain and we pick a vocabulary V = {*Professor*, *Name*, *Degree*, *Title*, *Address*, *Salary*, *hasProperty*} where *hasProperty* is a predicate symbol and others are constant symbols. So, we have $D = \{P, N, D, T, A, S\}$ and one conceptual relation $\rho = h$, therefore $\Re = \{\rho\}$. Here we have only one possible world *w* saying that a professor can have these properties.

A model of the language $M_1 = <\mathbf{S}, I_1>$, where $\mathbf{S} = <D, \mathbf{R}>$ and $\mathbf{R}$ is the resulting relation of applying ρ to *w*, so $\mathbf{R} = \{(P, N), (P, D), (P, T), (P, A), (P, S)\}$. Since I_1 assigns elements of **R** to predicate symbols in V, which is *hasProperty*, we assume that I_1 is defined as $I_1(hasProperty) = \{(P, N), (P, D), (P, T)\}$ since this model focuses on academic aspects of a professor. Similarly, another model $M_2 = <\mathbf{S}, I_2>$ and I_2 is defined as I_2(*hasProperty*) = {((*P*, *N*), (*P*, *A*), (*P*, *S*)} since this model focuses on the administrative aspects of a professor. Since both M_1 and M_2 are compatible with **K**, they are intended models of **L** according to **K** and $\mathbf{I_K(L)} = \{M_1, M_2\}$. In this case, if two information systems commit to the same conceptualization and they use the same vocabulary, they can agree with each other since the symbols have a consistent interpretation.

Now we assume that we have two conceptualizations, $\mathbf{C}_1$ and $\mathbf{C}_2$: $\mathbf{C}_1 = <D_1, W_1, \Re_1>$, $D_1 = \{P, N, D, T\}$, w_1 corresponds to "*Professor* has property *Degree* and *Title*", $W_1 = \{w_1\}$, and $\Re_1 = \{h\}$. Similarly, $\mathbf{C}_2 = <D_2, W_2, \Re_2>$, $D_2 = \{P, N, A, S\}$, w_2 corresponds to "*Professor* has property *Address* and *Salary*", $W_2 = \{w_2\}$, and $\Re_2 = \{h\}$. Given the same language **L** and vocabulary V = { *Professor*, *Degree*, *Title*, *Address*, *Salary*, *hasProperty* }, let $\mathbf{K}_1 = <\mathbf{C}_1, \Im_1>$, where $\Im_1$(*Professor*) = *P*, $\Im_1$(*Name*) = *N*, $\Im_1$(*Degree*) = *D*, $\Im_1$(*Title*) = *T*, and $\Im_1$(*hasProperty*) = *h*. Similarly, we have $\mathbf{K}_2 = <\mathbf{C}_2, \Im_2>$ where $\Im_2$(*Professor*) = *P*, $\Im_2$(*Name*) = *N*, $\Im_2$(*Address*) = *A*, $\Im_2$(*Salary*) = *S*, and $\Im_2$(*hasProperty*) = *h*. A model $M_1 = <\mathbf{S}_1, I_1>$ where $\mathbf{S}_1 = <D_1, \mathbf{R}_1>$, $\mathbf{R}_1 = \{(P, N), (P, D), (P, T)\}$, I_1(*Professor*) = *P*, I_1(*Name*) = *N*, I_1(*Degree*) = *D*, I_1(*Title*) = *T*, and I*1*(*hasProperty*) = {(*P*, *N*), (*P*, *D*), (*P*, *T*)}. Since M_1 is the only compatible model with $\mathbf{K}_1$, so $\mathbf{I}_{\mathbf{K}1}(\mathbf{L}) = \{M_1\}$. Similarly, we have $M_2 = <\mathbf{S}_2, I_2>$ where $\mathbf{S}_2 = <D_2, \mathbf{R}_2>$, $\mathbf{R}_2 = \{(P, N), (P, A), (P, S)\}$, I_2(*Professor*) = *P*, I_2(*Name*) = *N*, I_2(*Address*) = *A*, I_2(*Salary*) = *S*, and I_2(*hasProperty*) = {(*P*, *N*), (*P*, *A*), (*P*, *S*)}. Also, $\mathbf{I}_{\mathbf{K}2}(\mathbf{L}) = \{M_2\}$.

Now we look at the intersection of the two sets of intended models.

$\mathbf{I}_{\mathbf{K}1}(\mathbf{L}) \cap \mathbf{I}_{\mathbf{K}2}(\mathbf{L}) = \{M_1\} \cap \{M_2\} = \{<\mathbf{S}_1, I_1>\} \cap \{<\mathbf{S}_2, I_2>\}$,

$<\mathbf{S}_1> \cap <\mathbf{S}_2> = <D_1, \mathbf{R}_1> \cap <D_2, \mathbf{R}_2> = <\{P, N\}, \{(P, N)\}>$,

$I_1 \cap I_2$ = {I(*Professor*) = *P*, I(*Name*) = *N*, I(*hasProperty*) = *h*}.

Therefore, $\mathbf{I}_{\mathbf{K}1}(\mathbf{L}) \cap \mathbf{I}_{\mathbf{K}2}(\mathbf{L})$ = {<<{*P*, *N*}, {(*P*, *N*)}>, { I(*Professor*) = *P*, I(*Name*) = *N*, I(*hasProperty*) = *h* }>}. The interaction, i.e., the overlap of two sets of intended models, is the shared concepts as well as their shared properties. This means, since the two conceptualizations have overlapping, their intended models also overlap and the overlapping part consists of the shared concepts and shared properties, i.e., in this part the language has the same interpretation. Finally, this guarantees that the two conceptualizations can be integrated and it is possible that the information systems based on the two conceptualizations can communicate with each other.

(9) Ontology

Given a language **L** with ontological commitment **K**, an *ontology* for **L** is a set of axioms designed in a way such that the set of its models approximates as much as possible the set of intended models of **L** according to **K**.

The relationships between language, conceptualization, ontological commitment, and ontology are illustrated in the following Figure 3-13.

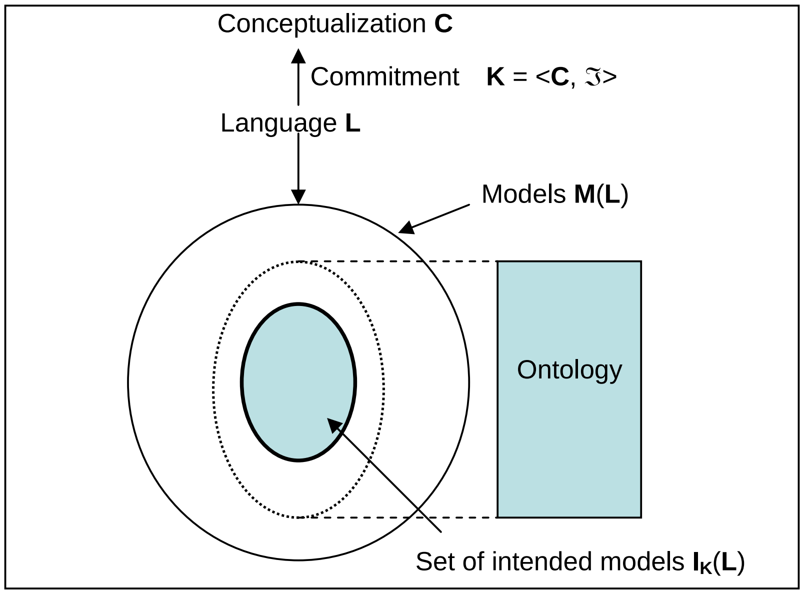

Conceptualization **C**
Commitment $\mathbf{K} = <\mathbf{C}, \Im>$
Language **L**
Models **M**(**L**)
Ontology
Set of intended models $\mathbf{I_K}(\mathbf{L})$

Figure 3-22. Language, conceptualization, ontological commitment, and ontology [Guarino, 1998].

(10) Ontological View

The above definition leads to an illusion that for one conceptualization there is one *single* ontology. However, this is not true since an "ontology" is a human-designed artifact, i.e., a type of model of the abstract conceptualization. When different designers are facing the same conceptualization, it is natural that multiple models will be created. Each model reflects a specific view of the conceptualization. Since the conceptualization can be viewed in various ways, actually there is not merely one unique "ontology" for it. Instead, different views of the conceptualization may exist. Each view can be formally and explicitly specified and we define the corresponding specification as an *ontological view*. Accordingly, its intensional interpretation is called an *ontological commitment of view*. There can be multiple ontological views for a single conceptualization. As for information systems, each system implies an ontological view of the conceptualization of the domain that it is built for.

(11) Integrate-able

Different languages can be employed for the specification of ontological views. Further, if two languages are employed for ontological views with partially overlapping intended models, it is possible for the corresponding ontological views to be semantically integrated. Formally, given one ontological view O with intended models $\mathbf{I_K}(\mathbf{L})$ and another ontological view O' with intended models $\mathbf{I_{K'}}(\mathbf{L'})$, O and O' are integrate-able (denoted by ◊) if and only if $\mathbf{I_K}(\mathbf{L})$ overlaps with $\mathbf{I_{K'}}(\mathbf{L'})$. That is,

$$(\mathbf{I_K}(\mathbf{L}) \neq \mathbf{I_{K'}}(\mathbf{L'})) \wedge (\mathbf{I_K}(\mathbf{L}) \cap \mathbf{I_{K'}}(\mathbf{L'}) \neq \emptyset) \leftrightarrow (O \lozenge O')$$

This can be illustrated by the following Figure 3-14:

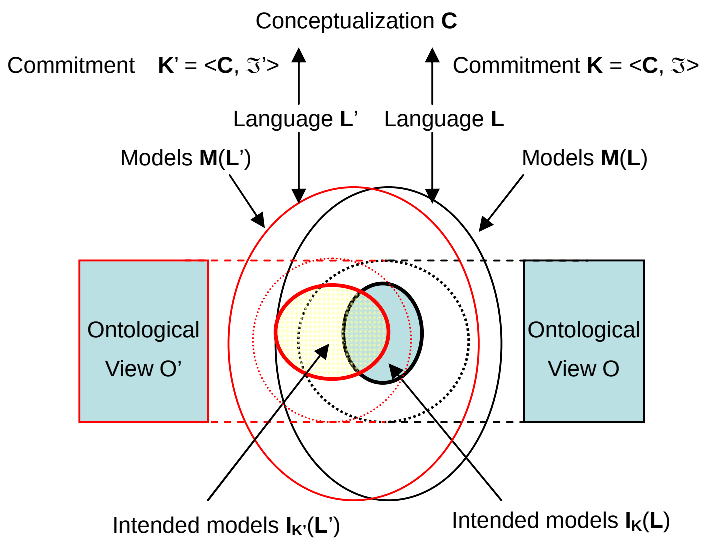


Figure 3-23. Different ontological views with different languages which sets of intended models overlap.

(12) Ontological View-driven Semantic Integration

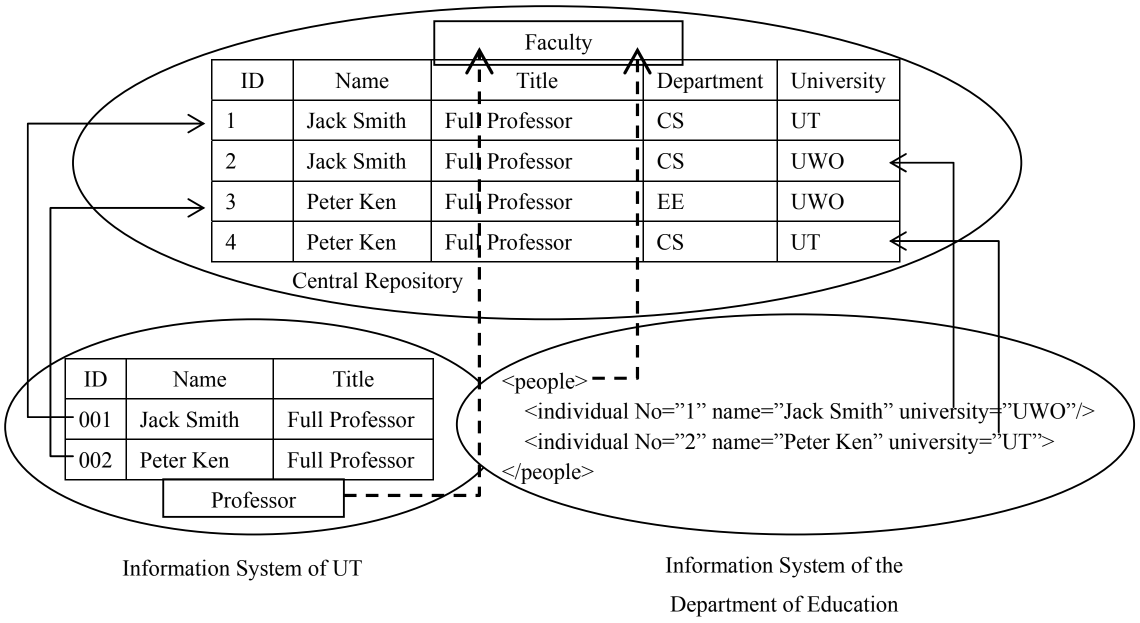
Ontology view-driven semantic integration is a mechanism to integrate information at the semantic level using the semantics carried by ontological views in a way that the overlapping parts, which mean the same concept references of the sets of intended models of multiple ontological views, are identified, modeled, persisted, and reused when performing information access and exchange.

3.4 Research Problem, Assumptions, and Hypothesis

3.4.1 A Case Study

Let's suppose that we are working in a domain *Education* and considering a real-world concept: *Faculty*. As human experts, we know exactly the meaning of the concept "*Faculty*" of a university department (note that here the concept from our conceptualization is identified by a unique name "*Faculty*") in the education domain, and we know that each concept has to be described by a set of properties. Let's assume that we determine a set of properties for the concept "*Faculty*": {*Name*, *Title*, *Department*, *University*} and the set is complete: no more properties are required. Each property has a clear meaning and is identified by a unique name.

Then, we assume that the information about four professors comes from two information systems. One information system is managed by a university UT (shortly named S_1), and the other one is maintained by the National Department of Education which manages many universities (shortly named S_2). Information in these systems denotes the same concept and reveals different instances (which may overlap) of that concept (*Faculty*), with different and independently adopted representations. In an ideal case, the information has been collected, cleaned, validated, normalized, and stored in a central information repository which owns a complete definition about "*Faculty*" and all instances, as depicted in the following Figure 3.15.



ID	Name	Title	Department	University
1	Jack Smith	Full Professor	CS	UT
2	Jack Smith	Full Professor	CS	UWO
3	Peter Ken	Full Professor	EE	UWO
4	Peter Ken	Full Professor	CS	UT

ID	Name	Title
001	Jack Smith	Full Professor
002	Peter Ken	Full Professor

```
<people>
   <individual No="1" name="Jack Smith" university="UWO"/>
   <individual No="2" name="Peter Ken" university="UT">
</people>
```

Figure 3-24. An integration scenario.

From this figure we can see that each instance of the concept has a unique identifier in the information system in which it resides. The identifiers are not helpful for the integration as they can just uniquely identify the entities in a technical sense in each information system but contain no business meanings. In the central repository another set of identifiers is assigned. From the central view, faculty with *ID* 1 and 3 come from the information system of UT, and faculty with *ID* 2 and 4 come from the information system of the Department of Education, as pointed out by the solid arrows. Even they share some identical values under some properties, such as *Name* and *Title*, we know that they are actually four different faculty instances, as implied by the unique *ID*s in the central repository.

However, note that information in both S_1 and S_2 is incomplete. For example, in S_1 information about *Department* is missing, and information about *University* is implicit. Similarly, in S_2 information about *Title* and *Department* is missing. Assuming that in some way we collected all the necessary information and put that in the central repository, we know that the information in this repository is the most complete and most accurate. Any answer we can get from this repository is perfect. If there is anything we cannot find from this repository, that "thing" actually does not exist.

This is the ideal case of semantic integration. It is more than semantic integration; in a sense it is actually a result of "physical" information integration. It can insure the most completeness, accuracy, and efficiency for any query issued to it.

Nevertheless, due to many technical, organizational, practical, legal, or business reasons, this solution is actually not applicable. For example, integrating so much information from various systems may result in a high cost of labour and performance pressure on the central server (such as the storage space and the query processing workload).

Going back, we consider a less ideal case; we don't maintain all information in a physically central repository, but keep it distributed. Then, some knowledge denoting the mapping from various information systems to the central repository can be discovered and maintained in the central repository. In this case, we define from where and how the required information comes instead of collecting the information itself. For example, we can have a piece of knowledge saying that "*Faculty*" can be a combination of "*Professor*" from S_1 and "*people*" from S_2, as depicted by the dashed arrows in Figure 3-15. This is a more feasible and applicable solution for the problem of information integration.

3.4.2 Problem Specification

As mentioned before, ontologies can provide much support for semantic integration (although this is not fully guaranteed). However, there are many cases where organizational, cultural, or infrastructural constraints hinder or even disallow the adoption of such semantic artifacts, i.e., there is a lack of explicit ontologies. In fact, the applications of ontologies pool mainly in several fields such as chemistry, biology, toxicology, environmental science, ecology, geography, etc., where much effort has been devoted to building ontologies to organize the rich knowledge in these fields. Information systems or integration systems in these fields can be built based on the available ontologies. Contrarily, lots of other information systems, such as traditional management information systems and E-commerce systems based on databases or flat data files, do not have pre-defined explicit ontologies at either domain level or application level, although to some extent each of them implements the (abstract and invisible) conceptualizations for the domains to which they belong through their internal mechanisms in terms of their information model, representation, storage, and processing.

In such cases, semantic integration at the information level is essential for the applications. Due to the lack of explicit ontologies (both the local ones and the global one), the recently developed ontology-based methodologies are not sufficient to support the integration of such systems. Therefore, new research is necessary to be conducted to bridge this gap.

Our research deals with *information systems*. An information system is a combination of an *information model* and a set of software components that operate the model. In this research we will ignore the software components and focus on the *information model* since we mainly consider information semantics.

Given a set of information models IM_1, IM_2, …, IM_n, their semantic integration includes two aspects:

(1) For any two elements e_i and e_j from IM_i and IM_j, $1 \le i, j \le n$ and $i \ne j$, if they refer to the same concept in terms of the domain of discourse, independent of the way they are represented, this fact can be discovered.

(2) For any element e_i from IM_i, $1 \le i \le n$, if it is required to be communicated to IM_j (if applicable), $1 \le j \le n$, it can be converted into another element (referring to the same concept) that is correct in both representation and semantics in IM_j such that IM_j can handle it in a semantically reasonable manner.

3.4.3 Short Summary on Conventional Solutions

In conventional schema matching-based information integration approaches, each information system has its own schema such as a database schema or a XML schema to represent its local conceptualization of a domain. The schemas can be understood and processed by computer-based applications. The matchings between different schemas are discovered by human experts or by automatic algorithms (note that usually the automatically discovered matches still require validation and confirmation from human experts) and are represented by some structure readable and operable by computers. Then, in an integration environment, if an information item *I* (following a modeling paradigm defined in IS_1) is required to be passed to system IS_2 from system IS_1, some mediator (a kind of software application) in the environment can get *I*, find the semantic relationship (here a specific schema matching) between *IS_1* and *IS_2*, convert it into a new representation *I'* following definition in *IS_2*, then pass it to *IS_2*. Now *IS_2* is able to correctly process *I'* since *I'* is following *IS_2*'s representation and is supposed to be denoting the same concept as *I*. In this category of solutions, there is a lack of semantics, i.e., two schema elements can be discovered to be similar and referring to the same concept, but it is unknown ***which concept*** they are referring to due to the lack of a concept model.

In ontology-driven information integration approaches, each information system has its own information model and explicitly represented ontology for its conceptualization of the domain. In some cases a global ontology for the domain can be used. Each system knows how to map the conceptual operation on its ontology to the structural operation on its internal model. Similarly, semantic mapping between ontologies can be discovered by human experts or by automatic algorithms (also requiring validation and confirmation from human experts) and stored in some way that computers can understand and process (such as mapping rules). Similarly, some software applications can handle these ontologies and semantic mappings to help involved systems achieve semantic integration. Ontology mapping or aligning techniques can be applied, but many valuable methods developed in schema matching cannot make a contribution, such as the instance-based methods (usually not many instances will be provided along with ontologies, even by definition ontologies can contain concept instances).

In many cases, the application of these approaches is limited due to the lack of explicit ontologies. Instead, schema-based approaches are more applicable because of the higher availability of information schemas.

3.4.4 Assumptions

Following lists a set of assumptions for this research. These assumptions are practical and reasonable. They provide a realistic foundation for the research and can help reduce the complexity of the problem.

- All of the information models are committed to intended models that overlap. This guarantees the possibility of semantic integration. However, the concepts and their relationships are not formally and explicitly modeled and represented.
- For each information system, there is an explicit information model that is used to organize the system's data and convert the data into information.

  The information models are not restricted to a particular modeling language or paradigm such as relational, XML, or Object-Oriented.
- The vocabularies used by the information models are based on natural languages.
- Based on each information model, an ontological view can be created.
  - An ontological view is an explicitly represented model.
  - The ontological view follows a specific modeling paradigm which is independent of the modeling paradigm adopted by the underlying information model.

3.4.5 Ontological Equivalence Mapping

In this research the semantic integration is conducted at the ontological view level. It is founded on a hypothesis. Before we present and prove the hypothesis, we formally define the *ontological equivalence mapping* between languages:

Given a source language $\mathbf{L}_S$ (which vocabulary is V_S) with an ontological commitment of view $\mathbf{K}_S = <\mathbf{C}, \Im_S>$ and a target language $\mathbf{L}_T$ (which vocabulary is V_T) with an ontological commitment of view $\mathbf{K}_T = <\mathbf{C}, \Im_T>$, the two languages share the same conceptualization $\mathbf{C} = <D, W, \Re>$; an *ontological equivalence mapping* is a function from V_S to V_T, m: $V_S \rightarrow V_T$ assigning symbols in V_T to the ones in V_S which share the same intensional interpretation, i.e., i) for constant symbols $c_S \in V_S$ and $c_T \in V_T$, $m(c_S) = c_T$ if and only if i) there exists a concept $d \in D$, such that $\Im_S(c_S) = \Im_T(c_T) = d$; ii) for predicate symbols $p_S \in V_S$ and $p_T \in V_T$, $m(p_S) = p_T$ if and only if there exists a conceptual relation $\rho \in \Re$ such that $\Im_S(p_S) = \Im_T(p_T) = \rho$.

It is obvious that an important task in semantic integration is to discover the ontological equivalence mapping between two ontological views, especially between the concepts within the ontological views.

The following Figure 3-16 illustrates such a mapping between languages:

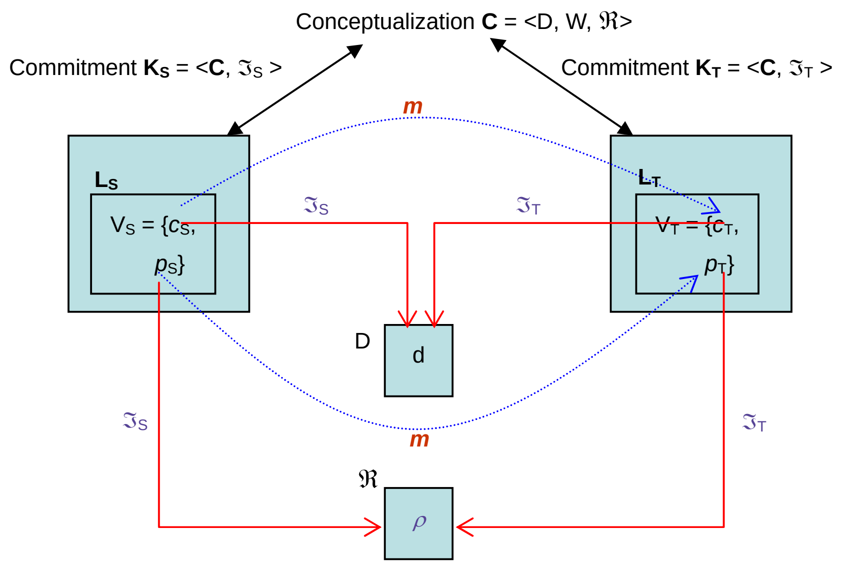

Conceptualization C = <D, W, ℜ>
Commitment K_S = <C, ℑ_S >
Commitment K_T = <C, ℑ_T >
m
L_S
V_S = {c_S, p_S}
L_T
V_T = {c_T, p_T}
ℑ_S
ℑ_T
D
d
ℑ_S
m
ℑ_T
ℜ
ρ

Figure 3-25. Ontological equivalence mapping between different languages for the same conceptualization.

The mapping can be bi-directional. If a symbol $s_S \in V_S$ is mapped to a symbol $s_T \in V_T$, we say that there is a *semantically equivalent relationship* (or *semantic equivalence relationship*) between s_S and s_T.

3.4.6 Hypothesis

In this context, we base our research on the following hypothesis:

If the semantically equivalent relationships between concepts (specified by symbols in languages) from multiple ontological views can be discovered, then these ontological views, as well as the information models from which the ontological views develop, can be semantically integrated.

To support this hypothesis, we introduce the following two propositions.

(1) A concept in a conceptualization can be externalized by a constant symbol in a language under an ontological commitment.

Prove:

According to the definition of the intended model, given a language **L** with an ontological commitment **K**, the set $\mathbf{I_K}(\mathbf{L})$ of all models of **L** that are compatible with **K** is defined as the set of intended models of **L** according to **K**. So, for any two models m_1 and m_2 in $\mathbf{I_K}(\mathbf{L})$, m_1 and m_2 are compatible with **K**. That is, for each constant symbol c in the vocabulary of **L**, there is $\mathbf{I}_1(c) = \mathfrak{I}(c)$ for m_1 where $\mathbf{I}_1$ is the interpretation function of m_1, and $\mathbf{I}_2(c) = \mathfrak{I}(c)$ for m_2, where $\mathbf{I}_2$ is the interpretation function of m_2, and $\mathfrak{I}$ is the interpretation function in **K**. That is, under the given ontological commitment **K** a constant symbol c is always interpreted as a concept in the domain of discourse.

On the other hand, it is guaranteed that c is interpreted as a single concept, e.g. C, under **K** since in any model **I** is a function. In other words, it is an explicitness of the intended model of concept C. Therefore, even C is implicit, c can be taken as its representative. c can be used for processing the concept that it represents since it is explicit. □

Based on this proof, it can be stated that the intended model of a concept can be made explicit by a constant symbol.

(2) The semantically equivalent relationship between symbols under an ontological commitment implies the same concept reference.

Prove:

Given symbols v_1 and v_2 from two ontological views such that v_1 maps to a concept c_1 in an intended model and v_2 also maps to a concept c_2 in another intended model (Proposition 1), if v_1 and v_2 have a semantically equivalent relationship, then they have the same semantics, i.e., the same concept reference. Therefore, it can be concluded that c_1 and c_2 are actually the same concept in the conceptualization. Consequently, information models corresponding to v_1 and v_2

are semantically equivalent. □

For example, if v_1 and v_2 are synonymous, it is already taken as a fact (by the definition of synonymy) that v_1 and v_2 mean the same thing, i.e. they refer to the same concept. Therefore, the information models corresponding to v_1 and v_2 are semantically integrated.

