REPRESENTING REAL-WORLD SEMANTICS IN OWL ONTOLOGIES

by

ANNA VLADIMIROVNA KRASNOPEROVA

B.Sc., The Ural State University, 1994
M.Sc., The Ural State University, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN BUSINESS ADMINISTRATION

in

THE FACULTY OF GRADUATE STUDIES

(Management Information Systems)

THE UNIVERSITY OF BRITISH COLUMBIA

March, 2006

ABSTRACT

The emergence of the Semantic Web as a future of the World Wide Web has created a strong interest in information system (IS) ontologies as a means of representing - in a formal and machine readable form - the knowledge about various domains, and the semantics of heterogeneous web sources in particular. For effective representation, sharing and reuse of real world domain knowledge, an IS ontology needs to properly convey beliefs about the real world. However, as the focus of IS ontologies is often on formalization and machine-readability, the question arises as to how well IS ontologies or ontology development languages allow the representation of a real world domain.

The OWL Web Ontology Language is an ontology development language recently proposed by the World Wide Web Consortium as one of the key components of the Semantic Web. Prior research identified several potential weaknesses of OWL in its ability to represent knowledge about real world domains, which may lead to limited expressiveness, ambiguity, inconsistency and lack of stability of representations. It suggested ways to improve the expressiveness of OWL by associating it with a philosophical ontology that deals with what exists in the real world.

This thesis continues research in that direction. It uses an established philosophical ontology – Bunge's ontology - for developing modeling rules and guidelines on how to better represent real world domains in OWL. The study conducts a comparative analysis of the key constructs of Bunge's ontology (things, properties, interactions, classes and composition) and the related OWL constructs, so as to propose a representation mapping between these constructs. Through the transfer of ontological assumptions of Bunge's ontology, this thesis develops general modeling guidelines and specific rules on how to model real world domain elements in OWL ontologies. In addition, this thesis proposes a meta-model – a set of high-level domain independent OWL classes and properties for OWL ontologies of real world domains – to facilitate the application of the proposed guidelines and to clarify the semantics of domain specific elements of ontologies. The applicability and the process of applying the proposed modeling approach are illustrated by specific examples.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my thesis supervisor, Dr. Yair Wand, for the on-going support of this research and his helpful advice and suggestions. I thank him for the inspiration for this thesis's topic and really appreciate his believing in my abilities as well as his encouragement by helping me believe in the potential and the importance of this work. I am also very grateful to Palash Bera for fruitful discussions, advice, and feedback on my work. A special thank you goes to my thesis defense committee members, Carson Woo and Andrew Burton-Jones, for their interest and appreciation of my work, and for their useful comments and suggestions.

I would like to extend my thanks to Ed Egan for his helpful comments, moral support, and continuous encouragement during the work on this thesis, as well as for providing me with valuable opportunities to participate in other interesting projects and to keep my professional skills up to date while working on this research.

A very special thank you goes to my husband, Alexey, and to my son, Andrey, for their love and support. This thesis would not be possible without my husband's understanding and continuous help in various aspects of life – from computer issues to household chores to child care - during the months of my studying in the program and working on this thesis. I especially appreciate my son's patience and having to put up with me being absorbed in reading research papers or busy working at the computer (which sometimes deprived him from playing computer games).

Finally, I am indebted to my beloved parents, Tatiana and Vladimir Balashov, my parents-in-law, Natalia and Vitaly Krasnoperov, and my sister Tatiana Balashova, whose long-distance support, encouragement, and invaluable help during their visits to us allowed me to concentrate on the work, and whose loving and caring attitude towards my son compensated him, at least in part, for the lack of attention from his busy mother.

1 INTRODUCTION

1.1 IS ONTOLOGIES AND OWL WEB ONTOLOGY LANGUAGE

During the last decade ontologies and ontological engineering have gained considerable popularity in a variety of research disciplines including artificial intelligence, knowledge engineering and representation, and computer science. Ontologies are now widely used in research and applications related to knowledge management, natural language processing, e-commerce, intelligent information integration, information retrieval, integration of heterogenous databases, bioinformatics, and, most recently, in newly emerging fields like the Semantic Web (Gomez-Perez *et al.*, 2004). A popular definition of an ontology was proposed by Gruber (1993): "An ontology is a formal explicit specification of a shared conceptualization which is a simplified view of the world". In simpler terms, a formalized ontology can be understood as set of (usually computer usable) definitions of concepts and relationships among concepts describing a particular domain. Ontologies provide a common vocabulary of an area and define, with different levels of formality, the meaning of the terms and relationships between them (Gomez-Perez *et al.*, 2004). They are used by people, databases, and applications that need to share subject-specific (domain) information – domain examples include medicine, tool manufacturing, real estate, automobile repair, financial management, and others.

A number of markup languages and ontology development languages have been developed, such as XML, RDF(S), OIL, DAML + OIL, and, more recently, OWL. Some languages have been developed for use in specific projects and applications, while others have gained wider acceptance and popularity in support of the goal of making the Web more accessible by computers. One such language – the OWL Web Ontology Language – is the focus of this thesis. OWL is a language for describing, publishing and sharing ontologies on the World Wide Web. It has been developed by the World Wide Web Consortium (W3C)[1] Web Ontology working group and has been approved (as of February 10, 2004) as a W3C recommendation, which is understood by industry and the Web community at large as an accepted Web standard (W3C press release, 2004). OWL is currently considered one of the key accepted Semantic Web technologies that together provide a framework for asset management, enterprise integration, and data sharing and reuse on the Web.

[1] World Wide Web Consortium (W3C) website: http://www.w3c.org

1.2 IS ONTOLOGIES AND REPRESENTATION OF REAL WORLD DOMAINS

One of the key purposes of developing ontologies is to model relevant aspects of some real world domain (so that representations can be agreed upon, shared and reused by applications and people). Often, the main focus of IS ontologies and Web-based ontology development languages is on formalization and machine-readability; that is, on expressing domain conceptualization (shared by a community) explicitly in a form that can be understood and processed by computers. However, for effective representation, sharing and reuse of domain knowledge, it is equally important that an IS ontology properly conveys beliefs about the real world – i.e. beliefs on what exists, might exist or happen, as perceived by a community of domain knowledge users (Bera & Wand, 2004). Clear, consistent and stable representation of real world domain elements and relationships in IS ontologies is essential for effective and efficient development, maintenance, alignment, sharing and reuse of ontologies by people and applications. Thus, important research questions arise as to what should be considered a clear and consistent representation of a real-world domain, how well a particular IS ontology or an ontology development language are suited for proper representation of a real world domain of interest, and how language constructs and functionality should be used to achieve a stable and consistent representation.

OWL language provides a number of fundamental constructs for defining classes, individuals, and properties, and for asserting properties of classes and individuals. OWL syntax allows the creation of ontologies that can be used in reasoning about classes, properties and individuals, to the degree permitted by the formal (logical) semantics of OWL. However, OWL is not specifically tailored to real world domain representation, but rather is intended as a general language providing generic constructs which allow representation of any concepts (whether "real world" or abstract) and the relationships between them in an ontology. No clear real world domain semantics or guidelines on how to use these constructs to properly represent real world domain knowledge are available for OWL. Modelers using OWL are allowed substantial freedom and flexibility (within OWL syntax rules) in how they can apply these basic constructs to represent domains of interest. On one hand this is an advantage of OWL, providing language flexibility, universality and applicability to multiple domains. On the other hand, such freedom, together with the lack of modeling guidelines and constraints, may become a drawback and can lead to a number of problems including limited expressiveness of an implemented ontology, inconsistent domain

representation by different ontologies, potential for ambiguous interpretation, and a lack of stability of ontologies (as significant changes may be required for an existing ontology when more domain knowledge is acquired, affecting applications using information described by this ontology). Bera and Wand (2004) discuss some specific examples of the above issues in the case of OWL ontologies modeling real world domains.

1.3 ONTOLOGICAL EXPRESSIVENESS OF THE OWL LANGUAGE

To identify possible causes of such problems and potential ways of alleviating them, Bera and Wand (2004) proposed using an established philosophical ontology – Bunge's ontology (Bunge, 1977, 1979) - as a benchmark for evaluating *OWL expressiveness* (i.e. how well OWL allows the representation of a real world domain of discourse). A philosophical ontology (such as Bunge's ontology) makes explicit commitments about what might exist and happen in a domain. It provides a number of high-level constructs (such as things, properties, classes, states, events, etc.) for describing real world phenomena. To evaluate OWL expressiveness, Bera and Wand (2004) apply the notion of *ontological expressiveness,* which had been introduced by Wand & Weber (1993) as a way to analyze and evaluate conceptual modeling grammars and languages. The ontological expressiveness approach has been used to evaluate a number of systems analysis and design methods, such as ERM, ARIS, UML etc. (e.g. Evermann & Wand, 2001a,b; Green & Rosemann, 2000; Wand *et al.*, 1999). The evaluation is done by exploring the mappings between a set of ontological concepts and the grammar's constructs. A grammar is *ontologically complete* if and only if every ontological construct can be mapped to a grammar construct (i.e. the mapping is *total*). A grammar is *ontologically clear* if and only if no two ontological concepts are mapped onto the same grammar construct, and all grammar constructs can be mapped to ontological concepts (Wand & Weber, 1993).

In the analysis of the ontological expressiveness of OWL, Bera and Wand (2004) identified a number of deficiencies in the mapping between OWL and Bunge's ontological constructs. In particular, they argue that 1) mapping between OWL and Bunge ontology is not total (ontological incompleteness) as some ontological constructs do not have an equivalent in OWL (construct deficit), and that 2) there are several situations when ontological incompleteness arises due to construct overload, construct redundancy or construct excess. These deficiencies are likely to lead to the problems discussed above, namely limited expressiveness, ambiguity, inconsistency and lack of stability of real world domain representations using OWL ontologies.

1.4 THESIS SCOPE AND OBJECTIVES

Bera and Wand (2004) proposed that evaluating IS ontologies (and OWL in particular) against a philosophical ontology can provide modeling guidance to support the creation of consistent and stable ways of describing domain knowledge. Such guidance can be provided by introducing ontologically grounded general representation guidelines, as well as specific guidelines on modeling certain ontological constructs in OWL. Also, the philosophical ontology can suggest the addition of new constructs to overcome the lack of expressiveness of the language.

This thesis continues the work cited above on the ontological analysis and improvement of the expressiveness of OWL. Specifically, the key objectives of this thesis are as follows:

- To conduct, based on ontological foundations, a comparative analysis of certain OWL and Bunge constructs so as to propose clear and consistent mapping between those constructs or their combinations (with respect to real world domains)
- To develop, based on the comparative analysis and proposed mapping, a high-level class structure (meta-model) and representational rules and guidelines for modeling real world domains in OWL ontologies

The proposed ontologically grounded guidelines and a high-level class structure (which will also be termed 'meta-model' in this thesis) are intended to help OWL modelers avoid, or at least alleviate, the earlier mentioned problems with ontologies and to facilitate the development of more clear, consistent, unambiguous, and stable OWL ontologies for real world domains.

The scope of this thesis is limited mainly to the representation issues related to static aspects of the world (which are described by such concepts of Bunge ontology as things, properties, and classes). Some dynamics-related concepts (such as interactions) are discussed when relevant to the analysis, but other dynamic concepts (states, events, etc) are outside of the scope of this work and are not discussed in detail.

Also, the scope of this work does not include detailed discussion of another OWL improvement method proposed by Bera & Wand (2004) – the addition of new constructs to OWL. Developing, implementing, and approving significant changes to the OWL language would require considerable time and effort from the many researchers and practitioners involved in the development and use of OWL and OWL ontologies, because OWL has been approved as a current Web standard recommended by the W3C and has already been widely used for developing ontologies in research and applications. Thus it is appropriate to limit

the scope of this research and not attempt to change the current OWL functionality[2], but rather suggest how the available constructs and functionality can be used in ways consistent with ontological foundations. Nevertheless, we believe that our recommendations can also provide some ideas as to what constructs can be added to OWL in the future versions of the OWL language in order to improve OWL expressiveness, and how such new constructs can be best implemented in OWL.

1.5 GENERAL APPROACH

To achieve the objectives of the thesis, we use a general approach that has been developed based on the ontological analysis method initially proposed by Wand and Weber (Wand & Weber 1993; see also Wand *et al.* 1995). This approach has been successfully employed in several prior contributions to the literature on the ontological analysis of various modeling languages (e.g. Evermann & Wand, 2001a,b; Green & Rosemann, 2000; Wand *et al.*, 1999). It has recently been formalized by Evermann and Wand (2005) in a general form in the case of conceptual modeling languages. As stated in Evermann & Wand (2005), the general idea of the approach is that the likelihood of creating correct domain models can be increased if the syntax of a modeling language is restricted to ensure that only possible configurations of a domain can be modeled. This can be achieved through first assigning domain (ontological) semantics to the modeling language constructs, and then subsequently restricting the syntax of the modeling language to respect assumptions and constraints in the domain (ontological assumptions).

Specifically, our analysis includes the following key steps:

1) Adopting a specific high-level ontology – Bunge's ontology, or more specifically, Bunge-Wand-Weber ontology (BWW),[3] – as a view of the real world (as perceived by someone). Thus, we assume that any real-world domain can be represented in terms of the general abstract constructs from Bunge ontology, and that the ontological assumptions and postulates made in Bunge's ontology are applicable to any real world domain
2) Comparing selected constructs of Bunge's ontology to OWL constructs so as to propose a mapping between these Bunge constructs and the OWL constructs or their certain combinations. This would allow the assignment of ontological semantics to the OWL

[2] This thesis is based on the OWL functionality as described in the OWL Language Guide (McGuinness *et al.,2004)*

[3] We are going to use Bunge's ontology, as adapted and extended by Wand & Weber (1989) for use in information systems research. This ontological model of information systems has been termed Bunge-Wand-Weber ontology in subsequent research literature.

constructs (or their combinations) when they are used for representing real world domains.

3) Identifying key ontological assumptions and rules that govern the elements of the domain and their relationships according to Bunge's ontology
4) Transferring the relevant assumptions identified in step 3 by means of the mapping proposed in step 2, and proposing general and specific modeling guidelines, as well as the set of high-level domain-independent classes and properties (a meta-model) for modeling real world domains in OWL. The ontological assumptions become rules which restrict the use of the language constructs and limit the kind of statements that can be made about a real-world domain in OWL.

The applicability and the process of using the proposed guidelines will be illustrated by an example. In addition, some ideas will be proposed on how ontology development tools and environments can be enhanced to support the proposed guidelines and rules, in order to facilitate development of OWL ontologies that are consistent with the proposed guidelines.

1.6 THESIS STRUCTURE

The remainder of the thesis is structured as follows. Chapter 2 provides background theoretical information relevant to this thesis and discusses related prior research. Specifically, section 2.1 is devoted to IS ontologies in general, including definitions, types of ontologies, and their applications. Section 2.2 gives a brief overview of the OWL Web Ontology Language. Key concepts and premises of Bunge's ontology are introduced in section 2.3, and section 2.4 provides a review of related prior research work in the area of ontological analysis and IS ontologies.

Chapter 3 discusses in more detail the general methodology employed in this thesis. Chapter 4 contains the main theoretical analysis of selected concepts of Bunge's ontology and their representation in OWL. We start with the discussion of things (Section 4.1), followed by a detailed examination of properties, which includes general issues related to properties in Bunge's ontology and in OWL as well as issues specific to representation of intrinsic and mutual properties in OWL (Section 4.2). Classes and classification-related issues are the subject of Section 4.3. Next, Section 4.4 discusses a number of issues related to the representation in OWL of ontological premises that govern the relationships between the key ontological constructs (things, properties and classes). Finally, in Section 4.5 we focus on the representation of composition relationships between things.

Chapter 5 summarizes some of the outcomes of the main analysis. Specifically, it provides a summary of the proposed meta-model and presents the key modeling process

steps recommended for modelers intending to use the proposed approach. An example demonstrating the applicability and the process of applying the proposed rules and meta-model is presented in Chapter 6. Finally, Chapter 7 concludes the thesis by summarizing the main ideas and outcomes of the work and discussing potential future research directions.

2 THEORETICAL BACKGROUND AND RELATED RESEARCH

This section includes some theoretical background information and discussion of prior research relevant to this thesis. First, we provide some general information on IS ontologies (such as their definition, categorization, and applications). Next, we briefly discuss main aspects of the OWL language and introduce OWL concepts relevant to this work. Following that, we introduce the concepts of Bunge's ontology. Finally in this section, we provide an overview of related prior research work, which we categorize in two groups: 1) Bunge-Wand-Weber ontological foundations and their applications to ontological analysis of information systems analysis and design (ISAD) modeling languages, and 2) research and practical guidelines (such as tutorials, guides and best practices) related to ontology development, and to the OWL language in particular.

2.1 IS ONTOLOGIES – AN OVERVIEW

2.1.1 What are IS ontologies?

The term "ontology" has been in use for many years. The Merriam-Webster dictionary[4] provides two abstract, philosophical definitions of ontology: 1) a branch of metaphysics concerned with the nature and relations of being, and 2) a particular theory about the nature of being or the kinds of existents. In philosophy, Ontology is "that branch of philosophy which deals with the order and structure of reality in the broadest sense possible" (Angeles, 1981).

A new notion of "ontologies", which is different from the original philosophical concept of "ontology", has emerged relatively recently and has been gaining popularity among different research disciplines, such as artificial intelligence, knowledge engineering, knowledge representation, qualitative modeling, language engineering, database design, information retrieval and extraction, and knowledge management and organization (Guarino, 1998; McGuinness, 2002; Noy and Hafner, 1997; Uschold and Gruninger, 1996).

Ontologies have become even more important – not only among researchers but also among practitioners and businesses - with the advent and the widespread usage of the World Wide Web and the new vision of the Semantic Web, which was first put forward by Tim Berners-Lee (Berners-Lee *et al.,* 2001). The Semantic Web is a vision for the future of the Web, in which information is given explicit meaning, making it easier for machines to

[4] http://www.m-w.com/cgi-bin/dictionary?book=Dictionary&va=ontology

automatically process and integrate information available on the Web (Heflin, 2004). In these contexts, ontologies are often termed 'formalized ontologies' or 'information system ontologies' (or simply 'IS ontologies').

Various definitions for these ontologies have been proposed in the literature. Comparative analyses of different definitions can be found in several research works (e.g. Gomez-Perez et al., 2004; Guarino & Giaretta, 1995). One of the most widely cited definitions is that by Gruber (1993): "An ontology is a formal explicit specification of a shared conceptualization, which is a simplified view of the world". Fensel (2001) explains this definition as follows: the term "shared conceptualization" refers to an abstract model (or a view) of a set of phenomena or a domain of interest which is shared by a community of agents (people or computational agents); the word "formal" indicates that an ontology is expressed with the use of some formal notation, and "explicit" usually means that the precision of concepts and their relationships is clearly defined.

In simpler terms, an ontology is a formal description of the concepts and relationships that can exist in a domain as viewed and shared by a group of users (e.g. people or computational agents). An ontology often includes a hierarchical description of concepts in a domain, along with descriptions of the properties of each concept and the relationships between concepts. It may also contain instances of concepts. An ontology with individual instances is sometimes termed a 'knowledge base' (Noy & McGuinness, 2001).

We can summarize that ontologies:

- are used to describe a specific domain
- clearly define terms and relations in that domain
- use some formal mechanism to represent these concepts
- are agreed upon by users in such a way that the meaning of the terms is used consistently
- are usually build cooperatively by different groups of people in different locations

The main focus of IS ontologies is on machine-readability, i.e. on expressing a "specification of a shared conceptualization" explicitly in a form that can be "understood" and processed by computers. A number of markup languages (such as XML, RDF, RDFS) and ontology languages (such as OIL, DAML + OIL, and, more recently, OWL) have been developed to this end. Some of those languages have gained wide acceptance in supporting the goal of making the Web more accessible by computers. The available languages differ in terms of their expressiveness and inference mechanisms, and are based on diverse

knowledge representation paradigms (such as frames or description logics). Gomez-Perez *et al.* (2004) review and compare a number of languages available for implementing ontologies, and give some recommendations regarding the suitability of these languages for different purposes and areas of application.

2.1.2 Types of IS Ontologies

The word 'ontology' has been used to describe artifacts with different degrees of structure - from simple taxonomies (such as the Yahoo! hierarchy) to metadata schemes (such as the Dublin Core) to logical theories (Heflin, 2004).

McGuinness (2002) classifies ontologies along a linear spectrum based on the level of detail and formalization in their specification. This spectrum proceeds from "simple ontologies", such as controlled vocabularies, catalogs, terms/glossary, thesauri, and taxonomies with informal "is-a" hierarchies, to more complex, structured, ontologies with formal subclass hierarchies, frames, and value restrictions, and finally to very expressive ontologies that use first order logic constraints between terms and more detailed relationships to represent domain knowledge facts.

According to Gomez-Perez *et al.* (2004), the ontology community often uses the term *'lightweight ontologies'* for ontologies that are mainly taxonomies, and the term *'heavyweight ontologies'* for those ontologies that model a domain in a deeper way and provide more restrictions on domain semantics. Lightweight ontologies usually include concepts, concept taxonomies, relationships between concepts, and properties that describe concepts. Heavyweight ontologies add axioms and constraints to lightweight ontologies to clarify the intended meaning of the terms in the ontology.

Heavyweight and lightweight ontologies can be modeled with different knowledge modeling techniques and can be implemented in a variety of languages (Uschold & Gruninger, 1996). As a result, ontologies differ in their degree of formalization and structure. Ontologies can be classified into *highly informal* if they are expressed in natural language, *semi-informal* if expressed in a restricted and structured form of natural language, *semi-formal* if expressed in an artificial and formally defined language, and *rigorously formal* if they provide meticulously defined terms with formal semantics, theorems and proofs of properties such as soundness and completeness (Gomez-Perez *et al.,* 2004).