The first proposition indicates that each ontological view has a specific *representation* based on a language since the ontological view is an explicit model. The second proposition shows that the *semantic similarity* between representations of models can be used to approximate the semantic equivalence relationships between the models themselves. Semantic similarity is a metric upon explicitly represented models computed from a syntactical, structural, or instance perspective.

A semantic similarity metric is a combination $<A, t>$ where A is an approach to compute the similarity between symbols and t is a threshold. The approach A can be viewed as a function A: $S \times S \rightarrow R$ where S is the set of symbols and R is the set of real numbers. If $A(s_1, s_2) > t$, $s_1, s_2 \in S$, then it can be confidently believed that two symbols are semantically equivalent, i.e., s_1 and s_2 have a semantic equivalence relationship. Such a metric implies that two models may have the same semantics because their representations are syntactically or structurally similar to each other, or their instances are similar.

3.4.7 Formulating the Problem

Generally, the information models adopt vocabulary from natural languages such as English. The constant symbols such as English words refer to concepts under an ontological commitment. This work takes such a fact as an assumption, i.e., our work does not deal with the cases that random symbols are picked for the information models.

We adopt similar ideas in *schema matching* to formulate the problem. We try to discover semantic relationships between the elements, mainly the semantic equivalence relationship between concepts of multiple ontological views. Before doing this, the information models using different modeling paradigms and representations need to be converted to ontological views. The following Figure 3-17 illustrates this idea.

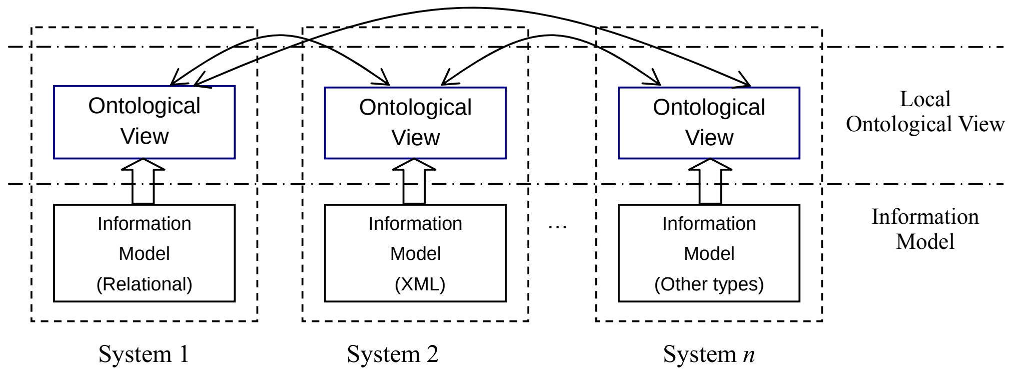


Figure 3-26. Semantic integration based on ontological views.

Given an ontological view O_1 with a set of concepts $C_1 = \{c_{11}, c_{12}, \ldots, c_{1n}\}$ and another ontological view O_2 with a set of concepts $C_2 = \{c_{21}, c_{22}, \ldots, c_{2m}\}$, the goal of ontological view matching is to discover the ontological equivalence mappings, i.e., pairs of matching concepts c_{1i} and c_{2j} such that c_{1i} and c_{2j} represent the same real world concept, $1 \leq \iota \leq n$, $1 \leq j \leq m$. We denote a concept mapping with $c_{1i} \rightarrow c_{2j}$ and the ontological equivalence mappings with $M = \{c_{1i} \rightarrow c_{2j} \mid c_{1i} \in C_1, c_{2j} \in C_2\}$.

Now we look into the concepts. Each concept c can be modeled (or specified by) as a set of *properties*, i.e., $c = \{p_1, p_2, \ldots, p_n\}$, where each p_i is a property, $1 \leq i \leq n$. We rely on the assumption that the similarity of properties indicates the semantic similarity of real-world objects abstracted by these concepts. That is, for two concepts $c_1 = \{p_{11}, p_{12}, \ldots, p_{1n}\}$ and $c_2 = \{p_{21}, p_{22}, \ldots, p_{2m}\}$ from two ontological views, if most of their properties can be discovered as similar, e. g., $p_{11} \approx p_{21}$ ($\approx$ denotes semantically similar), $p_{12} \approx p_{22}$, …, $p_{1k} \approx p_{2k}$, $k \leq \min \{n, m\}$ and k is a given threshold number, then it can be claimed that c_1 and c_2 are semantically equivalent (referring to the same real-world concept).

Chapter 4 Research Issues and Proposed Solutions

The objective of this research is to build a solid theoretical foundation and sound engineering solutions for ontological view-driven semantic integration in open environments. This chapter presents the major research issues and proposed solutions within the context of three main aspects: the architecture of the semantic integration enabled environment, ontological view modeling and representation, and semantic equivalence relationship discovery.

4.1 Architecture of Semantic Integration Enabled Environment

A major challenge to address in this research is the inherent distribution nature of open environments. Traditionally, a common domain ontology is specified as a solution for integrating schemas or local ontologies within the environment. The limitation of this approach is that centralized authority over the environment is usually not architecturally designable or feasible.

We propose a novel architecture that extends the traditional data/information architecture to a three layered architecture (see Figure 4-1), including:

(1) The data management and integration layer. This layer provides abstraction for the binary digits and organizes the digits into various types of elemental data such as numbers, characters, and strings. The management and integration of this layer are achieved by encoding standards, operation systems, and network communication protocols, ensuring that the binary digit streams are consistently interpreted as data of specific types.

(2) The information management and integration layer. This layer associates data to

information models, providing specifications to data and converting data into information. The management and integration of this layer are achieved by applications, including domain dependent applications such as word processors and domain independent applications such as database management systems. It guarantees that data with the same specifications can be manipulated in consistent ways.

(3) The semantics management and integration layer. This layer deals with the semantics of information, resolves semantic heterogeneities, and ensures that information with the same semantics is handled in a semantically consistent way. The management and integration of this layer are addressed by solutions proposed in this research.

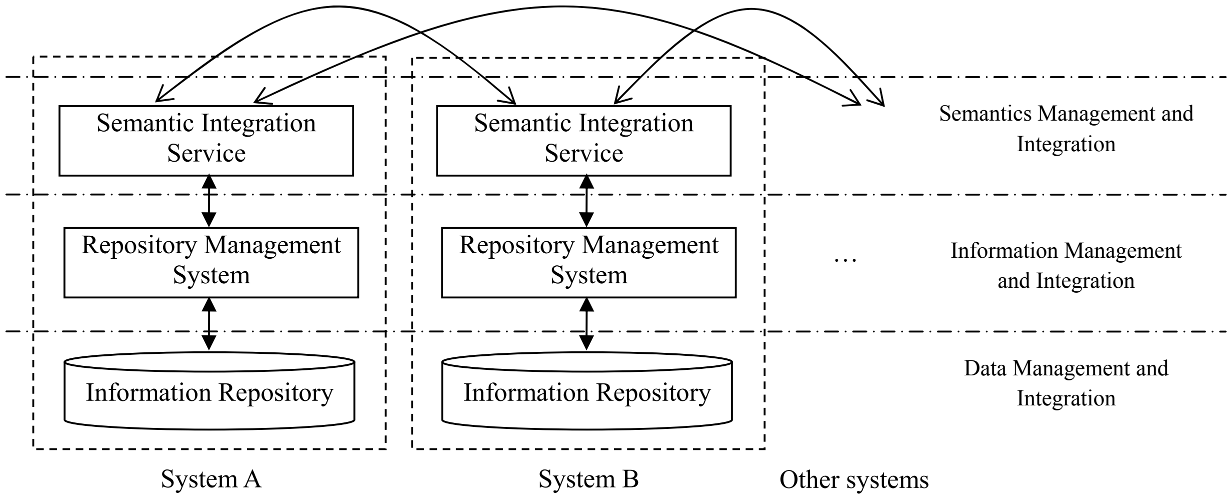

Semantic Integration Service
Semantic Integration Service
Semantics Management and Integration
Repository Management System
Repository Management System
…
Information Management and Integration
Information Repository
Information Repository
Data Management and Integration
System A
System B
Other systems

Figure 4-27. Architecture of the semantic integration enabled environment.

In this architecture, a *Semantic Integration Service* is attached to each information system, which converts a traditional information system into a semantic enhanced system. With the semantic integration service being attached, the requests regarding information semantics will be redirected to this service to resolve potential semantic heterogeneities. The service is responsible for mapping the concepts represented in the requests to compatible concepts (if possible) modeled in the ontological views.

4.2 Architecture of Semantic Integration Service

A further architecture for the semantic integration service is inspired by Act*. It provides the capabilities of representation perceiving (encoding), integrated result delivering (performance), internal knowledge storage (the memories), and semantics manipulating (retrieving, matching, etc). Based on these capabilities we define the architecture, as shown in the following Figure 4-2:

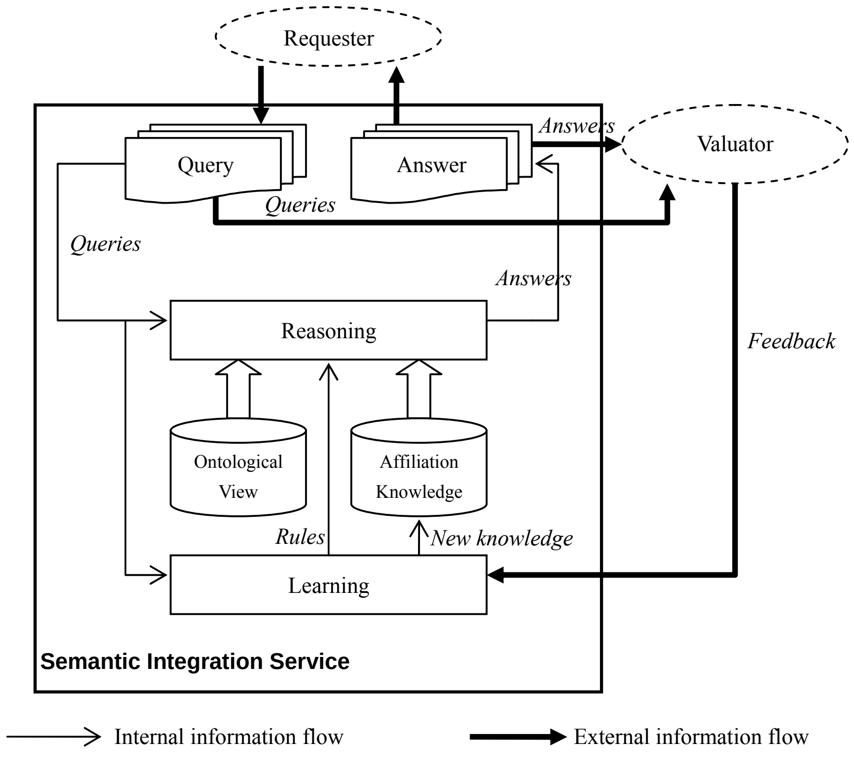


Figure 4-28. Architecture of semantic integration service.

A semantic integration service S can be described as a 5-tuple $S = (I, A, R, L, K)$, where I is the query set that it can accept; A is the answer set that it can generate; R is the reasoning component which can reason about its knowledge base by searching for facts and inferring semantics-matching rules; L is the learning component which can take feedback attached to the query/answer pair and improve the capability of the reasoning component; and, K is the internal knowledge of the service, including the ontological view of the local information system as well as affiliation knowledge that is helpful for reasoning.

The semantic integration service can be viewed as a request-response system: external requesters submit queries to it, and it generates answers as response to the requests.

The working process of the architecture is as follows:

- A requester issues a query.
- The service reasons out the inquiry to discover the semantic relationships.
- The service returns an answer to the requester.
- A valuator validates the answer as well as the query, and provides feedback to the service to enhance its capability.

In light of the proposed architecture, to enable semantic integration we identify the following research issues:

a) How to establish an ontological view of the conceptualization of the domain. The research issues include modeling and representing an ontological view associated with a given information model.

   The proposed solutions are presented in Section 4.3.

b) How to discover the semantic equivalence relationships between the concepts presented in the inquiries and the concepts modeled in ontological views.

   The proposed solutions are presented in Section 4.4 and 4.5.

4.3 Ontological View Modeling and Representation

4.3.1 **Requirements for Modeling**

The value of the ontological view concept is that it provides a common level of models beyond the original heterogeneous information models that use different modeling paradigms and representations. Fundamentally, a concept can be modeled as a structure of C = <P, *hasProperty*>, where P is a set of intrinsic concepts and *hasProperty* is a semantic relationship which associates P to C. An intrinsic concept is a concept that is semantically dependent on an extrinsic (contrary to intrinsic) concept. An intrinsic concept is not usually being processed solely by itself. A property is treated as an intrinsic concept. Therefore, it can also be stated that a concept is modeled by a set of properties. Many of the paradigms used to build information models, such as relational and Object-Orientation, follow the *concept-property* construct. Therefore, it will be normal to adopt the *concept-property* construct for modeling ontological views.

A modeling paradigm is necessary to model the ontological views. The modeling paradigm should support modeling:

(1) Concepts: extrinsic concept is a structure of intrinsic concepts with a *hasProperty* relationship.

(2) Properties: intrinsic concepts.

(3) Relationships between concepts such as *isA* and *partOf*.

4.3.2 Frame Paradigm

In our work we adopt the *frame* paradigm [Karp, 1993 and Minsky, 1975] to model the ontological views. Minsky's *frame* theory is a major milestone in the history of knowledge representation. Proposed in the 1970s, this theory suggests the idea of using object-oriented groups to define a frame which is the data structure to represent the stereotypical situations [Brachman and Levesque, 2004]. It can represent the world meaningfully and naturally, and is cognitively simple, intuitive, and understandable for domain experts. Frames have been widely used in artificial intelligence and knowledge-based systems. Frame-like structures, in combination with rules, are used extensively in expert systems [Aikins, 1993]. Some recent examples of applying frames to knowledge representation can be found in [Kiatisevi, et al., 2006 and Marinov, 2008].

As defined in the Open Knowledge-Base Connectivity (OKBC) specification[15], *frame* is one of the most widely-used ontology modeling paradigms. It is implemented in the core Protégé[16], a cutting-edge tool for creating, editing, browsing, and maintaining ontologies.

Some researchers view *frame* itself as a modeling language, comparing it to other modeling paradigms such as production rules, description logics, and semantic networks. We view *frame* as a modeling paradigm at the conceptual level. From the system's perspective, there should be a specification language that provides structures and semantics to encode frames. However, there is not yet a single standard frame specification language [Wang, et al., 2006].

In the *frame* theory, a frame models a concept which represents a collection of instances. Each frame has an associated collection of slots which can be filled by values or other frames. The slots define the different characteristics of the objects or relations through other objects. In particular, frames can have an IS-A slot which allows the assertion of a concept taxonomy.

Structurally, a frame has the following four-level structure:

- The highest level is literally FRAME, which is a primitive object that represents a concept in the domain of discourse.

- SLOT level captures the properties associated with the concept and relationships to other concepts (frames).

[15] http://www.ai.sri.com/okbc/spec.html

[16] http://protege.stanford.edu/index.html

- Within a SLOT, there is FACET level which captures the details of each SLOT. The FACET level contains multiple facets, with each specifying one aspect of the slot, such as data type, cardinality, and value range.

- Finally, DATA level (or INSTANCE level) provides specific information about each property for an instance of the concept. This level is provided to build a complete knowledge base. When modeling concepts, usually the DATA level is not used if the major focus is on the concept itself without concerning the instances of the concept.

Brachman and Levesque [Brachman and Levesque, 2004] introduced a simple formal representation formulism to express the frame's structure as follows:

```
(Frame-name
     <:IS-A frame-name>
     <slot-name1 filler1>
     <slot-name2 filler2>
     ...
)
```

According to this structure, a frame owns a list of slots into which values can be dropped. The items that go into them are called *fillers*. The fillers of slots that represent relationships are the names of other frames. The frames can have a slot "*:IS-A*" slot whose filler is the name of a more generic frame, meaning that the former frame is a specialization of the latter one.

The frame and slot names are atomic symbols (like numbers or strings without further structures). The fillers are either atomic values or the names of other frames.

4.3.3 **Modeling Ontological Views with Frame**

Support for logical inference is one of the most valued aspects for some knowledge representation paradigms in knowledge-based systems. For example, the OWL DL provides the description-logic reasoning capabilities that enable a reasoning engine to infer knowledge that is not explicitly represented in an ontology, including subsumption testing, equivalence testing, consistency testing, and instantiation testing.

Different from the knowledgebase systems where logical inference is an essential requirement, the information models within information systems focus mainly on modeling concepts and the characteristics of the concepts in the domain of discourse. Each concept is specified by its own (even other concepts can be involved to specify its characteristics), not defined by other concepts. Furthermore, the models focus on the stereotype instead of the individual instances. Therefore, the reasoning capability as provided by DL is not an essential element for modeling the ontological views based on the information models, and the instances can usually be ignored.

The concepts are the fundamental elements in the information models. A concept is modeled by a set of properties. Many of the paradigms used to build information models, such as relational and object-orientation, are following the *concept-property* construct.

As a knowledge modeling paradigm, frame provides a clear and explicit structure that is adequate at modeling the proposed ontological view model, in particular in describing the properties of concepts, which makes frame an ideal candidate for modeling the ontological views.

In an open environment the frame-based ontological views create a common level. This common level eliminates the structural and syntactic heterogeneities among the information models. For instance, relational database schemas and XML schemas use different structures and syntaxes. By converting them into frame-based ontological views they all follow the standard *concept-property* construct. With this commonness only semantic heterogeneities should be considered in the semantic integration.

There is some other research that also proposes to create a kind of concept model from the underlying information model [Boran, et al., 2007]. For example, D2RQ [17] supports lifting the basic relational database schema information into RDF to create RDF-based ontologies. It is a declarative language to describe mappings between a relational database schema and RDF ontologies. It uses these mappings to enable applications to access a RDF-view on a non-RDF database. RDF is a standard model for data interchange on the Web. It extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link. Such a way does not apply to our context for the following reasons:

- RDF focuses on the Semantic Web. Our work focus on integrating traditional information systems that are very different from the Web. There are no URIs in the systems.
- RDF is good at modeling things that interconnect to each other, resulting in a graph. There isn't an explicit structure showing that a set of “things” are the properties of a specific “thing”. In the information models it is important to describe that a concept has a set of properties. The properties are not treated as independent resources.
- RDF mixes the concepts’ properties and the property values together. Our modeling requires a clear separation between concepts and instances of the concepts.

[17] http://sites.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/

4.3.4 A Frame-based Ontological view Specification Language (FOSL)

4.3.4.1 Specification of Ontological Views

The ontological views must be explicitly specified in order to be used with information systems, i.e., delivered using some concrete representation.

The specification of an ontological view is composed of:

(1) symbols mapped to concepts (as an explicit representation of the intended model);

(2) symbols mapped to properties and their associated characteristics;

(3) symbols mapped to relationships between concepts; and

(4) symbols that logically connect (1), (2), and (3) with specific semantics.

Note that the *language* specifying the ontological views and the *language* specifying the conceptualizations (as defined in section 3.4) belong to different categories. The former contains the basic elements, syntactical rules upon the elements and the semantics to specify meaningful models. It is guaranteed that these elements and rules are commonly agreed upon by any semantic integration service within an environment. The latter refers to the vocabulary that is used to denote the concepts as well as the interpretation of the vocabulary. This language contains symbols that map to concepts, properties, and relationships. This section is focused on the former language.

An information model does not always explicitly describe concepts, properties, or relationships. However, some of its constructs usually imply these elements. For example, in a relational database schema (which is a type of information model), a table can be used to represent a concept; in an XML document, a node can represent a concept. Given that an information model M is specified by language $\mathbf{L}_M = <\mathbf{S}_M, \mathrm{I}_M>$ with vocabulary V_M and the ontological view model is specified by language $\mathbf{L}_O = <\mathbf{S}_O, \mathrm{I}_O>$ with vocabulary V_O, the creation of an ontological view is to find a mapping m between $\mathbf{L}_O$ and $\mathbf{L}_M$ such that $m(\mathrm{I}_O)\subseteq\mathrm{I}_M$. The mapping requires a set of rules for each modeling paradigm to identify:

- What constructs in the information model can be mapped to concepts;
- What constructs in the information model can be mapped to properties;
- What constructs in the information model can be mapped to facets of the properties;
- What constructs in the information model can be mapped to values of facets;
- What constructs in the information model can be mapped to relationships between concepts.

For example, as to a relational database schema,

- A table which has a primary key is a candidate of a concept;
- Each column in the table is a candidate of a property;
- The attributes of the column, such as data type, size, default value, null-able, are candidates of facets;
- The value of the attributes, such as *Integer* and *NULL*, are candidates of values of facets.
- A foreign key column implies a relationship to a concept indicated by the referred table;

- A table that has a combined primary key and each of which column is a foreign key implies a relationship between two concepts indicated by the referred tables.

By applying these rules, an ontological view can be constructed from a corresponding information model. These rules reveal the key requirements for the specification language, including the symbols and syntax indicating concepts, properties, facets, facet values, and relationships.

The explicit specification of ontological views following a specific modeling paradigm provides a common foundation that eliminates the heterogeneities residing in the underlying information models in terms of technical platform, modeling paradigm, specification syntax, etc. Later work, such as semantic integration, can just focus on the semantic aspect, i.e., the difference regarding various views of the domain conceptualization, based on a single modeling paradigm without concern for dealing with different ways of modeling and specifying the models.

4.3.4.2 Definition of FOSL

We propose the Frame-based Ontological view Specification Language (FOSL) to support specification of the above aspects. It is a logical language created from the following vocabulary:

(1) Constant symbols: the set of $FR \cup S \cup F \cup V$, where FR is a set of symbols referring to frames (concepts), S is a set of symbols referring to slots (properties), F is a set of symbols referring to facets, and V is a set of values that the facets can take.

(2) Variable symbols: there are four sets V_{FR}, V_S, V_F, V_V of variable symbols which ranges are FR, S, F, and V, respectively.

(3) Predicate symbols: the following predicate symbols are defined:

(a) A binary predicate *hasProperty* applied on $FR \times S$. *hasProperty*(fr, s) refers to a frame $fr \in FR$ with a slot $s \in S$.

(b) A triple predicate *hasFacet* applied on $FR \times S \times F$. *hasFacet*(fr, s, f) indicates that slot $s \in S$ has a facet $f \in F$ in a frame $fr \in FR$.

(c) A quad predicate *hasValue* applied on $FR \times S \times F \times V$. *hasValue*($fr$, s, f, v) indicates that the slot $s \in S$'s facet $f \in F$ has a value $v \in V$ in a frame fr$\in FR$.

(d) A binary predicate *isA* applied on $FR \times FR$. *isA*(fr_1, fr_2) indicates that frame $fr_1 \in FR$ is a type of frame $fr_2 \in FR$, i.e., the concept modeled by fr_1 is a specialization of the concept modeled by fr_2.

(e) A binary predicate *partOf* applied on $FR \times FR$. *partOf*(fr_1, fr_2) indicates that frame $fr_1 \in FR$ is a part of frame $fr_2 \in FR$, i.e., the concept modeled by fr_1 is a part of the concept modeled by fr_2.

The predicates *isA* and *partOf* specify two types of relationships between concepts selected to be defined in FOSL. The reasoning behind the choice is that these two types provide strict semantics that can be commonly agreed upon among multiple parties. Such relationships can be generally reasoned.

Other relationships are rather arbitrary, resulting in unpredictable semantics. For instance, a frequently used example is "*Student* takes *Course*" where *Student* and *Course* are two concepts and takes is a relationship. Here takes does not provide inferable semantics but only a human reader can understand its meaning. The reasoning for such relationships depends on domain-specific engines that are aware of the meaning of the relationships.

Even the predicate *hasFacet* implies *hasProperty* because when *hasFacet*(fr, s, f) holds we also have *hasProperty*(fr, s) (similar case applies to predicate *hasValue* and *hasFacet*), the individual *hasProperty* predicate is still necessary since it is not guaranteed that every information model is complete. That is, in some models it may be that only properties of a concept are listed but details of the properties are missing.

This redundancy also increases the readability of a specification written in FOSL in a way that a layered structure of the concept specification is presented and different reader interests can be well satisfied. For example, given a set of statements with *hasProperty* predicate, it is easy to grasp a general view of a concept, i.e., "this concept is described by this set of properties", without any unnecessary information involved. If a reader is interested in what a property is like, a set of statements with the *hasFacet* predicate will help. Furthermore, the statements with the *hasValue* predicate provide the lowest level of details for the facets.

4.3.4.3 Inference Rules

Now we define the inference rules that can be expressed by the language.

Inheritance Rule:

- *isA*(*subfr*, *superfr*) ← *isA*(*subfr*, *fr*) & *isA*(*fr*, *superfr*), i.e., a frame *subfr* specialized from another frame *fr* is also a specialization of that frame's generalized frame *superfr*.

- *hasProperty*(*subfr*, *s*) ← *isA*(*subfr*, *fr*) & *hasProperty*(*fr*, *s*), i.e., a generic frame's slots are inherited by its specialized frames.

- *hasFacet*(*subfr*, *s*, *f*) ← *isA*(*subfr*, *fr*) & *hasProperty*(*fr*, *s*) & *hasFacet*(*fr*, *s*, *f*), i.e., the facets of a slot of a generic frame are inherited by the same slot of its specialized frames.

- *hasValue*(*subfr*, *s*, *f*, *v*) ← *isA*(*subfr*, *fr*) & *hasProperty*(*fr*, *s*) & *hasFacet*(*fr*, *s*, *f*) & *hasValue*(*fr*, *s*, *f*, *v*), i.e., the value of a facet of a slot of a generic frame is inherited by the same facet of the same slot of its specialized frames.

Composition Rule:

- $\exists$ *fr*∈*FR* ← $\exists$ *partialfr*∈*FR* & *partOf*(*partialfr*, *fr*), i.e., there must exist a frame where another frame is a part of it.

- *partOf*(*partialfr*, *wholefr*) ← *partOf*(*partialfr*, *fr*) & *partOf*(*fr*, *wholefr*), i.e., if a frame *partialfr* is a part of another frame *fr*, it is also a part of a larger frame *wholefr* which has that other frame as a part of it.

4.3.4.4 XML-based Encoding

To explicitly encode ontological views we propose a human readable and machine process-able representation which enables:

(1) The ontological view created from an information model to be verified and refined by human experts;

(2) The semantic integration to be executed in an automated manner based on the analysis applied on the representations.

To this end we adopt an XML-based representation for FOSL. An ontological view can be modeled as a set of frames and represented in an XML document. The document is supported with multiple <*concept*> tags for concepts (frames), respectively. Under a <*concept*> tag the slots are divided into two categories and specified by <*relationships*> and <*properties*>. Under each category there are a collection of individuals, namely <*relationship*> and <*property*>. The *isA* and *partOf* predicates are represented as specific <*relationship*> nodes with pre-defined semantics.

The facets of each slot are tagged as <*facet*> which is described by two attributes: *name* and *value*. To uniquely identify each concept, there is also a sub-tag <*name*> under each <*concept*> tag denoting the identifier of each concept.

The following is the schema of the XML document derived from FOSL.

```
<?xml version="1.0" encoding="utf-16"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified" version="1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="ontological_view" type="ontological_viewType" />
  <xsd:complexType name="ontological_viewType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="concept" type="conceptType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="conceptType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="properties" type="propertiesType" />
      <xsd:element name="relationships" type="relationshipsType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="relationshipsType">
    <xsd:sequence>
      <xsd:element name="relationship" type="relationshipType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="relationshipType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="target_concept" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="propertiesType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="property" type="propertyType">
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="propertyType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="facets" type="facetsType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="facetsType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" name="facet" type="facetType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="facetType">
    <xsd:attribute name="name" type="xsd:string" />
    <xsd:attribute name="value" type="xsd:string" />
  </xsd:complexType>
</xsd:schema>
```

4.4 Semantic Equivalence Relationship Discovery

4.4.1 Short Summary on Matching Approaches

Generally, the information models adopt vocabulary from natural languages such as English. The constant symbols such as English words refer to concepts under an ontological commitment. This research takes this fact as an assumption, i.e., this research does not deal with the cases that random symbols are picked for the information models.

According to propositions (2) in Section 3.4.6, if the semantic equivalence relationships between symbols can be discovered, then it can be inferred that the symbols are referring to the same concept, therefore the semantic integration can be achieved. The semantic equivalence relationship is deduced from the *semantic similarity metric* between symbols. A semantic similarity metric is a combination $<A, t>$ where A is an approach to compute the similarity between symbols and t is a threshold. The approach A can be viewed as a function $A: S \times S \rightarrow R$ where S is the set of symbols and R is the set of real numbers. If $A(s_1, s_2) > t$, $s_1, s_2 \in S$, then it can be confidently believed that two symbols are semantically equivalent, i.e., s_1 and s_2 have a semantic equivalence relationship.

In the research of schema matching and ontology mapping, multiple approaches have been developed to discover the semantic relationships between elements of the schemas or ontologies. These approaches can be applied to ontological views.

Next, we briefly introduce three major categories of approaches.

(1) Linguistic (Syntactical) Matching

Linguistic matching utilizes vocabularies of languages to discover semantic equivalence relationships. Linguistic matching works on symbols that can be mapped to concepts under an ontological commitment. The languages adopt symbols based on a natural language foundation such as English. For example, assuming there is an intended model of some concept *c*, one may use an English word “Professor” as a symbol to model it in an information model. By modeling the symbol, “Professor” is mapped to that concept and by representing the string “Professor” is used as the name of a table in a relational model.

In linguistic matching, the principle is that the more syntactically similar two symbols are, the more likely they map to the same concept, the same property, or the same facet. To increase the precision of the comparison, the symbols will often be normalized and compared, sometimes with the help of natural language dictionaries to determine the synonym when the symbols are syntactically different.

Linguistic matching consists of two major steps: normalization and comparison.

(a) Normalization. Information models usually utilize similar symbols for the same concept, but with syntactical differences due to abbreviations, acronyms, or punctuations. To tolerate these differences, a normalization process is used to reduce the syntactical diversity. The process includes:

- Tokenization – The symbols are parsed into tokens by a customizable tokenizer using punctuation, upper case, special symbols, or digits.
- Expansion – Abbreviations and acronyms are expanded to the full form.
- Elimination – Tokens that are articles, prepositions, or conjunctions are marked to be ignored during comparison.

(b) Comparison. To determine the semantic equivalence relationship, the linguistic similarity between the symbols representing the concepts is computed. For example, the edit distance can be used to compute the similarity between two symbols. To enhance semantic matching, some natural language dictionaries such as WordNet can be employed. A dictionary is designed in a way that it relates words with different syntactical forms together if they refer to the same or similar concepts in a specific domain. It can be used, for instance, to determine the synonym when two symbols are syntactically different.

As an example, if the symbols “Engineering” and “Eng.” are used in different ontological views to model concepts in the education domain, a shared domain dictionary may tell that “Eng.” is usually an abbreviation of “Engineering”, therefore these two symbols should be referring to the same concept (the same faculty). As another example, both s_1 = “Research Center” and s_2 = “Research Centre” can be adopted as symbols to model concepts. A simple edit distance metric between s_1 and s_2 is 2, considering that one letter is removed from s_1 and another letter is inserted s_1 to make s_1 identical to s_2. By comparing the edit distance between s_1 and s_2, as well as edit distances between s_1 and other symbols, it is reasonable to conclude that s_1 is semantically equivalent to s_2.

(2) Structural (Semantic) Matching

The structural matching utilizes the semantic structures captured by the proposed *frame* model to discover the semantic equivalence relationships if syntactical matching cannot provide sufficient clues. The frame model’s tree-like structure will be utilized to consider the following cases:

(a) Atomic symbols (leaves) in two trees are similar if they are linguistically similar and the associated symbols in their respective vicinities (ancestors and siblings) are similar.

(b) Two non-leaf symbols are similar if they are linguistically similar, and the sub-trees rooted at the two symbols are similar.

(c) Two non-leaf symbols are similar if their leaf sets are similar, even if their immediate children are not. This is because the leaves represent the atomic information that the models ultimately contain.

These rules can support the inference that two symbols are semantically equivalent if their properties are very similar, even though they are syntactically different.

For example, consider the following two concepts based on frame:

Concept: Professor *isA* Person	**Concept**: Faculty *isA* Person
--- *hasProperty*: Title	--- *hasProperty*: AssignedTitle
-- *hasFacet*: Range	-- *hasFacet*: Range
- *hasValue*: Full, Associate, Assistant	- *hasValue*: Full, Associate, Assistant
--- *hasProperty*: Name	--- *hasProperty*: Name
-- *hasFacet*: Type	-- *hasFacet*: Type
- *hasValue*: Text String	- *hasValue*: Text String

The non-leaf symbols *Title* and *AssignedTitle* are syntactically similar, and since their sub-trees are similar, case (b) can be applied to conclude that they refer to the same property. The non-leaf symbols *Professor* and *Faculty* are syntactically different. However, since their sub-trees are structurally similar, from (c) it can be concluded that they refer to the same concept.

(3) Instance-based Matching

Instance-based matching approaches belong to another important category since the instances can provide much useful information.

Let O be an ontology model that has a concept hierarchy C. C can be expressed by the set $\{c_1, ..., c_n\}$ where $c_1, ..., c_n$ are concepts in O. Every concept in O may be instantiated by one or more instances, denoted by an expression of the form $[r_{k1}, ..., r_{km}]$ where $r_{k1}, ..., r_{km}$ are instances of concept c_k that belongs to the concept hierarchy C. We say that instances r_a in O_A and r_b in O_B, respectively, are equivalent, denoted as $r_a \equiv r_b$, when they represent the same (real-world) object; in this case, we also say that c_a and c_b map to each other, where c_a and c_b are the concepts under which instances r_a and r_b are classified, respectively.

In instance-based mapping semantic relations between concepts of two ontologies are determined based on the overlap of their instance sets. The basic idea is that the more significant the overlap of the common instances of the two concepts is, the more related these concepts are [Isaac, et al., 2007]. This is a very natural approach, as in most ontology formalisms the semantics of the relations between concepts is defined via the set of their instances. The idea for mapping is then simply that the higher the ratio of co-occurring instances for two concepts, the more they are related.

4.4.2 A Tree Similarity-based Approach

4.4.2.1 Introduction to Tree-based Similarity Discovery

An ontology model can be viewed as a concept structure representing some domain knowledge [Sanin, et al., 2007], and one commonly used form is a tree structure. The frame-based ontological view and each concept in an ontological view also have a tree structure. Approaches developed for comparing tree structures can be applied to discover possible semantic relationships.

Much of the research on comparing trees uses the editing cost from one tree to another to measure the similarity of two trees [Guegan and Hernandez, 2006]. The classical methods focus on the structural and geometrical characteristics of the trees, mainly considering the number of nodes affected by the tree editing operations [Allali and Sagot, 2005 and Guda, et al., 2002]. However, in a knowledge context where the trees are used to represent the concept structures, in addition to the structural characteristics of the trees, more attention must be paid to the concepts represented by the internal tree nodes. Therefore, besides the number of edited nodes, the positions and conceptual similarities of the affected nodes also have to be considered.

The similarity of two individual concepts can be relatively easily estimated by domain experts. As an example, based on common sense, concepts "*People*" and "*Human*" are often regarded as referring to the same meaning, i.e. their similarity degree is 1. On the other hand, concept "*Faculty*" does not always refer exactly to the same thing as "*Professor*" in the university domain. Roughly speaking, a similarity degree can be assigned to these two concepts, say, 0.9, meaning that under approximately 90% of the occasions they are describing the same group but not in other cases. Some research has also proposed various methods of determining conceptual similarity between individual concepts in a knowledge context [Han and Kamber, 2000 and Warin, et al., 2005].

Determining the similarity of various structures containing many concepts is another complicated research topic. For instance, given the following three trees in Figure 4-3 (which are modelling the concept structures about the university domain and are developed by different people) where relationships between concepts are identical ("*part-of*" in this example) and a list describing the similarities of individual concept pairs (e.g. sim(*People*, *Human*) = 1 and sim(*Faculty*, *Professor*) = 0.9) which can be provided by domain experts, how can we determine the extent that they are similar to each other and which two are more similar.

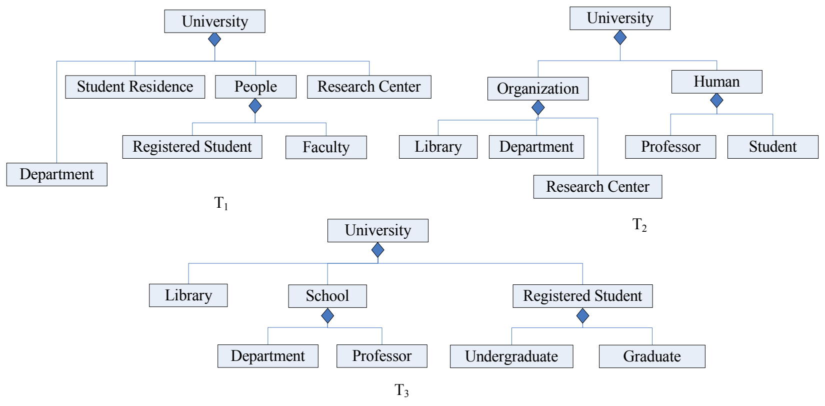


Figure 4-29. An example of multiple concept trees for the same domain.

Our work extends the classical tree editing operations and introduces the tree transformation operations. We propose four types of transformation operations which can transform one concept tree into another, and provide definitions for the cost of each operation considering the number of affected nodes, the scale of the node set, the conceptual significance of affected nodes, and the conceptual similarity of the node pairs (each node representing one concept) in a knowledge context. The degree of tree similarity is measured according to the tree transformation cost. This method can be applied to ontological view comparison to support semantic integration in cases where different ontological views for the same domain can be represented as trees.

4.4.2.2 Related Work

The tree is one of the most commonly used combinatorial structures in computer science. Research on comparing tree structures has a long history in many fields. It has been well studied in several diverse areas such as computational biology, structured text databases, image analysis, and compiler optimization [Bille, 2003]. In the research the edit cost (or edit distance) from one tree to another is employed to measure the similarity degree of two trees [Allali and Sagot, 2005; Guda, et al., 2002; Guegan and Hernandez, 2006; Jin, et al., 2005]. However, such research is mainly focused on finding matches based on the pure structure or geometry perspective without considering the conceptual semantics of the tree nodes in a knowledge context.

Tree pattern matching is another one frequently used methods. For example, some research has explored the algorithm of matching pattern discovery in an XML query [Bruno, et al., 2002 and Yao and Zhang, 2004] where they did not focus on the cost of matching. Another domain of using tree pattern matching is compiling where matching cost is defined through tree-rewriting rules and instruction types [Aho, et al., 1989].

Maedche et al. conducted in-depth research into the similarity between ontologies [Maedche and Staab, 2002]. In their research context, an ontology has a tree structure that is modelling a concept taxonomy. A method was developed to measure the similarity between ontologies based on the notions of lexicon, reference functions, and semantic cotopy. This method is based on an assumption that the same terms are used in different ontologies for concepts but their relative positions may vary. However, in many real ontologies different terms will be adopted to construct the concept taxonomies, although some of them have similar semantics. In these cases computing taxonomic overlap is not fully applicable and lexical level comparison becomes almost inapplicable. Furthermore, this research did not take the structural characteristics of trees into consideration.

Li et al. conducted similar research on measuring the similarity of ontologies (represented as trees) based on tree structure mapping [Li, et al., 2006]. They proposed a mapping method that combines the similarity of the inner structure of concepts in different ontologies and the language similarity of concepts. The similarity of concepts is computed from some lexical databases like WordNet. However, such a generic semantic similarity calculating algorithm is not perfectly applicable in domain-based concept systems. Furthermore, Li's work did not handle cases of crossing-layer mappings, which is common in tree mapping where similar terms may be placed in various layers within the trees.

Summarizing, to the best of our knowledge, no research has been fully done to measure the similarity of trees based on both structure comparing and concept comparing and then applied to ontological view comparison.

4.4.2.3 Definition for Concept Tree

A lot of research has been done on tree comparing, which has focused mainly on finding matches based on the pure structure or geometry perspective (e.g. [Guda, et al., 2002 and Jin, et al., 2005]) without considering the conceptual semantics of the tree nodes in a knowledge context.

We extend the traditional definition of trees for the sake of describing concept structures. The formal definition is given below:

Definition 1: Concept Tree. An (unordered and labelled) Concept Tree is a six-tuple $T = (V, E, L^V, \text{root}(T), D, M)$ where V is a finite set of nodes, E is a set of edges satisfying that $E \subset V \times V$ which implies an irreflexive and antisymmetric relationship between nodes, L^V is a set of lexicons (terms) for concepts used as node labels, $\text{root}(T) \in V$ is the root of the tree, D is the domain of discourse, and M is an injective mapping from V to L^V, M: $V \rightarrow L^V$ ensuring that each node has a unique label. For convenience, we simply call each term in L^V a concept with an agreement on their semantics. A mapping from a node v to a label l is simply written as a tuple $(v, l) \in M$.

A concept tree is acyclic and directed. If $(u, v) \in E$, we call u a parent of v and v a child of u, denoted as $u = \text{parent}(v)$ or $v = \text{child}(u)$. The set of all children of node u is denoted as $C(u)$. For two nodes $u_1, u_2 \in V$, if $(u_1, u_2) \in E^*$ holds, then we call u_1 an ancestor of u_2 and u_2 a descendant of u_1. The set of all descendants of node u is named $D(u)$.

The following conditions are satisfied by any concept tree:

(1) The root node does not have a parent node.

(2) Any node in V other than the root has one and only one parent node.

(3) For each non-root node u in V, there exists $(\text{root}(T), u) \in E^*$, where E^* is the transitive closure of E, meaning that no node is isolated from others.

(4) There is a unique directed path composed of a sequence of elements in E from the root to each of the other elements in V.

Definition 2: Conceptual Similarity Measure. A conceptual similarity measure $S_{L^{V_1}, L^{V_2}}$ is a set of mappings from two lexicon sets L^{V1}, L^{V2} used in different concept trees to the set of real numbers R, $S_{L^{V_1}, L^{V_2}}$: $L^{V1} \times L^{V2} \rightarrow R$, in which each mapping denotes the conceptual similarity between two concepts represented by these two lexicons. R has a range of (0, 1]. $S_{L^{V_1},L^{V_2}}$ is semantically reflexive and symmetric, i.e. for $l_1 \in L^{V1}$ and $l_2 \in L^{V2}$ we have $S_{L^{V_1},L^{V_2}}(l_1, l_1) = 1$ and $S_{L^{V_1},L^{V_2}}(l_1, l_2) = S_{L^{V_1},L^{V_2}}(l_2, l_1)$. For convenience, we simply use $w = s(l_1, l_2)$ to refer to the number value of conceptual similarity between two concepts from two trees T_1 and T_2. Intuitively, the larger w is, the closer the two concepts are and $w = 1$ means two concepts are actually identical (the terms used to denote the concepts are synonymous).

Conceptual similarity between two concepts can be given by domain experts or calculated based on some linguistic analysis methods. For instance, Mitra et al. use a linguistic matcher to assign a similarity score to a pair of similar concepts [Mitra and Wiederhold, 2002]. As an example, given the strings "*Department of Defense*" and "*Defense Ministry*", the match function returns match(*Defense*, *Defense*) = 1.0 and match(*Department*, *Ministry*) = 0.4, then it calculates the similarity between the two strings as: s("*Department of Defense*", "*Defense Ministry*") = (1 + 0.4)/2 = 0.7.

For $l_1 \in L^{V1}$ and $l_2 \in L^{V2}$, if there is no definition for l_1 and l_2 in the measure, we view l_1 and l_2 as totally different (disjoint) concepts. Such a concept pair will not be considered when two concept trees are being compared.

4.4.2.4 Tree Transformation Operations and Transformation Cost

Tree transformation operations can map one tree T into another one, T', as defined below.

(1) Deleting node *v* (denoted as delete(*v*))

If $v \neq \mathrm{root}(T)$, then $V' = V - \{v\}$, $E' = E - \{(u, v) \mid u = \mathrm{parent}(v)\} - \{(v, v_c) \mid v_c \in C(v)\} + \{(u, v_c) \mid u = \mathrm{parent}(v) \wedge v_c \in C(v)\}$, $L^{V'} = L^V - \{M(v)\}$, and $M' = M - \{(v, M(v))\}$.

It must be noted that when deleting one node, besides eliminating that node from the tree we still need to make its children nodes new direct children nodes of its parent node, which is different from deleting a sub-tree.

If v = root(T), the result of deleting is a forest $\{T[v_c] \mid v_c \in C(v)\}$. In a concept tree the root is usually a very general concept like "object", therefore we assume that all trees have a common root concept and restrict that the root is never allowed to be deleted.

The deleting operation is depicted in the following Figure 4-4:

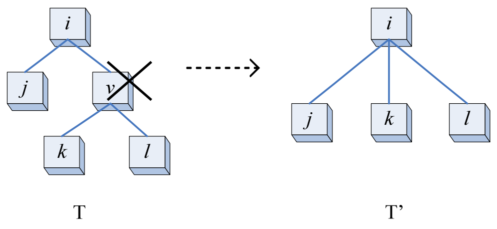


Figure 4-30. Deleting a node.

(2) Inserting node *v* under node *u* (denoted as insert$_u$(*v*))

We have $V' = V + \{v\}$, $E' = E + \{(u, v)\} + \{(v, u_c) \mid u_c \in C'(u)\} - \{(u, u_c) \mid u_c \in C'(u)\}$, $L^{V'} = L^V + \{l_v\}$, and $M' = M + \{(v, l_v)\}$, where l_v is the lexicon assigned to the new node v, and $C'(u) \subseteq C(u)$ meaning that some children nodes of u are changed to be children of the new node v. The elements contained in $C'(u)$ is determined by the context when performing the editing operation.

The inserting operation is depicted in Figure 4-5:

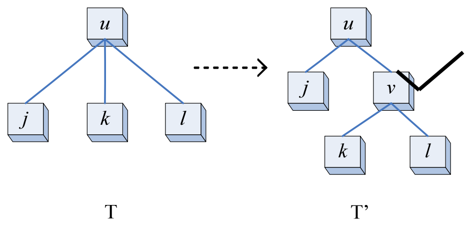


Figure 4-31. Inserting a node.

(3) Re-labelling node *v* (denoted as relabel$_{l_v \to l_v'}$(*v*))

This is a particular operation in a labelled tree. Re-labelling of v with label l_v is to assign v a new label l_v', keeping the positions of all the nodes unchanged. We have $L^{V'} = L^V - \{l_v\} + \{l_v'\}$ and $M' = M - \{(v, l_v)\} + \{(v, l_v')\}$, where l_v' is the new label assigned to v, as is depicted in the following Figure 4-6.

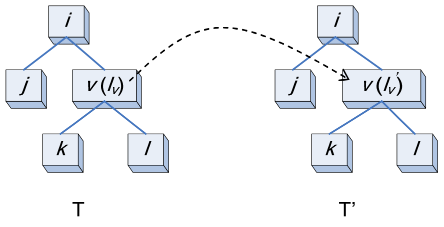


Figure 4-32. Re-labelling a node.

(4) Moving node *v* to be under node *u* (denoted as move$_u$(*v*))

This is an extended operation in a knowledge context that is not defined in classical tree editing operation sets. From Figure 4-7 we see that in the case of pure structured trees (a) and (b) two operations delete(E) and insert$_B$(E) can be performed to convert (a) to (b). However, when mapping a concept tree to another we cannot simply delete a node and then insert it since the concept represented by the node's label already exists in the tree.

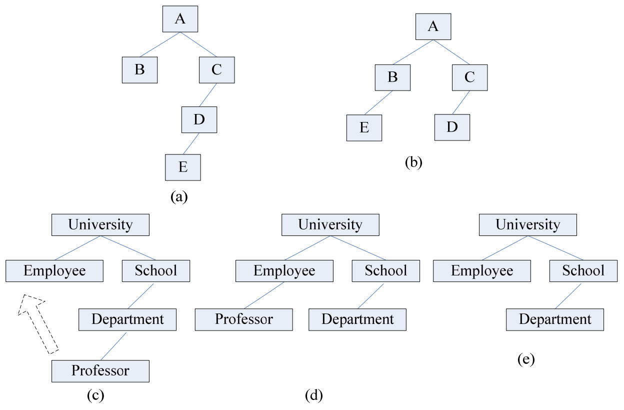


Figure 4-33. An example of a moving operation.

More specifically, in Figure 4-7 two trees (c) and (d) put the concept "*Professor*" in different positions and by moving node "*Professor*" to be under "*Employee*" we transform (c) to (d), instead of deleting "*Professor*" and then inserting it back (from (c) to (e) and then (e) to (d)).

The moving operation regulates that $V' = V$, $E' = E + \{(u, v)\} + \{(v, u_c) \mid u_c \in C'(u)\} + \{(\text{parent}(v), v_c) \mid v_c \in C(v)\} - \{(\text{parent}(v), v)\} - \{(v, v_c) \mid v_c \in C(v)\} - \{(u, u_c) \mid u_c \in C'(u)\}$, where $C'(u) \subseteq C(u)$ meaning that some children of node u are changed to be children of the node v based on the operation context.

Definition 3: Transformation Cost. Each transformation operation Op on tree T is mapped to a real number which is defined as the transformation cost of the operation and denoted as $\gamma(Op)$. The transformation cost reflects the extent of change it makes to the tree.

If $OP = \{Op_1, Op_2, \ldots, Op_k\}$ is a transformation sequence, then the transformation cost of the sequence is defined as $\gamma(OP) = \sum_{i=1}^{i=|OP|} \gamma(Op_i)$.

Definition 4: Tree Transformation Cost and Similarity Index. If OP is a transformation sequence mapping a tree T_1 to another tree T_2, then the tree transformation cost from T_1 to T_2 is defined as

$$\gamma(T_1 \rightarrow T_2) = \min\{\gamma(OP) \mid OP \text{ is a transformation sequence mapping } T_1 \text{ to } T_2\}.$$

Also, we define the similarity index of two trees T_1 and T_2 as

$$\gamma(T_1, T_2) = \min\{\gamma(T_1 \rightarrow T_2), \gamma(T_2 \rightarrow T_1)\}.$$

It is a measure representing the extent to which two trees are similar to each other. The higher the tree transformation cost and similarity index is, the less similar the two trees are and vice versa.

4.4.2.5 Computing of Transformation Cost

In a tree transforming process we need to count the total cost of all transformation operations. A tree transforming process that maps tree $T_1 = (V_1, E_1, L^{V1}, \text{root}(T_1), D,$

M_1) into $T_2 = (V_2, E_2, L^{V2}, \text{root}(T_2), D, M_2)$ based on $S_{L^{V_1},L^{V_2}}$ contains the following tasks:

(1) Compute the set of nodes to be deleted, D, in T_1.

$D = \{u \mid u \in V_1 \wedge M_1(u) \notin L^{V2} \wedge \neg\exists s(M_1(u), l_2) \in S_{L^{V_1},L^{V_2}} (l_2 \in L^{V2})\}$. That is, the nodes which labels are appearing in T_1 but T_2 and have no conceptual similarity with any labels in T_2 defined (the concepts represented by the nodes in T_1 are totally not contained by T_2).

(2) Compute the set of nodes to be inserted into T_1, I.

$I = \{v \mid v \in V_2 \wedge M_2(v) \notin L^{V1} \wedge \neg\exists s(l_1, M_2(v)) \in S_{L^{V_1},L^{V_2}} (l_1 \in L^{V1})\}$. That is, the nodes which labels are appearing in T_2 but T_1 and do not have conceptual similarity definition with any labels in T_1 (the concepts represented by the nodes in T_2 are totally not contained by T_1).

(3) Try every possible combination of the deletion and insertion operations and find the minimal cost.

(4) Compute the set of nodes to be moved within T_1 itself, M, and move them.

$M = \{u \mid u \in V_1 \wedge (M_1(u) \in L^{V2} \wedge M_1(\text{parent}(u)) \neq M_2(\text{parent}(M_2^{-1}(M_1(u)))) \wedge \neg\exists s(M_1(\text{parent}(u)), M_2(\text{parent}(M_2^{-1}(M_1(u))))) \in S_{L^{V_1},L^{V_2}}) \vee (\exists s(M_1(u), l_2) \in S_{L^{V_1},L^{V_2}} (l_2 \in L^{V2})) \wedge M_1(\text{parent}(u)) \neq M_2(\text{parent}(\mathcal{M}_2^{-1}(l_2))) \wedge \neg\exists s(M_1(\text{parent}(u)), M_2(\text{parent}(M_2^{-1}(l_2)))) \in S_{L^{V_1},L^{V_2}})\}$. That is, the nodes that are appearing in both T_1 and T_2, or which labels have conceptual similarity with labels defined in T_2, but which parents are neither the same nor similar.

(5) After the deleting, inserting, and moving operations are performed on T_1, T_1 now has the same structure with T_2, but still has some nodes with different labels (implying

different conceptual semantics). The final task is to compute the set of nodes to be re-labelled, R, and re-label them. $R = \{u \mid u \in V_1 \wedge M_1(u) \notin L^{V2} \wedge \exists s(M_1(u), l_2) \in S_{L^{V_1}, L^{V_2}}$ $(l_2 \in L^{V2})\}$. That is, the nodes that are appearing in both T_1 and T_2 with different labels, but the labels have conceptual similarity between them.

Let OP be the editing sequence containing operations in the above tasks, the transforming cost is computed as follows (using pure operation names):

$$\gamma_{T1 \to T2}(OP) = \min\{\sum_{i \in D} \gamma(delete(i)) + \sum_{i \in I} \gamma(insert(i)) + \sum_{i \in M} \gamma(move(i)) + \sum_{i \in R} \gamma(relabel(i))\}$$

The cost of each transformation operation (deleting, inserting, moving, and re-labelling) is a key issue for the measuring. The cost is affected by which level that the node resides in the tree structure, the scale of the node set, the number of descendants of the node, and the similarity of the two concepts (labels) attached to the two nodes. For example, first, a node at a higher layer contains richer semantics than does a lower node does, or, the concept it represents is more significant for the domain than a lower one. Therefore, when a node u is at a higher layer, the effect to the concept tree of deleting u or inserting a new node under u is bigger than that of deleting or inserting a node at a lower layer. Second, the more nodes a tree has, the less the effect will be when one node is deleted or inserted. That is, the larger the concept tree is, the less different it will be if it gets one new concept or loses one old concept. Third, a node with more descendants will cause greater change to the tree structure if it is deleted, or greater change is made if a node gets more descendants after it is inserted. Finally, the more similar the two concepts are, the less the cost will be to change one into the other.