The OWL language, which is the focus of this thesis, allows representation of both heavyweight ontologies and lightweight ontologies (depending on the language version used – OWL Lite, OWL DL, or OWL Full). OWL is based on a rigorous knowledge representation formalism (Description Logics), is formally defined, and is assigned clear

model-theoretic semantic. Thus, with respect to the degree of formalization, OWL ontologies should be classified as (at least) *semi-formal.*

2.1.3 Applications of IS ontologies

Ontologies can support a great variety of tasks in diverse research and application areas. Numerous applications of IS ontologies have been proposed and implemented in research prototypes and industry applications. Ontologies enable knowledge sharing and reuse where information resources can be communicated between human or software agents. Using ontologies, ontology-based tools can perform automated reasoning, and thus provide advanced services to intelligent applications such as conceptual/semantic search and retrieval, software agents, decision support, speech and natural language understanding, knowledge management, intelligent databases and electronic commerce (Heflin, 2004).

McGuinness (2002) describes various benefits and applications both of simple ontologies and of complex structured ontologies. An important advantage of simple ontologies is that they are not costly to build and many are already available in various forms. Even simpler ontologies can provide controlled vocabularies and can be used for website organization, navigation and browsing support, for search support (such as query reformulation and disambiguation), and sense disambiguation support. More complex structured ontologies can, in addition, provide a basis for inference and thus can be used in applications to enable consistency checking, interoperability support, semantic integration of heterogeneous information sources, and validation and verification of data. They can also support structured, comparative and customized information searches and exploit generalization/ specialization hierarchies.

Ontologies play a key role in the emerging Semantic Web, providing a way of representing the semantics of documents and enabling these semantics to be used by web applications and intelligent agents. The OWL use cases and requirements document (Heflin, 2004) discusses six representative use cases of web ontologies in such application areas as web portals, multimedia collection management, corporate website management, design documentation management, intelligent agents and services, and ubiquitous computing. These selected examples clearly demonstrate the potential and real benefits and usefulness of ontologies on the Web.

2.2 OWL WEB ONTOLOGY LANGUAGE

2.2.1 General information

In this section, we provide a brief discussion of key constructs and features of the OWL Web ontology language[5]. This OWL overview is based on the official OWL documentation from the World Wide Web Consortium (W3C) (Bechhofer *et al.,* 2004; Heflin, 2004; McGuinness *et al.,* 2004), as well as on several other available OWL guides and tutorials (Antoniou & van Harmelen, 2004; Horridge, 2004). Please refer to these sources for more detail on OWL syntax and functionality. Another excellent source of information on OWL is the W3C OWL website (http://www.w3.org/2004/OWL\) which provides links to various OWL resources.

The OWL Web Ontology Language is the most recent development in standard ontology languages from the World Wide Web Consortium (W3C). It is intended to enable publishing and sharing IS ontologies on the Web. OWL is based on RDF (Resource description Framework) and RDF Schema (RDFS), which are widely accepted as formal languages of meta-data describing any web resources. As an extension of RDF/RDFS, OWL uses some basic elements of RDF/ RDFS (such as rdf:subclassOf, rdfs:domain, etc.). It also provides constructs for defining and characterizing classes and properties of those classes, and for defining individuals and asserting properties about them. OWL language has a formal logical model and formal semantics which allow reasoning about classes, individuals, and their properties.

The OWL language is divided into the three increasingly expressive sublanguages: OWL Lite, OWL DL and OWL Full. OWL Lite is the least expressive sublanguage. It supports only a subset of OWL constructs and is intended to be used in situations where only a simple class hierarchy and simple constraints are needed. The OWL DL sublanguage is based on description logics and supports the maximum expressiveness without losing decidability and computational completeness[6] of reasoning systems. Tool builders have already developed powerful reasoning systems (based on description logics) which support ontologies constrained by the restrictions required for OWL DL. The complete OWL language, OWL Full, supports the same set of the OWL language constructs as OWL DL, but relaxes some of the constraints on OWL DL to provide maximum expressiveness and the syntactic freedom of RDF. However, it does not guarantee decidability or computational

[5] http://www.w3.org/2004/OWL\

[6] *Computational completeness* means all entailments are guaranteed to be computed. *Decidability* means all computations will finish in finite time.

completeness. One of the distinctions between OWL DL and OWL Full is that OWL DL requires disjointness of classes, properties, individuals and data values, i.e. in OWL DL, for example, a class cannot be at the same time an individual (or vice versa). The focus of this thesis is on OWL DL (unless noted otherwise).

Key OWL constructs are *classes*, *individuals* and *properties*. An OWL document consists of optional ontology headers plus any numbers of class axioms, property axioms, and facts about individuals. In the following sections we discuss these concepts in more detail.

2.2.2 OWL classes and individuals

Classes in OWL are intended to represent concepts in a domain of discourse. They provide an abstraction mechanism for grouping resources with similar characteristics. Every OWL class is associated with a set of *individuals* called the *class extension*.

OWL individuals represent objects in the domain of discourse. The individuals in the class extension are called *instances* of the class. Generally, it is intended that classes should correspond to naturally occurring sets of things in a domain of discourse and individuals should correspond to actual entities that can be grouped into these classes. For example, we can define a class Book with instances of this class (OWL individuals) representing some specific books.

Two OWL class identifiers are predefined: owl:Thing and owl:Nothing. The class extension of the owl:Thing class is the set of all OWL individuals; thus, every OWL class is a subclass of owl:Thing. The class extension of owl:Nothing is the empty set; so owl:Nothing is a subclass of every class.

The simplest way to define a class in OWL is just to declare it by name, for example:

```
<owl:Class rdf:ID="Human">
```

This definition is sufficient to allow, for example, the declaration of some OWL individuals to be instances of this class. OWL individuals are defined with individual axioms (also called '*facts*'). Facts about individuals in OWL include facts about class membership, facts about property values of individuals, and facts about individual identity (which assert whether individuals are same or different[7]). An individual can be minimally introduced by being declared a member of a class (either of the predefined top class owl:Thing or some other

[7] Unlike many languages, OWL does not have a "unique names" assumption. That is, in OWL, even if individuals have different names they can still be the same. OWL provides several constructs to make explicit statements regarding whether individuals are the same or different.

class defined in an ontology), for example:

```
<owl:Thing rdf:ID="SomeThing">
<owl:Human rdf:ID="John_Smith">
```

In the above example, the first statement introduces an individual SomeThing simply as an instance of owl:Thing (no further information about this individual has beeen provided yet). The second statement declares another individual, John_Smith, which is stated to be an instance of the class Human (note that this individual is automatically an instance of owl:Thing since any OWL class is the subclass of owl:Thing).

Declaring a class only by declaring its name does not provide much information about the class (other than its name). In general, OWL classes are further defined through *class descriptions,* which can be combined into *class axioms*. A class description describes an OWL class either by name (as was shown above) or by specifying the class extension (set of instances) of an unnamed (anonymous) class.

Defining classes in OWL by specifying the class extension means describing the conditions that must be satisfied by an individual for it to be a member of the class. For example, a class in OWL can be described

- by exhaustive enumeration of its individuals (using owl:oneOf construct for stating that the extension of a class consists of these and only these listed instances)
- as a set of all individuals which satisfy certain constraints on their properties (*property restrictions*).

Classes can also be defined as Boolean combinations of two or more class descriptions - union, intersection or complement - using the constructs owl:unionOf, owl:IntersectionOf, and owl:complementOf, respectively.

Class axioms contain components that state necessary and/or sufficient characteristics of class membership. OWL provides three language constructs for combining class descriptions into class axioms: rdfs:subClassOf, owl:equivalentClass, owl:disjointWith.

Using rdfs:subclassOf construct, classes in OWL can be organized into superclass-subclass hierarchies. OWL allows multiple inheritance - an individual can be an instance of different classes and a class can be a subclass of several other classes (this can be declared and/ or inferred based on formal semantics).

2.2.3 OWL properties

All OWL properties are binary relationships. They are used to assert general facts

about class instances and specific facts about individuals. There are two main types of properties in OWL[8]:

- *Object properties* relate individuals to individuals. For example, in some ontology describing people we can define an object property hasParent to relate individuals representing persons to other individuals - their parents.
- *Datatype properties* link individuals to data values (an XML schema datatype value or an RDF literal). For example, we may define a datatype property hasAge to represent the age of a person, i.e. to link an individual (person) to a nonnegative integer representing age.

Note, that properties in OWL have direction: a property links *a subject* (an OWL individual) to *an object* (an OWL individual or a data value), and the object is considered a value of this property for the subject. For readability, a predicate notation P(x,y) is often used to show that a pair (x,y) is linked by some property P, meaning that P is a property of an individual x with a value y (which is either an individual or a data value). A term *'property extension'* is sometimes used (in a similar fashion to 'class extension') to denote the set of (directed) subject-object pairs that are associated with a particular property.

Properties in OWL are described using *property axioms*. In its simplest form, a property axiom just declares the existence of a property by its name, for example

```
<owl:ObjectProperty rdf:ID="hasParent"/>
```

Properties may have a domain and/or a range specified (using class descriptions and XMLS schema datatypes for datatype property range). For example, in some ontology we may need to specify that the domain of a property hasAge is the class Human and the range is a set of nonnegative integers (represented as an XML Schema datatype):

```
<owl:Class rdf:ID="Human">

<owl:DatatypeProperty rdf:ID="hasAge">
  <rdfs:domain rdf:resource="#Human"
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonnegativeInteger"/>
</owl:DatatypeProperty>
```

It is important to remember, however, that domains and ranges in OWL are not

[8] OWL also has other types of properties, such as owl:AnnotationProperty and owl:OntologyProperty. They are intended for specific purpose of adding annotation information (metadata) to classes, individuals, object/ datatype properties and the ontology itself. The features of those property types are limited (compared to object/datatype properties), and these properties are usually ignored in reasoning. Thus, these properties are not considered in this study due to their specific purpose.

viewed as constraints to be checked but rather they are used as 'axioms' in reasoning. In particular, they allow the inference that an individual belongs to a class declared as some property domain (or range) based on the fact that an individual possesses that property. For example, for the hasAge property (introduced earlier), we could also assert that some individual ThingX possesses this property, e.g.:

```
<owl:Thing rdf:ID="ThingX">
   <hasAge rdf:datatype="http://www.w3.org/2001/XMLSchema#nonnegativeInteger">25</hasAge>
</owl:Thing>
```

Based on the above, most reasoning tools would infer that ThingX is an instance of the class Human (since this class is the domain of the property hasAge). Thus, domain and range constraints should be used with caution since they may lead to unintended implications.

It is important to note that properties in OWL are defined independent of classes. This means that one does not have to define classes to be able to define properties, and vice versa. By default, a property is assumed to be a binary relationship linking two individuals of the predefined class owl:Thing or an individual of the class owl:Thing and a data value. Thus, in general, any individual can (but does not have to) possess any property defined in an ontology, i.e. can have an arbitrary number (zero or more) of values for a particular property.

OWL provides a number of constructs to describe additional characteristics of properties, for example:

- owl:SymmetricProperty: $P(x, y) \Leftrightarrow P(y,x)$ (e.g. "is a sibling of")
- owl:TransitiveProperty: $P(x,y) \& P(y,z) \Rightarrow P(x,z)$ (e.g. "is taller than")[9]
- owl:FunctionalProperty: states that a property has at most one unique value for each individual (e.g. "age" property)
- owl:InverseFunctionalProperty: defines a property for which two different individuals cannot have the same value (e.g. "social security number")

OWL also allows the description of certain relationships between properties using the following constructs:

- rdfs:subpropertyOf: properties can be arranged in property-subproperty hierarchies. P is a *subproperty* of Q iff $P(x,y) \Rightarrow Q(x,y)$ (i.e. the property extension of P is a subset of the property extension of Q). P and Q should be either both datatype or both object properties.

[9] In OWL DL only object properties can be declared symmetric or transitive

- owl:EquivalentProperty: two properties P and Q can be declared *equivalent* which would mean they have the same property extension (set of pairs), i.e. $P(x,y) \Leftrightarrow Q(x,y)$
- owl:InverseOf: an object property may have a corresponding inverse property[10]. For example, we can declare that the property hasParent has an inverse property hasChild, which would mean the implication hasParent(X,Y) <=>hasChild(Y,X)

2.2.4 Property restrictions and class descriptions

As mentioned earlier, classes in OWL can be defined by specifying the class extension, i.e. by describing the conditions that must be satisfied by an individual for it to be a member of the class. One way to do that is to use *property restrictions*. A property restriction describes an *anonymous (unnamed) class* of all individuals that satisfy certain constraints on a property. Property restrictions are used as parts of class descriptions.

OWL has two types of property restrictions: value constraints and cardinality constraints. A *value constraint* puts constraints on the range of the property when applied to a particular class description (within the scope of a particular class axiom). A *cardinality constraint* puts constraints on the number of values a property can take, also in the context of a particular class description.

The following constructs can be used to specify value constraints:

1) owl:allValuesFrom (universal qualifier) – is used to specify the class of possible values (or a data range) that the property specified in a property restriction can take. For example, we can describe an anonymous OWL class of all individuals for which the hasParent property can only have values of the class Human[11]:

```
<owl:restriction>
        <owl:onProperty rdf:resource="#HasParent" />
        <owl:allValuesFrom rdf:resource="#Human" />
</owl:Restriction>
```

This restriction can be used in class descriptions. For example, it can be stated that the class Human is a subclass of the anonymous class of individuals that can have only human parents (if any). In other words, this would mean that humans can only have

[10] In OWL DL only object properties (not datatype properties) can have inverse properties

[11] Note that this does not imply that the property hasParent can only have values from this class Human. There may be individuals that possess the property hasParent but have values for this property that are not instances of the class Human. Such individuals will not be members of this anonymous class, however. Also note, that in OWL this constraint on the property is trivially satisfied if an individual has no values for the property P at all, thus this class includes individuals which have no parents at all.

parents who are also humans.

2) owl:someValuesFrom (existential qualifier) - is used to specify the existence of at least one value from a specified class for the property specified in the restriction. For example, we may specify an unnamed class of individuals who have at least one parent who is a student:

```
<owl:restriction>
        <owl:onProperty rdf:resource="#hasParent" />
        <owl:someValuesFrom rdf:resource="#Student" />
</owl:restriction>
```

3) Owl:hasValue constraint - is used to describe a class of all individuals for which the specified property has at least one value semantically equal to the specified value. For example, we may describe a class of individuals that have the individual referred to as Smith as their parent:

```
<owl:restriction>
        <owl:onProperty rdf:resource="#hasParent" />
        <owl:hasValue rdf:resource="#Smith" />
</owl:restriction>
```

As for the cardinality restrictions, in general it is assumed in OWL that any instance can have an arbitrary number (zero or more) of values for a particular property. OWL provides three cardinality restriction constructs for specifying constraints on a number of semantically distinct values for a property (within a context of a particular class description): owl:minCardinality, owl:maxCardinality, owl:Cardinality. For example, we can describe an anonymous class of individuals that have at least 1 and at most 4 children (using minimum and maximum cardinality restrictions on hasChild object property):

```
<owl:restriction>
  <owl:onProperty rdf:resource:='#hasChild"/>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonnegativeInteger">1
    </owl:minCardinality>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonnegativeInteger">4
    </owl:maxCardinality>
</owl:restriction>
```

Anonymous classes (such as property restrictions described above) can be used in class axioms in a variety of ways. For example, we can state that a particular class C is a

subclass of an anonymous class C1 defined as a property restriction (which would mean that all instances of the class C satisfy this restriction). We can also state that a certain class is equivalent to or disjoint with another class (anonymous or named) using owl:equivalentClass or owl:disjointWith constructs, respectively. Finally, both anonymous and named classes can be used in class definitions with Boolean operators on classes (owl:intersectionOf, owl:unionOf, owl:complementOf), which can be arbitrarily nested. Thus, it is possible in OWL to combine anonymous classes (property restrictions) and named classes in a variety of ways to create complex class definitions. By describing and combining classes, properties, and individuals using constructs and mechanisms provided in OWL, one can represent knowledge about a particular domain. In later sections, more detail and usage examples are provided for the OWL constructs and mechanisms relevant to this thesis.

2.2.5 Reasoning

OWL has a well-defined syntax and formal semantics, which allow applications to make inferences and provide automatic reasoning support. Antoniou & van Harmelen (2004) list the main aspects that can be reasoned about for ontological knowledge:

- *class membership*: for example, if x is an instance of a class C, and C is a subclass of D, then it can be inferred that x is an instance of D;
- *equivalence of classes*: for example, if classes A and B are equivalent and classes B and C are equivalent, then we can infer that A is equivalent to C;
- *consistency:* if, for example, we declared that x is an instance of A, and that A is a subclass of B, A is a subclass of C, and B and C are disjoint, then inconsistency in the ontology can be detected (since A should be empty but it also has the instance x),
- *classification*: if it is declared that certain property-value pairs are sufficient condition for membership of a class A, then if an individual x satisfies such conditions, it can be concluded that x is an instance of A.

Description logic is a subset of the predicate logic, for which efficient reasoning support is possible. For OWL DL ontologies, derivations such as above can be performed automatically by reasoning applications (such as FaCT[12] or RACER[13]). Reasoning support allows one to check the consistency of an ontology and of the represented knowledge, to check for unintended relationships between classes, to classify instances in classes automatically, and so on. Automated reasoning support is invaluable for designing large ontologies (where multiple modelers are involved), for integrating and sharing ontologies

[12] http://www.cs.man.ac.uk/~horrocks/FaCT/
[13] http://www.sts.tu-harburg.de/~r.f.moeller/racer/

from various sources, and for ontology-based applications. More specific examples of reasoning that can be performed based on OWL ontologies can be found in Bechhofer (2003).

2.3 BUNGE-WAND-WEBER ONTOLOGICAL FRAMEWORK

As mentioned in the introduction, following Bera & Wand (2004) and other research on ontological expressiveness of modeling languages (Wand & Weber, 1990a,b, 1993, 1995; Wand et al., 1999), this thesis applies a particular philosophical framework - Bunge's ontology (Bunge 1977, 1979) – to develop a representational mapping (between ontological and language constructs), a meta-model (a high-level class/property structure), and ontologically grounded modeling guidelines with the intent to improve the representation of real world domain semantics in OWL. This section briefly introduces some key concepts of Bunge's. More complete description of Bunge's ontology can be found elsewhere (e.g. Bunge, 1997, 1979; Wand & Weber, 1990a,b, 1993, 1995). The relevant concepts of Bunge's ontology are also discussed in more detail in other sections of the thesis as part of the analysis.

Bunge's ontology describes a set of high-level constructs that are intended to represent real world phenomena. In this work, we are using Bunge's ontology as adapted and extended by Wand & Weber (1990a,b, 1993, 1995) for the use in information systems research (sometimes referred to as the Bunge-Wand-Weber ontology, or 'BWW ontology'). Bunge's ontology has been chosen as a basis for this research for a number of pragmatic reasons: 1) it is based on solid philosophical foundations, is comprehensive and well formalized; 2) it has been adapted and extended for information systems (Wand & Weber 1990a,b, 1993, 1995; Wand *et al.,* 1995; Weber 1997); 3) it has been successfully applied to evaluation of modeling languages (e.g. Evermann & Wand (2001a,b, 2005); Green & Rosemann, 2000; Wand & Weber, 1993; Wand *et al.* 1999); 4) it has also been empirically shown to lead to useful outcomes (e.g. Burton-Jones & Weber, 1999, 2003; Bodart *et al.* 2001; Evermann, 2003; Gemino & Wand, 2000; Parsons & Cole, 2004). Numerous prior applications of Bunge's ontology in IS research are discussed later in section 2.4.1.

The fundamental concepts of Bunge's ontology are summarized in Table 1. Of these concepts, the most relevant ones for this thesis are the concepts of thing, property, class, kind, composition, attribute, functional schema, and interaction. These concepts will be discussed in more details in further sections of the thesis.

Table 1: Selected concepts of Bunge's ontology (adapted from Wand & Weber, 1993; Bera & Wand, 2004)

Concept	Description
Things	A thing is the elementary unit in the ontological model. The world is made of things. A distinction is made between *concrete (or substantial) things* (e.g. a book) and *conceptual things* (e.g. a mathematical set). It is assumed that any domain can be described by concrete things and the linkages between them
Properties	All things possess properties. A property that is inherently a property of an individual thing is called an *intrinsic property*. A property that is meaningful only in the context of two or more things is called a *mutual property*. For example, height is an intrinsic property of a person and salary is a mutual property between a person and a company. Properties *in general(or generic properties)* are those properties possessed by a set of things (e.g. "color"); properties *in particular (individual, or specific, properties)* are properties that can be represented as the value of a property in general (e.g. "blue in color")
Composition	A *composite* is a thing that is made up of other things. Composites possess *emergent properties* – properties not inherited from their components. For example, a computer has a property "processing power" not possessed by any of its components individually.
Law	Ontology postulates that things satisfy some *laws*. Laws are defined in terms of relations between properties. A particular form of law is *precedence*: property A precedes property B iff whenever a thing possesses B, it also possesses A. *A (state) law* is a restriction on the possible values of the components of a functional schema of a thing or their combinations.
Class, kind, and natural kind	Things can have one or more properties in common. A *class* is a set of things possessing a common property. A *kind* is a set of things possessing more than one common property. A *natural kind* is a set of things adhering to the same laws (which implies a set of properties as well since, by definition, laws relate properties)
Attribute	A property is modeled via an attribute function that maps a set of things into a set of values at a given time
Functional schema	Humans conceive of things in terms of models of things. Similar things can be represented by the same model. A *functional schema* is a formalization of *a view* of a set of similar things in terms of a set of attribute functions. For example, a person may be viewed as an employee, a customer or a taxpayer. Each view is modeled as a different set of attribute functions (usually functions of time).
State	The *state* of a thing is the vector of values for all attribute functions in a schema of a thing at a given time
Event	An *event* is a change of state of a thing. It is affected via a transformation (see below)
Transformation	A *transformation* is a mapping of a set of states into itself
History	The *history* of a thing is the chronologically ordered states that a thing traverses. Example – history of positions of an employee over a period of time
Interaction	*Interaction* is the ability of a thing to change the "history" (states traversed) by another thing. The two things are said to *interact* or to be *coupled*. Interactions usually give rise to mutual properties. The existence of interaction can be considered a mutual property of things, and conversely, the existence of a mutual property can indicate an interaction.