Based on the research of [Bille, 2003 and Kruskal, 1999] and above observations, we define the cost for each transformation operation as follows:

- Deleting cost.

$\gamma(delete(v)) = \frac{height(T) - depth(v) + 1 + |D(v)|}{|V|}$, where v is a non-root node, height(T) is a function calculating the height of tree T, depth(v) calculates the depth of node v, and $|D(v)|$ is the number of descendants of node v (including its direct children and indirect offspring). Intuitively, depth(root(T)) = 1, and depth(v) > 1 iff v is not the root. If v is a leaf node, D(v) = ∅ and |D(v)| = 0. When v is a leaf node at the lowest level (height(T) = depth(v)), deleting v will cause the minimal effect to the tree and γ(delete(v)) = 1/|V|. Note that here V refers to the original node set before the deletion.

- Inserting cost.

$\gamma(insert_u(v)) = \frac{height(T) - depth(u) + 1 + |D(v)|}{|V|}$, where $|D(v)|$ is the number of descendants that v gets after it is inserted. Note that here V refers to the original node set before the insertion. When u is at the lowest layer, inserting a new node v under u will result in the minimal cost γ(insert$_u$(v)) = 1/|V|.

- Moving cost.

$\gamma(move_u(v)) = \frac{1}{2}[\gamma(delete(v)) + \gamma(insert_u(v))] \times \frac{|V|-2}{|V|}$, where |$V$|>2 (the tree has a root and at least two non-root nodes) and $u \neq$ parent(v). Note that here insert$_u$(v) is performed on a tree without node v. In this definition we consider both deleting and inserting operations because the moving operation does generate effects similar to deleting and inserting, although not exactly the same. The factor 1/2 adjusts the cost of operations since the node is not truly deleted and inserted into the tree. Another factor (|V| - 2)/|V| adjusts the cost again to ensure that in an extreme case where v is the only node other than the root, its moving cost should be 0 (actually it cannot be moved) and when the number of nodes in the tree grows, the effect of the moving operation to the tree structure is less.

- Re-labelling cost.

This cost is heavily dependent on the similarity of two labels (concepts). The re-labelling cost is different from the deleting cost, inserting cost, or moving cost since the re-labelling operation does not result in the change of a tree structure. Kouylekov et al. [Kouylekov and Magnini, 2005] proposed a definition for substitution of two similar words w_1, w_2 as $\gamma(\text{insert}(w_2))\times(1 - \text{sim}(w_1, w_2))$ where insert(w_2) is the cost of inserting w_2 and sim(w_1, w_2) is the similarity between w_1 and w_2. This definition does not take the deletion of the original word into consideration, therefore when two words have no conceptual similarity the cost of substitution becomes the cost of insertion, neglecting the implicit deleting operation. In our work we give a more comprehensive definition.

Let the conceptual similarity measure between two labels l_{v1}, l_{v2} which are attached to node v be s, $0\leq s\leq 1$, we define:

$$\gamma(relabel_{l_{v1}\to l_{v2}}(v)) = [\gamma(delete(v)) + \gamma(insert_{parent(v)}(v))]\times(1-s)$$

We analyze two extreme cases: if $s = 1$, then re-labelling will only result in literal replacing without any loss of information, therefore the re-labelling cost is 0; if $s = 0$ (i.e., the two concepts are totally different), the re-labelling operation is equivalent to deleting v and inserting v again, the transformation cost is $\gamma(\text{delete}(v)) + \gamma(\text{insert}_{\text{parent}(v)}(v))$. In other cases, the cost will be between these two boundaries.

4.4.2.6 Cost Computing Algorithm

The cost computing algorithm is composed of a pre-processing phase and a transforming phase, as depicted below. The pre-processing phase finds the nodes that are to be deleted and inserted. In the transforming phase, an exhaustive method is used to try every possible transformation sequence to find the minimal cost.

A. The pre-processing phase.

Input: Tree $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$; Concept similarity measure set $S_{L^{V_1},L^{V_2}}$

Output: Sets of nodes to be deleted, ***D***, and inserted, ***I***

Algorithm:

```
1)  $D = \varnothing$;
2)  for every node $u$ in $V_1$
3)  {
4)     if(not exists any $l$ in $L^{V2}$ such that $M_1(u) = l$)
5)        if(not exists any $s(M_1(u), l)$ in $S_{L^{V_1},L^{V_2}}$ )
6)           add $u$ into $D$;
7)  }
8)  $I = \varnothing$;
9)  for every node $v$ in $V_2$
10) {
11)    if(not exists any $l$ in $L^{V1}$ such that $M_2(v) = l$)
12)       if(not exists any $s(l, M_2(v))$ in $S_{L^{V_1},L^{V_2}}$ )
13)          add $v$ into $I$;
14) }
15) return $D$ and $I$;
```

B. The transforming cost computing phase.

Input: Tree $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$; ***D, I***; Concept similarity measure set $S_{L^{V_1},L^{V_2}}$

Output: $\boldsymbol{\gamma(T_1 \rightarrow T_2)}$

Algorithm:

```
1)  find all permutations composed by elements in $D \cup I$ and store in $P$;
2)  $transformCost = +\infty$;
3)  for each permutation $p$ in $P$
4)  {
5)     backup $T_1$ and $T_2$;
6)     $editCost = 0$;
7)     for each element $u$ in $p$
8)     {
9)        perform deletion (if $u \in D$) or insertion (if $u \in I$) on $u$ if applicable;
10)       $editCost = editCost +$ ($\gamma$(delete($u$)) or $\gamma$(insert($v$)));
11)    }
12)    for each $u$ in $V_1$ but not in $p$
13)    { /* handle the nodes to be moved. */
14)       if(exists $l$ in $L^{V2}$ such that $M_1(u) = l$ or exists any $s(M_1(u), l)$ in $S_{L^{V_1},L^{V_2}}$ )
15)          if($M_1$ (parent($u$)) $\neq M_2$ (parent($M_2^{-1}(l)$)) and
```

```
16)             not exists any $s(M_1(\text{parent}(u)), M_2(\text{parent}(M_2^{-1}(l)))$ in $S_{L^{V_1},L^{V_2}}$)
17)             perform moving on $u$;
18)             $editCost = editCost + \gamma(\text{move}(u))$;
19)     }
20)     for each $u$ in $V_1$ but not in $p$
21)     { /* handle the nodes to be re-labelled. */
22)         if(exists $l$ in $L^{V2}$ such that exists any $s(M_1(u), l)$ in $S_{L^{V_1},L^{V_2}}$)
23)             perform re-labelling on $u$;
24)             $editCost = editCost + \gamma(\text{relabel}(u))$;
25)     }
26)     transformCost = min(transformCost, editCost);
27)     restore $T_1$ and $T_2$;
28) }
29) return transformCost;
```

In this algorithm, a backup operation and a restore operation are included, which are used to setup a common starting point each time a new operation sequence is tried.

The same algorithm can be used to compute the cost of converting T_2 into T_1, therefore the similarity index of T_1 and T_2 can be determined.

Following, we give the time complexity analysis of the algorithm: Given two trees $T_1 = (V_1, E_1, L_1^{V1}, \text{root}(T_1), D, M_1)$, $T_2 = (V_2, E_2, L_2^{V2}, \text{root}(T_2), D, M_2)$, and a conceptual similarity measure $S_{L^{V_1},L^{V_2}}$, let $|V_1|$ and $|E_1|$ be the number of nodes and edges in T_1, $|V_2|$ and $|E_2|$ be the number of nodes and edges in T_2, so the upper bound of $|S_{L^{V_1},L^{V_2}}|$ is $|V_1|\times|V_2|$. In the pre-processing phase, the times to search T_1, T_2 as well as $S_{L^{V_1},L^{V_2}}$ are:

$$|V_1|\times|V_2|\times|V_2| + |V_2|\times|V_1|\times|V_1|$$

Without loss of generality, we assume that two trees have similar sizes. That is, $|V_1| \approx |V_2| \approx n$. Therefore, we have $|E_1| \approx |E_2| \approx n\text{-}1$. The time complexity of the pre-processing phase is $O(n^3)$.

In the cost computing phase, on average half of the nodes in T_1 may be deleted and half of the nodes in T_2 need to be inserted, so the complexity of getting the permutations of $D \cup I$ is $O(n(n + 1)/2) = O(n^2)$. The average times of deleting and inserting nodes are n. When moving the nodes, on average $n/4$ nodes can be moved (half of the untouched nodes), and the time complexity of finding the position to move for each node is $O(n/4 + n/4) = O(n/2)$ (considering both the node itself and its parent node). The time complexity of the relabeling operations is $O(n/4)$. Therefore, the time complexity of the cost computing phase is $O(n^2) \times O(n + n/2 \times n/2 + n/2) = O(n^4)$.

To sum up, the time complexity of the algorithm is $O(n^3) + O(n^4) = O(n^4)$. Usually in an ontological view the number of concepts is limited and the comparison is often a one-time action, therefore the cost is acceptable although better tree comparison algorithms can be explored to reduce the cost.

4.5 Instance-based Approach

4.5.1 Introduction to the Approach

In the family of schema matching approaches, instance-based approaches [Doan, et al., 2001] can utilize the data instances which imply plenty of valuable clues for the potential attribute matches. When comparing concepts in different ontological views, the fact that data instances are maintained in the information repositories can be applied to increase the precision of discovering semantic equivalence relationship between the concepts.

One of the major issues of these approaches is the cost of manipulating a large quantity of raw data. One solution to increase the efficiency is to use instance representatives (with each representing a set of data instances) for the analysis instead of using all raw data. The clustering methods can be applied as a solution.

Some research also uses clustering methods to find closely related schema elements. For example, Pei et al. [Pei, et al., 2006] proposed a new approach for schema matching by clustering schemas on the basis of their contextual similarity and clustering attributes of the schemas that are in the same schema cluster to find attribute correspondences between these schemas. The approach also clusters attributes across different schema clusters using statistical information gleaned from the existing attribute clusters to find attribute correspondences between more schemas. Smiljanic et al. [Smiljanic, et al., 2006] presented a clustering-based technique for improving the efficiency of XML schema matching by partitioning schemas with clusters and reducing the overall matching load. In this work clustering is used to quickly identify regions, i.e., clusters, in the large schema repository which are likely to produce good mapping. This research has a different context than our work, i.e., they cluster the schema elements instead of clustering the data instances. Also, no work was done based on the concept of ontological views.

4.5.2 Instance-based Semantic Equivalence Relationship Discovery

The semantic similarities of the concepts in ontological views can be computed based purely on the representations of their properties. However, the data instances, when available, can provide many more useful clues to help discover the similarity of the properties regardless of how they are represented. The probability distribution (or probability density) is one of the often-used approaches to analyze the instance values. Basically, if two properties of two concepts have compatible data types (the data type can be known from the schema) and the probability distributions of their data instances are identical or very close, then it is reasonable to infer that these two properties are very likely to be semantically similar. In the following sections we discuss the problems to solve and the corresponding solutions.

4.5.2.1 Estimation of Probability Density of Data Instances

The first problem is how to estimate a probability density function $f(x)$ given a sequence of independent and identically distributed random variables x_1, x_2, …, x_n (data instances of a property) from this density f.

There is a rich collection of non-parametric density estimators, including kernel, spline, orthogonal, series, and histogram [Bean and Tsokos, 1980].

We adopt the Kernel density estimation method [Turlach, 1993 and Wasserman, 2005] to compute the probability distribution of the data instances. In statistics, Kernel density estimation is a non-parametric way of estimating the probability density function of a random variable. Different than many distributions, the Kernel density estimation is smooth and independent of end points. It just depends on the bandwidth. The definition of kernel density estimation is presented as follows.

If x_1, x_2, …, $x_N \sim f$ is an independent and identically-distributed random variables sample of a random variable, then the kernel density approximation of its probability density function is

$f_h(x) = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x - x_i}{h})$, where K is some kernel and h is the bandwidth (smoothing parameter). Quite often K is taken to be a standard Gaussian function with mean zero and variance 1, and h is computed according to the standard deviation (S) of the N values [Scott and Sain, 2004]:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, \quad h = \frac{1.06S}{\sqrt[5]{N}}.$$

4.5.2.2 Comparison of Probability Densities

After the probability densities of the properties are computed, it is necessary to compare them and check their similarity. The question here is how to compare different probability densities.

We employ the Kullback-Leibler (K-L) divergence approach [Kullback, 1987] to compare the probability densities. In the probability theory and information theory, the K-L divergence is a non-commutative measure of the difference between two probability densities.

For probability densities f_1 and f_2 of a continuous random variable, their K-L divergence is defined as

$$\delta(f_1, f_2) = \int_{-\infty}^{\infty} f_1(x) \log \frac{f_1(x)}{f_2(x)} dx .$$

Although a property is a continuous variable, in practice it should be manipulated as a discrete random variable in order to compute the K-L divergence. The solution is to sample a set of values (with each denoted as s_i) from the domain of two properties under comparison, then compute the probability of each value according to the probability density functions (denoted as $f_1(s_i)$ and $f_2(s_i)$), and finally compute the K-L divergence by

$$D_{K-L} = \sum_i f_1(s_i) \log \frac{f_1(s_i)}{f_2(s_i)}$$

4.5.2.3 Clustering of Data

In the instance-based analysis, another problem emerges when using original data instances to compute the probability densities and compare them. The computation cost is very high due to the large amounts of raw data. The solution is to cluster the data first, and then compute the probability densities based on the clustered data.

Cluster analysis [Kotsiantis and Pintelas, 2004], also called data segmentation, relates to grouping or segmenting a collection of objects (also called observations, individuals, cases, or data rows) into subsets or "clusters" such that those within each cluster are more closely related to one another than objects assigned to different clusters. Since the objects in each cluster are closer or similar to each other, it is reasonable to use one typical object within one cluster to represent the entire cluster. The typical object is a weighted cluster centre which can represent a set of values similar to the centre itself. The use of a typical object will significantly reduce the size of the problem.

Hierarchical clustering is one of the major methods of cluster analysis. Hierarchical clustering is subdivided into agglomerative methods, which proceed by a series of fusions of the n objects into clusters, and divisive methods, which separate n objects successively into finer clusters. A key component of the analysis is repeated calculation of distance measures between objects, and between clusters once objects begin to be grouped into clusters.

The initial data for the hierarchical clustering of N objects is a set of $\frac{N \times (N-1)}{2}$ object-to-object distances and a *linkage function* for computation of the cluster-to-cluster distances. The linkage function is an essential prerequisite for hierarchical clustering. Its value is a measure of the *distance* between two groups of objects, i.e. two clusters.

A commonly used linkage function is *complete linkage clustering*, in which distance between groups is defined as that of the furthest pair of individuals, where a pair consists of one member from each cluster. Mathematically, the complete linkage function—the distance $D(X, Y)$ between clusters X and Y is defined as $D(X, Y) =$

max($d(x, y)$), $x \in X$ and $y \in Y$, where $d(x, y)$ is the distance between elements $x \in X$ and $y \in Y$, and X and Y are two sets of elements (two clusters).

Complete linkage clustering is an agglomerative method. It starts from the clusters initially containing one element each and successively fuses them to generate larger clusters. Therefore, the two clusters with the lowest distance are joined together to form the new cluster. At each step, the clusters to be used are those that are, according to some pre-defined metric, most similar to each other.

The above discussion shows that the distance between elements is the foundation of cluster analysis. An important task in any clustering is to select an appropriate distance measure, which will determine how the similarity of the two elements is calculated. This will influence the shape of the clusters, as some elements may be close to one another, according to one distance and further away according to another.

At the information level, we consider generic metric space, not definitely pure Euclidean Space (i.e., it is only required that the distance between any pair of elements is known. It is not limited to the coordinates of points). A metric of a set X is a function (called the distance function or simply distance) d: $X \times X \rightarrow R$, where R is the set of real numbers. For all x, y, z in X, this function is required to satisfy the following conditions:

(1) $d(x, y) \geq 0$ (non-negativity)

(2) $d(x, y) = 0$ if and only if $x = y$ (identity of indiscernibles). Condition (1) and (2) together produce positive definiteness.

(3) $d(x, y) = d(y, x)$ (symmetry)

(4) $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

In an information system we usually face three types of data: numeric data, date-time, and text string. Therefore, we define the distance metric for the three types:

If *x*, *y* are values of concept instances on property *X*, the distance between *x* and *y*, *d*(*x*, *y*), is defined as:

- Euclidean distance in Euclidean one dimensional space, $|x - y|$, if the type of *X* is numeric;
- Euclidean distance in Euclidean one dimensional space, |absolute_time(*x*) – absolute_time(*y*) |, if the type of *X* is date-time, where absolute_time is a function to map each date time to a long integer;
- Edit distance of string, if the type of *X* is text string. The edit distance *d*(*x*, *y*) is the minimal cost for a sequence of edit operations to transform *x* to *y*.

  The edit operations include:

  (1) Replace one character in *x* by a character from *y*;

  (2) Delete one character from *x*;

  (3) Insert one character from *y*.

  The cost model is defined as:

  $$c(a,b) = \begin{cases} 1, & \text{if } a \neq b \\ 0, & \text{if } a = b \end{cases}$$

  a and *b* can be ε (null character) meaning inserting a new character *b* or deleting an existing character *a*.

After the clusters are created, we expect to use the representative data instance in each cluster, i.e. the cluster centre, to represent the entire set of data instances in the following analysis. This is known as a 1-median problem [Drezner, et al., 1986] which is defined as follows:

Given a universe *U*, a finite multi-set of points *P*, and a metric *d*, a 1-median is a point $m \in U$ that minimizes the objective function

$$\sum_{p \in P} d(p,m)$$

In this definition, *m* is a valid member in *U* but not definitely a point in *P*. It is an optimal one to represent others since the median point is relatively closer to other points (in terms of the selected distance metric).

The basic idea of the algorithm of finding the 1-median point is: for a point $p \in P$, let $S(p) = \sum_{x \in P} d(p,x)$, then conduct a series of comparisons between $S(p)$, $p \in P$ to find a point *q* that minimizes the value of *S*. The point *q* is the cluster center under the 1-median's definition.

Chapter 5 Implementation and Result Validation

5.1 Implementation

The implementation of this research includes two stages: 1) mapping the proposed solution to technologies; and 2) creating engineering solutions using the adopted technologies. Section 5.1.1 focuses on the first stage. The second stage is described in section 5.1.2 and 5.1.3.

5.1.1 **Mapping of Proposed Solution to Technology**

In section 4.3, we propose to use *frame* as the paradigm to model the ontological views. The fundamental elements within *frame* include:

- Frames representing concepts.
- Slots representing properties of concepts or relationships with other concepts.
- Facets representing characteristics of properties.
- Data representing instances of a concept.

These elements should be mapped to constructs provided by an implementation technology (except the Data element that may be unnecessary in some modeling situations) to guarantee that the solution can be supported by the technology. We consider two types of technologies:

- Relational model.
- RDF-based model.

The following sections show how the mapping is achieved.

5.1.1.1 Mapping to Relational Model

Objects in the real world can be abstracted as data with specifications in the computational world. A data model provides a uniform way to specify and represent data. The relational model [Codd, 1990] was the first data model theoretically founded and well thought out. It has become the foundation of the relational database technology.

The fundamental assumption of the relational model is that all data is represented as mathematical *n*-ary **relations.** Briefly, the relational model structures the logical view of data around two mathematical constructs: domains (i.e., data types) and relations. The name *relational* comes from "relation" as known and widely used in mathematics, although in database theory the definition of relation is slightly extended.

A *domain* is simply a set of values, together with its associated operators. It is equivalent to the notion of a *type* in programming languages.

A relation over the domains D_1, D_2, ..., D_n is simply a subset of the Cartesian product; the usual notation is R "*included in*" $D_1 \times D_2 \times ... \times D_n$. An element of the Cartesian set is called a tuple. A database is a collection of "relation valued" variables, together with the set of integrity constraints that the data must satisfy. A relation can also be viewed as a structure describing the relationships between things in the real world.

Each domain that defines a relation is associated with a string label (that is called *attribute name*). An *attribute* is then the association between an attribute name and a domain. In other words, an attribute has a name and a domain. A relation header is then a set of attribute names. A tuple becomes the mapping between each attribute name in the relation header and a value. And a relation is a set of tuples, all corresponding to the relation header.

A key of a relation is composed of one or more attributes. The value of a key uniquely identifies each tuple. A relation may have many keys, each of which is called a candidate key. Every relation has at least one candidate key. One candidate key is selected as the primary key.

A foreign key is composed of one or more attributes whose values are used elsewhere as primary key values. The primary key and foreign key are defined on the same domain but do not necessarily have the same attribute names.

Besides the structure of data, the relational model also defines the means for data manipulation (relational algebra or relational calculus) and the means for specifying and enforcing data integrity (integrity constraints).

Mapping between the frame to the relational model enables the adoption of a relational model as an implementing technology. According to the definition of frame and relational model, the following mapping rules are defined:

- A frame is mapped to a relation.
- A slot of a frame representing a property is mapped to an attribute of a relation.
- A facet of a slot is mapped to a domain.
- A set of values on all slots representing an instance is mapped to a tuple.
- The set of one or more slots that uniquely identify an instance is mapped to a primary key.
- A slot representing a relationship to another frame is mapped to a foreign key or a relation, all of which attributes are foreign keys.

The relational model is the foundation of relational database systems. To apply the relational model,

- *Type* is used to implement a domain. A type may be the set of integers, the set of character strings, the set of dates, or the two boolean values *true* and *false*, and so on. The corresponding type names for these types might be the strings "int", "char", "date", "boolean", etc.

- Attribute is the term used in the theory for what is commonly referred to as a *column* in a relational database.

- The database systems provide rich characteristics, besides name and type, for attributes, e.g., value range, null-able, default value, etc.

- *Table* is commonly used in place of the theoretical term *relation*. A table structure is specified as a list of column definitions, each of which specifies a unique column name and the type of the values that are permitted for that column.

- A tuple is basically the same thing as a *row*.

Based on these rules, the frame model is mapped to the relational model, which is further implemented by the relational database technology.

5.1.1.2 Mapping to XML-based Models

XML is a standard for specifying data on the Web in a structured manner. Strictly speaking, XML is a formalism of encoding information. An XML document is a flat file with a rigid structure to specify concepts. It may follow the concept-property paradigm and be compatible with a relational model.

An XML schema is helpful for defining the valid structure of an XML document.

A concept can be mapped to an element within an XML document. The element may have multiple attributes, each of which is corresponding to a property of the concept.

Another way is to map a property to a sub-element of an element within an XML document. The different situations show that the structure of XML can be quite arbitrary in terms of how the concepts are modeled. For example, each of the following two XML fragments shows a valid modeling of the concept *product*:

```
<product>
      <name>Donut</name>
      <price>1.99</price>
</product>
```

```
<product name="donut" price="1.99" />
```

The frame model is compatible with either case. However, creating a frame-based ontological view from such arbitrary models may pose a significant challenge. In our solution, we assume that the XML documents follow a given format:

```
<concept-group>
      <concept-name attribute-list />
      <concept-name attribute-list />
      <…>
</concept-group>
<relationship-group>
      <relationship-name>
            <subject-concept-name identifier-attribute-list />
            <object-concept-name identifier-attribute-list />
      </relationship-name>
</relationship-group>
```

The following is an example about products:

```
<products>
      <product name="donut" price="1.99"/>
      <product name="cookie" price="1.49" />
</products>
<times>
      <time name="weekend" start="Saturday" end="Sunday">
</times>
<product_time_rules>
      <product_time_rule>
            <product name="donut" />
            <time name="weekend" />
      </product_time_rule>
</product_time_rules>
```

It still follows the concept-property structure. A concept in a frame is mapped to an element in an XML document. The properties of a concept is mapped to attributes of an element. A relationship is denoted by a specific element that has multiple sub-elements indicating the subject concept and object concept in the relationship. Note that in an XML document the instance data is embedded. A schema can be extracted from a valid XML document. The facets of a property are not directly specified in an XML document but can be specified in the schema as further attributes of an attribute within the document.

In the current stage we only consider binary relationships but the solution can be extended to support multi-arity relationships.

A wrapper is created to convert the XML-based model into an ontological model. The wrapper can be enhanced to support more formats.

5.1.1.3 Mapping to RDF Model

RDF is a general model for conceptual description of the modeling of information that is implemented in Web resources, using a variety of syntax formats.

The RDF data model is similar to classic conceptual modeling approaches such as the Entity-Relationship model, as it is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions. These expressions are known as triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object.

A frame model can be mapped to a RDF model guided by the following rules:

- A frame is mapped to a subject, representing a concept.

- A property is mapped to an object, and the predicate that associates the subject and the object is defined as a "has-property" relationship.

- An object can be a resource on the Web. Therefore, the object can be a subject in another statement. In this sense, a property is mapped to a subject, and its facet is mapped to an object. The predicate that associates the subject and the object is defined as a "has-facet" relationship.

- An instance is mapped to an object with a predicate "has-instance" between a concept and itself. The instance further acts as a subject, and the property values act as objects. The predicate between the instance and value is defined as a relationship "has-*property*" where *property* refers to a specific property on which the instance gets a value.

XML can be used as a serialization format of RDF. With such mappings an ontological view model can be implemented as an RDF model, and further represented with XML. In our engineering solution, RDF is not actually used since we focus on typical information systems and these systems are usually not Web-based. They concern information but such information is not treated as Web resources.

5.1.2 Prototype Environment

The proposed solutions are applied to a collaborative intelligence prototype environment. Collaborative intelligence refers to a mechanism for semantically integrating decentralized business intelligence and providing a comprehensive knowledge foundation which can be utilized to achieve various goals. In another sense, collaborative intelligence can be viewed as a kind of artifact that is produced by a specific mechanism by collecting distributed intelligence, resolving semantic heterogeneities, and converting to an expected form.

The collaborative intelligence mechanism is used to improve the product promotion/advertisement domain and facilitate collaborative promotion. QSR (Quick Service Restaurant) is a typical business that requires collaborative promotion. In such a business, the promotion is achieved by displaying multimedia contents on digital displays installed in many stores. The multimedia contents contain information about various products, such as images, product names, prices, effective dates of promotions, etc. The decision about what product to promote, when and where to promote, and what multimedia content to play should be made based on the overall knowledge of the entire business. Information systems managing the media assets, product inventory, sales transactions, device schedules, etc. were originally developed in a separate manner without considering collaboration in the future. The systems share some common concepts within the business; however, they adopt different ways of modeling the domain, resulting in heterogeneous information models.

A basic promotion criterion requires promoting a particular product more than another, i.e., playing a media asset representing that product more frequently, if that product reaches higher inventory level. To achieve such objectives the information systems should be semantically integrated, so the scattered information can be exchanged and understood by each system and therefore final decisions can be made. Collaborative promotion, as a business strategy, is expected to be applied to achieve more efficient and flexible promotion decisions.