2.4 RELATED PRIOR RESEARCH

In this section we provide a condensed overview of the prior research relevant to the thesis topic, specifically in the areas of ontological analysis and development of IS ontologies. Prior research related to this thesis can be categorized in the following groups:

1) research on the ontological foundations for IS research and application of those foundations to ontological analysis of some information analysis and design (ISAD) modeling languages,
2) research and practical guidelines (such as tutorials, guides and best practices) related to ontology development and applications, and to the OWL language in particular.

The next two sections discuss the prior work in each of these areas in more detail.

2.4.1 Ontological foundations for IS research and their applications

The stream of the research work that applies philosophical ontology to IS research was initiated by Wand and Weber (1989, 1990a,b). Starting from the fundamental premise that an information system is intended to be a representation of some perceived real world system, they proposed to turn to a philosophical ontology which provides a set of constructs to describe a generalized view of reality. They adopted a specific philosophical ontological framework, Bunge's ontology (Bunge 1977, 1979), and adapted and extended it for use in information systems research. Based on Bunge's ontology, Wand & Weber (1990a, 1993) proposed a set of the ontological models of information systems, one of which – the *representation model* - defines a set of core concepts that can be used to describe the structure and behavior of information systems. This model has been termed in subsequent research literature as *Bunge-Wand-Weber ontology* (or BWW ontology) and has found a number of applications in IS research since then.

One of the most popular applications of the BWW ontology is the evaluation of conceptual modeling techniques and languages. The ontological analysis method is based on the notion of the *ontological expressiveness* proposed by Wand and Weber (Wand & Weber, 1993). The method is used for the evaluation of conceptual modeling grammars with respect to their capability to represent properly the elements of real-world systems, and for the identification of potential representational deficiencies. This approach is based on exploring mappings between the set of ontological concepts and the grammar's constructs, so as to evaluate modeling grammar's ontological completeness and clarity.

A grammar is *ontologically complete* if and only if every ontological construct can be mapped to a grammar construct (i.e. the mapping is total). A grammar is *ontologically clear*

if and only if no two ontological concepts are mapped onto the same grammar construct, and all grammar constructs can be mapped to ontological concepts (Wand & Weber, 1993). Ontological clarity can be undermined by such deficiencies as *construct overload* (if one grammar construct represents more than one ontological construct), *construct redundancy* (if more than one grammar construct represents the same ontological construct), and *construct excess* (when a grammar construct is present that does not map into any ontological construct). Through the mapping between ontological and grammar constructs, the ontological semantics can be assigned to a language, which also helps to derive modeling guidelines prescribing how language constructs should be used to model elements of a real-word domain to improve ontological clarity and completeness.

The ontological analysis approach has been employed for the evaluation of several systems analysis and design and conceptual modeling languages, such as data flow diagrams (Wand & Weber, 1993), NIAM (Weber & Zhang, 1996), Entity-Relationship (ER) diagrams (Wand *et al.*, 1999), ARIS (Green & Rosemann, 2000), and more recently – UML (Evermann, 2003; Evermann & Wand (2001a,b); Evermann & Wand, 2005).

Other applications of the BWW ontology in IS research include the analysis of the object concept in object-oriented modeling (Wand, 1989), clarifying the notion of data quality dimensions by anchoring them in ontological foundations (Wand and Wang, 1996), developing guidelines for choosing classes in conceptual modeling (Parsons & Wand, 1997), proposing ontologically grounded two-layer data model in which instances are allowed to exist separately of classes (Parsons & Wand, 2000), and applying the ontological concept of property precedence to the semantic reconciliation of heterogeneous data sources (Parsons & Wand, 2004).

In addition to the theoretical research, a number of empirical studies have been conducted, aiming to test theoretical propositions derived from the applications of Bunge's ontological foundations to the evaluation of modeling grammars. In general, these studies investigated whether the use of mappings and guidelines derived based on the ontological analysis leads to better models, specifically, the models that are better understood and are more useful for model designers or users. Some specific issues tested in those studies included the impact of using relationships with attributes in ER modeling on users' problem-solving performance (Burton-Jones & Weber, 1999, 2003); the use of optional properties compared to the use of subclassification with only mandatory properties (Bodart et al., 2001; Gemino, 1998; Gemino & Wand, 2000); an experimental evaluation of representing property precedence in conceptual modeling (Parsons & Cole, 2004); an experimental evaluation of

the specific benefits to domain understanding induced by UML models developed in conformance with the ontologically grounded modeling rules (Evermann, 2003).

Overall, the results of the empirical studies in this area have been encouraging. The studies provided support to the idea that models built in accordance with the guidelines and recommendations developed based on Bunge's ontological foundations lead to better models especially in cases when more in-depth (rather than surface-level) understanding of the models was required.

To summarize, a lot of prior research work in this field has focused on the ontological evaluation of languages. This thesis continues the research in this area and applies the ontological analysis method to an ontology development language – OWL. Bera and Wand (2004) conducted an initial ontological analysis of the OWL language, highlighted a number of weaknesses of OWL with respect to the ontological clarity and completeness, and proposed a high level mapping between OWL constructs and ontological constructs as well as some general guidelines on how the clarity and completeness of OWL models can be improved. This thesis follows up on and expands the work started by Bera and Wand, aiming to employ the results of the analysis and the mapping between the ontological and OWL constructs in a constructive and prescriptive way - to develop specific ontologically grounded guidelines on how to use OWL language in an ontologically better way for modeling real world domains.

2.4.2 Prior work on IS ontology development methodology and guidelines

IS ontologies have been receiving ever increasing attention in various research disciplines, including artificial intelligence, computer science, knowledge management, Semantic Web research, and others. Relevant to this thesis topic is the research related to the certain aspects of ontological engineering. A term 'ontological engineering' is used to refer to the set of activities that pertain to the ontology development process, the ontology lifecycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them (Gomez-Perez *et al.,* 2004).

The focus of this thesis is mainly on the conceptual aspects of modeling real world domains with OWL ontologies (rather then on technical, implementation or application issues). Therefore, in this section we conscentrate on prior work that is concerned more with conceptual modeling issues and is relevant to OWL. For more detailed discussion of the state of the art in the ontological engineering field, various ontology development languages, methods and methodologies for ontology construction, and other issues in the area of ontological engineering, we refer the readers to the book by Gomez-Perez *et al.,* 2004

(which, in turn, provides plenty of other references).

A number of research publications on IS ontologies (e.g. Chandrasekaran *et al.*, 1999; Gruber, 1993; Gruninger & Fox, 1995; Guarino & Giaretta, 1995; Uschold & Gruninger, 1996) is concerned with the general issues related to ontologies such as what is an ontology and what is not, why ontologies should be created, how they can be used, and what criteria should be followed to build an ontology. Other research work (e.g. Guarino, 1998; Smith, 2003) is focused mainly on the philosophical aspects of ontologies and ontological commitment.

Yet other ontology research focuses on the development of the so called *upper-level, or top level, ontologies* – i.e. ontologies which describe very general concepts that are common across the domains and give general notions under which all the terms in the existing ontologies should be modeled. Among such upper level ontologies are the top level ontologies of universals and particulars (Guarino & Welty, 2000; Gangemi et al. 2001), Sowa's top-level ontology (Sowa, 1997), Cyc's upper ontology (Lenat & Guha, 1990), and SUO/ SUMO (Pease & Niles, 2002). The topic of this thesis is (to some extent) related to the upper-level ontology research since one of the thesis objectives is to propose (based on a general philosophical ontology) certain upper-level classes and properties to be recommended for inclusion into every OWL ontology representing a real world domain. However, this thesis work is different from other research on upper-level ontologies since the thesis does not attempt to develop a comprehensive upper-level ontology, but rather uses an established philosophical ontology to inform IS ontology development. While the thesis proposes certain upper-level classes and properties for OWL ontologies modeling real world domains, the intent is to keep the suggested 'upper-level' structure relatively simple and intuitively clear so that it could be used by domain modelers more easily. Other formal upper-level ontologies (as those mentioned above) are much more comprehensive and demand more background knowledge and experience from modelers (and thus maybe harder to apply).

There also exists a number of IS ontology development publications that adopt a more pragmatic (rather than theoretical or philosophical) perspective and that are intended to provide more specific guidance to ontology developers (both researchers and practitioners). For example, a popular ontology development guide by Noy and McGuinness (2001) discusses an ontology development methodology for declarative frame-based systems (using as an example the well popularized wine ontology and the Protégé-2000 ontology development environment). This paper proposes the steps in the ontology development

process and addresses some complex issues pertaining to the definition of class hierarchies and properties of the classes. Other research publications propose alternative ontology development methodologies (e.g. Gomez-Perez, 1998; Gruninger and Fox, 1995; Uschold & Gruninger, 1996).

As for the OWL related publications, not much theoretical research on the methodologies for developing OWL ontologies is available (partly due to the relatively recent introduction of the language). Many existing theoretical research papers on OWL are not focused on guidelines for modelers on how develop conceptual models in OWL ontologies, but rather are concerned more with model-theoretic or technical issues such as the underlying formal logical model of OWL, formal reasoning and machine readability aspects of OWL ontologies (Carroll & De Roo, 2004; Horrocks *et al.,* 2003; Patel-Schneider *et al.* 2003), reasoners and querying capabilities for OWL ontologies (e.g. V. Haarslev & R. Möller, 2003; Haarslev et al., 2004), and formalized extensions to OWL (e.g. Horrocks & Patel-Schneider, 2004; Horrocks et al. 2005; Fikes *et al.* 2004).

Among a limited number of publications discussing representational guidelines for OWL modelers, are the several documents from the World Wide Web Consortium (W3C), which aim to develop and promote best practices for the OWL community[14]. These documents discuss some frequently arising modeling issues and needs in ontology development, suggest possible modeling patterns, and discuss some advantages and disadvantages of those patterns. Some discussed modeling issues include: modeling N-ary relations in OWL (Noy & Rector, 2004), modeling part-whole relationships (Rector & Welty, 2005), modeling specified values (value partitions and value sets) in OWL (Rector, 2004). To the best of our knowledge, we reviewed most of the existing OWL tutorials and best practice documents (currently available from the OWL W3C website[15] as well as from other publication sources) in an attempt to keep up-to-date with OWL functionality and conceptual modeling issues discussed within the OWL community.

[14] Some of these documents are working drafts and are still being developed collaboratively by members of W3C consortium (Semantic Web Best Practices and Deployment Working Group).
[15] http://www.w3.org/2004/OWL/

3 METHODOLOGY

As briefly discussed earlier, the key objective of this thesis is to develop, based on a specific philosophical ontology (Bunge's ontology), some ontologically grounded modeling guidelines and a high-level class structure (a meta-model) that would facilitate the development of ontologically clear, consistent, and expressive OWL ontologies of real world domains. The methodology we use is based on the theoretical foundations for information systems analysis and design developed by Wand and Weber (Wand & Weber 1990, 1993, 1995) and has been employed by various researchers in many subsequent studies in the conceptual modeling and ISAD areas (as discussed in section 2.4.1). One of the most recent applications of the approach (and most similar to the one used in this thesis) is the research conducted by Evermann and Wand (Evermann, 2003; Evermann & Wand, 2001a,b, 2005) on the ontological analysis of UML and the use of UML for conceptual modeling.

In the introduction we briefly presented the key steps comprising our analysis. Here, we discuss the process in more detail. Specifically, the approach is as follows. In order to develop theoretically-grounded modeling guidelines and a meta-model for representing real world domains in OWL, we make an ontological commitment to Bunge's ontology. In other words, we assume that constructs and assumptions of Bunge's ontology provide sufficient descriptive power for representing phenomena of any real world domain.[16] Under this assumption, we consider a subset of the ontological constructs from Bunge's ontology (limiting the focus of this work mainly to the static aspects of the world) and propose how they can be modeled using OWL constructs (or their combinations) so as to preserve the ontological assumptions and to ensure that relationships among the ontological constructs are reflected in their representations in OWL ontologies. This analysis results in a mapping between the ontological and the language constructs.

Such a mapping allows the assignment of ontological semantics to OWL language constructs (or their combinations). In addition, such mapping can be used to transfer certain ontological assumptions to the language. As pointed by Evermann (2003), an established high level domain ontology (such as Bunge's ontology) may suggest that certain situations are possible in the real world while others are not. By virtue of mapping between modeling language constructs and ontological constructs, some combinations of language elements may therefore describe possible real world situations while others may describe impossible

[16] To be more specific, by 'real world' we mean beliefs on what exists, might exist or happen as perceived by a community.

ones. To achieve a proper representation, rules or constraints that relate ontological concepts (ontological assumptions), by virtue of the mapping, must also hold for the mapped language constructs in order to allow only models of possible real-world situations. Thus, we can transfer the ontological rules and constraints to a modeling language (like OWL) by creating the respective modeling rules and guidelines which specify how language constructs should be used to model elements of real world domains. Thus, we can limit the kinds of statements that can be made about real world domains in OWL to ensure that the ontological assumptions are preserved in the resulting OWL ontologies and that the language constructs have well-defined semantics.

Encouraged by the prior successful applications of the above approach to other modeling languages (ER, UML and others) and some empirical support (as discussed in section 2.4.1), we believe that the mapping and the guidelines resulting from our analysis can potentially lead to better representations of real world domains in OWL ontologies. If modelers are guided by the proposed mappings and follow the proposed ontologically grounded modeling rules and guidelines, then they would be more likely to develop more expressive, clear, consistent, and stable OWL ontologies of real world domains and would be less likely to model situations that are not reflected in a real world domain or to represent the same domain elements using different language constructs in different ontologies. Since we assume that the constructs of Bunge's ontology allow the proper representation of domain knowledge, then if we are able to represent those constructs in OWL in a consistent and clear way, we can expect the resulting OWL ontologies also to represent the domain more adequately and be more stable (from person to person and over time).

Bera and Wand (2004) note some potential problems with OWL ontologies when modeling real world domains including potential inconsistency in domain fact representation and interpretation, difficulty in modeling some domain information, and instability of OWL ontologies developed from domain information. These problems may arise due to a lack of the well-defined real world ontological semantics in OWL, and can also be attributed to the fact that the OWL language is not specifically intended for modeling real world domains. Rather, OWL is intended as a general purpose web ontology language with basic constructs (which allow the representation of various concepts and their properties) and a number of operators for describing these constructs and relationships between them, with a main focus on machine readability and reasoning (thus OWL focuses on formal logical semantics rather then real world ontological semantics). The approach in this thesis, focused specifically on the representation of real world domains in OWL and grounded in philosophical ontological

foundations, is intended to help modelers to avoid or alleviate the earlier mentioned problems when modeling real world domain, without modifying or extending the existing OWL language.

The scope of this thesis is limited a subset of ontological constructs.. The selected constructs (such as things, properties, classes, interaction, composites) mainly describe the static structure of the world. The representation of other constructs of Bunge's ontology (such as states or events), while briefly mentioned in this thesis where relevant, is generally outside of the scope of this work and is a topic of future research.

It is also important to note that this work does not aim to provide a full interpretation mapping (i.e. a mapping from *all* OWL constructs to Bunge's ontological constructs). This restriction of the scope is, in part, due to the fact that the main purpose of the OWL language is to allow the development of machine-readable ontologies. Thus, some OWL constructs are purely implementation-related and can not be assigned real-world ontological semantics (as they are not intended for real world representation).

One more methodological consideration pertains to the issue of achieving ontological completeness and clarity. As mentioned earlier, a grammar is *ontologically complete* if and only if every ontological construct can be mapped to a grammar construct, i.e. the mapping is total. A grammar is *ontologically clear* if and only if no two ontological concepts are mapped onto the same grammar construct, and all grammar constructs can be mapped to ontological concepts (Wand & Weber 1993). Since OWL only has three key constructs (classes, individuals and properties), there is no way to represent each ontological construct using a separate OWL construct (without extending the language by adding new constructs). Evermann (2003) stresses a similar issue noting that to achieve a bijective mapping between ontological and modeling language constructs (i.e. a one-to-one mapping of all constructs, which provides maximum ontological clarity) it may be necessary to map language constructs to ontological constructs only if they appear *in a particular context.* For example, the same language construct, depending on its usage context, may represent two different real-world elements (ontological concepts). To achieve ontological clarity we can use modeling rules and constraints on the meta-model so as to map a certain language construct, when used in the first context, to the one ontological concept, and map it, when in the second context, to the other ontological concept. This thesis research adopts a similar approach in the case of OWL in order to alleviate the problem of ontological incompleteness and construct overload. Specifically, it proposes modeling rules and guidelines and meta-modeling elements to specify in what context a certain OWL construct (or a combination of

constructs) represents a particular ontological construct. Also, some naming conventions are proposed as an additional means to clarify the semantics of OWL elements used to model certain ontological elements.

It is also worth mentioning that since IS ontologies (and OWL ontologies in particular) are intended to represent knowledge (including real world domain knowledge) in a machine-readable form the ontology development process usually involves two types of issues: conceptual modeling issues and implementation-related or machine-processing issues. While machine readability and implementation isues are very important, conceptual modeling aspects of ontology development are equally important. Despite the advances in computer application support for ontological engineering activities, people are still heavily involved in ontology development and have to understand, analyze and agree upon ontologies. Ontologies also need to be maintained, merged and expanded, and used in applications, which also requires a clear understanding of what knowledge an ontology represents and how. All this also requires human involvement, even though some semi-automatic tools for building and maintaining ontologies have been developed. Thus, the issues of the effective conceptual representation of a domain of interest and of the proper human comprehension and unambiguous interpretation of the resulting ontologies are very relevant to the ontological engineering field.

Furthermore, some ontologies may be intended to represent only real world domain information while others may attempt to combine both aspects (e.g. real world domain and application aspects). Our analysis focuses on the conceptual modeling aspects of real-world modeling and is not intended to address implementation-related or machine-readability aspects of the OWL modeling process. However, to make our proposed rules and meta-model applicable in practice we follow the existing language syntax and try to keep in mind, as much as possible, certain implementation, machine-understandability and reasoning issues (including some current ontology usages and practices which may affect the way ontologies are developed and applied). Therefore, we hope that using the proposed guidelines for ontology development in OWL should not affect machine-understandability of the resulting ontologies. At the same time, our analysis does not exclude or proscribe using OWL constructs to represent non-real world related aspects in OWL. Rather, one of the goals of the meta-model and guidelines proposed in this thesis is to distinguish between the situations when OWL constructs are used in ontologies for modeling real-world elements and when they are used for other modeling purposes so as to achieve better ontological clarity and consistency.

Finally, it is worth noting that this work takes the current OWL language as given and does not attempt to extend or modify it. Nevertheless, the outcomes of our analysis do suggest some ideas for possible extensions and modifications to OWL, as well as for some potentially useful functionality for OWL ontology development environments and tools which can improve the expressiveness of the language and facilitate the development of better models.

4 ANALYSIS - REPRESENTING MAIN ONTOLOGICAL CONSTRUCTS IN OWL

This chapter discusses how some key constructs of Bunge's ontology can be represented in OWL. According to Bunge's ontology, our world consists of a static structure of things with their properties, changes in things and interactions of things. This thesis mainly focuses on the representation in OWL ontologies of the ontological constructs that describe the static structure of the world (such as things, intrinsic and mutual properties, classes and kinds), leaving the discussion of the concepts related to change and dynamics (such as states or events) to future research. However, the dynamics-related concept of interactions among things will be discussed as it is relevant for the modeling of other constructs (such as mutual properties).

Specifically, this chapter provides a representation mapping for the basic constructs stated above and develops recommendations for modelers (in a form of guidelines and modeling rules) for representing these constructs in OWL based on the relevant ontological rules and assumptions. The term *'guideline'* is used for more general recommendations stating how certain constructs should be represented or some general constraints on representation, while the term *'modeling rule'* is employed for more detailed recommendations, which are usually OWL-specific and provide more detailed suggestions on how general guidelines can be implemented using OWL constructs and mechanisms. As an additional outcome of the analysis, we propose several upper-level classes and properties which together may be viewed as a meta-model for developing OWL ontologies representing real world domains, and which we recommend (as part of our rules and guidelines) to include in all such OWL ontologies.

4.1 REPRESENTATION OF THINGS

4.1.1 General guidelines and modeling rules

Our analysis begins with the concept of a *thing* - the most fundamental, elementary, concept in Bunge's ontology. Bunge's ontology distinguishes between *concrete, or substantial, things*, and *conceptual things* (such as mathematical concepts such as sets or functions or other abstract concepts). Following Parsons & Wand (2000), we assume that information modeling reflects humans' view of existing or possible reality and apply the notion of thing to anything perceived as a specific entity by someone, whether existing in

physical reality or only imagined. Thus, for example, both a bank account and an imagined product will be considered concrete things. We will use the terms 'Bunge thing' or 'substantial thing' to denote ontological things and distinguish them from OWL things (i.e. OWL individuals in general), which are discussed below.

OWL provides the construct of *individual,* which is an instance of the top level class owl:Thing or any its subclasses (i.e. OWL classes).[17] It is possible in OWL to declare an individual, and, if desired, to specify its properties.

We propose to map Bunge things to OWL individuals. That is, entities in a real-world domain that are Bunge things should be represented in OWL ontologies as OWL individuals:

Guideline 1*: Substantial things (Bunge things) in a domain should be modeled in OWL ontologies as OWL individuals.*

However, note that in general, not every OWL individual (i.e. an instance of the top class owl:Thing) can be interpreted as some Bunge thing. OWL does not place any restrictions on what can be modeled using the construct of individual. In current OWL usage practice, OWL individuals are often used to represent all sorts of things - substantial things, conceptual things, properties, property values, and so on. This situation has been identified by Bera & Wand (2004) as a construct overload issue – one of the problems that undermines the ontological clarity of the resulting models (Wand & Weber 1993).

While it is unpractical to require restricting the use of individuals only for modeling ontological things (in the current OWL syntax), the above problem can be alleviated as suggested in the following guideline for modeling real-world ontologies in OWL:

Guideline 2: *OWL ontologies intended to model real world domains should clearly distinguish between OWL individuals representing substantial things (in the ontological sense) and OWL individuals representing other concepts (i.e. non-substantial, or conceptual, things).*

To implement the above two guidelines in OWL we propose to declare two upper-level classes (subclasses of the owl:Thing class) – one for representing all substantial things (and classes of substantial things), and the other one for representing anything other than

[17] Note that every OWL individual is an instance of a top level class owl:Thing, that is, every OWL individual is an 'OWL thing'. The concept of Bunge thing (or substantial, ontological, thing) is different from the OWL thing (OWL individual), which can potentially represent anything in OWL (not just substantial things).

substantial things (in Bunge's sense). Since in Bunge's ontology no thing can be a substantial thing and not be a substantial thing at the same time, these two upper-level classes should be declared as disjoint. This will help to ensure that thes classes do not overlap (i.e. no OWL individuals can be declared instances of both classes). Ontology development environments such as Protégé OWL would not allow modelers in this case to define classes (or individuals) that are subclasses (or instances) of both of these upper-level classes (such classes would be found inconsistent since they would not have any instances). Also, reasoning tools (such as Racer[18]) would be able to detect such inconsistencies in an ontology.

This implementation suggestion is summarized in the following modeling rule:

Modeling Rule 1: *In order to distinguish between OWL individuals representing substantial things and OWL individuals used for other purposes, an OWL ontology intended to model a real world domain should include two disjoint upper-level classes:*

1) Substantial_Thing[19] *class – the extension of this class would consist of all OWL individuals that represent substantial things*

2) Non_Substantial_Thing *class – the extension of this class would consist of all OWL individuals that are used to represent anything other than substantial things*[20].

Several corollaries follow from the above rule and the fact that the two upper-level classes should be disjoint, specifically:

Corollary 1: *Substantial things should be modeled as OWL individuals that are instances of the class* Substantial_Thing *or its subclasses; OWL individuals used for other purposes should be made instances of the* Non_Substantial_Thing *class or its subclasses.*

Corollary 2: *Any OWL class, all instances of which are intended to represent substantial things, should be made a subclass of the* Substantial_Thing *class. OWL classes used for other purposes should be made subclasses of the* Non_Substantial_Thing *class*

Corollary 3: *No OWL individual in an ontology can represent both a substantial thing and non-substantial thing at the same time*

Corollary 4: *No OWL class can (other than built-in top class* owl:Thing*) can include both OWL individuals representing substantial things and OWL individuals representing non-*

[18] http://www.sts.tu-harburg.de/~r.f.moeller/racer/

[19] The names chosen for the upper level classes are just our suggestions; other class names can be used

[20] We include the word 'thing' in this class name since the instances of this class would still be things in OWL sense (even though they are not substantial things in the Bunge-ontological sense).

substantial things

Corollary 5: *Other OWL constructs (such as OWL properties) should not be used to represent substantial things*

If the proposed guidelines and rules are followed, then ontological substantial things can be mapped to a subset of OWL individuals - specifically, individuals that are members of the class extension of the upper-level class Substantial_Thing (representation mapping). On the other hand, every OWL individual that is an instance of the class Substantial_Thing would correspond to some ontological substantial thing (interpretation mapping). Thus, OWL individuals from the class Substantial_Thing are assigned ontological real-world semantics.

4.1.2 Example implementation in OWL

To illustrate the implementation of the guidelines and rules proposed in the previous section, we can declare the proposed two disjoint upper-level classes in OWL as follows[21]:

```
<owl:Class rdf:about="#Substantial_Thing">
    <owl:disjointWith>
        <owl:Class rdf:ID="Non_Substantial_Thing" />
    </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Non_Substantial_Thing">
    <owl:disjointWith rdf:resource="#Substantial_Thing" />
</owl:Class>
```

Now, if in some ontology (e.g. about people) we want to represent some specific ontological thing, for example, a person John Smith, we can minimally represent it as an OWL individual declared to be an instance of the class Substantial_Thing:

```
<Substantial_Thing rdf:ID="John_Smith">
```

In principle, this declaration in OWL is sufficient to represent an individual and allows assertions about properties of this individual (as will be discussed later). Note that OWL syntax does not require to declare classes (such as Person) first to be able to declare individual things. Without a doubt, classes are very useful for modeling and exist in any ontology, and OWL individuals can be declared or inferred to be instances of certain classes, However, in principle, OWL individuals are not required to be declared instances of any classes (other than of the top level default class owl:Thing) and can be asserted to possess

[21] These classes will automatically be subclasses of the built-in top OWL class owl:Thing

properties independent of class membership. We will discuss this issue in more details in later sections (when analyzing classes and properties) and will later introduce more rules regarding classes and properties.

For now, as far as classes are concerned, we just want to show that if, for example, one needs to represent some class of substantial things, such as a class Person, it should be modeled in accordance with our rules (Corollary 2) as a subclass of the upper-level class Substantial_Thing[22]:

```
<owl:Class rdf:ID="Person">
        <rdfs:subClassOf>
                <owl:Class rdf:ID="Substantial_Thing" />
        </rdfs:subClassOf>
</owl:Class>
```

On the other hand, sometimes the concepts that do not represent real world substantial things still may need to be represented using OWL classes and individuals (for example, due to some implementation or reasoning related issues). For instance, a domain may include a concept of "Delivery". According to our rules, such concepts should be modeled as subclasses and individuals of the Non_Substantial_Thing class:

```
<owl:Class rdf:ID="Delivery"
        <rdfs:subClassOf>
                <owl:Class rdf:ID="Non_Substantial_Thing" />
        </rdfs:subClassOf>
</owl:Class>
```

In later sections, we will present more examples of what kind of concepts may need to be modeled as non-substantial things and how they can be related to substantial things. For now, we proceed to the discussion of the concept of properties and their representation in OWL.

4.2 REPRESENTATION OF PROPERTIES

In this section we compare the ontological property concept with the OWL property construct and propose methods to represent ontological properties in OWL in such a way that real world semantics and the ontological assumptions involving properties are preserved

[22] A class can also be declared as a subclass of another class, which in turn is a subclass of the Substantial_Thing class. For example, if we already declared a class Person to be a subclass of Substantial_Thing, we could declare another class, Woman, as a subclass of Person, which would imply that Woman is also a subclass of Substantial_Thing.

to as great an extent as possible.

4.2.1 Properties in OWL and ontological properties – a comparison

According to Bunge's ontology, every *thing* possesses *properties* and properties are always attached to things. Properties can be *intrinsic* - possessed by the thing alone, or *mutual* – shared properties of two or more different substantial things.

Properties of things exist whether or not humans are aware of them, and things are known to us through their properties. However, according to Bunge's ontology, humans conceive of things in terms of *models of things* (or conceptual things), and conceive of properties of things in terms of *attributes* (which are properties of models of substantial things). An attribute may or may not reflect a substantial property or a number of properties. For example, the height of a person is an attribute that reflects a substantial ontological property, while the name of a person does not actually represent any specific substantial property but rather it is an attribute that stands for the individual as a whole (Wand et al., 1999).

Keeping in mind the distinction between the property and the attribute notions, in our further discussion of properties both terms are used interchangeably whenever talking about representation of ontological properties in OWL. More specifically, when talking about the representation of ontological properties in OWL, we actually mean the representation of the attributes which we are aware of and which we ascribe to things to model the ontological properties that we believe these things possess (since humans can only conceive of properties of things via attributes).

The property construct is available in the OWL language as well. There are two main types of properties in OWL: *object properties* and *datatype properties*[23]. A *property P* in OWL is a directed binary relation P(x,y) that links a subject *x* (which is an OWL individual) to an object *y* which is either an OWL individual (if P is an object property) or a data value (if P is a datatype property). The object (*y*) is considered a value of the property P for the subject - individual *x*.

The analysis of the OWL syntax rules, existing OWL modeling practices and OWL ontologies[24] reveals that in general there is no clear correspondence between OWL properties and ontological properties. For example, Bera & Wand (2004) point out that one

[23] Other types of OWL properties such as annotation properties and ontology properties are not considered in this work since they are implementation related and not essential for real-world domain representation issues.

[24] Many existing OWL ontologies are available from the W3C website (http://www.w3.org/2004/OWL/#ontologies) and from the Protégé system website (http://protege.stanford.edu/plugins/owl/ontologies.html)

of the deficiencies of OWL from the ontological analysis perspective (construct overload) is that OWL does not distinguish between intrinsic and mutual properties. To expand on that, we highlight several other issues related to the existing practices of modeling properties in OWL, which may lead to problems from the ontological analysis standpoint when modeling real world domains:

Issue 1: In OWL, properties are used to describe links (or relations) between two OWL individuals or between an individual and a data value, where OWL individuals *do not* necessary represent substantial things. That is, some OWL properties are not associated with OWL individuals that represent substantial things and do not represent ontological (substantial) properties.

Issue 2: Even in the cases when OWL properties do represent ontological properties, there are no consistent guidelines regarding, for example, whether a datatype or an object property should be used to represent a mutual or an intrinsic property. At a first glance, it may seem that datatype properties (which connect individuals to data values) should be used for representing intrinsic properties, while object properties (which connect individuals to individuals) should only be used to represent mutual properties. However, this is not always the case and not always possible to do in OWL (as we will show in later sections on properties). For example, while many intrinsic properties (e.g. person's name, birth date or age) can be represented in OWL using datatype properties with a suitable XML Schema datatype (e.g. string, date or positive integer), in other cases it may be necessary or more advantageous (e.g. for reasoning purposes) to model intrinsic properties using OWL object properties.

Specifically, an OWL object property can be used to represent an intrinsic property in general, while the ontological intrinsic properties in particular (or property values) are sometimes represented using special OWL classes and their individuals. This is currently a common practice used for representing such properties that have finite enumerated collections of values (for example, wine color could be white, red or rose; clothing size could be small, medium, or large, and so on) [25]. Such properties can be represented in OWL using one of the two (at least) distinct ways: 1) as datatype properties with enumerated sets of data values (e.g. of string type) as property ranges, or 2) as object properties with the sets

[25] The famous wine ontology (http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf) provides an example of such representation. It has so-called "wine descriptor" properties of wine (such as hasColor, hasBody, hasSweetness) which relate OWL classes representing wines (i.e. substantial things) to OWL classes (e.g. WineColor) representing color and components of taste, such as sweetness, body and flavor (i.e. wine properties).

of property values represented as special OWL classes or their instances. As a part of W3C best practice guidelines, Rector (2004) discusses different representation patterns for such value collections and their advantages and disadvantages. A related document by Noy (2004) addresses the issue of using classes vs. using individuals as property values for representing property values.

The choice of one of the two representations (datatype property versus object property with 'value' classes/individuals) is mainly driven by implementation aspects and the intended ontology usage. It is also motivated by the goals of improving automatic reasoning and facilitating ontology editing and sharing. Ontological (conceptual) considerations are often not taken into account in the current practices. However, as Bera & Wand (2004) pointed out, the use of classes and individuals for representing property values is another example of construct overload problem in OWL. To alleviate this problem, they suggested that such "special" classes or instances should be distinguished from the classes and instances representing substantial things. This thesis actually incorporates the above suggestion in Guideline 2 and Modeling Rule 1, proposing two disjoint upper-level classes Substantial_Thing and Non_Substantial_Thing to be included in real world domain ontologies.

Issue 3: Another general issue with modeling properties in OWL is that mutual properties in Bunge's ontology can be shared by two *or more* things, while OWL properties are always *binary* relations. Thus, it is not possible to use a single OWL property to represent a mutual property shared by more than two things. Some workarounds are required such as, for example, creating special OWL classes to represent N-ary relations for N>2 (see Noy & Rector (2004) for a discussion of possible representation patterns for higher order relationships in OWL).

A similar problem exists in some other conceptual modeling languages where higher power relationships are often represented as sets of binary relationships (Wand *et al.,* 1999). However, replacing an N-nary relationship with a set of binary relationships may lead to the loss of information. Wand *et al.* (1999) also criticize this approach from the ontological standpoint and recommende that the same construct should be used to represent binary and higher order relations (i.e. mutual properties).

Issue 4: The problems with representing mutual properties may potentially arise even in situations when one needs to represent a mutual property shared by only two things. In many cases, properties occur together or have some dependency on each other. For example, we may define an object property EnrolledIn linking a person (student) to a university he is

enrolled in, that is, this would be a mutual property shared by a person (an individual) and a tertiary institution (an individual)[26]. However, another mutual property shared by these two individuals is '*having a start date (at a particular university)*', which is a property in general (e.g. named hasStartDate) with specific start date values for instances (e.g. corresponding to different universities). In OWL there is no direct way (i.e. using only the construct of property for properties) to represent such a mutual property since one property statement (fact) can only link either two individuals or one individual and a data value. In our case, however, we need to link *three* items: two individuals (an institution and a student) and a data value (specific start date). A workaround way to represent this in OWL would be to create an additional class, e.g. University_Student (representing the relation between a university and a student) and to associate this class with the properties such as hasUniversity, HasStudent, hasStartDate. This method is similar to the use of association classes in UML or relationship entity types in ERM and will be explored later in more detail.

Issue 5: A final issue with properties is that often different OWL constructs can be used to represent ontological properties in different ontologies. Specifically, some ontological properties may be represented as properties in one ontology, but as classes in another. For example, one ontology may have a property *"being a student"* while another ontology may not have such a property but rather have a class Student. In the latter case, declaring that an individual X is an instance of the class Student can also be interpreted as the individual X possessing a general property *"being a student"* [27]. On one hand, this may be considered a problem, as noted by Bera & Wand (2004). They criticized the possibility of using multiple constructs in OWL to represent the same ontological concept of property (as the 'construct excess' problem) as this may undermine the ontological clarity of OWL models. On the other hand, as we will discuss in more detail later, in Bunge's ontology classes (and kinds) are defined in terms of properties: a class is a set of things that possess a common property. Therefore, for each property, in principle there exists a corresponding ontological class (or kind) of the ontological things possessing this property, which is termed *the scope of the property*. Thus, in the above example, the extension of the class Student actually can be viewed to represent the scope of the ontological property *"being a student"*.

[26] We assume that a student can be enrolled in several tertiary institutions.

[27] According to Bunge's ontology, a property 'being a student' is actually an attribute representing a number of properties that comprise 'being a student'; in other words, it represents a compound property (Wand et al., 1999) which is a combination of all these properties.

To summarize our comparative analysis of properties in OWL versus ontological properties, we have shown that in OWL syntax and in the current practices, the concept of property in OWL does not fully correspond to the ontological notion of property. Thus, it would be incorrect to simply map ontological properties to OWL properties. Rather, we propose that ontological properties should be mapped to a subset of OWL properties that satisfy certain conditions. In the next sections we consider several cases of ontological properties, and propose how they can be represented in OWL to alleviate ontological completeness and clarity problems (including those noted by Bera & Wand (2004)). We will propose modeling guidelines and rules, which, if followed, allow the mapping of ontological properties to OWL properties when they are used in a particular context and in particular combinations with other OWL constructs in some cases (thus achieving "mapping in context", as recommended by Evermann (2003)).

4.2.2 General guidelines on representation of ontological properties in OWL

This section focuses on some general issues related to the representation of the ontological (substantial) properties in OWL. In sections 4.2.3 and 4.2.4 intrinsic and mutual properties will be discussed separately in more detail, and relevant guidelines and rules are proposed.

Bunge's ontology distinguishes between *properties in general* (generic properties) and *properties in particular* (individual, or specific, properties). *Properties in general* are properties possessed by a set of things, e.g. "color", "speed", "salary", etc. An *individual property* is one that can be represented as a value of some property in general, such as "blue in color", speed of 100mph" or "salary of $2000" (Bunge, 1997, p.63; Evermann, 2003). Similarly, many ontological properties in general can be represented as OWL properties, while ontological properties in particular can be represented as values of the OWL properties that represent the corresponding properties in general. For example, one can define a generic property NumberOfChildren with integer values representing actual number of children (property values) for specific persons.

However, as noted in the comparison of OWL and ontological properties (issue 2, section 4.2.1), several ways to represent property values are possible in OWL. Depending on modeler's needs, reasoning requirements, and the intended ontology usage, property values in OWL can be represented in one of the several ways: as XML datatype values, as OWL classes, or as OWL individuals. We do not intend to impose unnecessary restrictions on

OWL usage since this can make the resulting ontologies less useful from the pragmatic standpoint and our rules less applicable in real-life modeling. Thus, we acknowledge the existence of different ways of representing property values in our rules. However, in accordance with Guideline 2 and Modeling Rule 1 (which require distinguishing OWL classes and individuals representing substantial things from OWL classes and individuals modeling anything else) we require that OWL classes and individuals intended to represent property values should be distinguished from OWL individuals and classes representing substantial things.

To summarize the above discussion on properties, we propose a following general guideline on modeling properties in OWL:

Guideline 3:

- *In OWL ontologies modeling real world domains, ontological properties in general should be modeled as OWL properties, and ontological properties in particular should be modeled as property values of those OWL properties that represent the corresponding properties in general;*
- *Depending on a property type (e.g. intrinsic or mutual) and model usage and reasoning requirements, ontological properties in particular (property values) may be represented either as XML datatype values or as special OWL classes and their individuals;*
- *If ontological properties in particular (property values) are modeled using OWL classes/ individuals, then such classes and individuals should be clearly distinguished from OWL classes and individuals that represent substantial things.*

Since we proposed the two upper-level classes Substantial_Thing and Non_Substantial_Thing to distinguish between OWL classes/individuals representing substantial and non-substantial things, the next corollary follows:

Corollary 6: *OWL classes (individuals) representing property values should be subclasses (instances) of the upper-level class* Non_Substantial_Thing

Since (as discussed in previous section) not all OWL properties represent ontological properties, the implication of the Guideline 3 for the representation mapping between OWL properties and ontological properties is that ontological properties can be mapped to a *subset* of OWL properties that conform to certain constraints, depending on ontological property

type (intrinsic or mutual) and intended model usage requirements[28]. These constraints will be discussed in more detail as modeling rules later in this section and in the next sections.

Bunge's ontology distinguishes between intrinsic and mutual properties. Therefore, being able to distinguish between these two types of property in OWL ontologies would enhance a model's ability to represent real-world domain faithfully. Thus, we propose another guideline:

> ***Guideline 4:*** *OWL ontologies modeling real world domains should distinguish among OWL properties that are used to represent the following groups of properties:*
>
> - *Ontological intrinsic properties of substantial things*
> - *Ontological mutual properties of substantial things*
> - *Other OWL properties, i.e. properties that are not intended to represent substantial properties but are used for other purposes in the ontology*

The third category of properties is intended for OWL properties that do not represent substantial intrinsic or mutual properties. Among those are, for example, properties that modelers may need to use to represent certain relationships between constructs, or some properties that are not properties of substantial things (for example, implementation related properties). For example, later in this thesis (section 4.5), we propose a pair of mutually inverse object properties isComposedOf and IsComponentOf, which are intended for representing composite/component relationships between a pair of OWL individuals modling some substantial things. These properties are not domain-specific; rather, we view them as upper-level or meta-properties (we suggest that such properties are defined in any ontology that needs the representation of composite/component relationships). Guideline 4 suggests that such properties should be distinguished from domain-specific mutual and intrinsic properties.

Unfortunately, OWL does not provide a mechanism to group properties or to specify a user-defined property type or category to facilitate the above categorization. As one way to distinguish among these categories of properties (without introducing changes to the OWL syntax), we suggest the use of naming conventions (such as prefixes) in OWL property

[28] We recognize that the important role of OWL ontologies is to provide machine readable representation of a domain and to facilitate reasoning. Therefore, in our ontological interpretations and guidelines we are also trying, whenever possible to acknowledge certain desirable representation patterns used in OWL, provided they can be interpreted ontologically and allow for ontologically consistent representation.

names[29]. In this thesis, we will use the following prefixes:

- **"ip_"** - for OWL properties representing intrinsic properties (e.g. "ip_Size"),
- **"mp_"** - for OWL properties representing mutual properties (e.g. "mp_Distance").

Additional naming conventions will be introduced in later sections.

The last rule in this section deals with the concept of 'property domain' in OWL. OWL properties can have a domain declared. By definition, a property domain in OWL is the set of OWL individuals **X** such that for each X there is at least one **y**[30] that is a value of P for the individual X (i.e. P(X, y) holds). Note that the role of the domain in OWL is to allow a certain type of inference, it is not a constraint to be checked. Specifically, if a property P is stated to have a domain D (usually as a named or anonymous class), and for some X a statement of a form P(X, y) is declared (or inferred) in an ontology, then, according to OWL, it can be inferred that X is an instance of the domain D.

In Bunge's ontology, properties are possessed by substantial things and according to Modeling Rule1, OWL individuals that represent substantial things should always be declared (or inferred) to be instances of the upper-level class Substantial_Thing (or its subclasses). Therefore, we propose another modeling rule regarding the domain of OWL properties that are intended to represent ontological properties:

> ***Modeling Rule 2:*** *If an OWL property is intended to represent an ontological (substantial) property, then the domain of such property should be either the* Substantial_Thing *class or its subclasses.*

The next section (4.2.3) continues the discussion of property representation and focuses on intrinsic properties in more detail, while section 4.2.4 is devoted to the discussion of representation of mutual properties.

4.2.3 Representation of intrinsic properties in OWL

In Bunge's ontology, intrinsic properties are properties that depend on one thing only (for example, height or color). In Guideline 3, we proposed that ontological substantial properties in general (including intrinsic properties) should be modeled in OWL ontologies as OWL properties, while properties in particular (specific properties possessed by

[29] OWL also allows the use of labels, which could be used to have more human-readable (natural) alternative names for properties that can be used by applications (e.g. Name vs. ip_Name). Thus, the use of prefixes should not significantly affect ontology usability in applications.