Figure 5-1 illustrates the physical architecture of a collaborative promotion environment:

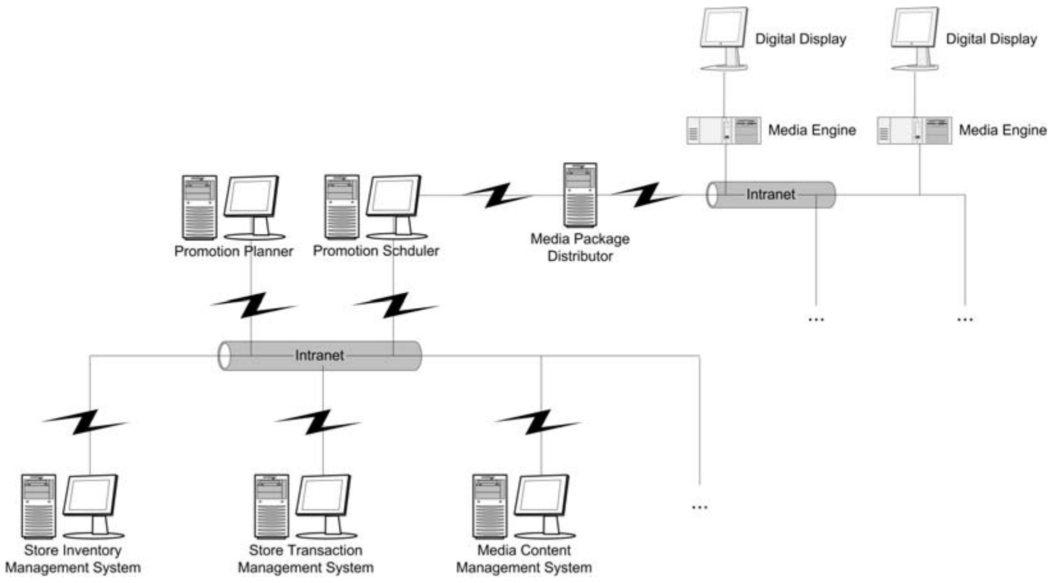
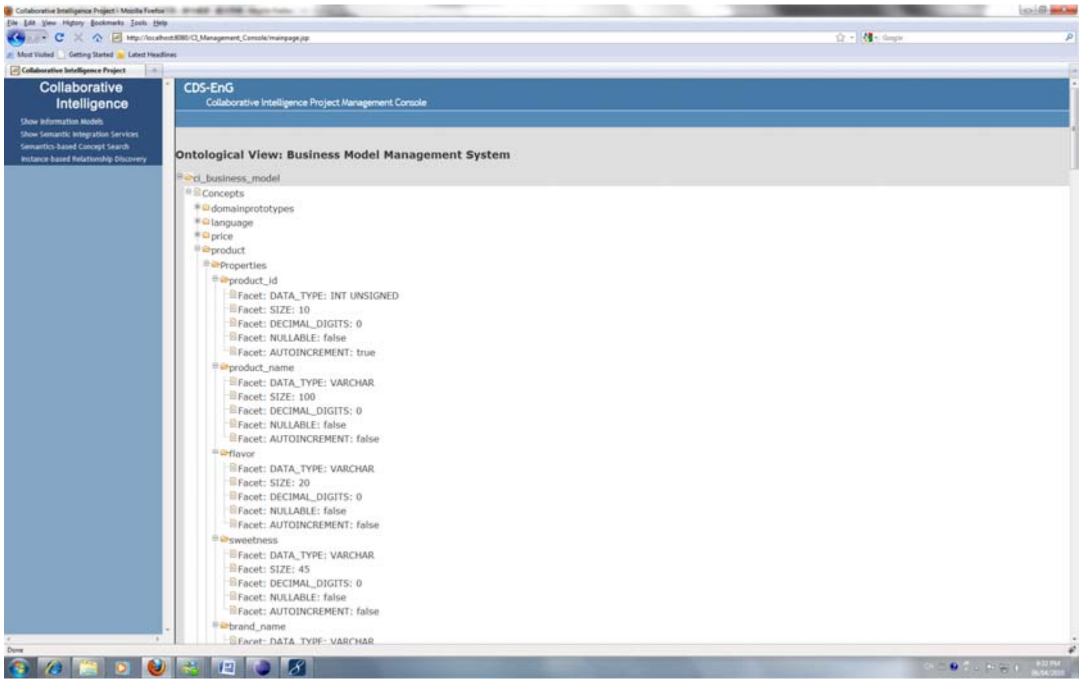

Digital Display
Digital Display
Media Engine
Media Engine
Promotion Planner
Promotion Schduler
Media Package Distributor
Intranet
...
...
Intranet
...
Store Inventory Management System
Store Transaction Management System
Media Content Management System

Figure 5-34. Architecture of a collaborative promotion environment.

In the environment some systems (e.g. store inventory management system, store transaction management systems, etc.) provide essential information and some other systems (e.g. promotion planner, promotion scheduler) utilize the information to make decisions. Information maintained by these systems, such as the promotion schedule, can also be used by other systems for further analysis and decision making.

The access to the information models of these systems is guaranteed. A Web-based management console is deployed to the environment. The console is able to list the systems within the system. Each system is identified by a unique system ID and a system name. From the console, users can browse each system's information model and the created ontological view.

The following screenshot shows the interface of the management console that presents an ontological view created from the business model management system.


Collaborative Intelligence Project - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/CI_Management_Console/mainpage.jsp
Most Visited Getting Started Latest Headlines
Collaborative Intelligence Project
Collaborative Intelligence
Show Information Models
Show Semantic Integration Services
Semantics-based Concept Search
Instance-based Relationship Discovery
CDS-EnG
Collaborative Intelligence Project Management Console

```
Ontological View: Business Model Management System
ci_business_model
  Concepts
    domainprototypes
    language
    price
    product
      Properties
        product_id
          Facet: DATA_TYPE: INT UNSIGNED
          Facet: SIZE: 10
          Facet: DECIMAL_DIGITS: 0
          Facet: NULLABLE: false
          Facet: AUTOINCREMENT: true
        product_name
          Facet: DATA_TYPE: VARCHAR
          Facet: SIZE: 100
          Facet: DECIMAL_DIGITS: 0
          Facet: NULLABLE: false
          Facet: AUTOINCREMENT: false
        flavor
          Facet: DATA_TYPE: VARCHAR
          Facet: SIZE: 20
          Facet: DECIMAL_DIGITS: 0
          Facet: NULLABLE: false
          Facet: AUTOINCREMENT: false
        sweetness
          Facet: DATA_TYPE: VARCHAR
          Facet: SIZE: 45
          Facet: DECIMAL_DIGITS: 0
          Facet: NULLABLE: false
          Facet: AUTOINCREMENT: false
        brand_name
          Facet: DATA_TYPE: VARCHAR
```

Figure 5-35. Screenshot of the management console.

5.1.3 **Implementation of Services**

5.1.3.1 Registration Service

Collaborative intelligence is produced in an open environment. In this environment multiple information systems maintain business intelligence. A semantic integration service is expected to be attached to each information system and be in charge of resolving semantic heterogeneities.

To facilitate the systems' awareness of the existence of semantic integration services, a registration service is deployed to the environment. The registration acts as a yellow-page. A look-up against the registration service results in the address information of one or more semantic integration services.

The mechanism to access the registration service is built-in knowledge for all the semantic integration services. Technically, the registration service is provided as a

Web Service, therefore, its end point URL, methods provided within the service, and usage of the methods are commonly known by each semantic service. The end point URL is a publicly available configuration entry. The usage of each method include the meaning and functionality of the method, the meaning of input parameters, and the meaning of the returned result.

The registration service can provide the following functions:

- Support registration of a semantic integration service.
- Support de-registration of a semantic integration service.
- Maintain a unique ID / name pair for each registered semantic integration service.
- Maintain an end point URL for each registered semantic integration service.
- Return a list of ID / end point URL pairs upon a request.

5.1.3.2 Semantic Integration Service

A semantic integration service, technically a Web Service, is attached to an information system as a plug-in. The semantic integration service is responsible for discovering semantic equivalence relationships between concepts from various information models. The discovered semantic equivalence relationships will serve to resolve semantic heterogeneities.

The functionality for the semantic integration service includes:

(1) Accessing the information system to get its information model as well as the instance data.

(2) Creating an ontological view based on an information model.

(3) Performing analysis on the concepts and instance data to discover semantic relationships among the concepts from various ontological views.

(4) Managing the concept model implied by the information system to which it is attached. It can answer two questions:

- Given a concept specification (as an income request), it can tell which concept within its ontological view that is possibly the same as the income concept and how possible it is;
- If an income concept is possibly the same as one internal concept, how their properties are the same as each other, respectively.

(5) Contacting the registration service to know what other services also reside in the environment.

(6) Fetching an ontological view from another service (using the address got from the registration service), performing analysis on its own ontological view as well as the other and discovering potential equivalence relationships.

The discovered relationships are treated as alignments. We adopt a tailored version of the INRIA [INRIA, 2010] format to represent the alignments. The format is specified as follows.

- Alignment class

  The *Alignment* class describes a particular alignment between two ontological views. Its properties are the following:

  ov1: (value: OntologicalView) the first ontological view to be aligned;

  ov2: (value: OntologiclaView) the second ontological view to be aligned;

  map: (value: Cell*) the set of equivalence relationships between concepts of the ontological views.

- Cell class

  concept1: (value: Concept) the reference of a concept of the first ontological view;

  concept2: (value: Concept) the reference of a concept of the second ontological view;

  measure: (value: float number between 0 and 1) the confidence in the assertion that the relationship holds between the first and the second concept.

5.2 Validation

The proposed solutions are validated based on the application prototype for the collaborative promotion / advertisement domain. The validation focuses on the following aspects of the solutions:

(1) Completeness of the frame-based ontological view specification language.

This is to validate whether the language can express all mandatory elements from regular information models that will be used in semantic integration, i.e., it contains all constructs that are needed.

Traditionally, the validation of modeling is a process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model [AIAA, 1998]. This is also related to another characteristic of modeling: expressiveness, which refers to the power of modeling scenarios.

In our work, we assume that the accuracy of the original modeling is guaranteed. That is, validating the model itself is not necessary. Also, we don't worry about the expressiveness of the original modeling paradigm since we assume that it is expressive enough to model the scenarios which the application cares about.

Therefore, we only focus on the generated ontological view model and validate the degree to which it reflects the original model.

(2) Richness of the frame-based ontological view specification language.

Richness refers to the power of built-in abstraction directly, i.e., the direct constructs provided for modeling things at a different abstraction level.

(3) Completeness of the approaches. There are two aspects to validate:

(i) The approach for creating ontological views from information models. This is to validate that the approach can capture all mandatory elements from the identified information models.

(ii) The approach for discovering semantic equivalence relationships between concepts from different ontological views. This is to validate that all relationships can be discovered.

(4) Soundness of the solution. It contains the following aspects:

(i) Whether the adoption of ontological views can address the schematic and syntactic heterogeneities of the information models and create a common platform for semantically integrating the information models.

(ii) Whether the discovered semantic equivalence relationships can address the semantic heterogeneities of the information models.

(iii) Whether the overall architecture can achieve its purpose, i.e., if the same concept is modeled and represented in different ways among the information systems, this fact can be identified, and therefore, a search request regarding a specific concept can be collaboratively satisfied by multiple information systems within the environment that have models of that concept.

5.2.1 Analytical Validation

5.2.1.1 Completeness of the Frame-based Ontological View Specification Language

The FOSL language is based on the *frame* modeling paradigm. The completeness of the language lies in three aspects:

- The completeness of the modeling paradigm, i.e., if the modeling paradigm is able to model all mandatory elements, i.e., extrinsic concepts, intrinsic concepts (properties), characteristics of properties, and relationships.
- The vocabulary and syntax of the language, i.e., if the above elements can be specified by the constructs of the language.
- The transferability with other modeling languages, i.e., if all necessary constructs of another modeling language can be mapped to the constructs of the language.

Generally speaking, an information model is an abstraction and formal representation of a domain of discourse. An information model is developed following a specific modeling paradigm (also referred to as a knowledge representation paradigm). The information implied by a model is relying highly on how the symbolic system is interpreted. Considering the availability of an interpretation, any formal or informal representation can express some information. In other words, any data structure in computer systems can be a specific representation of a model. For example, we can use a one-dimensional array [0, 1, 2, …] to represent different characters in the game of chess, e.g., use 0 to represent the king and 1 for the queen. We can also employ a three-dimensional array [[5, 4, 0], [5, 5, 1], …] to represent a chess game, each element of which represents the character at a specific position (e.g., the king is at the cell of 5th row and 4th column). Such a representation can express specific worlds, but too much information is implied by the simple array formalism and a complicated interpretation is required. A good specification language should make the implied information as explicit as possible.

In terms of the elements to model, many of the modeling paradigms, including first order logic [Sumllyan, 1995], description logic [Badder and Sattler, 2001], production rules [Klahr, et al., 1987], conceptual graph [Sowa, 2005], semantic network [Kendal and Creen, 2007], F-logic [Kifer, et al., 1995], entity-relationship model, object-oriented model, RDF, etc., model the world around the notion of *concept*. That is, these paradigms are able to specify individual elements that can be mapped to concepts. Some other paradigms do not have the explicit notion of *concept*. State-space is the earliest representation formalism used extensively in Artificial Intelligence [Barr and Feigenbaum, 1981]. It represents the structure of a problem in terms of the alternatives available at each possible state of the problem. It uses specific forms to represent the states that involve objects. Explicit interpretation is necessary to explain how the objects and relationships are arranged in the states. Specific applications are required to decide how the transitions can occur between the states.

In a procedural representation [Barr and Feigenbaum, 1981], knowledge about the world is contained in procedures—small programs that know how to do specific things, how to proceed in well-specified situations. For instance, in a parser for a natural language understanding system, the knowledge that a *noun phrase* may contain articles, adjectives, and nouns is represented in the program by calls to routines that know how to process articles, nouns, and adjectives. In this paradigm concepts are not stated explicitly and thus is neither typically extractable in a form that humans can easily understand, nor reusable by other programs.

Many expert systems use decision-making rules [Kendal and Creen, 2007] that can be represented using the **IF…THEN** format, that is

IF <*situation*> **THEN** <*action*>

Other clauses such as **OR** and **ELSE** can also be used with this construct to show alternative situations or different courses of action. Rules in a knowledgebase system (KBS) stand along as statements of truth or fact and can be used by an inference engine to reach other true conclusions. This representation does not provide a standard way to specify concepts in the *situation* and *action* part.

Propositional logic is one approach for representing knowledge in many expert systems. In this approach, the elementary building blocks, propositions, are *atomic statements* that cannot be decomposed any further, e.g., "It is raining", "Tom is a student". Logical connectives like "and", "or", "not" can be used to build propositional formulas. Similarly, there is no standard way to specify concepts in the propositions.

Among the paradigms that have the notion of concept, first order logic and production rules do not differentiate concepts and instances of concepts. Others can specify concepts and instances separately. For example, in conceptual graphs, each concept has a concept type and referent such as [Person: Tom].

All the paradigms that have the notion of *concept* also support the notion of *relationship* that associates concepts. For example, in first order logic a relationship can be represented as *sell*(Store, Product).

Most of the paradigms do not provide facilities to model further details such as properties of concepts as well as characteristics of properties. Properties and the further characteristics actually refer to relationships with specific meanings. The entity-relationship model, object-oriented model, and frame provide means to model all these aspects. Production rule has the *entity-attribute-value triple* structure, which can be viewed as a form to represent properties of concepts. The entity-relationship and object-oriented paradigm can model characteristics of properties but the capability is not complete. It is completed at the supporting technology level such as the relational database and application written with specific OO languages, but not at the modeling level.

Many of the modeling paradigms also model the behavioral/logical aspects besides the informational aspects. The exceptions are state space, conceptual graph, and semantic network. In the implementations, usually the informational aspects are supported by persistence technologies and the behavior aspects are supported by applications. In the modeling of ontological views the behaviors of concepts are not required.

The degree of structured of a modeling paradigm means how different elements are represented separately so each one of them can be differentiated from others and treated individually. The procedural representations embed model of the world within programs so it is hard to extract the individual elements. Similarly, the rule-based methods and propositional logic do not define internal structures for the sentences. First order logic is more structured in a sense that atomic formulas are interpreted as statements about relationships between objects. Other modeling paradigms are quite structured since they provide separate structures for different types of elements.

Model Implication means the degree that the model requires interpretation for humans' understanding. A well structured paradigm is usually explicit in terms of the meaning of the internal constructs, which makes the models easier to understand. An exception is the state space which can be highly structured but how each state represents the world requires lots of interpretation.

Some paradigms do not have general-purposed supporting technologies for model persistence and reuse by applications, therefore they are not considered in the validation.

The following table presents a summary of the features of various modeling paradigms. It shows that frame provides the most complete features for our modeling purpose.

Table 1. Comparison of various modeling paradigms.

Features / **Modeling paradigms**	**Has Notion of Concept**	**Differentiate Concepts and Instances**	**Has Notion of Relationship**	**Has Notion of Property**	**Has Notion of Property Characteristics**	**Has Notion of Behavior / Logic**	**Structured**	**Model Implication**	**General-purpose Supporting Technology**
State space	No	No	No	No	No	No	High	High	No (interpreted by applications)
Procedural Representation	No	No	No	No	No	Yes	Low	High	No (interpreted by applications)
Rule-based methods	No	No	No	No	No	Yes	Low	High	No (implemented by specialized systems)
Propositional logic	No	No	No	No	No	Yes	Low	Medium	No (implemented by specialized systems)
First order logic	Yes	No	Yes	No	No	Yes	Medium	Medium	Prolog
Description Logic	Yes	Yes	Yes	No	No	Yes	High	Low	OWL
Production rules	Yes	No	Yes	Yes	No	Yes	High	Low	Prolog
Conceptual Graph	Yes	Yes	Yes	No	No	No	High	Low	No (implemented by specialized systems)
Semantic Network	Yes	Yes	Yes	No	No	No	High	Low	No (implemented by specialized systems)
F-Logic	Yes	Yes	Yes	No	No	Yes	High	Low	No (implemented by specialized systems)
Entity-Relationship Model	Yes	Yes	Yes	Yes	Yes (not complete)	No	High	Low	Relational database

Object-Oriented	Yes	Yes	Yes	Yes	Yes (not complete)	Yes	High	Low	Object-Oriented languages
Frame	Yes	Yes	Yes	Yes	Yes	No	High	Low	Not required

According to the definition of ontological views, a complete specification language should provide constructs to denote concepts, properties of concepts, characteristics of properties, and relationships to specify the objects to be modeled. We examined two languages that are practically used in specifying information models since our work is based on the existing information systems: relational (implemented by SQL) and XML schema. They are well supported by mature persistence technologies.

The following table presents the comparison between FOSL, SQL, and XML schema elements. It shows that FOSL has the complete set of constructs for modeling the expected elements and all the constructs can be mapped to the counterparts within SQL and XML schema.

Table 2. Comparison of FOSL, SQL, and XML

Representation Language / **Language Construct** / **Modeling Object**	**FOSL**	**SQL**	**XML Schema**
World	Ontological_View	database	schema
Concept	Concept	table	element
Property	Property	column	attribute
Relationship	Relationship	foreign key	embedded element (complexType)

Property/Relationship Characteristics		Facet	column attribute	element attribute
Characteristics	Name	Name	column name	*name* attribute
	Identity	(Not necessary)	primary key, unique key	*key* element, *unique* element
	Auto-Increment	Auto_Increment	auto_increment/identity	
	Data type	Data_Type	type	*type* attribute
	Default value	Default_Value	default	*default* attribute
	Fixed value	Fixed_Value		*fixed* attribute
	Optional	Nullable	null/not null	*use* attribute
	Restriction on values		check	*restriction* element, *minInclusive* elemnt, *maxInclusive* element
	Restriction on a set of values		check	*restriction* element, enumeration element
	Restriction on a series of values		check	*restriction* element, *pattern* element
	Restriction on string length	Size	column length	*restriction* elemment, *length* element, *minLength* element, *manLength* elemnt
	Restriction on data types	Decimal_Size	column length	*restriction* element, *fractionDigits* elemnt, *totalDigits* element
	Relationship cardinality	Cardinality	(Implicit by model)	*maxOccur* attribute, *minOccur* attribute

5.2.1.2 Richness of the Frame-based Ontological View Specification Language

The *frame* paradigm focuses on modeling concepts, properties, relationships, and characteristics of properties. The FOSL language provides corresponding vocabulary and constructs to explicitly and directly represent each of them, providing sufficient abstraction:

- The symbol "Concept" identifies a concept with a unique identity.

- The symbol "Property" identifies a property with a unique identity for a concept.
- The construct of a concept associating a set of properties represents the relationship "has-property".
- The symbol "Facet" identifies a unique facet of a property.
- The construct of a property associating a set of facets represents the relationship "has-facet".
- The symbol "Value" identifies a value of a facet.
- The construct of a facet associating a value represents the relationship "has-value".
- The symbol "Relationship" identifies a *n*-ary relationship between concepts.
- The symbol "IS-A" indicates a generalization/specialization relationship between two concepts.
- The symbol "PART-OF" indicates a whole-part relationship between two concepts.

The following segment shows a partial specification of a concept:

```
Concept: product
    Property: id
        Facet: DATA-TYPE    Value: INT UNSIGNED
        Facet: SIZE    Value: 10
    Property: name
        Facet: DATA-TYPE    Value: VARCHAR
        Facet: SIZE    Value: 100
    Relationship:
        IS-A: sellable-item
```

Note that the language itself only provides sufficient way to specify the abstraction of a world. The world can be abstracted with different granularity and this depends on the purpose and capability of the modeler. The responsibility of the language is to specify any abstraction but not to guarantee a proper modeling granularity.

5.2.2 **Empirical Validation**

5.2.2.1 Completeness of Ontological View Creation

A component within a semantic integration service is responsible for creating an ontological view from an information model. The approach adopted by the component uses the following heuristic rules to analyze the information model:

- A relational table that has a primary key and extra columns is identified as a concept. Each column is identified as a property of the concept.

- Using MySQL, 5 types of facets can be identified: data type, size, decimal digits, nullable, auto-increment.

- A foreign key within a table (the subject concept) referring to another table (the object concept) is identified as a relationship between the two concepts. The relationship is simply named as "has" since there is a lack of explicit semantics of the foreign keys in a relational database.

- A table which columns are all members of a foreign key is identified as a relationship between the two concepts represented by the two referred tables. The relationship is named following the table name due to the lack of explicit semantics.

- An element in an XML document is identified as a concept. Attributes of an XML element are identified as properties of the concept. This limitation is based on an assumption of the acceptable structure of the XML document.

- The facets of the properties depend on the availability of the schema of the XML document. Without the schema, it is infeasible to extract the facets from an XML document due to the lack of information.
- The embedding of one XML element within another element is identified as a relationship between the concepts represented by the two elements.

This approach is applied upon several information models deployed in the collaborative promotion prototype environment. Among them, some systems adopt the relational model (using MySQL DBMS) and one system adopts XML-based model. These systems are described as follows:

(1) Business Model Management System. This system manages business model of the QSR domain. The business model contains essential business concepts and business rules for this domain, such as products, languages, prices, times, resources, time rules, resource rules, etc. This system adopts the relational database as the persistence technology.

(2) Media Management System. This system manages the information about media assets that can be displayed to achieve the promotion purposes. The media assets have a set of properties and are described by some keywords. The media assets are digital files; therefore the properties about the physical files are also managed. This system adopts the relational database as the persistence technology.

(3) Promotion Management System. This system manages the promotions. It works as a consumer of the business model management system. A promotion specifies what product to promote, when to promote, and where to promote. This system adopts the relational database as the persistence technology.

(4) Information Model of Intelligent Media. This system manages informational level model of the multimedia assets that can be displayed to achieve promotion purposes.

The information level model encapsulates the low level visual features of the multimedia content. This system adopts the relational database as the persistence technology. Note that this system does not involve in the semantic integration. Instead, it uses other systems' semantic integration services to achieve integration.

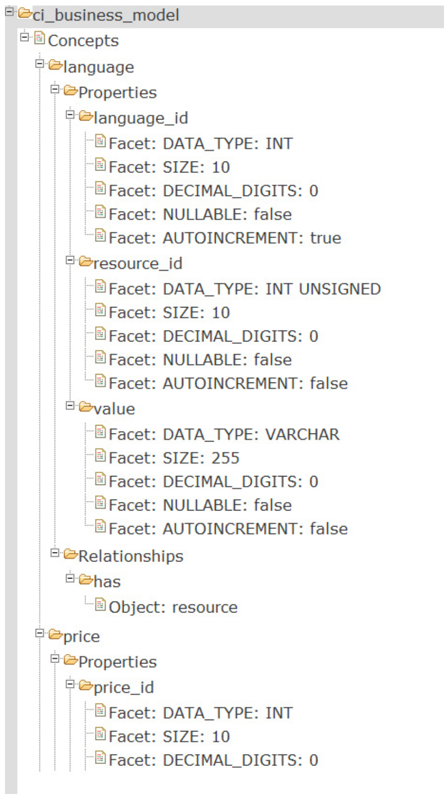
(5) Inventory Management System. Product is a major concept that this domain concerns. The inventory of products is one of the most essential aspects to be managed for the business. This system manages the inventory of products, inventory locations and the stocking history. This system adopts the relational database as the persistence technology.

(6) Transaction Management System. Sales transactions of products can serve to show how the products are sold during specific time, which can further serve as indications of what products to promote more or less. This system manages sales-related information including products, POS machines, operators, detailed transactions and receipts. This system adopts the relational database as the persistence technology.

(7) Scheduling Management System. This system manages the promotion schedules in terms of what product to promote, which media to be used for the promotion, what time to display the media, and on which resources to display the media. This system adopts XML as the persistence technology.

Details of these systems' information models are provided in Appendix A.

The following figure shows a segment of the created ontological view of the business model management system. It extracts the elements that represent concepts as well as the relationships from the underlying information model.



```
ci_business_model
  Concepts
    language
      Properties
        language_id
          Facet: DATA_TYPE: INT
          Facet: SIZE: 10
          Facet: DECIMAL_DIGITS: 0
          Facet: NULLABLE: false
          Facet: AUTOINCREMENT: true
        resource_id
          Facet: DATA_TYPE: INT UNSIGNED
          Facet: SIZE: 10
          Facet: DECIMAL_DIGITS: 0
          Facet: NULLABLE: false
          Facet: AUTOINCREMENT: false
        value
          Facet: DATA_TYPE: VARCHAR
          Facet: SIZE: 255
          Facet: DECIMAL_DIGITS: 0
          Facet: NULLABLE: false
          Facet: AUTOINCREMENT: false
      Relationships
        has
          Object: resource
    price
      Properties
        price_id
          Facet: DATA_TYPE: INT
          Facet: SIZE: 10
          Facet: DECIMAL_DIGITS: 0
```

Figure 5-36. A segment of an ontological view.

The created ontological view is persisted in an XML document for further usage. The following is a segment of the document for the business model management system:

```
<concepts>
  <concept name="language">
    <properties>
      <property name="language_id">
        <facet name="DATA_TYPE" value="INT"/>
        <facet name="SIZE" value="10"/>
        <facet name="DECIMAL_DIGITS" value="0"/>
        <facet name="NULLABLE" value="false"/>
        <facet name="AUTOINCREMENT" value="true"/>
      </property>
      <property name="resource_id">
```

```
            <facet name="DATA_TYPE" value="INT UNSIGNED"/>
            <facet name="SIZE" value="10"/>
            <facet name="DECIMAL_DIGITS" value="0"/>
            <facet name="NULLABLE" value="false"/>
            <facet name="AUTOINCREMENT" value="false"/>
        </property>
        <property name="value">
            <facet name="DATA_TYPE" value="VARCHAR"/>
            <facet name="SIZE" value="255"/>
            <facet name="DECIMAL_DIGITS" value="0"/>
            <facet name="NULLABLE" value="false"/>
            <facet name="AUTOINCREMENT" value="false"/>
        </property>
    </properties>
    <relationships>
        <relationship name="has">
            <object_concept name="resource"/>
        </relationship>
    </relationships>
</concept>
```

The results show that the created ontological views correctly reflect the model based on the design of the original relational database or the XML document. This provides well-founded support for the semantic equivalence relationship discovery in a later stage.

5.2.2.2 Completeness of Semantic Equivalence Relationship Discovery

Adopting a benchmark is helpful for validation from an empirical perspective. Information integration, as an application of semantic integration, has been an active area of research since the early 80's and has produced a rich collection of techniques and approaches to integrate heterogeneous information. As a result, determining the quality and applicability of a solution is a difficult task. It has been the focus of several studies (e.g. [Do, et al., 2002]). The lack of available test data and benchmark makes such validation more challenging.

THALIA[18] is the first publicly available testbed and benchmark of integration technologies allowing the objective comparison of integration solutions [Hammer, et al., 2005]. It provides a collection of over 25 data sources representing university course catalogs from computer science departments around the world. THALIA provides a set of benchmark queries as well as a scoring function for ranking the performance of an integration system. It focuses on syntactic and semantic heterogeneities.

The following table shows a sample course catalog from the CS department at Brown University providing information such as course number, instructor, title, time and location in a tabular format.

Course	**Instructor**	**Title/Time**	**Room**
CS002	Stanford	Concepts & Challenges of CS C hr. MWF 10-11	Salomon 001
CS004	Usas	Intro to Scientific Computing K hr. T,Th 2:30-4	MacMillan 117
CS016	Tamassia	Intro to Algorithms & Data Structures D hr. MWF 11-12	CIT Lubrano
CS018	Klein	CS: An Integrated Approach J hr. T,Th 1-2:30	CIT 227
CS022	Lysyanskaya	Intro. to Discrete Mathematics B hr. MWF 9-10	CIT 165
CS032	Reiss	Intro. to Software Engineering K hr. T,Th 2:30-4	CIT 165, Labs in Sunlab

One benchmark query is to find synonyms: attributes with different names that convey the same meaning, for example, “instructor” vs. “lecturer”.

THALIA provides a useful guide for empirically validating our solution. However, it cannot be directly adopted for the following reasons:

- It collects data from the web pages of the universities. The course catalog

[18] http://www.cise.ufl.edu/research/dbintegrate/thalia/

information in the pages is quite un-structured (some is even in plain text) or not well-structured (like Brown's case where two information items are combined into one column). This leads to a different context with our research, where we deal with well-structured information models.

- The benchmark queries are heavily relying on the application background. For example, one query asks to list all *database courses* that *carry* more than *10 credit hours*. Our work is not limited to any specific application or specific domain.

- The source data is collected and output as XML. The XQuery technique is used to conduct XML-based queries. Our work does not rely on any specific technique.

- It requires the integration solution to have the capability of processing queries. In our solution the semantic integration only deals with semantic heterogeneities. Query processing is the capability of the original information systems.

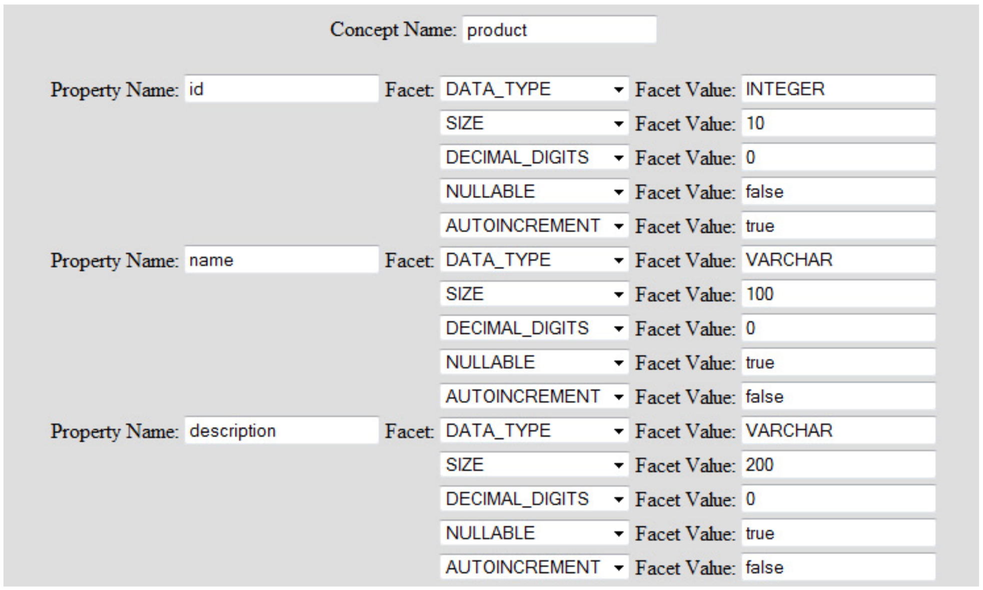
In the work we adopt some ideas of THALIA to design the criteria to validate the solutions. We focus on the query that checks if the key heterogeneities that exist in the underlying information models are well addressed:

- Synonyms: different names that convey the same meaning. It includes two aspects:
    - Concept: different symbols, used as concept identifiers, refer to the same concept.
    - Property: different symbols, used as property identifiers, refer to the same property.

In the environment, the concept of "product" is a core concept for the QSR domain and it is modeled commonly in the business model management system, media management system, promotion management system, inventory management system, transaction management system and the scheduling management system in different ways. The concept of "resource" (digital device to play multimedia contents) and "time" (indicating when to play the contents) are modeled in the business model management system, promotion management system and scheduling management system in different ways. Our intention is to find out all such relationships.

The approaches developed for discovering semantic equivalence relationships are applied in two manners:

(1) A Web-based management console for the integration system provides one page to allow a human user to define a concept to be processed. In the page the user enters the name of the concept, properties of the concept, and a set of facets for each property in the form of name/value pair. These create a frame structure which is actually a representation for the concept from a conceptualization. The following figure shows the information that a user enters:



Concept Name: product

Property Name:	id	Facet:	DATA_TYPE	Facet Value:	INTEGER
			SIZE	Facet Value:	10
			DECIMAL_DIGITS	Facet Value:	0
			NULLABLE	Facet Value:	false
			AUTOINCREMENT	Facet Value:	true
Property Name:	name	Facet:	DATA_TYPE	Facet Value:	VARCHAR
			SIZE	Facet Value:	100
			DECIMAL_DIGITS	Facet Value:	0
			NULLABLE	Facet Value:	true
			AUTOINCREMENT	Facet Value:	false
Property Name:	description	Facet:	DATA_TYPE	Facet Value:	VARCHAR
			SIZE	Facet Value:	200
			DECIMAL_DIGITS	Facet Value:	0
			NULLABLE	Facet Value:	true
			AUTOINCREMENT	Facet Value:	false

Figure 5-37. The frame-based representation of a concept.

It is assumed that the intended concept is known by the information systems in the environment, but is modeled and represented in different ways. The heterogeneities need to be addressed at the ontological view level. Merely using the name of the concept it is not sufficient to identify which elements within the information models of the systems mean the same thing with the one defined from the management console. We apply the tree similarity-based approach to conduct the semantics-based search, considering the properties as well as their facets. The following table shows the transformation costs of the concept from the management console and the business model management system:

Concept	**Transformation Cost**
price	1.1420396187560367
product	0.28946908531683
product_feature	1.549417980921546
product_rule	0.3890182097794555
resource	1.998957541249725
resource_rule	0.47903603509139847
time	1.5462285233203237
time_rule	0.3710146447170669

The lowest transformation cost indicates that the concept from the management console matches best with the "product" concept in the business model management system.

The results on other systems show that synonyms in terms of concept identifiers and property identifiers can be successfully discovered. The scheduling management system is an exception since its ontological view does not contain sufficient information (no facets for the properties are available).

(2) While the instance data, e.g., the data in the relational tables, is available, we compare the representation of two concepts from two ontological views by applying the instance-based approach on their instance data. This approach does not assume any domain knowledge about the concept modeling. It examines different permutations of the properties of two concepts to make sure that every possible matching candidate is checked. The similarity degree varies for each permutation pair.

However, there must be one permutation pair reaching the lowest distance if the data sets of the properties appearing in the permutations have a very similar probability density. This identifies the similarity of the two concepts while other permutation pairs can be ignored.

The following example shows one concept named "product" from the business model management system and another concept named "products" from the promotion management system. It is possible that a linguistic-based approach discovers that "product" and "products" may be the same according to their spelling forms. The instance-based approach does not require any linguistic or domain-based knowledge.

product

Property	Name
property_id	LONG
property_name	STRING
flavor	STRING
sweetness	STRING
brand_name	

products

Property	Name
pr_id	LONG
pr_name	STRING
pr_description	STRING

The following tables show some property matching candidates:

Source Property Name	Source Property Data Type	Target Property Name	Target Property Data Type	KL Divergence
product_id	LONG	pr_id	LONG	0.0
product_name	STRING	pr_name	STRING	0.0
flavor	STRING	pr_description	STRING	0.003275840774546316 3
Similarity Index				0.0032758407745463163
Source Property Name	**Source Property Data Type**	**Target Property Name**	**Target Property Data Type**	**KL Divergence**
product_id	LONG	pr_id	LONG	0.0
flavor	STRING	pr_name	STRING	0.00319282355720962
product_name	STRING	pr_description	STRING	4.145749971432248E-5

Similarity Index				0.004851421585385914

Source Property Name	**Source Property Data Type**	**Target Property Name**	**Target Property Data Type**	**KL Divergence**
product_id	LONG	pr_id	LONG	0.0
flavor	STRING	pr_name	STRING	0.00319282355720962
sweetness	STRING	pr_description	STRING	0.028018566333710967
Similarity Index				0.04681708483638088

These tables show that two pairs of properties, (product_id, pr_id) and (product_name, pr_name) can be well matched and the similarity index indicates that these two concepts are identical even though the property flavor and pr_description are, in fact, not the same.

The results on the systems show that synonyms can be successfully identified, i.e., the concepts of "product", "resource" and "time" modeled in the business model management system, promotion management system and scheduling management system can be discovered using the instance data.

5.2.2.3 Soundness of the Solution

The solution is applied to produce collaborative intelligence in an open environment. It is able to address the schematic and syntactic heterogeneities of the information models, and identify the same concept that is modeled and represented in different ways in different information systems.

This environment contains an extra intelligent multimedia system, besides the traditional information systems maintaining the operational data of the QSR business. The intelligent multimedia system is able to identify what an image represents, such as "Apple Fritter" using image processing technology and low-level feature matching. The intelligent multimedia system is closely integrated with a business model management system that maintains fundamental business concepts. The business model management system tells what this thing is, for example, a "product". Then, this concept is sent to various information systems to examine how it is modeled and represented using the semantic integration services deployed into each system.

The results are collected by the management console, which in turn contacts each system using the specific concepts that are managed by the business model system. Each system returns some information related to those concepts. On the management console side the media object is able to collect complete information which is converted into a kind of intelligence about the specific product. Such intelligence is utilized in the later stage to decide what product to promote and where to display a multimedia asset to realize the promotion.

The following figure 5-5 shows the overall architecture of the collaborative intelligence system:

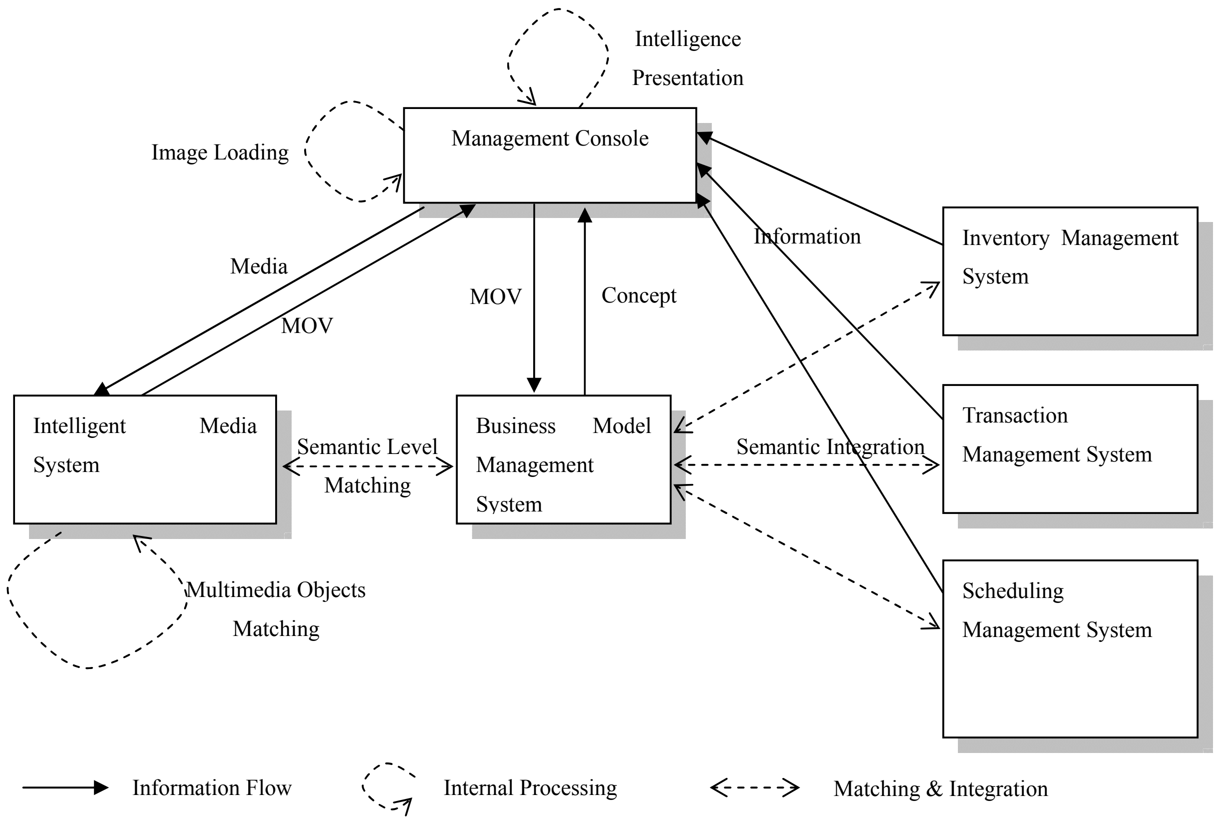


Figure 5-38. Architecture of a collaborative intelligence system.

In the figure MOV means “Multimedia Ontological View”. It is an ontological view containing the objects and relationships discovered from an image.

Semantic level matching is utilized to match concepts from the ontological view created from the intelligent media system and concepts from the ontological view created from the business model management system. Note that semantic level matching is logically conducted between the intelligent multimedia system and the business model management system, but there is no direct communication between these two systems. Instead, the matching is supervised by the management console, i.e., the management console gets the created multimedia ontological view, and sends it to the business model management system's semantic integration service.

The management console uses the concepts to perform semantic integration and gets to know how these concepts are modeled in other systems. Similarly, the semantic integration is logically between the business model management system and other information systems (inventory management system, transaction management system, scheduling management system, etc) but there is no direct communication between these systems. The management console supervises the integration, i.e., it sends the concepts returned from the business model management system to another system's semantic integration service. The service will discover if the same concepts are modeled within it and how the concepts are modeled.

Chapter 6 Conclusion and Future Work

Semantic integration, as an important factor for successful information integration, has grown into one of the most active research areas. Our work on semantic integration fits into its evolution by extending the traditional ontology-driven approaches to an ontological view-driven approach to overcome the grand challenges that were not thoroughly addressed by the traditional approaches. The most significant advancement is the removal of the assumption about the availability of explicit ontologies. With the concept of ontological view we provide a formal way to explicitly specify the concepts within a conceptualization with rich details based on various information models. This work establishes a solid foundation for semantic integration in an open environment.

The main contributions of this work are listed as follows.

(1) It conducts a thorough review on semantic integration-related topics and presents a full picture of the state-of-the-art of the research in this domain. It clarifies the meanings of some important terms including conceptualization, concept, model, representation, schema, semantics, ontology, ontological view, ontological integration, semantic heterogeneity, information integration, semantic integration, ontology-driven semantic integration and ontological view-driven integration. It examines the semantics of information from the structural and intensional perspective and discusses how to discover the semantics. This work also proposes a classification of the views on semantic integration, including the structural view at elemental data level and structure level and semantic view at data level, concept level and knowledge level. Several architectures of semantic integration at the application level are discussed.

(2) The schema-based structural approaches and ontology-driven semantic

approaches regarding information integration are analyzed in the review. Discussions on their advantages and limitations are presented. With structural approaches, the information schemas are available and it can be discovered that two or more schema elements have the same meaning and they can match. However, there is no clue about what concept they refer to due to the lack of a concept model. In semantic approaches, the semantics is explicitly specified by establishing concept models such as ontologies, and the focus is that two or more ontology elements refer to the same concept if they can be discovered to be semantically identical. However, the application of these approaches is limited since in many domains there are no explicit ontologies available.

(3) It provides the formal definition for domain, possible world, domain space, conceptual relation, conceptualization, intended structure, ontological commitment of logical language, compatible model of language, intended model and ontology based on Guarino's work [Guarino, 1998]. Then, it analyzes that there is no a unique explicit "ontology" for a conceptualization. Instead, different views of the conceptualization may exist. Thus, the notion of ontology is extended to the notion of ontological view. This notion is used to facilitate the semantic integration where no "ontology" is available. It also defines the ontological equivalence mapping and the semantically equivalent relationship. It proves that a concept in a conceptualization can be externalized by a constant symbol in a language under an ontological commitment, and the semantically equivalent relationship between symbols under an ontological commitment implies the same concept reference. This becomes the foundation of the following semantic relationship discovery algorithms.

(4) It proposes a novel architecture of semantic integration enabled environment that extends the traditional data/information architecture to a three layered architecture including the data management and integration layer, the

information management and integration layer, and the semantics management and integration layer. In the architecture, a semantic integration service is attached to each information system, which converts a traditional information system into a semantics enhanced system. The architecture for the semantic integration service inspired by Act* is proposed.

(5) It adopts *frame* as the modeling paradigm of the ontological view. An ontological view can be created from the information model of an information system. In an open environment the frame-based ontological views create a common level that eliminates the structural and syntactic heterogeneities among the information models. With this commonness only semantic heterogeneities should be considered in the semantic integration. It proposes a frame-based ontological view specification language (FOSL) and uses XML to explicitly encode the ontological views.

(6) It proposes a tree similarity-based approach and an instance-based approach to compute the semantic similarities between concepts represented in different ontological views adopting the frame's tree-like structure or available data instances. Such similarities can be used to discover the semantic equivalence relationships between concepts.

(7) It implements the proposed solutions in a collaborative intelligence prototype environment. Several aspects of the solutions, including the completeness and richness of FOSL, completeness of the ontological view creation approach, completeness of the semantic equivalence relationship discovery approach, and soundness of the solution are validated from the analytical and empirical perspectives.

Our future work will focus on several aspects:

(1) Improving the automatic ontological view creation based on regular information models, providing visual editing of ontological views, and providing efficient model

validation to ensure the consistency of ontological views. Based on these efforts the semantic integration service layer can keep being improved.

(2) Extending the ontological view's tree structure to a graph, with further attention to the relationships between concepts. New definitions for graph transformation operation and transformation cost are to be explored. Meanwhile, more types of relationships among concepts have to be considered, which require further consideration on the semantics of the relationships.

(3) Applying and evaluating other approaches for density estimation, probability density comparison, and clustering as well as richer collection of linkage functions and distance metrics. A more sophisticated evaluation engine combining multiple approaches will also be investigated to improve the discovered results in terms of the semantic equivalence relationships between concepts within different ontological views. Furthermore, semantic relationship types other than the equivalence relationships, such as generalization or specialization, will also be taken into consideration to enhance the capability of the semantic integration service.

References

[Aho, et al., 1989] Aho, A. V., Ganapathi, M., and Tjiang, S. W. K.. Code Generation Using Tree Matching and Dynamic Programming. ACM Transactions on Programming Languages and Systems, Vol. 11, Issue 4, pp. 491-516, October 1989.

[AIAA, 1998] American Institute of Aeronautics and Astronautics (AIAA). Guide for the Verification and Validation of Computational Fluid Dynamics Simulations. AIAA-G-077-1998, Reston, VA, 1998.

[Aikins, 1993] Aikins, J. S.. Prototypical Knowledge for Expert Systems: a Retrospective Analysis. Artificial Intelligence, Vol. 59, pp.207-211, Elsevier, 1993.

[Allali and Sagot, 2005] Allali, J. and Sagot, M.. Novel Tree Edit Operations for RNA Secondary Structure Comparison. In Proceedings of IWABI 2004, pp. 412-425, Bergen, Norway, September 17-21, 2004.

[Ambite and Knoblock, 2000] Ambite, J. L. and Knoblock, C. A.. Flexible and Scalable Cost-Based Query Planning in Mediators: A Transformational Approach. Artificial Intelligence Journal, Vol. 118, pp. 1-2, 2000.

[Anderson, 1983] Anderson, J. R.. The Architecture of Cognition. Cambridge, MA: Harvard University Press, 1983.

[Arens, et al., 1993] Arens, Y., Chee, C. Y., Hsu, C. –N., and Knoblock, C. A.. Retrieving and Integrating Data from Multiple Information Sources. International Journal of Intelligent and Cooperative Information Systems, 2(2): 127-158, 1993.

[Arens, et al., 1996] Arens, Y., Knoblock, C. A., and Hsu, C. –N.. Query Processing in the SIMS Information Mediator. In A. Tate (ed), Advanced Planning Technology. Menlo Park: AAAI Press, 1996.

[Arroyo, 2007] Arroyo, S.. Ontology and Grammar of the SOPHIE Choreography Conceptual Framework – An Ontological Model for Knowledge Management. Journal of Universal Computer Science, Vol. 13, No. 9, pp.1157-1183, 2007.

[Baader, et al., 2002] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2002.

[Badder and Sattler, 2001] Badder, F. and Sattler, U.. An Overview of Tableau Algorithms for Description Logics. Studia Logica, Vol. 69, No. 1, pp. 5-40, 2001.

[Barr and Feigenbaum, 1981] Barr, A. and Feigenbaum, E. A.. The Handbook of Artificial Intelligence, Volume I. Heuristech Press, Stanford, California, USA and William Kaufmann, Inc., Los Altos, California, 1981.

[Batini, et al., 1986] Batini, C., Lenzerini, M., and Navathe, S. B.. A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, 18: 323-364, 1986.

[Bean and Tsokos, 1980] Bean, S. J. and Tsokos, C. P.. Developments in Nonparametric Density Estimation. International Statistical Review, 48, pp. 267-287, 1980.

[Beard and Smith, 1998] Beard, K. and Smith, T.. A Framework for Meta-Information in Digital Libraries. In A. Sheth, W. Klas (eds), Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media, McGraw Hill, pp. 341-365, 1998.

[Bell and Sethi, 2001] Bell, G. S. and Sethi, A.. Matching Records in a National Medical Patient Index. CACM 44(9): 83-88, 2001.

[Berlin and Motro, 2001] Berlin, J. and Motro, M.. Autoplex: Automated Discovery of Content of Virtual Databases. In Proceedings of 9th International Conference on Cooperative Information Systems (CoopIS), Lecture Notes in Computer Science, vol. 2172. Springer, Berlin Heidelberg New York, pp. 108-122, 2001.

[Berners-Lee, et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O.. The Semantic Web. Scientific

American, 2001(5), 2001.

[Bille, 2003] Bille, P.. Tree Edit Distance, Alignment Distance and Inclusion. Technical Report Series TR-2003-23, ISSN 1660-6100, IT University of Copenhagen, March 2003.

[Bishr, et al., 1999] Bishr, Y. A., Pundt, H., Kuhn, W., and Radwan, M.. Probing the Concept of Information Communities - A First Step toward Semantic Interoperability. In M. Goodchild, Max Egenhofer, R. Fegeas, and C. Kottman, editors, Interoperating Geographic Information Systems, pp. 55-69. Kluwer Academic, 1999.

[Boran, et al., 2007] Boran, A., O'Sullivan, D., and Wade, V. P.. A Case Study of an Ontology-Driven Dynamic Data Integration in a Telecommunications Supply Chain. In Proceedings of the Workshop on First Industrial Results of Semantic Technologies, co-located with ISWC 2007 + ASWC 2007, pp. 1-13, Busan, Korea, November 11th, 2007.

[Bornhövd, 1998] Bornhövd, C.. MIX - A Representation Model for the Integration of Web-based Data. Technical Report, No. DVS98-1, Computer Science Dept., Darmstadt University of Technology, November 1998.

[Bouquet, et al., 2003] Bouquet, P., Serafini, L., and Zanobini, S.. Semantic Coordination: A New Approach and an Application. In Proceedings of ISWC 2003, LNCS 2870, pp.130-145, 2003.

[Brachman and Levesque, 1984] Brachman, R. J. and Levesque, H. J.. The Tractability of Subsumption in Frame-based Description Languages. In Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84), pp. 34-37, 1984.

[Brachman and Levesque, 2004] Brachman, R. J. and Levesque, H. J., Knowledge Representation and Reasoning, Morgan Kaufmann publishers, 2004.

[Broekstra, et al., 2001] Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F., and Horrocks, I.. Enabling Knowledge Representation on the Web by Extending RDF Schema. In Proceedings of the 10th World Wide Web Conference, pp. 467-478, Hong Kong, China, May 1-5, 2001.

[Bruno, et al., 2002] Bruno, N., Srivastava, D., and Koudas, N.. Holistic Twig Joins: Optimal XML Pattern Matching. In Proceedings of the 2002 ACM SIGMOD, pp. 310-321, Madison, Wisconsin, USA, June 4-6, 2002.

[Calvanese, et al., 1998(1)] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., and Rosati, R.. Knowledge Representation Approach to Information Integration. In Proceedings of AAAI Workshop on AI and Information Integration, pp. 58-65. AAAI Press / The MIT Press, 1998.

[Calvanese, et al., 1998(2)] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., and Rosati, R.. Description Logic Framework for Information Integration. In Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, pp. 2-13, 1998.

[Calvanese, et al., 2002] Calvanese, D., Giacomo, D. G., and Lenzerini, M.. A Framework for Ontology Integration, In Proceedings Of the 1st Semantic Web Working Symposium at the Emerging Semantic Web, pp. 201-214, 2002.

[Castano, et al., 2001] Castano, S., De Antonellis, V., and De Capitani di Vemercati, S.. Global Viewing of Heterogeneous Data Sources. IEEE Transactions on Data Knowledge Engineering, 13(2): 277-297, 2001.