[30] If P is an object property, then y is an OWL individual. If P is a datatype property, then y is a datatype value.

substantial things, or property values) should be represented as values of the OWL properties representing the respective properties in general. However, as discussed earlier, different ways of representing property values are possible in OWL, for example, as XML datatype values, as OWL classes or as OWL individuals (depending on the intended model usage, as well as on the application and reasoning needs). The choice of the property value representation method, in turn, determines whether an OWL object property or an OWL datatype property should be used for representing an intrinsic property.

In the remainder of this section, we will discuss in more detail (using simple examples for illustration) two most typical cases of intrinsic property representation in OWL ontologies which may require different representation approaches. For each case, we propose some additional modeling guidelines and rules complementing and extending earlier proposed guidelines and rules for property representation.

4.2.3.1 Case 1: Simple cases of generic intrinsic properties with value manifestations – representation using OWL datatype properties

A common group of intrinsic properties are generic properties (such as "weight", "color", "age", "name") which are manifested in things (individuals) by specific values (e.g. literal or numeric). Possessing a specific value for a generic property implies possessing the generic property itself.[31] For example, a person (a substantial thing) possesses a generic property 'weight' and also has a specific value for his/her weight (in some units), which can be viewed as a specific property (property in particular), e.g. "has weight of 100 kg"[32].

Such intrinsic properties are usually represented in OWL as datatype properties with the range of values defined as a suitable XML Schema datatype (e.g. integer, string or date). For example, 'name' is an intrinsic (generic) property possessed by persons (a set of substantial things represented by a class Person). Individuals also have specific names (such as Alex or Maria), which are specific intrinsic properties preceded by the generic property "name". In OWL we can represent the generic property "name" as an OWL datatype property (e.g. ip_PersonsName); specific names for individuals of the class Person would then be represented as values of the property ip_PersonName. In accordance with Corollary 2, we would make the class Person a subclass of the upper level class Substantial_Thing. We also define the domain of the property ip_PersonName to be a class Person (Rule 2).

---

[31] This is an example of what Parsons and Wand (2003) term as "value manifestation", which is a type of property precedence law

[32] For simplicity, here we assume that there is no need to represent units (e.g. kg or US$) but only focus on the representation of a generic property (e.g. weight) and its specific values for individuals (e.g. 100). If one needs to represent values with units, then object properties and special value classes would be required (as in case 2).

Below is a representation of this example in the OWL RDF/XML syntax[33]:

```
<!--Declaring a class Person – a subclass of the class Substantial_Thing (Corollary 2) -->
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Substantial_Thing" />
  </rdfs:subClassOf>
</owl:Class>

<!--Declaring an OWL property ip_PersonName (Guideline 3,4) with the class Person as a domain
(Rule 2) and string datatype as range-->

<owl:DatatypeProperty rdf:about="#ip_PersonName">
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
</owl:DatatypeProperty>

<!—Example statement about a specific individual (represented as OWL individual "MikeSmith", an
instance of the class Person) having a specific property value ("Michael Smith") for the property
ip_PersonName -->

<Person rdf:ID="MikeSmith">
  <ip_PersonName rdf:datatype = http://www.w3.org/2001/XMLSchema#string">Michael  Smith
  </ip_PersonName>
</Person>
```

The above approach (i.e. using OWL datatype properties with a suitable XML datatype as range) is suitable for modeling many intrinsic properties. This approach is best suited for representing those properties which can have a large (or unlimited) number of possible numeric or string values (i.e. specific properties). However, in other cases this method of representation may be too restrictive from the implementation and reasoning standpoint (since, for example, reasoning on datatypes is currently less developed and less powerful in OWL compared to reasoning on classes and individuals). An alternative representation, popular in current OWL practices, is to use special value classes and individuals of those classes to represent property values (where properties themselves are represented as OWL object properties rather than datatype properties).

In the next section we discuss a specific type of intrinsic properties - generic properties with enumerated collection of property values - for which the aforementioned alternative representation can be useful (and is often preferred by OWL practitioners; see, for example, Noy, 2004; Rector, 2004). In order to make this representation consistent with

[33] We assume that the upper-level class Substantial_Thing has already been declared (as proposed in section 4.1.1) and only show an excerpt from the ontology pertaining to classes and properties of interest here.

the ontological considerations, we propose additional modeling rules guiding his representational pattern. We also show that the earlier proposed general guidelines and rules are still applicable, irrespective of how property values are represented (as datatype values or as OWL classes or individuals).

4.2.3.2 Case 2: Intrinsic properties with enumerated collections of values

Some generic intrinsic properties of things can be manifested by a specified (enumerated, usually finite) collection of values. For example, wine has color which can be white, red or rose; clothes have size which can be small, medium or large, and so on. Rector (2004) and Noy (2004) review several modeling patterns for representing such specified value collections of properties in OWL and discuss advantages and disadvantages of these patterns.

As already mentioned in the previous section, one way to represent such properties with enumerated collections of values is to use an OWL datatype property to represent a generic property (such as "color" or "clothing size") and use the OWL enumeration construct (owl:oneOf) in combination with a suitable XML Schema datatype to represent a list of possible values for the property. In other words, the range of values for the datatype property representing a generic intrinsic property would be a custom-defined enumerated datatype - a list of values of some predefined XML datatype (such as string or integer). For example, we can represent a generic intrinsic property "clothing size" and its values in the OWL/RDF syntax in the following way (conforming to our rules and guidelines):

```
<!--Declaring meta-model classes Substantial_Thing, Non_Substantial_Thing -->
 <owl:Class rdf:about="#Substantial_Thing">
   <owl:disjointWith>
    <owl:Class rdf:about="#Non_Substantial_Thing"/>
   </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Non_Substantial_Thing">
   <owl:disjointWith rdf:resource="#Substantial_Thing"/>
</owl:Class>

<!-- Declaring a class Clothes to be a subclass of the upper level class Substantial_Thing (Corollary
1)->
 <owl:Class rdf:ID="Clothes">
        <rdfs:subClassOf>
                <owl:Class rdf:ID="Substantial_Thing" />
        </rdfs:subClassOf>
 </owl:Class>
```

```
<!--- Declaring a datatype property ip_ClothingSize representing intrinsic generic property "closing
size" (hence the use of the prefix ip_). Property domain is the class Clothes (a subclass of
Substantial_Thing class; Rule 2). Property range is the enumerated datatype – list of values of XML
datatype String (Small, Medium, Large) -->

 <owl:DatatypeProperty rdf:about="#ip_ClothesSize">
    <rdfs:domain rdf:resource="#Clothes" />
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Small</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Medium</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil" />
                <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Large</rdf:first>
              </rdf:rest>
            </rdf:rest>
          </rdf:rest>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
 </owl:DatatypeProperty>

<!--- Statements about individuals can be made regarding property values. For example, Dress1 is an
OWL individual - an instance of the class Clothes and has a property value "Small" for the property
ip_ClothingSize (this represents a specific ontological intrinsic property "having clothing size 'small'"
for this particular clothing item which is a substantial thing (represented by an OWL individual) -->

<Clothes rdf:ID="Dress1">
        <ip_ClothingSize rdf:datatype = "http://www.w3.org/2001/XMLSchema#string">Small
        </ip_ClothingSize>
</Clothes>
```

The example above is just a special case of the earlier discussed case 1, and thus the same guidelines and rules are applicable. That is, a class Clothes (individuals of which represent substantial things – clothing items) is declared as a subclass of the upper-level class Substantial_Thing (Corollary 2); an OWL datatype property with the "ip_" prefix (ip_ClothesSize) is used to model the intrinsic generic property 'clothing size' (Guidelines 3, 4). We also declare the domain of the property to be the class Clothes (a subclass of the class Substantial_Thing, Rule 2).

Rector (2004) discusses other possible methods of representing enumerated collections of values for properties. In particular, individuals of the specially defined value classes can be used to represent enumerated property values. In this case, we would have to use OWL object properties, rather than datatype properties, to represent generic intrinsic

properties (since we need to link OWL individuals representing substantial things to OWL individuals representing property values). Applying this approach to the earlier discussed clothing size example, we represent a generic intrinsic property 'clothing size' as an OWL object property (ip_ClothingSize). Again, as per Rule 2, the domain for this property is the class Clothes (a subclass of the Substantial_Thing class). However, the range of this property is now not an XML datatype (as in earlier examples) but a specially defined OWL class, for example, Clothing_Size_Value. This class is declared to have several instances - OWL individuals representing property values (e.g. Small, Medium, Large)[34]. The OWL RDF/XML representation of this example is as follows:

```
<!--Declaring meta-model classes Substantial_Thing, Non_Substantial_Thing, and Property_Value -
->
    <owl:Class rdf:about="#Substantial_Thing">
      <owl:disjointWith>
        <owl:Class rdf:about="#Non_Substantial_Thing"/>
      </owl:disjointWith>
    </owl:Class>
    <owl:Class rdf:about="#Non_Substantial_Thing">
      <owl:disjointWith rdf:resource="#Substantial_Thing"/>
    </owl:Class>

    <!--The class Property_Value is declared as subclass of the Non_Substantial_Thing (corollary 6)
      -->
    <owl:Class rdf:ID="Property_Value">
      <rdfs:subClassOf>
        <owl:Class rdf:ID="Non_Substantial_Thing"/>
      </rdfs:subClassOf>
    </owl:Class>

<!—Declaring a class of substantial things – Clothes class-->
  <owl:Class rdf:ID="Clothes">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Substantial_Thing"/>
    </rdfs:subClassOf>
  </owl:Class>

<!-- Declaring a class representing enumerated set of property values for an intrinsic property
'clothing size' and instances of this class representing individual property values>
  <owl:Class rdf:ID="Clothes_Size_Value">
      <rdfs:subClassOf rdf:resource="#Property_Value"/>
  </owl:Class>
    <Clothing_Size_Value rdf:ID="Small"/>
```

[34] Often, subclasses of value classes are created, with individual values still represented by instances (i.e. subclasses Small, Medium and Large can be created). Also, sometimes, such subclasses themselves are used to represent values. However, the latter is only possible in OWL Full. Rector (2004) discusses both methods and analyses their advantages and disadvantages. For this thesis it is not relevant which of the two approaches is used (our guidelines and rules will hold for both methods) thus we do not discuss this issue in more detail.

```
    <Clothing_Size_Value rdf:ID="Medium"/>
    <Clothing_Size_Value rdf:ID="Large"/>

<!—Declaring an object property ip_ClothingSize representing the intrinsic property 'clothing size' (with the domain Clothes and the range – class Clothing_Size_Value -->
    <owl:ObjectProperty rdf:ID="ip_ClothingSize">
      <rdfs:domain rdf:resource="#Clothes"/>
      <rdfs:range rdf:resource="#Clothing_Size_Value"/>
    </owl:ObjectProperty>

 <!-- A statement about a particular Clothes individual (e.g. Dress1) having a specific value for the property ip_ClothingSize'-->

   <Clothes rdf:ID="Dress1">
     <ip_ClothingSize rdf:resource="#Small"/>
    </Clothes>
```

The diagram shown in Figure 1 illustrates this representation in a graphic form:

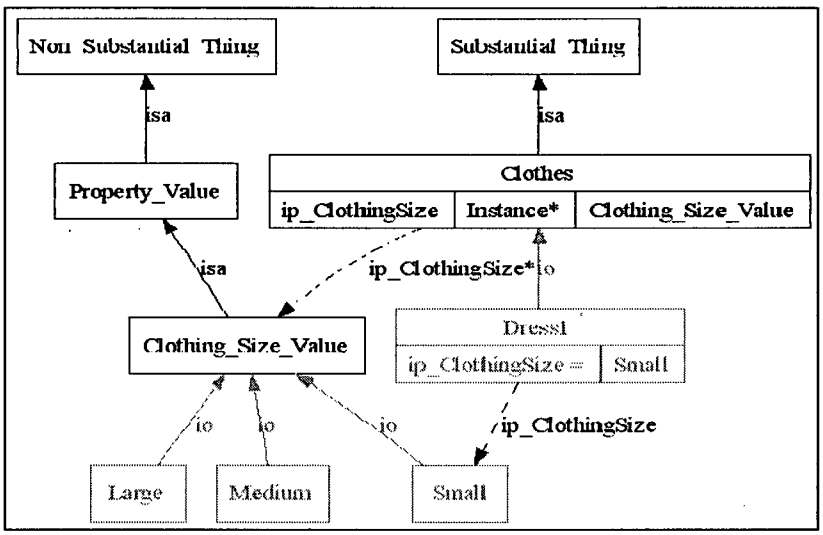

Figure 1: Representing enumerated property values using value classes and instances

The choice of a method for representing property values (as datatypes vs. as classes or individuals) is often related more to the application and implementation issues than to the conceptual modeling aspects. We acknowledge that there may be compelling reasons for OWL developers and modelers to choose a specific implementation, and thus do not suggest to proscribe completely any of these approaches. However, we recommend that our proposed rules and guidelines (which are applicable to both cases) are followed to help achieve more ontologically consistent representation.

In addition, we would like to propose one more modeling rule to ensure better ontological consistency and clarity in a case when OWL classes or individuals are used to

represent property values. Guideline 3 states that if OWL classes or OWL individual are used to represent property values of generic ontological properties (represented by OWL properties), then such special purpose classes and individuals should be distinguished from OWL classes and individuals representing substantial things. Also, Modeling Rule 1 proposes that in order to distinguish between substantial things and other concepts in an ontology, two disjoint upper level classes, Substantial_Thing and Non_Substantial_Thing are used. Combining those guidelines and rules, we summarize our recommendation for the property representation case approach using value classes in the following modeling rule:

Modeling Rule 3: *If an OWL class (and its instances) is used to represent a collection of property values for some OWL property representing an intrinsic generic property of substantial things, then*

- *This intrinsic generic property should be represented as an OWL object property (rather than an OWL datatype property)*
- *The domain of this property should be the class* Substantial_Thing *or some of its subclasses*
- *The range of this property should be defined as the OWL class that is used to represent property value collection*
- *The OWL class representing the property value collection should be declared a subclass of the* Non_Substantial_Thing *upper-level class (to distinguish it from substantial thing classes)*

The last suggestion in this section is intended to help further separate OWL classes and individuals used for representing property values (as discussed above) from OWL classes and individuals representing substantial things. Specifically, we propose that an upper-level class, Property_Value, is created in every ontology (as a subclass of the upper-level class Non_Substantial_Thing), and that any OWL classes representing property values should be declared subclasses of this class. For example, the class Clothing_Size_Value in the earlier example would be declared a subclass of the class Property_Value (which also makes it a subclass of the Non_Substantial_Thing class). This suggestion hepls clearly identify those special purpose value classes (by grouping them under the same upper level class), and thus will help achieve better ontological clarity and consistency (since there will be more clear distinction and an indication of the specific type and purpose of these classes, separating them from the substantial thing classes). This suggestion is summarized in Modeling Rule 4:

Modeling Rule 4: *If OWL classes and individuals are used in an OWL ontology modeling a real world domain, then a special upper-level class* Property_Value *should be included in the ontology as follows:*

- *This upper level class* Property_Value *should be declared a subclass of the upper-level class* Non_Substantial_Thing
- *Any OWL class used to represent a collection of property values for some ontological property should be declared a subclass of the upper level class* Property_Value *(and thus also a subclass of the upper-level class* Non_Substantial_Thing*)*

To summarize, in all the discussed cases of intrinsic properties (irrespective of whether property values for an intrinsic property are represented in OWL as predefined XML Schema datatype values, enumerated datatype values, or as OWL classes and individuals) the same key guidelines are applicable:

- A generic intrinsic property should be represented as an OWL property (a datatype or an object property depending on the selected value representation method)
- The domain of the property should always be some subclass(es) of the Substantial_Thing upper-level class
- The range of the property depends on which property value representation method is chosen (it could be an XML datatype or a special OWL class, which should be clearly distinguished from the substantial thing classes)

Assuming that the guidelines proposed above are followed, we can map ontological intrinsic properties to a subset of the OWL properties, specifically, to those properties that have as a domain a class Substantial_Thing or some its subclass(es), and have as a range either an XML datatype (if property values are represented as datatype values) or a special value class - a subclass of the Property_Value class, which in turn is a subclass of the Non_Substantial_Thing class (if OWL individuals are used to represent property values).

4.2.4 Representation of mutual properties in OWL

This section is devoted to the discussion of how ontological mutual properties can be represented in OWL clearly and consistently with ontological assumptions. The discussion is structured as follows. First, we briefly review the concepts of Bunge's ontology which are relevant for the analysis of mutual properties (section 4.2.4.1). Next, section 4.2.4.2 discussess some general considerations and possible choices of the representation method available in OWL. Section 4.2.4.3 provides some theoretical foundations to justify the

method of mutual property representation that we consider preferred. Then, in section 4.2.4.4, the proposed method for representing bundles of mutual properties is further illustrated using an example, and modeling guidelines and rules for their representation are proposed. Finally, in section 4.2.4.5 we provide some analysis and ontological considerations related to the use of an alternative method of representation of mutual properties – direct linking of OWL individuals representing substantial things using OWL object properties[35], and suggest some cases in which the use of such representation method is acceptable from the ontological standpoint.

4.2.4.1 Relevant concepts of Bunge's ontology

In Bunge's ontology, *mutual,* or *relational, properties* are properties that depend on two or more things. For example, a property of being a university student is a mutual property since it depends on the existence of both a person and a tertiary institution (Wand *et al.,* 1999).

Closely related to the concept of mutual property is a notion of *interaction.* According to Bunge's ontology, things can interact. Wand *et al.* (1999) provide a clear and concise explanation of interaction and its relation with mutual properties. Following Bunge's ontology, they state that when two things interact, one may cause the other to change. Changes to things are manifested as changes to properties, which are modeled via changes in the values of attribute functions, i.e. changes of state. The existence of an interaction can be considered a mutual property of things, and conversely, the existence of a mutual property can indicate an interaction. For example, a mutual property that a person is employed by a company implies that the existence of the company affects the state of the person (and vice versa). If the company ceases to exist, the person loses the property of being employed by that company, and similarly, if the employee quits, the (set-valued) attribute of the company that shows its list of employees will change *in value* (Wand *et al.,* 1999). A mutual property that reflects an interaction is termed a *binding mutual property* (Bunge 1977, p.102). A binding mutual property implies that some changes in one thing are related to (precede, are accompanied by, or are followed by) changes in the other thing. A property that does not imply an interaction is termed *non-binding*, for example, "thing A is behind thing B", "thing A is older than thing B".

In the following sections we discuss different cases of mutual properties and consider several possible mechanisms by which mutual properties of substantial things can be

[35] This method of representation is quite popular in current OWL practices and thus deserves special consideration.

modeled in OWL. Based on the analysis of these mechanisms, we develop representation patterns and propose some rules and guidelines for representing mutual properties in order to improve the ontological consistency and clarity of the resulting OWL ontologies.

4.2.4.2 Mutual properties - general representational considerations

By definition, mutual properties are shared by two or more things. Modeling Rule 1 states that ontological substantial things should be modeled as OWL individuals that are instances of the upper-level class Substantial_Thing or its subclasses. Thus, in order to model mutual properties properly and to comply with the earlier proposed guidelines, we need to use some OWL mechanism that allows associating two or more OWL individuals (as well as classes to which these individuals belong). Also, Guideline 3 states that ontological properties should be modeled as OWL properties conforming to certain constraints, which depend on a type of property as well on the model usage and implementation considerations. In the interest of ontological clarity, it is preferable that, like intrinsic properties, mutual properties are modeled using the same language construct, that is OWL property construct (as stated in guideline 3). These considerations guide our choice of modeling pattern for mutual properties.

Unlike other modeling languages (such as ER or UML), OWL does not provide any predefined construct for linking several entities (similar to UML association classes or relationships in ER). However, two mechanisms in OWL allow linking OWL individuals (and thus can be considered as candidate methods for representing of mutual properties):

1) **Using OWL object property construct to link two OWL individuals**
   - For example, an object property EnrolledIn (Student, Univesity) links student individuals to university individuals (represented by OWL individuals). A statement of a form EnrolledIn (StudentX, UniversityU) in an ontology would mean that a specific StudentX is enrolled in a UniversityU ('being enrolled' is a mutual property here)
   - However, the problem with this method is that it only allows linking two individuals. Also, it does not allow the representation of bundles of related mutual properties shared by the same set of things, as will be demonstrated later.
2) **Defining special classes (relation classes) to link two or more OWL individuals**:
   - Two or more OWL individuals (or classes) representing substantial things can be linked via a special OWL class. An instance of this class would represent a relation between a set of OWL individuals (things) that are involved in this relation (i.e. share some mutual properties).

- OWL individuals representing things that share certain mutual properties would be linked to the same instance of the respective relation class (i.e. an instance of the relation) using a special OWL object property (thus, two or more things would be indirectly linked together via being linked to the same relation class instance)[36]

The first approach (using OWL object properties) may seem easier and more straightforward (compared to the second one), and it is popular in current OWL practice. However, we will show that from the ontological standpoint it has some drawbacks which limit its usefulness for representing mutual properties in an ontologically consistent way, and that the second approach is more universal and allows better ontological interpretation. In the next two sections we provide some theoretical considerations guiding our choice of the representation method for mutual properties, and analyze both mechanisms in more detail to propose rules and guidelines on their use. We argue that most cases of mutual properties can be represented using a second method (i.e. using special classes to link individuals), while the use of the first method (i.e. using OWL object properties to directly link two OWL individuals representing substantial things) should be limited to some specific cases only.

4.2.4.3 Analysis and theoretical considerations

The issue of the representation of mutual properties in conceptual models has been discussed in several prior research works in the area of the ontological analysis of conceptual modeling languages.

In the ontological analysis of the Entity-Relationship Model (ERM) and of the relationship construct in particular, Wand et al. (1999) proposed that all mutual properties should be represented using relationship construct, whereas the entity construct should only be used to represent substantial things in a domain. In that paper, the authors further suggest that each mutual property should be represented by a separate relationship construct.

In the research on the use of UML for conceptual modeling (Evermann 2003; Evermann & Wand 2001a,b) Evermann and Wand propose a slightly different approach for interpreting and modeling mutual properties – the approach based on interactions among things. In particular, the researchers note that according to Bunge's ontology mutual properties usually occur together, and that many mutual properties arise out of interactions among things. Most interactions give rise to some mutual properties. For example, a student's enrollment at a university (that can be viewed as a result of the '*enrollment*' interaction) gives rise to some mutual properties such as '*tuition fee balance*' and

[36] A discussion of such representation of N-ary relationships in OWL can be found in Noy & Rector (2004).

'registration status'. A *'book loan'* interaction between a library member and a library leads to the acquisition of the mutual properties such as *'DateOut'* and *'DateDue'* by both the library and the library member. For UML conceptual models, Evermann and Wand (2001a,b) propose that such sets, or bundles, of mutual properties arising out of the same interaction should be modeled together - as attributes of a single UML association class (whereas ordinary classes should only be used to represent classes of substantial things). They interpret a UML association class itself as a set of related concurrent mutual properties arising out of the same interaction. Consequently, the researchers require that different association classes should be used for the sets of mutual properties that are not necessary concurrent (e.g. they arise out of different interactions). Consequently, two ordinary UML classes (that represent substantial thing classes) may be linked by more than one association class.

This thesis adopts (with some modifications) a similar approach as a basis for modeling of mutual properties in OWL. Specifically, we agree with Evermann and Wand (following Bunge's ontology) that identifying and analyzing interactions between things can help determine what mutual properties are shared by these things and how these mutual properties are grouped together. Thus, we suggest that if an interaction between some substantial things is within the scope of a model and if such an interaction gives rise to a set of (usually concurrent) mutual properties, then these properties should be modeled together. Since OWL does not provide a separate construct similar to the construct of 'association class' in UML, we propose, as an alternative, to define special purpose classes - 'relation' (or 'interaction') classes to perform a role similar to association classes in UML or to relationships with attributes in ERM. The next section discusses the proposed method of representing bundles of mutual properties in OWL in more detail, provides an ontological interpretation for the elements of the proposed representation model, and formulates modeling guidelines and rules related to representation of mutual properties in OWL.

4.2.4.4 Modeling bundles of mutual properties in OWL using interaction (relation) classes

In this section we first discuss a simple example to illustrate the suggested approach for modeling mutual properties, and then summarize the approach in more general form proposing relevant modeling rules and guidelines.

Let us consider the following example: a person is employed by a company. From the standpoint of Bunge's ontology, a company and an employee (a person) are both substantial

things which interact (e.g. via *'employment'* interaction), and, as a result, acquire and share a number of mutual properties, such as *'job title'*, *'salary'*, *'employment start date'*, and others. Note that we may also consider a more general mutual property - *'involved in an employment interaction'* – also shared by these two things (a company and an employee). According to Bunge's ontology, the relationship between this more general property and the more specific properties in a bundle ('job title', 'salary', etc.) is that of *property precedence* – the possession of any of the more specific mutual properties (e.g. 'having a certain salary' or "having job title') by a set of things implies the possession of the preceding generic one ('involved in employment interaction'). This observation regarding the existence of a general property preceding a bundle of more specific mutual properties helps in the interpretation of some constructs used in the modeling approach proposed below.

The employee-employer relationship example is presented schematically in Figure 2. We have two substantial things, each of those things possesses some intrinsic properties (here *'name'* and *'address'* properties are shown). Both things also share several mutual properties (arising out of an employment interaction) such as those shown on the diagram[37].

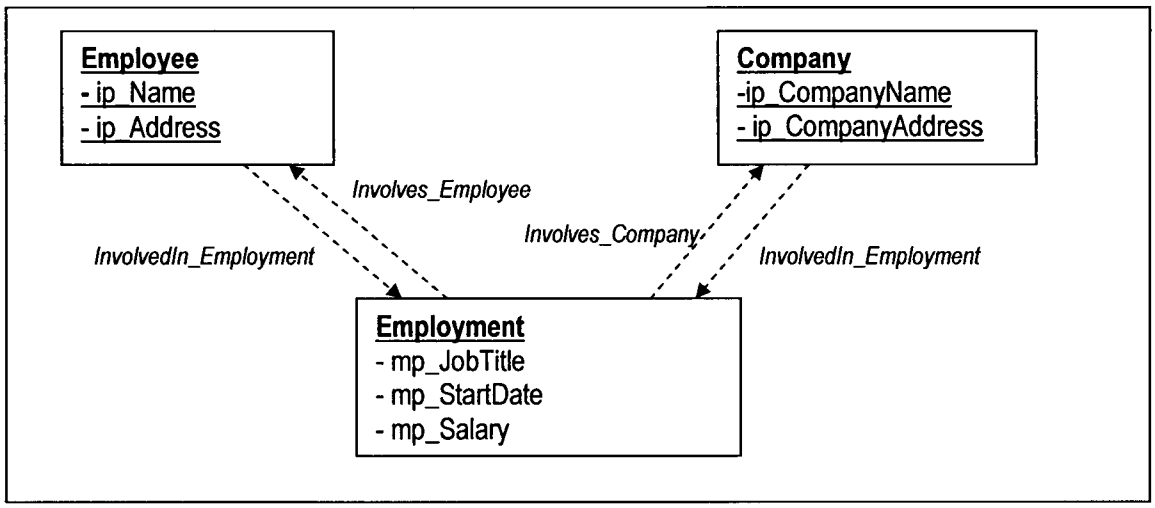

Figure 2: Schematic representation of the employment interaction example

In OWL, we can define two classes – Employee and Company; the instances of these classes (OWL individuals) would represent specific employees and companies (i.e. substantial things). Intrinsic properties of each of these classes would be represented as OWL properties in accordance with Guidelines 3, 4 and Rules 3, 4 (datatype or object properties depending on the selected property value representation method, as discussed in section 4.2.3).

[37] As proposed earlier, we use prefixes in property names to indicate the type of ontological property represented (such as 'mp_' for mutual properties or 'ip_' for intrinsic properties)

To represent in OWL the mutual properties shared by these substantial things, we propose defining a special class, e.g. Employment, which corresponds to the 'employment' interaction (and relates Employee and Company classes). This class does not represent any substantial thing (or a class of substantial things) but rather stands for a bundle of mutual properties arising out of the same interaction between the two substantial things (we can also view this class as a representation of the interaction itself). The mutual properties in this bundle (e.g. 'job title' or 'start date') would be represented as OWL properties (in accordance with Guideline 3). To represent that these mutual properties are shared by the substantial things Employee and Company and arise out of a particular interaction, the OWL properties representing them are associated with the interaction class Employment (and its individuals, respectively), rather than directly with the Employee or Company classes and instances[38].

In addition, we suggest that the classes representing substantial things (i.e. Employee and Company) should be linked to the interaction class Employment (and, thus, indirectly, to the mutual properties they share) using a special OWL object property such as InvolvedIn_Employment. This OWL object property not only serves a practical purpose of linking substantial things to a bundle of mutual properties they possess but also can be interpreted ontologically as a representation (using OWL property construct) of the earlier discussed generic mutual property 'being involved in employment interaction', which precedes a bundle of mutual properties represented as properties associated with the Employment relation class.

To summarize the example above, we propose the following approach to represent a bundle of mutual properties shared by substantial things and acquired as a result of an interaction:

- A special 'relation' (or interaction[39]) class is created to represent the 'connection' existing between those things; this class can be viewed as a representation of the bundle of properties as a whole, and as a representation of the interaction among these things;
- Individual mutual properties in a bundle are represented as OWL properties and are

[38] This way of representation ensures that for each shared generic mutual property both an employee and a company individuals will have the same value. As we discussed in section 4.2.1 (issue 4) if we were to associate each mutual property from a bundle with employee and company classes separately (rather than indirectly through a relation class), this could lead to a company and an employee instances sharing these properties to possess different values for these properties. Also, the 'bundling' of the properties (concurrency) would not be represented in such case.

[39] The term "interaction class" was suggested by P. Bera as an alternative to the term 'relation class' and was subsequently used in a recent paper by Bera *et al.* (2005) (which incorporates some results of this thesis). Therefore, for consistency with this and future work we will use the term 'interaction class'

associated (through class axioms and individual facts) with the interaction class representing the bundle,

- OWL classes/individuals representing the interacting substantial things (which share the bundle of mutual properties) are connected to the relation class (and its instances) through a special OWL object property. This property can be interpreted as a representation of an ontological mutual property of 'being involved in the relationship of interest' (or of 'being involved in a specific interaction'), which precedes each of the more specific mutual properties in the bundle.

To formalize the approach, we propose several guidelines and modeling rules (which also take into consideration the earlier suggested guidelines and rules).

Guideline 5: *In OWL ontologies modeling real world domains, a set of mutual properties of substantial things arising out of the same interaction should be represented as OWL properties associated with the specially defined OWL class – an interaction class*

Guideline 6: *Each interaction class represents a set of related concurrent mutual properties (usually arising out of the same interaction). Different interaction classes should be used if sets of properties are not concurrent and/or pertain to different interactions.*

Since interaction classes are special purpose classes and their instances do not represent substantial things, then in accordance with Modeling Rule 1 and Corollary 2, they should be declared subclasses of the Non_Substantial_Thing upper level class:

Modeling Rule 5: *Interaction classes should be modeled as subclasses of the upper level class* Non_Substantial_Thing *(since they do not represent substantial ontological things)*

We can further distinguish interaction classes from other types of classes in OWL ontologies by creating an additional upper level class for this type of classes as well as by using naming conventions (such as prefixes in class names). These suggestions are summarized in the following rule:

Modeling Rule 6: *To further distinguish interaction classes from other types of classes in OWL ontologies, additional methods can be employed:*

- *A special upper-level class* Substantial_Thing_Interaction, *can be created as a subclass of the upper-level class* Non_Substantial_Thing. *All interaction classes then would be modeled as subclasses of this class* Substantial_Thing_Interaction *(which would also automatically make*

them subclasses of the Non_Substantial_Thing *class);*

- *Also, naming conventions can be used in naming interaction classes and instances for easier identification (e.g. a prefix* I_ *or* R_ *(which stands for 'interaction' or 'relation')*

The next rule places some restrictions on how individual mutual properties in a bundle should be modeled:

Modeling Rule 7: *Each individual mutual property in a bundle of concurrent properties (represented by some interaction class) should be modeled as an OWL property in accordance with the following rules:*

- *The domain of each property should be the interaction class representing the bundle*
- *Use of a prefix (e.g.* mp_*) is recommended in the mutual property name to distinguish it from other types of properties (to conform to Guideline 4)*

The following modeling rule proposes how OWL classes that represent substantial things participating in some interaction (and thus, sharing some properties) should be linked to their respective interaction classes (which represent the bundles of mutual properties that these things share as a result of the interaction). Such linking allows tracing interactions in which a specific thing is involved and determining which mutual properties this thing possesses as a result of being involved in a certain interaction (i.e. mutual properties are linked to substantial things indirectly, through the respective interaction classes):

Modeling Rule 8: *A special OWL object property should be defined to link OWL classes (and their instances) that represent substantial things sharing a set of mutual properties to the interaction class that represents this set of shared mutual properties:*

- *This OWL object property represents the ontological mutual property of having the relationship of interest (or participating in the respective interaction that gives rise to that set of mutual properties);*
- *The domain of this OWL object property should be defined as a union of the classes that represent interacting substantial things related by the respective interaction class; the range of this object property should be the respective interaction class;*
- *Use of naming conventions (e.g. a prefix* 'InvolvedIn_' *combined with the interaction class name) is recommended for such object property to explicitly show that the substantial things possessing it are involved in a particular interaction (to which this object property links them).*

For instance, in the earlier example we defined the property InvolvedIn_Employment to link Employee and Company instances to instances of the interaction class Employment (which in turn is associated with a number of mutual properties shared by Employee and Company)

To improve model understandability, we also recommend that interaction classes should be linked back to the classes of substantial things involved in the interaction. This can be achieved by introducing additional object properties to link the interaction class to each substantial thing class participating in the interaction. In the employment example above, we can create two properties Involves_Company and Involves_Employee, which would link instances of the interaction class Employment to the corresponding instances of Company and Employee, respectively. Such linking would help determine what types of instances participate in what types of interactions as well as trace the relationships at instance level. The next rule formulates this suggestion in a general form:

Modeling Rule 9: *For each class of substantial things involved in an interaction, a special OWL object property should be defined to link the OWL interaction class that represents the shared set of mutual properties back to the OWL classes (and their instances) representing the involved substantial things sharing this set of mutual properties:*

- *The domain of this object property should be the respective interaction class; the range of this property should be defined as a union of the classes that represent interacting substantial things that participate in this interaction;*
- *Use of naming conventions (e.g. a prefix* 'Involves_' *combined with the respective substantial thing class name) is recommended for such object properties to show explicitly that this property links the interaction class to a specific class of things involved in the interaction.*

We have implemented in OWL the above discussed example of employment relation between a company and an employee in accordance with the proposed rules. We also included several instances (individuals) to this example ontology to demonstrate how interaction instances and their specific mutual properties should be declared and linked with the respective instances representing substantial things. The OWL RDF/XML implementation of this example and some diagrams of ontology classes, instances and properties are included in Appendix A.

4.2.4.5 Notes on using OWL object properties to directly link OWL individuals representing substantial things

In the previous section we proposed a general method of modeling sets of mutual properties shared by some substantial things using interaction classes. This method allows consistent modeling of mutual properties irrespective of how many things (two or more) share these properties and how many concurrent mutual properties are in a set.

As discussed in section 4.2.4.2, an alternative method of 'direct linking' of OWL individuals is using OWL object properties. Recall that an OWL object property is a directed binary relation **P(*x*,*y*)** where ***x*** and ***y*** both are OWL individuals. Recall also that the linked individuals (**x** and **y**) play different roles in this binary relation - the individual ***y*** is considered a value of the property **P** for the individual ***x*** (for example, compare: isParentOf(x,y) vs. isChildOf (y,x), isOwnerOf (PersonX, PetY) vs. hasOwner (PetY, PersonX)).

In principle, OWL object properties can be used for representing some mutual properties. More specifically, using OWL object property construct, one can directly connect *two* OWL individuals representing substantial things that share some mutual property. For example, in the example discussed earlier of a person employed by a company, the property of 'being employed' could be modeled by defining an OWL object property IsEmployedBy (Employee, Company) with the class Employee as its domain and the class Company as its range (both would be modeled as subclasses of the class Substantial_Thing). In addition, we could also define an inverse object property Employs (Company, Employee), which, in fact, would be a representation of the same mutual property but from the company's perspective (i.e. property names reflect the roles of the employee and company in the relationship, respectively).

This way of representing relationships between OWL individuals is commonly used in current OWL practices. However, from the ontological standpoint the use of OWL object property construct for representing mutual properties has a number of limitations.

First, as noted earlier, this method only allows linking *two* OWL individuals, whereas in Bunge's ontology mutual properties can be shared by any number of things. Even though in some cases N-ary relations can be replaced by a set of binary ones, the use of such binary representations instead of the N-ary one may lead to a loss of information conveyed by a model (Wand *et al.*, 1999).

In addition, this method of representation does not allow 'grouping' concurrent properties and their specific values, which may lead to ambiguities and information loss. For example, the same person may be employed by several companies and for each of these

companies he has a different start date, job title and salary. Using only OWL object properties as direct binary links between employee individual and different company individuals we would not be able to correctly represent these properties in such a way that allows determining which values of these properties are related to which employment (i.e. with which company) for this person. For example, this person would have multiple values for properties such as WorksFor and hasSalary (represented as statements of a form WorksFor (PersonX, Company1), WorksFor (PersonX, Company2), hasSalary (PersonX, $40,000), hasSalary (PersonX, $50,000). However there is no way here to determine which salary a person has in which company because we only have binary properties.

A similar issue (related to the use of binary links between classes) was raised by Evermann (2003) with respect to the use of simple (or 'ordinary') associations for modeling mutual properties when using UML for conceptual modeling[40]. He argues that simple associations are employed in UML to enable message passing, which is a design related concept that does not have a Bunge-ontological equivalent. For this reason as well as in the interest of ontological clarity[41], he proposed that UML associations are ontologically excessive and should not be employed for conceptual modeling of mutual properties – such properties should only be modeled as attributes of association classes. For example, according to this recommendation the mutual property of 'being employed' would not be modeled in UML as a simple association between Person and Company classes, but instead an association class should be used with the attributes such as 'Job Title', 'Salary', etc. (which represent the mutual properties resulting from this interaction). Another rule for UML proposed in (Evermann, 2003) states that "Every ordinary association must be an association class" (Evermann, 2003, p.72).

The use of OWL object properties for modeling mutual properties of two things is somewhat similar to the use of simple associations in UML for modeling mutual properties. However, in this thesis research we refrain from completely proscribing the use of OWL object properties for direct linking of individuals representing substantial things, as this is a quite useful and widely employed OWL mechanism, which in many cases enables concise representation and efficient reasoning[42]. Rather, we propose that the use of OWL object properties in this context (i.e. for linking substantial thing instances and classes) should be

[40] In UML models, classes can be either linked by simple associations (without the use of association classes), which is somewhat similar conceptually to OWL object properties, or via association classes

[41] Specifically, this research tried to avoid the situation where mutual properties are modeled by two different UML constructs – associations and attributes of association classes.

[42] For example, additional property characteristics may be easily defined for OWL object properties such as transitivity or symmetry (which may be more difficult to represent when using the interaction class approach)

restricted to a number of particular situations only and be guided by certain rules.

Specifically, we suggest that in ontologies representing real-world domains, OWL object properties can be used to associate two substantial things directly only in the following two cases: 1) when only the fact of the existence of the relationship between two substantial things (a property in general) needs to be represented (and interaction details and its related mutual properties are outside of the scope), or 2) when a non-binding mutual property between two things has to be represented. Below, each of these cases is discussed in more detail.

1) Cases when only the existence of a general relationship (mutual property) between only two substantial things needs to be represented

Details of certain interactions between things (and thus certain mutual properties arising out of them) may be outside of the scope of a particular ontological model. For example, let us consider a mother-child relationship between two persons. A modeler developing an ontology about people may not be interested in the interaction that gave rise to this relationship (i.e. 'child birth') and, thus, would not need to represent in this ontology all the details of the 'child birth' interaction and of the associated mutual properties (such as birth date and time, birth place, etc.). In such a case, one may only need to represent the existence of the relationship between a pair of OWL individuals representing persons (i.e., the fact that persons have mothers and/or are mothers of some persons). In other words, a modeler may only want to be able to state that a particular person is a mother of (or has as a mother) another particular person. For such cases, it is sufficient to use of a pair of mutually inverse OWL object properties such, as for example, isMotherOf (PersonX, PersonY) and hasMother (PersonY, PersonX). Using interaction classes would be too cumbersome and unnecessary in this case.

Note that in the cases as above OWL syntax does not require defining inverse properies for OWL object properties[43]. However, from the ontological standpoint we argue that if a general mutual property (the existence of some relationship) is represented by an OWL object property, then a pair of mutually inverse properties should be modeled in the ontology so that each of the two individuals representing substantial things sharing this mutual property is associated with the OWL property representing this mutual property (rather than just one of the two individuals). Similarly, at the instance level, if both properties are defined then the statements regarding the relationship can be made for both

[43] That is, an OWL ontology (e.g. some ontology about people) would still be a valid ontology even if only one of the two mutually inverse properties (e.g. only isMotherOf) is defined.

mothers and children (i.e. for all substantial things sharing a property rather than just for one side of the relationship)[44].

The representation approach outlined above is summarized in the following rule:

Modeling Rule 10: *In OWL ontologies modeling real world domains, if an OWL object property is used to represent the existence of a relationship between* ***two*** *substantial things (i.e. a mutual property of having this relationship), then*

- *Two mutually inverse OWL object properties should be defined to link two individuals - instances of two classes A and B respectively, where A and B are classes of instances representing substantial things having a relationship*
- *One of these two properties should have the class A as its domain and the class B as its range, while the other property should have the class B as its domain and the class A as its range*
- *Use of naming conventions (such as a prefix 'mp_')) is recommended for both these object properties to indicate that they represent a mutual property (existence of the relationship) shared by the two things*

For example, there may be two classes in some ontology, Pet and PetOwner, the instances of which represent substantial things in our domain (i.e. specific pets and pet owners, respectively). One may want to represent a general 'pet ownership' relationship (i.e. a shared mutual property) between pet owners and their pets. In accordance with Rule 10, one would define two mutually inverse properties, e.g. mp_hasPet (with the domain PetOwner and the range Pet), and mp_hasPetOwner (with the domain Pet and the range PetOwner). To represent relationships between specific pets and pet owners, onw would include the assertions (facts) in the ontology of the form mp_hasPet (X, Y) and mp_hasPetOwner (Y, X).

To conclude the discussion of this example, we would like to address an additional issue related to ontology modification or expansion. Even though some interaction and its related mutual properties may be considered outside of the scope of the ontology by one modeler, another modeler may decide to reuse and expand the initial ontology and may consider this particular interaction to be within the scope of this expanded model. In this case, he or she can expand the initial ontology by using the 'interaction class method' of representing mutual properties as in section 4.2.4.4 (i.e. add a special interaction class to represent the set of mutual properties arising out of the interaction and link this interaction

[44] Of course, one needs to ensure consistency in the values for mutually inverse properties, i.e. HasMother (X, Y) implies isMotherOf (Y, X) if properties are declared to be mutually inverse in OWL (otherwise the ontology would be inconsistent). Reasoning applications are able to detect such inconsistencies.

class to the relevant substantial thing classes using OWL object properties).