[Chalupsky, 2000] Chalupsky, H.. OntoMorph: A Translation System for Symbolic Logic. In A. G. Cohn, F. Giunchiglia, and B. Selman (eds), KR2000: Principles of Knowledge Representation and Reasoning, pp. 471-482, San Francisco, CA, 2000.

[Chandrasekaran, et al., 1999] Chandrasekaran, B., Josephson, J. R., and Richard Benjamins, V.. Ontologies: What are they? Why do we need them? IEEE Expert (Intelligent Systems and Their Applications), 14(1): 20-26, 1999.

[Chawathe, et al., 1994] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J.. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In Proceedings of the 10th Meeting of the Information Processing Society of Japan, pp. 7-18, Tokyo, Japan, October 1994.

[Codd, 1990] Codd, E. F.. The Relational Model for Database Management: Version 2. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1990.

[Collet, et al., 1991] Collet, C., Huhns, M. N., and Shen, W.. Resource Integration Using A Large Knowledge Base in Carnot. IEEE Computer, 24(12): 55-62, 1991.

[Crubzy, et al., 2003] Crubzy, M., Pincus, Z., and Musen, M. A.. Mediating Knowledge between Application Components. In Proceedings of the Semantic Integration Workshop of the Second International Semantic Web Conference (ISWC-03), Sanibel Island, Florida, 2003.

[De Bruijn, et al., 2003] De Bruijn, J., Ding, Y., Arroyo, S., and Fensel, D.. Semantic Information Integration in the COG Project. Technical Report, Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria, 2003.

[Dempsey and Heery, 1997] Dempsey, L. and Heery, R.. Specification for Resource Description Methods Part 1: A Review of Metadata: A Survey of Current Resource Description Formats. Work Package 3 of Telematics for Research Project DESIRE (RE 1004), March, 1997.

[Dey, et al., 2001] Dey, A., Abowd, G., and Salber, D.. A Conceptual Framework and A Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. Human-Computer Interaction, 16: 97-166, 2001.

[Dhamankar, et al., 2004] Dhamankar, R., Lee, Y., Doan, A. H., Halevy, A., and Domingos, P.. iMAP: Discovering Complex Semantic Matches between Database Schemas. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD'04), pp. 383-394, New York, USA, ACM Press, 2004.

[Do, et al., 2002] Do, H. -H., Melnik, S., and Rahm, E.. Comparison of Schema Matching Evaluations. In Proceedings of the 2nd International GI-Workshop on Web and Databases, pp. 221-237, Erfurt, Germany, 2002.

[Doan and McCann, 2003] Doan, A. and McCann, R.. Building Data Integration Systems: A Mass Collaboration Approach. In Proceedings of the IJCAI-03 Workshop on Information Integration on the Web, 2003.

[Doan, et al., 2000] Doan, A. H., Domingos, P., and Levy, A.. Learning Source Descriptions for Data Integration. In Proceedings of WebDB Workshop. pp. 81-92, 2000.

[Doan, et al., 2001] Doan, A. H., Domingos, P., and Halevy, A.. Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In Proceedings of ACM SIGMOD Conference, pp. 509-520, 2001.

[Doan, et al., 2002] Doan, A., Madhavan, J., Domingos, P., and Halevy, A.. Learning to Map between Ontologies on the Semantic Web. In Proceedings of the 11th International Conference on World Wide Web, pp. 662-673, 2002.

[Doan, et al., 2003] Doan, A., Lu, Y., Lee, Y., and Han, J.. Object Matching for Data Integration: A Profile-Based Approach. In Proceedings of the IJCAI-03 Workshop on Information Integration on the Web, 2003.

[Drezner, et al., 1986] Drezner, Z., Thisse, J., and Wesolowsky, G. O.. The Minimaxmin Location Problem. Journal of Regional Science, 26, pp. 87-101, 1986.

[Farquhua, et al., 1995] Farquhua, A., Fikes, R., Pratt, W., and Rice, J.. Collaborative Ontology Construction for Information Integration. Technical Report KSL-95-63, Knowledge Systems Laboratory, Stanford University, 1995.

[Farquhua, et al., 1997] Farquhua, A., Fikes, R., and Rice, J.. The Ontolingua Server: A Tool for Collaborative Ontology Construction. International Journal of Human-Computer Studies, 46(6): 707-727, 1997.

[Fellbaum, 1998] Fellbaum, C., editor. WordNet: an Electronic Lexical Database. Language, Speech, and Communication Series. MIT Press, Cambridge, MA, 1998.

[Fensel and Brodie, 2004] Fensel, D. and Brodie, M.. Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin, 2nd edition, 2004.

[Fensel, et al., 1998] Fensel, D., Decker, S., Erdmann, M., and Studer, R.. Ontobroker: The Very High Idea. In 11th International Flairs Conference (FLAIRS-98), pp. 131-135, Sanibal Island, USA, 1998.

[Fernandez, et al., 1997] Fernandez, M., Gomez-Perez, A., and Juristo, N.. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In Proceedings of AAAI97 Spring Symposium, Workshop on Ontological Engineering, pp. 33-40, 1997.

[Fernandez, et al., 1999] Fernandez, M., Gomez-Perez, A., Sierra, A. P., and Sierra, J. P.. Building a Chemical Ontology Using METHONTOLOG and the Ontology Design Environment. IEEE Intelligent Systems, 14(1), 37-46, 1999.

[Fisseha, 2003] Fisseha, F.. The Basics of Ontologies. Nordic Agricultural Ontology Service (AOS) Workshop, February 28, 2003.

[Gangemi, et al., 1998] Gangemi, A., Pisanelli, D., and Steve, G.. Ontology Integration: Experiences with Medical Terminologies. In N. Guraino (ed), Formal Ontology in Information Systems, pp. 163-178. IOS Press, 1998.

[Ganter and Wille, 1999] Ganter, B. and Wille, R.. Formal Concept Analysis – Mathematical Foundations. Springer, 1999.

[Garcia-Molina, et al., 1995] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., and Widon, J.. The TSIMMIS approach to Mediation: Data Models and Languages. In Next Generation Information Technologies and Systems (NGITS-95), Naharia, Israel, 1995.

[Genesereth, et al., 1997] Genesereth, M. R., Keller, A. M., and Duschka, O. M.. Infomaster: An Information Integration System. In Proceedings of 1997 ACM SIGMOD International Conference on Management of Data, pp. 539-542, Tucson, Arizona, May 1997.

[Giunchiglia and Yatskvich, 2004]Giunchiglia, F. and Yatskevich, M.. Semantic Matching. Knowledge Engineering Review, 18(3): 265-280, 2004.

[Goh, 1997] Goh, C. H.. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources. PhD Thesis, MIT, 1997.

[Goh, et al., 1994] Goh, C., Madnick, S., and Siegel, M.. Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment. In Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94), pp. 337-346, Gaithersbury, 1994.

[Gruber, 1993] Gruber, T. R.. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2): 199-220, 1993.

[Gruber, 1995] Gruber, T. R.. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies, 43(5/6): 907-928, 1995.

[Gruninger, 1996] Gruninger, M.. Designing and Evaluating Generic Ontologies. In ECAI96's Workshop on Ontological Engineering, 1996.

[Guarino, 1998] Guarino, N.. Formal Ontology and Information Systems. In N. Guarino, editor, Formal Ontology in Information Systems, In Proceedings of FOIS'98, pp. 3-17, Trento, Italy, June 1998. IOS Press, Amsterdam.

[Guarino, et al., 1999] Guarino, N., Masolo, C., and Vetere, G.. Ontoseek: Content-based Access to the Web. IEEE Intelligent Systems, 14(3): 70-80, 1999.

[Guda, et al., 2002] Guda, S., Jagadish, H. V., Koudas, N., Srivastava, D., and Yu, T.. Approximate XML Joins. In Proceedings of the 2002 ACM SIGMID, pp. 287-298, Madison, Wisconsin, USA, June 4-6, 2002.

[Guegan and Hernandez, 2006] Guegan, M. and Hernandez, N.. Recognizing Textual Parallelisms with Edit Distance and Similarity Degree. In Proceedings of EACL 2006, The Association for Computer Linguistics, Trento, Italy, April 3-7, 2006.

[Gunther and Voisard, 1998] Gunther, O. and Voisard, A.. Metadata in Geographic and Environmental Data Management. In A. Sheth, W. Klas (eds), Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media, McGraw Hill, pp. 57-87, 1998.

[Haas, et al., 1999] Haas, L. M., Miller, R. J., Niswonger, B., Tork Roth, M., Schwarz, P. M., and Wimmers, E. L.. Transformation Heterogeneous Data with Database Middleware: Beyond Integration. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 22(1): 31-36, 1999.

[Hai, 2005] Hai, D. H.. Schema Matching and Mapping-based Data Integration. Ph.D. Thesis. University of Leipzig, German, 2005.

[Hakimpour and Geppert, 2002] Hakimpour, F. and Geppert, A.. Global Schema Generation Using Formal Ontologies. S. Spaccapietra, S. T. March, and Y. Kambayashi (eds): ER 2002, LBCS 2503, pp. 307-321, Springer-Verlag Berlin Heidelberg, 2002.

[Hakimpour and Timpf, 2001] Hakimpour, F. and Timpf, S.. Using Ontologies for Resolution of Semantic Heterogeneity in GIS. Proceedings 4th AGILE Conference on Geographic Information Science, pp.385-395, Brno, Czech Republic, 2001.

[Hamill, et al., 1997] Hamill, S., Dixon, M., and Read, B. J.. Classifying Schematic and Semantic Heterogeneities in Interoperating Database Systems. RAL Report RAL-97-xxx, 1997.

[Hammer, et al., 1997] Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Breunig, M., and Vassalos, V.. Template-based Wrappers in the TSIMMIS System. SIGMOD Record, 26(2): 532-535, 1997.

[Hammer, et al., 2005] Hammer, J., Stonebraker, M., and Topsakal, O.. THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches. In Proceedings of the 21st International Conference on Data Engineering (ICDE2005), pp. 485-486, Tokyo, Japan, April 2005.

[Han and Kamber, 2000] Han, J. and Kamber, M.. Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor. Morgan Kaufmann Publishers, August 2000.

[Heflin, et al., 1999] Heflin, J., Hendler, J., and Luke, S.. SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078, Institute for Advanced Computer Studies, University of Maryland, 1999.

[Hess and Iochpe, 2004] Hess, G. N. and Iochpe, C.. Ontology-driven Resolution of Semantic Heterogeneities in GDB Conceptual Schemas. In Proceedings of the Brazilian Symposium on GeoInformatics, pp. 247-263, 2004.

[Hovy and Nirenbury, 1992] Hovy, E. H. and Nirenbury, S.. Approximating an Interlingua in a Principled Way. In Proceedings of the DARPA Speech and Natural Language Workshop, Arden House, NY, 1992.

[Hovy, 1998] Hovy, E.. Combining and Standardizing Large Scale, Practical Ontologies for Machine Translation and Other Uses. In First International Conference on Language Resources and Evaluation (LREC), pp. 535-542, 1998.

[Hovy, 2003] Hovy, E. H.. Using an Ontology to Simplify Data Access. Communications of the ACM, Special Issue on Digital Government. Vol. 46, pp. 47-49, 2003.

[Hu, et al., 2007] Hu, C., Zhang, X., Zhao, Q., and Zhao, C.. Ontology-Based Semantic Integration Method for Domain-Specific Scientific Data. In Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Vol. 03, pp. 772-777, July 30-Aug. 1, 2007.

[INFOSEC, 1999] National Information Systems Security (INFOSEC) Glossary, NSTISSI No. 4009, January 1999 (Revision 1).

[INRIA, 2010] INRIA, A Format for Ontology Alignment. http://alignapi.gforge.inria.fr/format.html, 2010.

[Isaac, et al., 2007] Isaac, A, van der Meij, L., Schlobach, S., and Wang, S.. An Empirical Study of Instance-based Ontology Matching. In Proceedings of the 6th International Semantic Web Conference (ISWC 2007), pp. 253-266, Busan, Korea, November 11-15, 2007.

[Jasper and Ushold, 1999] Jasper, R. and Ushold, M.. A Framework for Understanding and Classifying Ontology Applications. In Proceedings of the 12th Banff Knowledge Acquisition for Knowledge-based

Systems Wrokshop, pp. 16-21, 1999.

[Jin, et al., 2005] Jin, J., Sarker, B. K., Bhavsar, V. C., Boley, H., and Yang, L.. Towards a Weighted-Tree Similarity Algorithm for RNA Secondary Structure Comparison. In Proceedings of HPC Asia 2005, pp. 639-644, IEEE Computer Society, Beijing, China, November 30-Decemeber 3, 2005.

[Kalfoglou and Schorlemmer, 2003] Kalfoglou, Y. and Schorlemmer, M.. Ontology Mapping: the State of the Art. The Knowledge Engineering Review, Vol. 18:1, pp. 1-31, Cambridge University Press, 2003.

[Karp, 1993] Karp, P. D.. The Design Space of Frame Knowledge Representation Systems. Technical Note 520, AI Center SRI International, Menlo Park, CA, 1993.

[Kendal and Creen, 2007] Kendal, S. and Creen, M.. An Introduction to Knowledge Engineering. Springer-Verlag London Limited, 2007.

[Kiatisevi, et al., 2006] Kiatisevi, P., Ampornaramveth, V., and Ueno, H.. A Frame-based Knowledge Software Tool for Developing Interactive Robots. Artificial Life and Robotics, Vol. 10, No. 1, pp. 18-28, July 2006.

[Kifer, et al., 1995] Kifer, M., Lausen, G., and Wu, J.. Logical Foundation of Object-Oriented and Frame-based Languages. Journal of ACM, Vol. 42, pp. 741-843, 1995.

[Kirk, et al., 1995] Kirk, T., Levy, A. Y., Sagiv, Y., and Srivastava, D.. The Information Manifold. In Proceedings of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments, Stanford University, pp. 85-91, 1995.

[Kitakami, et al., 1996] Kitakami, H., Mori, Y., and Arikawa, M.. An Intelligent System for Integrating Autonomous Nomenclature Databases in Semantic Heterogeneity. In Database and Expert System Applications, CEXA'96, No. 1134 in Lecture Notes in Computer Science, pp. 187-196, Zurich, Switzerland, 1996.

[Klahr, et al., 1987] Klahr, D., Langley, P., and Neches, R.. Production System Models of Learning and Development. Cambridge, Mass.: The MIT Press, 1987.

[Klein and Noy, 2003] Klein, M. and Noy, F. N.. A Component-based Framework for Ontology Evolution, In Proceedings of the IJCAI'03 Workshop: Ontologies and Distributed Systems, Acapulco, Mexico, 2003.

[Klein, 2001] Klein, M.. Combining and Relating Ontologies: An Analysis of Problems and Solutions. In A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, and M. Uschold (eds), Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA, Aug. 4-5, 2001.

[Kotsiantis and Pintelas, 2004] Kotsiantis, S. and Pintelas, P.. Recent Advances in Clustering: A Brief Survey. WSEAS Transactions on Information Science and Applications, 1(1), pp. 73-81, 2004.

[Kouylekov and Magnini, 2005] Kouylekov, M. and Magnini, B.. Recognizing Textual Entailment with Tree Edit Distance Algorithms. In Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment, pp. 17-20, Southampton, UK, 2005.

[Kruskal, 1999] Kruskal, J. B.. An Overview of Sequence Comparison. In D. Sankoff and J. Kruskal (eds): Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Chapter One, CSLI Publications, 1999.

[Kullback, 1987] Kullback, S.. The Kullback-Leibler Distance. The American Statistician, 41, pp. 340-341, 1987.

[Lambrix and Tan, 2006] Lambrix, P. and Tan, H.. SAMBO -- A System for Aligning and Merging Biomedical Ontologies. Web Semantics: Science, Services and Agents on the World Wide Web, Vol.4, No.3, pp.196-206, September 2006.

[Larson, et al., 1989] Larson, J. A., Navathe, S. B., and Elmasri, R.. A Theory of Attribute Equivalence in Databases with Application to Schema Integration. IEEE Transactions on Software Engineering, 16(4): 449-463, 1989.

[Lenzerini, 2002] Lenzerini, M.. Data Integration: a Theoretical Perspective. In Proceedings of Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.

233-246, Madison, Wisconsin, USA, 2002.

[Levy, 1999] Levy, A. Y.. Answering Queries Using Views: A Survey. Technical Report, University of Washington, 1999.

[Levy, 2000] Levy, A. Y.. Logic-based Techniques in Data Integration. In Jack Minker (editor) Logic Based Artificial Intelligence, Kluwer Academic Publisher, 2000.

[Levy, et al., 1996] Levy, A., Rajaraman, A., and Ordille, J. J.. Querying Heterogeneous Information Sources using Source Descriptions. In Proceedings of the 22nd International Conference on Very Large Databases, VLDB-96, pp. 251-262, Bombay, India, 1996.

[Li and Chang, 2000] Li, C. and Chang, E.. Query Planning with Limited Source Capabilities. In Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000), pp. 401-412, 2000.

[Li and Clifton, 1994] Li, W. and Clifton, C.. Semantic Integration in Heterogeneous Databases Using Neural Networks. In Proceedings of 20th International Conference on Very Large Databases, pp. 1-12, 1994.

[Li, et al., 2000] Li, W., Clifton, C., and Liu, S.. Database Integration Using Neural Network: Implementation and Experiences. Knowledge Information System, 2(1): 73-96, 2000.

[Li, et al., 2005] Li, L., Wu, B., and Yang, Y.. Agent-based Ontology Integration for Ontology-based Applications. In Proceedings of Australasian Ontology Workshop (AOW 2005), the 18th Australian Joint Conference on Artificial Intelligence, Conferences in Research and Practice in Information Technology (CRPIT) series by Australian Computer Society, Vol. 58, pp. 53-59, 2005.

[Li, et al., 2006] Li, S., Hu, H., and Hu, X.. An Ontology Mapping Method Based on Tree Structure. In Proceedings of the Second International Conference on Semantics, Knowledge, and Grid (SKD 2006), pp. 87-88, Guilin, Guangxi, China, 1-3 November, 2006.

[Madhavan, et al., 2001] Madhavan, J., Bernstein, P. A., and Rahm, E.. Generic Schema Matching with Cupid. In Proceedings of the 27th International Conference on Very Large Databases, pp. 49-58, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc, 2001.

[Maedche and Staab, 2002] Maedche, A. and Staab, S.. Measuring Similarity between Ontologies. In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, Ontologies and the Semantic Web, Lecture Notes in Computer Science, Vol. 2473, pp. 251-263, 2002.

[Magnini, et al., 2003] Magnini, B., Serafini, L., and Speranza, M.. Making Explicit the Semantics Hidden in Schema Models. 2nd International Semantic Web Conference, Workshop on Human Language Technology for the Semantic Web and Web Services, Sanibel Island, Flordia, USA, 20th October, 2003.

[Marinov, 2008] Marinov, M.. Using Frames for Knowledge Representation in a CORBA-based Distributed Environment. Knowledge-Based Systems, Vol. 21, Issue 5, pp. 391-397, July 2008.

[McCann, et al., 2003] McCann, R., Doan, A., Kramnik, A., and Varadarajan, V.. Building Data Integration Systems via Mass Collaboration. In Proceedings of the International Workshop on Web and Databases (WebDB-03), 2003.

[McGuiness, et al., 2000] McGuiness, D., Fikes, R., Rice, J., and Wilder, S.. An Environment for Merging and Testing Large Ontologies. In A. Cohm, F. Giunchiglia, B. Selman (eds), In Proceedings of KR 2000, pp. 483-493. Morgan Kaufmann, 2000.

[McGuinness, et al., 2000] McGuinness, D. L., Fikes, R., Rice, J., and Wilder, S.. An Environment for Merging and Testing Large Ontologies. In A. G. Cohn, F. Giunchiglia, and B. Selman (eds), KR2000: Principles of Knowledge Representation and Reasoning, pp. 483-493, San Francisco. Morgan Kaufmann, 2000.

[Meersman, 1995] Meersman, R.. An Essay on the Role and Evolution of Data (base) Semantics. In Meersman R, Mark L. (eds), Database Application Semantics, Proceedings of IFIP WG 2.6 Working Conference on Database Application Semantics, 1995.

[Melni, et al., 2002] Melnik, S., Garcia-Molina, H., and Rahm, E.. Similarity Flooding: A Versatile

Graph Matching Algorithm and Its Application to Schema Matching. In Proceedings of the 18th International Conference on Data Engineering (ICDE'02), pp. 117-128, Washington, DC, USA, IEEE Computer Society, 2002.

[Mena, et al., 2000] Mena, E., Illarramendi, A., Kashyap, V., and Sheth, A.. OBSERVER: An Approach for Query Processing in Global Information systems based on Interoperation across Pre-existing Ontologies. International Journal of Distributed and Parallel Databases (DAPD), 8(2): 223-272, 2000.

[Mendling, et al., 2005] Mendling, J., de Laborda, C. P., and Zdun, U.. Towards Semantic Integration of XML-based Business Process Models. In Proceedings of the Semantic Model Integration Workshop (SMI 2005) in conjunction with the 3rd International Conference on Professional Knowledge Management (WM 2005), pp. 513-517, Kaiserslautern, Germany, April 13, 2005.

[Menzel, 2008] Menzel, C.: An Account of Abstract Possible Worlds, Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/entries/actualism/possible-worlds.html, 2008.

[Minsky, 1975] Minsky, M.. A Framework for Representing Knowledge. In P. Winston (ed.), The Psychology of Computer Vision. New York: McGraw-Hill, pp. 211-277, 1975.

[Mitra and Wiederhold, 2002] Mitra, P. and Wiederhold, G.. Resolving Terminological Heterogeneity in Ontologies. In Proceedings of the ECAI Workshop on Ontologies and Semantic Interoperability, 2002.

[Mitra, et al., 1999] Mitra, P., Wiederhold, G., and Jannink, J.. Semi-automatic Integration of Knowledge Sources. In Proceedings of Fusion'99, Sunnyvale, USA, 1999.

[Mudumbai, 1997] Mudumbai, S.. ZEBRA: Customizable, Extensible Metadata-based Access to Federated Image Repositories. M. S. Thesis, Department of Computer Science, University of Georgia Online Computer Library Center, 1997.

[Newcomb, 2003] Newcomb, S. R.. A Semantic Integration Methodology. Extreme Markup Languages 2003, Montreal, Quebec, August 4-8, 2003.

[Nodine, et al., 1999] Nodine, M., Bohrer, W., and Ngu, A. H. H.. Semantic Brokering over Dynamic Heterogeneous Data Sources in Infosleuth. In Proceedings of the 15th International Conference on Data Engineering (ICDE 1999), pp. 358-365, 1999.

[Noy and Musen, 2000] Noy, N. F. and Musen, M., PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), Austin, TX, 2000. AAAI/MIT Press, pp. 450-455, 2000.

[Noy and Musen, 2001] Noy, N. F. and Musen, M., Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In Proceedings of the IJCAI Workshop on Ontologies and Information Sharing, pp. 63-70, 2001.

[Noy and Musen, 2003] Noy, N. F. and Musen, M. A., The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. International Journal of Human-Computer Studies, 59(6): 983-1024, 2003.

[Noy, 2003] Noy, F. N.. What Do We Need for Ontology Integration on the Semantic Web, Position Statement. In Proceedings of the Workshop on Semantic Integration, jointed held with the 2nd International Semantic Web Conference, pp. 175-176, Sanibal Island, Florida, USA, 2003.

[Noy, 2004] Noy, N. F.. Semantic Integration: A Survey of Ontology Based Approaches. SIGMOD Record, 33(4): 65~70, 2004.

[Ovsiannikov, et al., 1998] Ovsiannikov, I., Arbib, M., and McNeill, T.. Annotation Software System Design. 1998. URL: http://rana.usc.edu:8376/~ilya/at.ps.gz

[Parent and Spaccapietra, 1998] Parent, C. and Spaccapietra, S.. Issues and Approaches of Database Integration. Communication of the ACM, 41(5): 166-178, 1998.

[Partridge, 2002] Partridge, C.. The Role of Ontology in Semantic Integration. OOPSLA 2002, Seattle, USA.

[Pei, et al., 2006] Pei, J., Hong, J., and Bell, D.. A Novel Clustering-Based Approach to Schema Matching. Advances in Information Systems (Lecture Notes in Computer Science, Volume 4243), Springer Berlin/Heidelberg, pp. 60-69, 2006.

[Pinto and Martins, 2004] Pinto, H. S. and Martins, P. J.. Ontologies: How Can They Be Built? Knowledge and Information Systems, 6 (4), pp. 441-464, 2004.

[Pinto, 1999] Pinto, H. S.. Some Issues on Ontology Integration. In Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), pp. 7-1-7.12, Stockholm, Sweden, August 2, 1999.

[Plantinga and Davidson, 2003] Plantinga, A. and Davidson, M.. Essays in the Metaphysics of Modality. Oxford University Press US, 2003.

[Posner, 1989] Posner, M. I. (ed). Foundations of Cognitive Science. MIT Press, Cambridge, Massachusetts, London, England, 1989.

[Prasad, et al., 2002] Prasad, S., Peng, Y., and Finin, T.. Using Explicit Information to Map Between Two Ontologies. In Proceedings of the AAMAS Workshop on Ontologies in Agent Systems, pp. 52-57, 2002.

[Preece, et al., 1999] Preece, A., Hui, K ., Gray, W., Marti, P., Bench-Capon, T., Jones, D., and Cui, Z.. KRAFT Architecture for Knowledge Fusion and Transformation. In Proceedings of the 19th SGES International Conference on Knowledge-based Systems and Applied Artificial Intelligence (ES'99), Berlin, Springer, 1999.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A.. A Survey of Approaches to Automatic Schema Matching. The International Journal on Very Large Databases (VLDB), 10(4): 334-350, 2001.

[Reichman, et al., 1999] Reichman, O. J., et al. A Knowledge Network for Biocomplexity: Building and Evaluating A Metadata-based Framework for Integrating Heterogeneous Scientific Data. National Science Foundation Award # DEB99-80154. (Available at: http://knb.ecoinformatics.org).

[Rizopoulos, 2004] Rizopoulos, N.. Automatic Discovery of Semantic Relationships between Schema Elements. In Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004). Volume I - Databases and Information Systems Integration. pp. 3-8, 2004.

[Rodriguez and Egenhofer, 2003] Rodriguez, A. and Egenhofer, M.. Determining Semantic Similarity among Entity Classes from Different Ontologies. IEEE Transactions on Knowledge and Data Engineering, 15(2): 442-456, 2003.

[Roth and Schwartz, 1997] Roth, M. T. and Schwartz, P.. Don't Scrap It, Wrap It! In Proceedings of the 23rd International Conference on Very Large Databases, pp. 266-275, 1997.

[Sanin, et al., 2007] Sanin, C., Szczerbicki, E., and Toro, C.. An OWL Ontology of Set of Experience Knowledge Structure. Journal of Universal Computer Science, Vol. 13, No. 2, pp. 209-223, 2007.

[Sciore, et al., 1994] Sciore, E., Siegel, M., and Rosenthal, A.. Using Semantic Values to Facilitate Interoperability among Heterogeneous Information Systems. ACM Transactions on Database Systems, 19(2): 254-290, 1994.