The presence in the expanded ontology of both the mutually inverse properties directly linking substantial thing classes (and representing the general existence of some relationship) and of the interaction class with its associated detailed mutual properties creates redundancy, which, albeit allowed by OWL[45], however requires extra efforts from modelers to ensure consistency in the respective property value statements for individuals. For example, we may end up having both a pair of properties IsEmployedBy / Employs linking Employee and Company instances directly as well as the interaction class Employment linking the Employee and Company instances indirectly to a set of mutual properties (salary, job title etc). In this case, consistency needs to be ensured both at the class and at the instance level. To help enforce such consistency more complicated OWL mechanisms can be employed, such as recently proposed SWRL rules language statements (Horrocks & Patel-Schneider, 2004), which allow the representation of the dependence between property values in the of so called rules (for example, we can state that having certain property values for the mutually inverse properties for the individuals representing certain substantial things implies that each of these individuals is also linked to the same instance of the interaction class that models a set of mutual properties pertaining to this interaction).

From the conceptual modeling standpoint, such redundancy is undesirable since it undermines ontological clarity and consistency, and thus may lead to problems with model understanding and interpretation by people and, consequently, by applications. Therefore, we recommend that wherever possible modelers should use the more universal 'interaction class' approach if there is any possibility that in the future more details about interactions and their associated mutual properties may need to be represented or if more than two interaction participants sharing properties may come into the scope of the ontology (the latter would also require the use of interaction classes rather than direct linking using object properties).

2) Using OWL object property for modeling non-binding mutual properties of two things

According to Bunge's ontology, some mutual properties – non-binding properties - are not associated with any interaction, for example, "thing A is behind thing B" or "thing A is older than thing B". We propose that in cases when such a non-binding property is shared

[45] OWL allows redundancy, and redundant statements are often included in OWL ontologies to allow greater flexibility and efficiency in reasoning. However, redundancy may lead to ambiguity and logical inconsistency if not used carefully. It is up to modelers to ensure the logical consistency of the ontology, and reasoning tools are able to perform consistency checking.

by only two things, then it can be represented using an OWL object property, for example, isBehind (thingA, thingB) or isOlderThan (thingA, thingB). Therefore, we propose the following rule:

> ***Modeling Rule 11:*** *A non-binding mutual property shared by two substantial things can be represented using an OWL object property to link the two OWL individuals representing those two things.*

Note however, that irrespective of whether a mutual property is binding or non-binding and whether interaction is outside of the scope, in cases when more than two things share some property, we would still need to use the relation (or interaction) class approach to link more than two things indirectly (since OWL object properties can only link two things)

To summarize the analysis of mutual property representation, we have discussed two main methods of mutual property representation: 1) *using interaction (relation) classes* – a more universal approach applicable for indirect linking of two or more things sharing any number of mutual properties, and 2) *using OWL object properties for direct linking of two things only* – an approach applicable only in specific cases. While these two approaches employ and combine OWL constructs in different ways to represent mutual properties and things possessing them, in both approaches the ontological construct of mutual property is represented by OWL property construct, which is consistent with Guideline 3 and with the proposed mapping of ontological properties to a subset of OWL properties. Guided by the ontological foundations, we have roposed some rules and guidelines recommending in which situations and how each of these representation methods should be used.

4.3 REPRESENTATION OF CLASSES

The concept of class is widely used in conceptual modeling, object-oriented design, data modeling, knowledge representation, and ontological engineering. This is not surprising since in everyday life people automatically classify things into categories and relate to the things by these classes (Parsons & Wand, 2000). We usually conceive of a thing as an instance of a certain type of things. The first step in constructing a conceptual model or an ontology is usually the identification of a set of the fundamental concepts to describe a domain. These concepts are represented in the model as classes or types. Classification involves forming concepts (also termed categories or classes) to abstract common characteristics of instances and assigning new instances into these categories (Parsons & Wand, 1997).

The discussion of the representation of classes is structured as follows. First, we provide a brief description of classes and mechanisms of their definition in OWL. Then, we discuss how classification is understood in the Bunge's ontological model and in some other theories such as the concept theory. We review classification-related concepts of Bunge's ontology, such as classes, kinds and functional schema, and compare them to OWL classification related constructs and mechanisms. Based on this comparative analysis and theoretical foundations from ontology and classification theory, we propose recommendations on modeling classification related aspects of real world domains in OWL in a way consistent with ontological and cognitive foundations. In addition, we propose an ontological interpretation for some of the classification-related functionality in OWL.

4.3.1 Classes in OWL – an overview

To facilitate further analysis and comparison of classification in OWL and in Bunge's ontology, this section briefly reviews what classes are in OWL ontologies and how they can be defined.

OWL is based on the Description Logics (DL) and differs from other conceptual modeling and object-oriented modeling languages (such as ERM, UML or OOM) in its approach to representing classes and associating them with properties. For example, in object-oriented languages, such as UML, a class is a *description* of a set of objects that share the same attributes and operations. The classes play a role of templates or 'object factories' – objects depend on classes and cannot exist without them. Also, in object-oriented models properties are always linked to classes while in OWL this is not necessarily the case (as we will see in the next section). Objects are created from class templates, that is, an object as a

class instance possesses all the methods and attributes defined for its class (Evermann, 2003). Thus, an object-oriented class (e.g. UML class) is a description of a number of similar objects.

In OWL, classes are used to represent concepts in a domain of discourse. They provide an abstraction mechanism for grouping resources with similar characteristics. The OWL language guide (McGuinness et al., 2004) states: *"Many uses of an ontology depend on the ability to reason about individuals. In order to do this in a useful fashion we need to have a mechanism to describe the classes that individuals belong to and the properties that they inherit by virtue of class membership. We can always assert specific properties about individuals, but much of the power of ontologies comes from class-based reasoning."*

Every OWL class is associated with a set of *individuals* called the *class extension*. Individuals usually represent objects in the domain of discourse. The individuals in the class extension are called *instances* of the class. Generally, it is intended that classes should correspond to naturally occurring sets of things in a domain of discourse and individuals should correspond to actual entities that can be grouped into these classes (McGuinness et al., 2004).

Classes in OWL are described by so-called *class descriptions*, which are combined into *class axioms*. A *class description* describes an OWL class, either by a class name or by specifying the class extension of an unnamed (or anonymous) class. *Class axioms* contain components that state necessary and/or sufficient characteristics of class membership.

In particular, OWL allows the following types of class descriptions:

1) class identifier

A class in OWL can simply be declared by name, without specifying any further information about it. For example, we can declare a class Human as follows:

```
<owl:Class rdf:ID="Human">
```

If a class is declared in this way, statements can be made about individuals being instances of this class, for example:

```
<Human rdf:ID="Mike">
```

However, this basic representation does not tell much about the concept represented by the class or about common properties of its instances, and thus, while valid syntactically, is not particularly useful on its own.

2) exhaustive enumeration of individuals that together form the instances of a class

(i.e. enumerating class extension)

A class in OWL can be defined by enumeration of all its instances (using a special constructs owl:oneOf and rdf:parsetype="Collection"). For example, the following syntax defines an unnamed class of all continents (which are represented as OWL individuals[46]):

```
<owl:Class>
   <OWL:one of rdf:parseType="Collection"
   <owl:Thing rdf:about="#Eurasia">
   <owl:Thing rdf:about="#Africa">
   <owl:Thing rdf:about="#NorthAmerica">
   <owl:Thing rdf:about="#SouthAmerica">
   <owl:Thing rdf:about="#Australia">
   <owl:Thing rdf:about="#Antarctica">
<\owl:Class>
```

3) property restriction

As already discussed briefly in section 2.2.4., a property restriction describes an anonymous (unnamed) class of all individuals that satisfy certain constraints on a property. Property restrictions are used as parts of class descriptions. OWL distinguishes two kinds of property restrictions: *value constraints* and *cardinality constraints.* A *value constraint* puts constraints on the range of the property when applied to a particular class description (within the scope of a particular class axiom). A *cardinality constraint* puts constraints on the number of values a property can take, in the context of a particular class description.

Through the use of property restrictions as parts of class axioms, classes can be associated with properties. That is, necessary and/or sufficient conditions for class membership can be specified in a class definition through constraints on OWL properties, specifically, on the allowed ranges of property values (value constraints) and/or on the number of values allowed for specific properties (cardinality constraints).

For example, one can consider an object property hasParent that links an OWL individual to another OWL individual that represent its parent. Using the existential qualifier owl:someValuesFrom on this object property, we may specify an unnamed class of individuals who have at least one parent who is a student (i.e. the set of individuals that possess at least one value for the object property hasParent that is an instance of the class Student):

```
<owl:restriction>
        <owl:onProperty rdf:resource="#hasParent" />
        <owl:someValuesFrom rdf:resource="#Student" />
</owl:restriction>
```

[46] The syntax <owl:Thing rdf:about="..."> declares an OWL individual as an instance of the built-in top class owl:Thing.

The instances of this anonymous class would be all OWL individuals that satisfy the specified condition.

Similarly, we can consider another OWL object property, hasChild, which would link an OWL individual to other OWL individuals that represent its children. Using minimum and maximum cardinality constraints on this object property, we can specify an anonymous class of individuals that have at least 1 and at most 4 children:

```
<owl:restriction>
   <owl:onProperty rdf:resource:='#hasChild"/>
        <owl:minCardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema
          #nonnegativeInteger">1</owl:minCardinality>
   <owl:maxCardinality
     rdf:datatype="http://www.w3.org/2001/XMLSchema
     #nonnegativeInteger">4</owl:maxCardinality>
</owl:restriction>
```

As will be shown later in section 4.3.4, the OWL method of defining classes based on property restrictions is the most relevant to this work, since we have adopted the ontological view of classes based on sets of shared properties, and thus require a representation mechanism in OWL that would allow the association of classes with properties.

4) Set operators: intersection, union and complement of class descriptions

OWL allows the construction of more complex (nested) class descriptions by combining class descriptions (of anonymous or named classes) using set operators. The three types of class description combinations – *intersection*, *union*, and *complement* – correspond to the standard set-theoretic operators - the AND, OR and NOT operators on class extensions, respectively. OWL provides three language constructs implementing these operators - owl:intersectionOf, owl:unionOf, and owl:complementOf. They can be used in nested class descriptions, with either one (complement) or more than one class (union, intersection).

An owl:intesectionOf statement describes a class for which the class extension contains precisely those individuals that are members of the class extension of *all* the class descriptions in the list. It is analogous to the logical conjunction. For example, one can state using this construct that the class WhiteCar is exactly the intersection of the class Car and of the set of things that are white in color (i.e. of an anonymous class defined as a property restriction of a form hasValue="#White" on the property hasColor).

An owl:unionOf statement describes an anonymous class the class extension of which consists of those individuals that belong to *at least one* of the class extensions of the class

descriptions in the list. It is analogous to the logical disjunction. For example, the class extension of the class Fruit can be stated (using owl:unionOf) to include the instances of both the extension of the SweetFruit class and the extension of the NonSweetFruit class.

An owl:complementOf statement describes an anonymous class for which the class extension contains exactly those individuals that *do not* belong to the class extension of the class description that is the object of the statement. It is analogous to the logical negation. For example, the class of all things that are "not meat" can be represented in OWL as the class defined as the *complement* of the class Meat, i.e. the extension of this class "non-meat" contains all individuals that do not belong to the class Meat.

Class descriptions serve as building blocks for defining classes through *class axioms*. Class axioms contain components that state necessary and/or sufficient characteristics of class membership. OWL provides three language constructs for combining class descriptions into class axioms: rdfs:subClassOf, owl:equivalentClass, and owl:disjointWith.

Using rdfs:subclassOf construct, we may state - in a class axiom form - that a subclass relation exists between two OWL classes. If some class description C1 is stated to be a *subclass* of some other class description C2, this means that the set of individuals in the class extension of C1 is a subset of the set of individuals in the class extension of C2. Subclass relations provide *necessary conditions* for belonging to a class (i.e. an instance of the class C1 is *necessarily* an instance of the class C2). An OWL class can be included in any number of rdfs:subclassOf axioms.

The construct owl:equivalentClass allows expressing in a class axiom that the two class descriptions involved have the same class extension (i.e. that both class extensions contain exactly the same set of individuals)[47]. Such equivalence class axioms allow expressing *necessary and sufficient conditions* for class membership. In particular, as will be shown later, one can define a named class as a set of individuals sharing some properties by stating that this named class is equivalent to an anonymous class defined as an intersection of certain property restrictions representing some constraints on the above shared properties.

4.3.2 Theoretical foundations - classification in Bunge's ontology and concept theory

This section briefly reviews some theoretical foundations related to classification that guide our further analysis and the development of modeling rules and guidelines for classes.

[47] Note that equivalence here means that class extensions (i.e. sets of class instances) are the same. However, the concepts represented by classes may not be equal (i.e. intensional meaning of classes may be different). For example, "Graduate student" and "Research Assistant" may be different concepts but the class extensions might be the same in a context of some university.

In particular, we discuss how classification is viewed in Bunge's ontology and mention some basic concepts of classification (or concept) theory.

In Bunge's ontology, the primary constructs are *things* and *properties*. Classification implies the existence of things and their properties (Parsons & Wand, 2000). Things can have one or several properties in common. Furthermore, their properties might be subject to the same laws. This gives rise to the concepts of *class*, *kind* and *natural kind*. The following definitions taken from Wand et al. (1999) and Evermann (2003) are based on original definitions from Bunge's works (Bunge 1977, 1979):

- *A class* in Bunge's ontology is a set of things possessing a common property
- *A kind* is a set of things having several properties in common
    - Note that if the shared set of properties is finite, then a kind is also a class (since we can always consider a compound property of possessing all the properties in this finite set defining a kind).
- *A natural kind* is a set of things adhering to the same laws
    - Since by definition laws relate properties, a natural kind implies a set of properties as well. As laws determine possible states, a natural kind is the set of things that exhibit like behavior.

It is important to note that all the above concepts (class, kind and natural kind) refer to *sets of things*, not descriptions or templates of things. Also, they all are defined over *existing* sets of things. This implies that in Bunge's ontology, there can not be a class, a kind or a natural kind without any members.

Another concept of Bunge's ontology that is relevant to the discussion of classification is the concept of *functional schema.* Recall that according to Bunge's ontology, humans conceive of things in terms of models of things. Similar things can be represented by the same model. A *functional schema* is a set of state functions, which are usually functions of time indicating the value of the properties of a thing at a particular point in time (although other frames of reference are possible). These state functions represent properties in general of the things, whereas the values of these functions express individual properties of things (Bunge 1977; Evermann & Wand, 2001). Any thing can be described by more than one such schema depending on a model's purpose. Because a functional schema is just a model created for a certain purpose, the properties represented in a functional schema depend upon the circumstances and the purpose of modeling things. For example, a person may be viewed as an employee, a customer, or a taxpayer, and each of these views can be represented by different functional schemas focusing on the properties important for a

specific view (Wand *et al.*, 1999).

There exists an important relationship between the concepts of class, kind and natural kind and the concept of a functional schema. As pointed out in Wand *et al.* (1999), a functional schema is based on a representation of a *partial* set of properties and, hence, can be conceived as *a model of a class or kind.* In other words, a functional schema is a model of a set of things that have similar properties and laws. This observation will be important for our further analysis of classes in OWL.

Classification issues are also the subject matter of another research area – *classification, or concept, theory.* This discipline has emerged from the intersection of cognitive psychology and linguistics. Classification theory addresses questions of what are concepts, classes, and categories, why they are important and how they are acquired (Parsons & Wand, 1997). According to the concept theory, a class identifies a measure of similarity among its instances. One way in which similarity can be operationalized is through the sharing of properties.

Parsons & Wand (1997, 2000) note in their research on the application of the classification theory and ontology to IS modeling that two major functions of classification are recognized in the concept theory. First, classification provides *cognitive economy.* By identifying a group of entities as similar in some way(s) through classification (for example, focusing on a number of shared properties), some knowledge can be stored within the class rather than repeated for all instances. In that respect, a classification is useful when there are meaningful differences among classes, where "meaningfulness" depends on a model's purpose or context (Parsons & Wand, 1997). Second, classification enables *inference.* It is often possible to classify an instance based on a strict subset of its properties (also termed *class identifying properties*), and to infer the presence of other, unobserved, properties (*inferred properties*) by virtue of that classification (Parsons & Wand 1997; Parsons & Wand, 2000).

To summarize the theoretical discussion, in both ontology and classification theory the notion of thing or instance is fundamental and precedes the notion of classes. In other words, humans first recognize that things exist, then based on various considerations (which are usually dealt with in cognition) form classes to organize their knowledge about the properties of individual things. Classes (or concepts) are abstractions created by humans in order to describe useful similarities among things, and the particular choice of classes (a view) depends on the application (Parsons & Wand, 2000).

4.3.3 Comparing classes in OWL to the ontological notion of classes

Comparing classification-related concepts in OWL to those in Bunge's ontology, we can make a number of important observations.

First, classes, kinds, or natural kinds in Bunge's ontology are all *sets of things* (satisfying certain constraints such as possessing a common property, a number of common properties, or exhibiting the same type of behavior). They are *not* descriptions or templates for sets of things. OWL construct equivalent to that would be the concept of *class extension*. The class as a whole in OWL is intended to represent some concept in a domain of interest, whereas a class extension is a set of the class instances (individuals). Therefore, we propose to map classes, kinds, and natural kinds to OWL class extensions (representation mapping):

Guideline 7: *Bunge's ontological classes, kinds, or natural kinds as sets of substantial things correspond to OWL class extensions (i.e. sets of OWL individuals representing substantial things).*

Recall that in Modeling Rule 1 we proposed that OWL classes with individuals representing substantial things should be distinguished from the OWL classes and individuals that do not represent substantial things. We suggested the implementation of this distinction in OWL using two disjoint upper-level classes Substantial_Thing and Non_Substantial_Thing. From this recommendations and the Guideline 7 above follows another representation rule - a necessary condition for an OWL class extension to be considered a representation of an ontological class or kind:

Modeling Rule 12: *If the class extension of some OWL class is intended to represent an ontological class, kind, or natural kind, then such class should be declared a subclass of the upper-level class* Substantial_Thing.

However, it is important to note that not every class extension in OWL would be a representation of some ontological class or kind. First, as we mentioned earlier, some OWL classes and individuals do not represent ontological substantial things, therefore their class extensions will not correspond to any ontological classes or kinds. Also, while OWL provides a number of ways to describe OWL classes using class descriptions and class axioms, some of these descriptions do not have an ontological interpretation and may result in classes with extensions that do not correspond to any ontological classes, kinds or natural kinds.

For example, OWL allows defining a class simply by name, without specifying any further information about it and without associating any property restrictions with this class.

Then, individuals can simply be declared to be instances of such a class using individual axioms. For example, we can declare a named class Student, and also declare that some individuals are instances of this class. Such declaration would be syntactically valid in OWL:

```
<owl:Class rdf:ID="Student">
<Student rdf:ID="John_Smith">
<Student rdf:ID="Jane_Doe">
```

However, from the Bunge's ontological standpoint, the class Student defined simply by name is not a proper model (functional schema) of the set of things that are students. On one hand, one may argue that we may interpret a set of things that are students as a class, which is a set of things possessing a common property of "being a student", or even as a natural kind (since students exhibit similar behavior – attend classes, pay tuition, etc.). However, the above OWL class as a model does not represent any properties (except for the very generic one of "being a student"), therefore it is not very useful from the ontological and cognitive standpoint. Without any property information, such representation supports neither cognitive economy nor inference abilities as no information about common properties is provided (Parsons & Wand, 1997).

Similarly, OWL allows defining a class simply by the enumeration of its instances. Again, as in the previous example, no information about properties would be represented in this type of a class description, therefore, such model is not very useful and does not correspond to the ontological notion of classes, which is based on properties. Additional problem is that any enumeration of individuals can be declared to be a class in OWL, while not all sets of things can be considered classes or kinds in the ontological sense, let alone the issue of whether it would be a 'good' or 'useful' class.

Due to the above considerations, we recommend that these two methods of defining classes in OWL (only by name or only by enumeration) should not be used as a sole means of modeling classes the extensions of which are intended to represent ontological classes, kinds or natural kinds[48]:

Modeling Rule 13: *In OWL ontologies representing real world domains, if an OWL class is intended to model an ontological class or kind, then it should not be defined only by class name or only by direct enumeration of instances (i.e. without representing any information*

[48] Rather, we will propose later in this section that using class definitions based on property restriction class descriptions is a preferred method from the ontological and cognitive standpoint for ontological classes/kinds.

about common properties of class instances)[49]

Modeling Rule 1 and Corollary 2 require that all classes the instances of which represent substantial things should be modeled as subclasses of the upper-level class Substantial_Thing. From that and the above Modeling Rule 13 follows another corollary:

Corollary 7: *In OWL ontologies representing real world domains, subclasses of the class* Substantial_Thing *shoul7 not be defined simply by class name or by enumeration of instances (i.e. without representing any information about common properties of class instances)*

The requirement of the above Rule 13 (and Corollary 7) may be viewed as too restrictive by some OWL ontology developers because some lightweight ontologies are often modeled simply as class hierarchies without including any class axioms about properties of those classes. We agree that for such lightweight ontologies (taxonomies) the issue of modeling properties may not be relevant, but the main focus of this research is the development of more expressive ontologies (in OWL DL), which would include not only classes but also properties and possibly individuals. For such ontologies, we consider the Rule 13 relevant and appropriate. In fact, the OWL language guide (McGuinness, *et al.* 2004) also stresses the importance of properties stating (when discussing properties) that the "world of classes and individuals would be pretty uninteresting if we could only define taxonomies" and that "properties let us assert general facts about the members of classes and specific facts about individuals".

Another potential issue may arise in cases where modelers may believe that a certain group or type of things is different from other things (and thus deserves being modeled as a class) but they may not be able to point out specific properties which differentiate this set of things from other things. While we acknowledge that this problem may occur, we still believe that it would be beneficial for modelers (and would improve the resulting ontology) to identify one or more properties (intrinsic or mutual, which could also be participation in some interaction) that things in this class share and model them accordingly.

4.3.4 Modeling functional schemas and ontological classes/kinds in OWL

As mentioned earlier in section 4.3.2, in Bunge's ontology classes, kinds and natural kinds (which are all sets of things) can be modeled using functional schemas. A functional

[49] Note that our recommendations pertain to the conceptual modeling aspects of OWL ontologies. We do not proscribe using any class description types for classes that do not represent substantial things or for implementation related concepts. Furthermore, these types of class descriptions (named or enumerated classes) can still be present in an ontology provided that additional property based class descriptions for these classes are also included.

schema is a model of a set of things that have similar properties and laws, and hence it can be conceived as a model of a class or a kind (Wand et al., 1999). Ontological properties represented in a functional schema (via state functions) depend upon the circumstances and the purpose of modeling.