[Scott and Sain, 2004] Scott, D. and Sain, S.. Multi-dimensional Density Estimation. Handbook of Statistics, Volume 24: Data Mining and Computational Statistics, 2004.

[Shapiro, 2006] Shapiro, S.. Classical Logic. The Stanford Encyclopedia of Philosophy (Winter 2006 Edition). Edward N. Zalta (ed.), http://plato.stanford.edu/archives/win2006/entries/logic-classical/.

[Sheth and Larson, 1990] Sheth, A. and Larson, J.. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys, 22(3): 183-236, 1990.

[Sheth, 1998] Sheth, A. P.. Changing Focus on Interoperability in Information Systems: from System, Syntax, Structure to Semantics. In M. F. Goodchild, M. J. Egenhofer, R. Fegeas and C. A. Kottman (eds), Interoperating Geographic Information System, Kluwer, 1998.

[Shvaiko and Euzenat, 2004] Shvaiko, P. and Euzenat, J.. A Survey of Schema-based Matching Approaches. Technical Report DIT—04-087, Information e Telecomunicazioni, University of Trento, 2004.

[Skuce, 1997] Skuce, D.. How We Might Reach Agreement on Shared Ontologies: A Fundamental Approach. In Proceedings of AAAI97 Spring Symposium, Workshop on Ontological Engineering, pp.

114-119, 1997.

[Smiljanic, et al., 2006] Smiljanic, M., van Keulen, M., and Jonker, W.. Using Element Clustering to Increase the Efficiency of XML Schema Matching. In Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06), pp. 45, 2006.

[Sowa, 1997] Sowa, J. F.. Electronic Communication in the Onto-std Mailing List, December 4, 1997.

[Sowa, 1999] Sowa, J. F.. Knowledge Representation: Logical, Philosophical, and Computational Foundations. Thomson Learning, 1999.

[Sowa, 2005] J. F, Sowa. Conceptual Graphs website, http://www.jfsowa.com/cg/index.htm, 2005.

[Spaccapietra, et al., 1992] Spaccapietra, S., Parent, C., and Dupont, Y.. Model Independent Assertions for Integration of Heterogeneous Schemas. VLDB Journal 1: 81-126, 1992.

[Stuckenschmidt and van Harmelen, 2005] Stuckenschmidt, H. and van Harmelen, F.. Information Sharing on the Semantic Web. Springer-Verlag Berlin Heidelberg, 2005.

[Stuckenschmidt and Wache, 2000] Stuckenschmidt, H. and Wache, H.. Context Modeling and Transformation for Semantic Interoperability. In Proceedings of the International Workshop on Knowledge Representation Meets Databases (KRDB 2000), pp. 115-126, 2000.

[Stumme and Maedche, 2001] Stumme, G. and Maedche, A.. FCA-MERGE: Bottom-up Merging of Ontologies. In Proceedings of the 17th International Conference on Artificial Intelligence IJCAI 2001, pp. 225-234, Seattle, WA, 2001.

[Sumllyan, 1995] Sumllyan, R.. First-Order Logic. Courier Dover Publications, 1995.

[Tan, et al., 2006] Tan, H., Jakoniene, V., Lambrix, P., Aberg, J., and Shahmehri, N.. Alignment of Biological Ontologies Using Life Science Literature. In Proceedings of the International Workshop on Knowledge Discovery in Life Science, Singapore, pp. 1-17, 2006.

[Theodoratos, 2002] Theodoratos, D.. Semantic Integration and Querying of Heterogeneous Data Sources Using a Hypergraph Data Model. In: B. Eaglestone, S. North, and A. Poulovassilis (eds): BNCOD 2002, LNCS 2405, Springer-Verlag Berlin Heidelberg, pp. 166-182, 2002.

[Tomberlin and van Inwagen, 1985] Tomberlin, J. E. and van Inwagen, P. (ed). Alvin Plantinga, Springer, 1985.

[Turlach, 1993] Turlach, A. B.. Bandwidth Selection in Kernel Density Estimation: A Review. CORE and Institut de Statistique, pp. 23-493, 1993.

[Uschold and Gruninger, 1996] Uschold, M. and Gruninger, M.. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, 11(2): 93-155, 1996.

[Uschold and King, 1995] Uschold, M. and King, M.. Towards a Methodology for Building Ontologies. In IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.

[Vetere and Lenzerini, 2005] Vetere, G. and Lenzerini, M.. Models for Semantic Interoperability in Service-Oriented Architectures. IBM Systems Journal, Vol. 44, No. 4, pp. 887-903, 2005.

[Visser, 2004] Visser, U.. General Approach of Buster. In Intelligent Information Integration for the Semantic Web, Springer Berlin / Heidelberg, pp. 37-51, 2004.

[Visser, et al., 1997] Visser, P. R. S., Jones, D. M., Bench-Capon, T. J. M., and Shave, M. J. R.. An Analysis of Ontological Mismatches: Heterogeneity versus Interoperability. In AAAI 1997 Spring Symposium on Ontological Engineering, Stanford, USA, 1997.

[Wache, 2003] Wache, H.. Semantic Mediation for Heterogeneous Information Sources. PhD Thesis, University of Bremen, German, 2003.

[Wache, et al., 1999] Wache, H., Scholz, T., Stieghahn, H., and Konig-Ries, B.. An Integration Method for the Specification of Rule-Oriented Mediators. In Y. Kambayashi and H. Takakkura, editors, Proceedings of the International Symposium on Database Applications in Non-Traditional Environments (DANTE'99), pp. 109-112, Kyoto, Japan, 1999.

[Wache, et al., 2001] Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hubner, S.. Ontology-Based Integration of Information – A Survey of Existing Approaches. In Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, pp. 108-117, Seattle, USA, 4-5 August, 2001.

[Wang, 2008] Wang, Y.. Ontology-Driven Semantic Transformation for Cooperative Information Systems, PhD thesis, University of Western Ontario, 2008.

[Wang, et al., 2006] Wang, H., Noy, N., Rector, A., Musen, M., Redmond, T., Rubin, D., Tu, S., Tudorache, T., Drummond, N., Horridge, M., and Seidenberg, J.. Frames and OWL Side by Side. In Proceedings of the 9th International Protégé Conference, July 24-26, 2006, Stanford University, USA.

[Warin, et al., 2005] Warin, M., Oxhammark, H., and Volk, M.. Enriching An Ontology with WordNet based on Similarity Measures. In Proceedings of the MEANING-2005 Workshop, Trento, Italy, February, 2005.

[Wasserman, 2005] Wasserman, L.. All of Statistics: A Concise Course in Statistical Inference. Springer Texts in Statistics, 2005.

[Wick, et al., 2008] Wick, M. L., Rohanimanesh, K., Schultz, K., and McCallum, A.. A Unified Approach for Schema Matching, Coreference and Canonicalization. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 722-730, Las Vegas, Nevada, USA, Aug. 24-27, 2008.

[Wiederhold, 1992] Wiederhold, G.. Mediators in the Architecture of Future Information Systems. Computer, 25(3): 38-49, 1992.

[Wiederhold, 1994] Wiederhold, G.. An Algebra for Ontology Composition. In Proceedings of 1994 Monterey Workshop on Formal Methods, pp. 56-61. U.S. Naval Postgraduate School, Monterey CA, September 1994.

[Yao and Zhang, 2004] Yao, J. T. and Zhang, M.. A Fast Tree Pattern Matching Algorithm for XML Query. In Proceedings of International Conference on Web Intelligence (WI 2004), pp. 235-241, 2004.

[Zeng, 2008] Zeng, Y.. Recursive Object Model (ROM): Modeling of Linguistic Information in Engineering Design. Computers in Industry. 59(6), pp. 612-625, 2008.

Appendix A

Information Models of the Systems in the Prototype Environment

(1) Business Model Management System

Element Name: language

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
language_id	INT	10	0	false	true
resource_id	INT UNSIGNED	10	0	false	false
value	VARCHAR	20	0	false	false

Element Name: price

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
price_id	INT	10	0	false	true
product_id	INT UNSIGNED	10	0	false	false
resource_id	BIGINT	19	0	false	false
start_date	DATE	10	0	false	false
end_date	DATE	10	0	false	false
forever	CHAR	1	0	false	false
value	VARCHAR	255	0	false	false

Element Name: product

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
product_id	INT UNSIGNED	10	0	false	true
product_name	VARCHAR	100	0	true	false
flavor	VARCHAR	20	0	true	false
sweetness	VARCHAR	45	0	true	false
brand_name	VARCHAR	45	0	true	false

Element Name: product_rule

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
product_rule_id	INT UNSIGNED	10	0	false	true
product_rule_name	VARCHAR	100	0	true	false
product_rule_type	VARCHAR	255	0	true	false

Element Name: product_rule_map

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment

product_rule_parent_id	INT UNSIGNED	10	0	false	false
product_rule_child_id	INT UNSIGNED	10	0	false	false

Element Name: product_rule_product_map

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
product_rule_id	INT UNSIGNED	10	0	false	false
product_id	INT UNSIGNED	10	0	false	false

Element Name: resource

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
resource_id	INT UNSIGNED	10	0	false	true
resource_name	VARCHAR	100	0	true	false
aspect_ratio	VARCHAR	20	0	true	false
orientation	VARCHAR	20	0	true	false
resolution	VARCHAR	20	0	true	false
marketing_zone_id	INT	10	0	true	false

Element Name: resource_rule

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
resource_rule_id	INT UNSIGNED	10	0	false	false
resource_rule_type	VARCHAR	255	0	false	false
resource_rule_name	VARCHAR	100	0	true	false

Element Name: resource_rule_map

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
resource_rule_parent_id	INT UNSIGNED	10	0	false	false
resource_rule_child_id	INT UNSIGNED	10	0	false	false

Element Name: resource_rule_resource_map

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
resource_rule_id	INT UNSIGNED	10	0	false	false
resource_id	INT UNSIGNED	10	0	false	false

Element Name: time

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
time_id	INT UNSIGNED	10	0	false	true

time_name	VARCHAR	100	0	true	false
time_type	VARCHAR	20	0	true	false
start	VARCHAR	60	0	true	false
end	VARCHAR	60	0	true	false

Element Name: time_rule

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
time_rule_id	INT UNSIGNED	10	0	false	true
time_rule_name	VARCHAR	100	0	true	false
time_rule_type	VARCHAR	255	0	true	false

Element Name: time_rule_time_map

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
time_rule_id	INT UNSIGNED	10	0	false	false
time_id	INT UNSIGNED	10	0	false	false

(2) Media Management System

Element Name: asset

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
asset_id	BIGINT	19	0	false	true
user_given_name	VARCHAR	255	0	true	false
client_id	BIGINT	19	0	true	false
create_date	DATETIME	19	0	true	false
owner_library_id	BIGINT	19	0	true	false
current_config_id	BIGINT	19	0	true	false
product_id	INT UNSIGNED	10	0	true	false
media_file_id	BIGINT	19	0	true	false

Element Name: asset_meta_value_map

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
asset_id	BIGINT	19	0	false	false
meta_value_id	BIGINT	19	0	false	false

Element Name: media_file

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
media_file_id	BIGINT	19	0	false	true

server_given_name	VARCHAR	255	0	true	false
create_date	DATETIME	19	0	true	false
media_content	LONGTEXT	2147483647	0	true	false
mime_type	VARCHAR	255	0	true	false
file_size	BIGINT	19	0	true	false
file_type	VARCHAR	255	0	true	false

Element Name: media_library

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
media_library_id	BIGINT	19	0	false	true
name	VARCHAR	255	0	true	false
client_id	BIGINT	19	0	true	false
association_service	VARCHAR	255	0	true	false
is_third_party	BIT	1	0	true	false
service_url	VARCHAR	255	0	true	false
is_deleted	BIT	1	0	true	false
create_date	DATETIME	19	0	true	false

Element Name: meta_value

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
meta_value_id	BIGINT	19	0	false	true
meta_tag_name	VARCHAR	255	0	true	false
meta_tag_value	VARCHAR	255	0	true	false

Element Name: prod

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
id	INT UNSIGNED	10	0	false	true
name	VARCHAR	100	0	true	false
description	VARCHAR	255	0	true	false

Element Name: system_keywords

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
asset_id	BIGINT	19	0	false	false
keyword	VARCHAR	255	0	false	false

Element Name: thumbnail

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
thumbnail_id	BIGINT	19	0	false	true
asset_id	BIGINT	19	0	true	false
file_type	VARCHAR	255	0	true	false
width	INT	10	0	false	false
height	INT	10	0	false	false
media_file_id	BIGINT	19	0	true	false
media_library_id	BIGINT	19	0	true	false
asset_config_id	BIGINT	19	0	true	false

Element Name: user_keywords

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
asset_id	BIGINT	19	0	false	false
keyword	VARCHAR	255	0	false	false

(3) Promotion Management System

Element Name: daypart

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
dp_daypart_id	INT UNSIGNED	10	0	false	true
dp_name	VARCHAR	128	0	false	false
dp_state	INT UNSIGNED	10	0	false	false
dp_client_id	INT UNSIGNED	10	0	false	false

Element Name: daypart_time_to_play

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
dpttp_daypart_time_to_play_id	INT UNSIGNED	10	0	false	true
dp_daypart_id	INT UNSIGNED	10	0	false	false
ttp_time_to_play_id	INT UNSIGNED	10	0	false	false

Element Name: history

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
hi_history_id	INT UNSIGNED	10	0	false	true
hi_user_id	INT UNSIGNED	10	0	false	false

hi_date	INT UNSIGNED	10	0	false	false
hi_action	VARCHAR	255	0	false	false
hi_comment	VARCHAR	255	0	false	false
hi_status	VARCHAR	32	0	false	false

Element Name: prev_media_asset

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
pma_prev_media_asset_id	INT UNSIGNED	10	0	false	true
pr_promotion_id	INT UNSIGNED	10	0	false	false
pma_asset_id	INT UNSIGNED	10	0	false	false
pma_library_id	INT UNSIGNED	10	0	false	false

Element Name: products

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
pr_id	INT UNSIGNED	10	0	false	true
pr_name	VARCHAR	100	0	true	false
pr_description	VARCHAR	255	0	true	false

Element Name: promotion

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
pr_promotion_id	INT UNSIGNED	10	0	false	true
pr_name	VARCHAR	128	0	false	false
pr_client_id	INT UNSIGNED	10	0	false	false
pr_product_id	INT UNSIGNED	10	0	false	false
pr_layout_id	INT UNSIGNED	10	0	false	false
pr_region_id	INT UNSIGNED	10	0	false	false
pr_media_request_id	INT UNSIGNED	10	0	false	false
pr_source	VARCHAR	128	0	false	false
pr_bm_irt_id	INT UNSIGNED	10	0	false	false
pr_status	VARCHAR	32	0	false	false
pc_promotion_config_id	INT UNSIGNED	10	0	false	false
pr_note	TEXT	65535	0	false	false

Element Name: promotion_conf_daypart

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment

pcd_promotion_conf_daypart_id	INT UNSIGNED	10	0	false	true
pc_promotion_config_id	INT UNSIGNED	10	0	false	false
dp_daypart_id	INT UNSIGNED	10	0	false	false

Element Name: promotion_conf_time_to_play

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
pcttp_promotion_config_time_to_play_id	INT UNSIGNED	10	0	false	true
ttp_time_to_play_id	INT UNSIGNED	10	0	false	false
pc_promotion_config_id	INT UNSIGNED	10	0	false	false

Element Name: promotion_config

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
pc_promotion_config_id	INT UNSIGNED	10	0	false	true
pc_start_date	INT UNSIGNED	10	0	false	false
pc_end_date	INT UNSIGNED	10	0	false	false
pr_promotion_id	INT UNSIGNED	10	0	false	false

Element Name: promotion_history

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
pr_promotion_history_id	INT UNSIGNED	10	0	false	true
pr_promotion_id	INT UNSIGNED	10	0	false	false
pc_promotion_config_id	INT UNSIGNED	10	0	false	false
hi_history_id	INT UNSIGNED	10	0	false	false

Element Name: promotion_resource_allocation

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
promotion_id	INT UNSIGNED	10	0	false	false
resource_id	INT UNSIGNED	10	0	false	false

Element Name: resource_groups

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment

rg_resource_group_id	INT UNSIGNED	10	0	false	false
pc_promotion_config_id	INT UNSIGNED	10	0	false	false
rg_is_exclusive	BIT	1	0	false	false

Element Name: resources

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
re_id	INT UNSIGNED	10	0	false	true
re_name	VARCHAR	60	0	false	false
re_model	VARCHAR	60	0	true	false

Element Name: time_to_play

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
ttp_time_to_play_id	INT UNSIGNED	10	0	false	true
ttp_start_time	VARCHAR	32	0	false	false
ttp_end_time	VARCHAR	32	0	false	false
ttp_days_of_week	SMALLINT UNSIGNED	5	0	false	false

(5) Inventory Management System

Element Name: inventory

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
product_id	INT UNSIGNED	10	0	false	false
location_id	INT UNSIGNED	10	0	false	false
quantity	INT UNSIGNED	10	0	false	false
load_date	DATE	10	0	false	false

Element Name: load_records

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
record_id	INT UNSIGNED	10	0	false	true
load_date	DATE	10	0	false	false
operator	VARCHAR	60	0	false	false
product_id	INT UNSIGNED	10	0	false	false
location_id	INT UNSIGNED	10	0	false	false
load_quantity	INT UNSIGNED	10	0	false	false
comments	VARCHAR	255	0	true	false

Element Name: location

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
location_id	INT UNSIGNED	10	0	false	false
location_type	VARCHAR	45	0	false	false
location_number	VARCHAR	45	0	false	false
location_name	VARCHAR	100	0	false	false

Element Name: products

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
product_id	INT UNSIGNED	10	0	false	true
product_name	VARCHAR	100	0	true	false
product_type	VARCHAR	45	0	true	false
comments	VARCHAR	255	0	true	false

(6) Transaction Management System

Element Name: operation_assignments

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
o_id	VARCHAR	10	0	false	false
pos_number	VARCHAR	20	0	false	false

Element Name: operators

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
o_id	VARCHAR	10	0	false	false
o_first_name	VARCHAR	45	0	false	false
o_hourly_rate	DECIMAL	10	2	false	false
o_last_name	VARCHAR	45	0	false	false

Element Name: pos_machines

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
pos_number	VARCHAR	20	0	false	false
pos_model	VARCHAR	45	0	true	false
install_date	DATE	10	0	true	false
counter_no	INT UNSIGNED	10	0	true	false

Element Name: products

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
p_id	INT UNSIGNED	10	0	false	true

p_name	VARCHAR	100	0	true	false
p_price	DECIMAL	10	2	true	false
p_note	VARCHAR	255	0	true	false

Element Name: receipts

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
r_id	INT UNSIGNED	10	0	false	true
r_number	VARCHAR	45	0	false	false
start_time	DATETIME	19	0	true	false
o_id	VARCHAR	10	0	true	false
pos_number	VARCHAR	20	0	true	false
total_price	DECIMAL	10	2	true	false
payment_way	VARCHAR	45	0	true	false
payment	DECIMAL	10	2	true	false
change	DECIMAL	10	2	true	false

Element Name: transactions

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
t_id	INT UNSIGNED	10	0	false	true
sales_time	DATETIME	19	0	false	false
p_id	INT UNSIGNED	10	0	false	false
quantity	INT UNSIGNED	10	0	false	false
r_id	INT UNSIGNED	10	0	false	false

(7) Scheduling Management System (XML)

Element Name: res

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
aspection_ration	N/A	N/A	N/A	N/A	N/A
id	N/A	N/A	N/A	N/A	N/A
location	N/A	N/A	N/A	N/A	N/A
name	N/A	N/A	N/A	N/A	N/A
orientation	N/A	N/A	N/A	N/A	N/A
resolution	N/A	N/A	N/A	N/A	N/A

Subelements:

None

Element Name: prod

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
description	N/A	N/A	N/A	N/A	N/A
id	N/A	N/A	N/A	N/A	N/A
name	N/A	N/A	N/A	N/A	N/A
price	N/A	N/A	N/A	N/A	N/A

Subelements:

None

Element Name: media

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
id	N/A	N/A	N/A	N/A	N/A
location	N/A	N/A	N/A	N/A	N/A
name	N/A	N/A	N/A	N/A	N/A
prod	N/A	N/A	N/A	N/A	N/A
resolution	N/A	N/A	N/A	N/A	N/A

Subelements:

None

Element Name: time

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
id	N/A	N/A	N/A	N/A	N/A
name	N/A	N/A	N/A	N/A	N/A
type	N/A	N/A	N/A	N/A	N/A
value	N/A	N/A	N/A	N/A	N/A

Subelements:

None

Element Name: schedule

Attribute Name	Data Type	Size	Decimal Digits	Nullable	Auto Increment
id	N/A	N/A	N/A	N/A	N/A

Subelements:

Element Name	Type
prod	Simple

Element Name	Type

time	Simple

Element Name	Type
media	Simple

Element Name	Type
resources	Complex res

Original XML Document:

```
<data>
  <resources>
    <res id="1" name="res1" resolution="800x600" aspection_ration="4:3"
orientation="LANDSCAPE" location="store1"/>
    <res id="2" name="res2" resolution="1024x768" aspection_ration="16:9"
orientation="LANDSCAPE" location="store2"/>
    <res id="3" name="res3" resolution="600x800" aspection_ration="3:4"
orientation="PORTRAIT" location="store3"/>
  </resources>

  <products>
    <prod id="1" name="Donuts" description="" price="1.99"/>
    <prod id="2" name="Apple Fritter" description="" price="2.99"/>
    <prod id="3" name="Honey Dip" description="" price="2.49"/>
  </products>

  <media_assets>
    <media id="0001" name="donuts.mgp" location="/repository/media/"
resolution="800x600" prod="1" />
    <media id="0002" name="applefritter.mgp" location="/repository/media/"
resolution="800x600" prod="2" />
    <media id="0003" name="honeydip.mgp" location="/repository/media/"
resolution="800x600" prod="3" />
  </media_assets>

  <times>
    <time id="1" type="DAY" name="Monday" value="1" />
    <time id="2" type="DAY" name="Tuesday" value="2" />
    <time id="3" type="DAYTIME" name="lunch" value="11:00-13:00" />
    <time id="4" type="FLIGHTDATE" name="Christmas"
value="12/23/2009-12/26/2009" />
  </times>

  <schedules>
    <schedule id="1">
```

```
      <prod id="1" />
      <time id="2" />
      <media id="0003" />
      <resources>
        <res id="1" />
        <res id="2" />
      </resources>
    </schedule>
  </schedules>
</data>
```

Curriculum Vitae

Name Yunjiao Xue

Post-secondary Education and Degrees

●	Sept. 2006 - present	**Ph.D. candidate in Engineering Science** The University of Western Ontario, London, Ontario
●	Sept. 2003 - June 2006	**Ph.D. in Computer Science** Fudan University, Shanghai, P.R.China
●	Sept. 1999 - July 2002	**Master of Science in Computer Science** Fudan University, Shanghai, P.R.China
●	Sept. 1995 - July 1999	**Bachelor of Science in Computer Science** Fudan University, Shanghai, P.R.China

Honours and Awards

- **Scholarships**

2008 - 2010	Natural Sciences and Engineering Research Council of Canada Industrial Postgraduate Scholarships (NSERC IPS)
2006 - 2010	Western Graduate Research Scholarship - Engineering
2001 - 2002	Graduate Fellowship of Fudan University (first class)
2001 - 2002	Intel Scholarship
2000 - 2001	Liu Yongling Scholarship
2000 - 2001	Graduate Fellowship of Fudan University (second class)
2000 - 2001	Inchcape Hong Kong University China Scholarship Programme
2000	IBM Scholarship for Outstanding Students of China
2000	Johnson & Johnson Scholarship
1996 - 1997	Motorola Scholarship

- **Awards**

2010	CDS-EnG A/IR&D Collaboration Award
2008	CDS-EnG AR&D Collaboration Award
2007	CDS-EnG Best Research Achievements Award
2007	CDS-EnG Best Active Research Award

Related Work Experience

- October 2006 - present, Guest Worker, National Research Council Canada (NRC), London, Ontario, Canada
- October 2006 - present, Research Assistant at EK3 Technologies Inc., London, Ontario, Canada
- September 2006 - April 2010, Teaching Assistant at The University of Western

Ontario, London, Ontario, Canada

- July 2002 - July 2006, Lecturer, School of Information Science and Engineering, Fudan University, Shanghai, P.R.China

Related Publications

1. **Yunjiao Xue**, Hamada H. Ghenniwa, Weiming Shen. "Frame-based Ontological View for Semantic Integration". Recommended as a candidate paper for possible publication in the Elsevier Journal of Network and Computer Applications, 2010.
2. **Yunjiao Xue**, Hamada H. Ghenniwa, Weiming Shen. "A Frame-based Ontological View Specification Language". Proceedings of the 14th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2010), pp. 228-233, Shanghai, China, April 14-16, 2010.
3. **Yunjiao Xue**, Hamada H. Ghenniwa, Weiming Shen. "Instance-based Domain Ontological View Creation towards Semantic Integration". International Journal of Expert Systems with Applications, 38 (2011): pp. 1193-1202. DOI: 10.1016/j.eswa.2010.05.012.
4. **Yunjiao Xue**, H. H. Ghenniwa, W. Shen. "A Tree Similarity Measuring Method and its Application to Ontology Comparison". Journal of Universal Computer Science, Vol. 15, No. 9, pp. 1766-1781, 2009.
5. **Yunjiao Xue**, H. H. Ghenniwa, W. Shen. "Ontological View-driven Semantic Integration in Collaborative Networks". 10th IFIP Working Conference on VIRTUAL ENTERPRISES (PRO-VE'09), Thessaloniki, Greece, October 7-9, 2009.
6. **Yunjiao Xue**, H. H. Ghenniwa, W. Shen. "Instance-based Domain Ontological View Creation". Proceedings of the 13th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2009), pp. 344-349, Santiago, Chile, April 22-24, 2009.
7. Y. D. Wang, H. H. Ghenniwa, W. Shen, **Yunjiao Xue**. "Service-Oriented Coordinated Intelligent Rational Agent Model for Distributed Information Systems". Proceedings of the 13th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2009), pp. 350-355, Santiago, Chile, April 22-24, 2009.
8. **Yunjiao Xue**, C. Wang, H. H. Ghenniwa, W. Shen. "A New Tree Similarity Measuring Methodology and its Application to Ontology Comparison". Proceedings of the 12th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2008), pp. 258-263, Xi'an, China, April 16-18, 2008.
9. **Yunjiao Xue**, H. H. Ghenniwa, W. Shen. "Measuring Business Process Similarity based on Graph Transformation Cost". 2nd International SeMSoc Workshop – Business Oriented Aspects concerning Semantics and Methodologies in Service-Oriented Computing, 5th International Conference on Service Oriented Computing/ICSOC 2007, Vienna, Austria, September 17-20, 2007 (Accepted).
10. **Yunjiao Xue**, H. H. Ghenniwa, W. Shen. "An Extended Methodology for Tree Similarity Measuring and Its Application on Ontology Integration". *Workshop on Collective Intelligence on Semantic Web (CISW 2007)*, *IEEE/WIC/ACM Joint International Conference on Web Intelligence and Intelligent Agent Technology 2007*, Silicon Vally, USA, November 2-5, 2007 (Accepted).