We propose that functional schemas, which are models of Bunge's classes or kinds, can be modeled in OWL as OWL class definitions based on class descriptions and class axioms that include property restrictions on OWL properties (these properties, in turn, represent the respective ontological properties shared by Bunge class members and modeled by state functions in the considered functional schema). Modeling ontological functional schemas as OWL classes defined by property restriction class descriptions, also allows modeling ontological classes and kinds as sets of things – they would correspond to the class extensions of the OWL classes representing the respective functional schemas. The proposed representation mapping is schematically depicted in Figure 3:

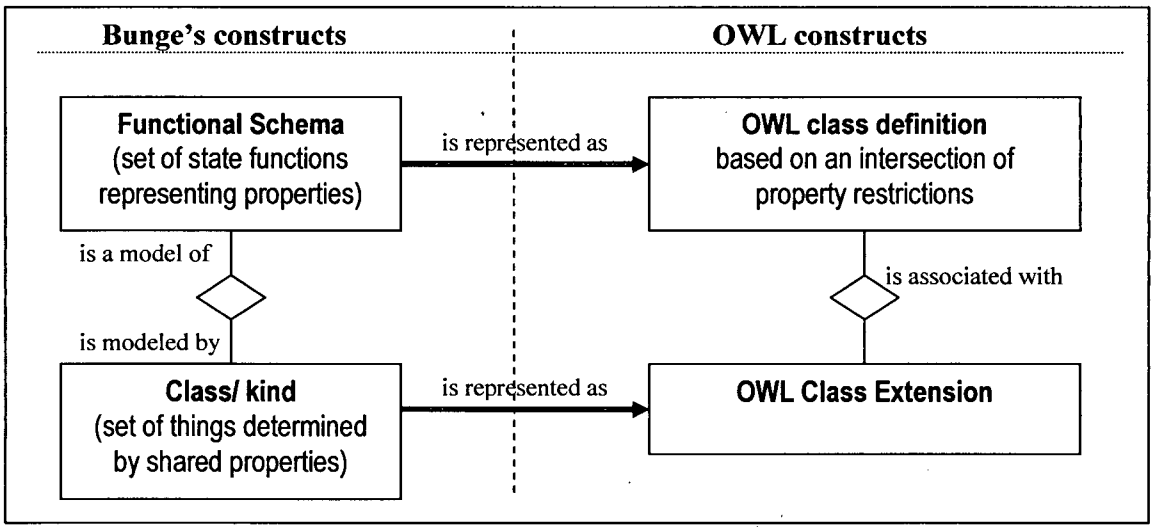

Figure 3: Mapping of Bunge's classes/kinds and functional schemas to OWL constructs

To illustrate the proposed approach for modeling functional schemas and Bunge classes or kinds), we first discuss a simple example. Then, we propose some rules on modeling classes and kinds in OWL.

For example, we can consider a class of *persons* described by a functional schema that includes state functions representing common intrinsic properties of instances (persons), for example, 'name', 'date of birth', and 'gender'. In OWL, we can represent this functional schema as follows. We declare a named class Person and three OWL properties representing the above ontological properties[50]:

[50] We are using our earlier proposed naming conventions to indicate that properties are intrinsic.

1) a datatype property ip_PersonName (with the range of values of the XML datatype String),

2) a datatype property ip_DateOfBirth (with the range of values of the XML datatype Date),

3) a datatype property ip_Gender (with the enumerated range of values: "M" or "F" which stands for male or female)[51].

Recall that in OWL, properties and classes are in general independent of each other. In order to associate classes and properties, i.e. define classes by specifying conditions on certain properties, one needs to use property restrictions in class descriptions. In accordance with our functional schema, we need to state that every instance of the class Person *necessarily* possesses each of the properties of the schema. Specifically, each Person instance has to possess some value for each of the three general properties that are modeled by state functions of the schema (i.e. a specific property corresponding to each general property). In this simple example, we assume that each person possesses exactly one value for the 'name', 'date of birth', and 'gender' properties; therefore, in OWL representation we can use owl:Cardinality constraint in restrictions on each of the properties[52].

In order to specify a necessary condition for all instances of some class in terms of a restriction on some property in OWL, we need to declare that the class (i.e. class Person in our case) is a subclass of an anonymous class of all things that satisfy this property restriction. For example, if a class Person is declared to be a subclass of all things that have name, this implies that all instances of the class Person have name. In our case we need to state that all instances of the class Person satisfy a certain restriction (of possessing a property, e.g. cardinality=1) for each of the three properties. Thus, we need to state (using class axioms) for each of the three properties, that the class Person is a subclass of the anonymous class defined by the respective property restriction (in this example, the "cardinality=1" restriction). Or, equivalently, we can state that the class Person is a subclass of the intersection of the three anonymous classes, each of which is defined by the respective property restriction. This situation is illustrated schematically in Figure 4 below. The instances of the class Person are also instances of each of the three anonymous classes defined by property restrictions: things that have name, things that have date of birth, and things that have gender. The OWL RDF/XML representation of this example can be found in Appendix B.

---

[51] The values for the property 'gender' could also be represented using special value classes and individuals as discussed in section 4.2.3.2 (in this case an object property would have to be used). Whether a datatype or object property is used is not relevant in this discussion and does not affect the proposed approach.

[52] If more than one value is allowed for a state function, then we can use minimum and maximum cardinality constraints in property restrictions in class descriptions.

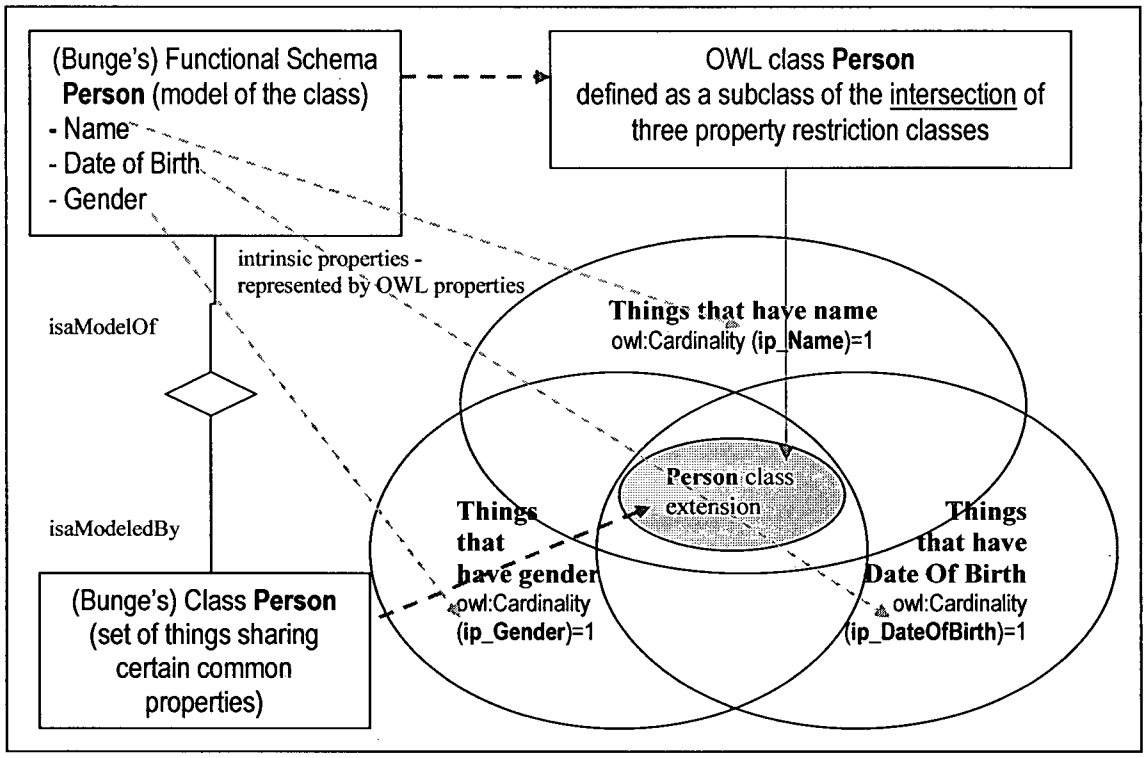

Figure 4: Schematic illustration of modeling ontological classes in OWL (an example)

Note that like in the example above, a functional schema usually represents a *partial* set of common properties of a set of things (it can be considered a view of a set of similar things). That is, for example, not all things that have gender, date of birth and name are necessary persons (e.g. it could be a pet). Therefore, we modeled this case in OWL as a *necessary* condition rather than necessary and sufficient one (necessary and sufficient condition would mean that the Person class extension is the same set as (or "equals" to) the intersection of the above three anonymous property restriction classes). If for the class in question there exists a subset of properties sufficient for classifying an instance (such properties are termed *class identifying properties* in Parsons & Wand (1997)), then in OWL ontology we can define a class axiom for that class that represents necessary and sufficient conditions for class membership (and not just the necessary ones). OWL provides a construct owl:equivalentClass to declare that the class Person is a class *equivalent* to the intersection of the anonymous classes defined by the respective property restrictions on the class identifying properties.

Note that while we used a simple cardinality constraint in the above example, in other cases other types of restrictions may have to be used (such as minimum/ maximum cardinality constraints or value restrictions). For example, in the context of the above example we may want to represent information about a class Female_Person of all persons

who are female (actually a subclass of the class Person). One of the necessary conditions in our OWL representation could be that all instances of this class possess a particular value ('F') for the property ip_Gender, i.e. they share a specific ontological property 'being of female gender'. In our case, we can represent this common property of the class instances in OWL using a restriction of the type 'owl:hasValue="F"' for the property ip_Gender[53]. That is, we can state that the class FemalePerson is a subclass of the intersection of the three anonymous classes; two of these classes are defined by property restrictions 'cardinality=1' on properties ip_Name and ip_DateOfBirth respectively, and the third anonymous class is defined by a property restriction 'owl:hasValue="F"' on the property ip_Gender (meaning that every instance of this anonymous class has a value "F" for the property ip_Gender). Alternatively, we can state that FemalePerson is a subclass of the class Person and of the anonymous class defined by a property restriction 'owl:hasValue="F"' on the property ip_Gender. The OWL RDF/XML representation of this class definition can be found in Appendix B (together with the Person class representation)

Generalizing the approach illustrated by the above examples, we propose that since ontological classes or kinds are defined in terms of properties (and can be modeled by functional schemas), they should be represented in OWL using class descriptions based on property restrictions. Specifically, we propose the following modeling rule:

Modeling Rule 14: *An ontological class or kind C, modeled by a functional schema with the state functions modeling some common properties P_1, ..., Pn of this class/kind C, can be represented in OWL as the class extension of an OWL class defined as follows:*

- *A named OWL class (e.g.* ClassC*) should be created;*
- *Each of the properties P_1, ..., Pn should be modeled by a suitable OWL property (in accordance with Guidelines 3-7 and Modeling Rules 2-11 on property representation);*
- *Class axiom(s) for the ClassC should be included that state (or imply) that all instances of the ClassC necessarily possess each property Pi[54];*
- *To achieve that, such axioms should state that the classC is a subclass of the anonymous class defined by suitable property restriction on the property Pi, for each Pi. Or, alternatively, the ClassC can be declared to be a subclass of the intersection of the anonymous classes defined by suitable property restrictions for each of the properties Pi.*

[53] In OWL, using this hasValue restriction would imply that all instances satisfying the restriction possess a specific value 'F' for the property hasGender, which in turn implies that they possess the general property hasGender itself (i.e. cardinality is not equal to 0)

[54] More specifically, usually instances would possess some value (or values) for the general property Pi.

The next rule proposes an additional representation guideline in the case when a set of class identifying properties for a class needs to be represented:

Modeling Rule 15*: If a set of ontological properties P_1, ..., P_k is a subset of common properties of a class or kind C that is sufficient to classify a thing as an instance of the class C (i.e. P_1, ..., P_k are class identifying properties), then this information can be represented in OWL in the following way:*

- *A class axiom for the OWL class representing the class C should be defined to represent the fact that possessing properties P_1, ..., P_k is a necessary and sufficient condition for individuals to be members of the class C*
- *This class axiom should state that the class C is equivalent to the intersection of the anonymous classes defined by suitable property restrictions for each of the properties P_1, ..., P_k (where each property restriction should imply the possession of the respective property Pi by all the instances of the class C).*

Note that anontological interpretation can be assigned to the anonymous classes that are defined by certain property restriction based class descriptions and that are also subclasses of the upper-level class Substantial_Thing[55]. Bunge's ontology has a notion of *the scope of a property* which is defined as the set of things that possess the property: **Scope(P)={x | x possesses P}.** If P is an OWL property representing some ontological property, then the extension of the anonymous OWL class defined by a suitable property restriction would actually represent the scope of the property P, i.e. a set of OWL individuals (substantial things) which possess the property P[56]. The property restriction should be such that it implies the possession of at least one value for the property, for example such restricction as owl:SomeValuesFrom, owl:hasValue, owl:Cardinality=N (N>=1), or owl:minCardinality =N (N>=1).

A class in Bunge's ontology (a set of things possessing a finite number of common properties) can then be interpreted in terms of scopes of properties that define a class - it is an intersection of the scopes of the properties shared by all class instances when the set of properties is class-defining or is a subset of such intersection if the set of properties is not

[55] That is, this ontological interpretation is only applicable to the classes that represent ontological substantial things.

[56] Since the property P represents an ontological property, we can assume that only instances of the Substantial_Thing class are allowed to possess it (i.e. the property domain is a subclass of the Substantial_Thing class). More precisely, the scope of P would be an intersection of the Substantial_Thing class and an anonymous class defined using one of the above mentioned property restrictions.

class-defining but is a partial set of properties shared by instances of the class.

According to Bunge's ontology, classes/ kinds are defined as sets of things possessing a common property (or properties), and therefore there can not exist empty (i.e."instanceless") classes or kinds. We propose a rule reflecting this ontological assumption, i.e. that all OWL classes representing ontological classes or kinds should have some instances, i.e. they should have the non-empty class extension:

Modeling Rule 16: *Every OWL class representing an ontological class or kind (and thus, modeled as a subclass of the class* Substantial_Thing*) should have or imply non-empty class extension.*

Note that Rule 16 means that even if individuals are not explicitly modeled in an OWL ontology, it should still be possible in principle for the class to have instances, i.e. the definition of the class should be logically consistent and imply non-empty class extension. Reasoning tools available for OWL ontologies (such as Racer) are usually able to detect inconsistent classes based on their definitions, i.e. classes that can not have any instances.

The last rule in this section addresses the issue of so called 'optional properties' or 'zero cardinality' constraints on properties in class definitions. Specifically, in OWL a cardinality constraint or a minimum cardinality constraint in a property restriction class description is allowed to be zero. Such a zero cardinality constraint in a property restriction would mean that instances of the anonymous class defined by this property restriction *may* but *do not have* to possess a value for the property in restriction (thus they do not have to possess the property itself). For example, we could include a statement in an ontology using a zero minimum cardinality constraint for the property hasChild[57] to state that persons may have zero or more children. Such a property restriction, if used in the description of the class Person, would mean that some persons may have children while others may not have any.

If such a zero cardinality restriction is used as part of a class definition in OWL, this would mean that the class definition (in terms of properties) contains properties that are not necessarily possessed by all its instances of the class. In Bunge's ontology, not possessing a property is not a property, and every thing that possesses a property in general possesses a

[57] This object property used earlier links a person individual X to onther person individuals Y that represent its children. Cardinality of one or more means that there is at least one individual Y that is a child of X.

particular individual property[58]. Also, in Bunge's ontology classes are defined in terms of properties possessed by *all* the things that are instances of that class. A better way of modeling in the above example from the ontological standpoint would be to create a subclass PersonWithChildren and include in its definition a suitable property restriction that states that all instance of the class have at least one child (i.e. hasChild would be a 'mandatory' common property for all instances of this subclass).

A similar issue of optional properties or zero cardinality/multiplicity has been raised in prior research on the ontological analysis of conceptual modeling languages. For example, in the ontological analysis of the relationship construct in the ER modeling Wand *et al.* (1999) propose that class definitions in conceptual models should not include optional properties and that 'null' attributes have no ontological meaning. This implies that all attributes included in a class model (irrespective of whether they represent intrinsic or mutual properties) should have values for all possible instances. As an alternative to using optional properties or null values in class descriptions, Wand et al. (1999) propose creating subclasses with mandatory properties arguing that this would reduce semantic ambiguity and improve consistency of the resulting models. Similar rules have been proposed for UML–based conceptual models by Evermann (2003) who also argues against using optional attributes and zero cardinality constraints of attributes in class descriptions. In addition, Bodart *et al.* (2001) in their empirical study on the use of optional properties argue that optional properties should be used with discretion. The experiments in this study provide support to the proposition that using subclasses with mandatory properties is preferrable to the use of optional properties in conceptual models intended for accurate and complete representation and deep (rather than surface-level) understanding of the domain.

Therefore, based on the ontological considerations and guided by prior research, we propose a similar rule for OWL suggesting that zero cardinality or zero minimum cardinality constraints should be avoided in definitions of classes that model ontological classes or kinds defined in terms of shared properties:

> ***Modeling Rule 17:*** *If an OWL class represents an ontological class or kind (defined by a set of common properties) then the property restrictions used in the class description for the respective OWL properties (which model common class properties) should not imply imply optional possession of a property (e.g. a zero cardinality constraint or zero minimum cardinality constraint). Instead, subclassification with property restrictions implying 'mandatory' possession of properties by all instances of the subclass is preferable.*

[58] Often modeled as value of attribute function representing a property in general.

Note however, that this rule does not imply that instances are not allowed to possess properties other than the properties of their classes. On the contrary, one of the advantages of OWL compared to other modeling approaches (such as ERM or UML) is that individuals and properties are independent of classes, and that individuals are allowed to possess properties other that properties of the classes of which they are declared to be instances. Rule 17 simply suggest that for the classes that represent ontological classes and kinds (or functional schemas that model them), class definitions based on property restrictions should include only those properties that are common to all instances of the class (i.e. properties shared by all instances of that class), and should not include 'optional' properties which are only possessed by some instances of the class (rather, in many cases subclassification with 'mandatory' properties is preferable from the ontological standpoint).

4.3.5 Additional classification-related issues

4.3.5.1 Subclassification and class hierarchy – ontological considerations

Relevant to the discussion on the representation of classes is the issue of subclassification. As in many conceptual modeling approaches, classes in OWL ontologies can form class hierarchies based on subclass-superclass (or so called 'IsA') relationships. In OWL, if a class description $\mathbf{C_1}$ is defined as a *subclass* of a class description $\mathbf{C_2}$, then the set of individuals in the class extension of $\mathbf{C_1}$ should be a subset of the set of individuals in the class extension of $\mathbf{C_2}$. OWL provides the construct rdfs:subClassOf for declaring that one OWL class is a subclass of another OWL class.

This section briefly discusses some issues related to subclass-superclass relationships in Bunge's ontology and proposes additional rules regarding the representation of subclasses in OWL ontologies that model real world domains.

As mentioned earlier, classes and kinds are defined in Bunge's ontology through sets of properties shared by *all* their instances. Based on Bunge's ontology, Evermann & Wand (2001b) mention two main ways by which subclasses can be 'created' from their superclasses: 1) by adding new properties to the set of properties of the superclass, or 2) by specialization of some properties of the superclass.

In the first case, if a class or a kind C_1 is defined by a set of common properties $\{P_1, \ldots, P_n\}$, then a subclass C_2 of the class C_1 can be defined by adding one or more additional properties $\{P_{n+1}, \ldots, P_k\}$ to the initial set of properties, so that the subclass C_2 will be defined by the combined set of properties $\{P_1, \ldots, P_n, P_{n+1}, \ldots, P_k\}$. Every instance of the class C_2

would possess all the properties of the class C_1 plus those additional properties. For example, a class of blue cars (e.g. BlueCar) is a subclass of all cars (Car), and an instance of the class Car becomes an instance of the class BlueCar via the acquisition of the property 'blue in color'.

The second way subclasses can be defined is through the *specialization* of properties. Specialization is related to the ontological concept of *property precedence,* which is a type of ontological law. In Bunge's ontology, laws are relations between or restrictions on properties. Property P_1 is said to *precede* P_2 iff for every thing possessing P_2, x also possesses P_1 (P_2 is said *to be preceded* by P_1).

By definition, every instance of a subclass of some class has to possess all the properties of that superclass. If, for example, some class Cp is defined by a property {P}, and there exists another property Q that is *preceded* by the property P (for example, Q can be a specialization of P), then Cq - a set of things possessing Q (also termed *the scope of property* Q) would be a subclass of the class Cp (since possessing the property Q implies possessing the property P). For example, we may consider a class of things that can move (property P= 'can move') and a class of things that can fly (property Q='can fly'). The property 'can move' precedes the property 'can fly', i.e. all things that can fly also can move. Therefore, the class 'things that can fly' is a subclass of the class 'things that can move'. This idea can be generalized to the case of classes defined by more than one property.

In general, for any two classes OWL allows declaring that one class is a subclass of another. It does not require explicit modeling of relationships between properties of those two classes, and thus, it may not be clear whether the difference of a subclass from its superclass is due to the acquisition of new properties or due to the specialization or constraining of some properties of the superclass. This problem of a lack of semantics and potential ambiguities in superclass-subclass relationships has been raised in prior research on the ontological analysis of modeling languages (Wand *et al.,* 1999, Evermann & Wand, 2001b) and on classification (Parsons & Wand, 1997). Specifically, based on ontological and cognitive foundations, Parsons & Wand (1997) argue that in order to achieve cognitive economy (which is one of the reasons for people to use classification) in selection of classes, a principle of *nonredundancy* should be followed when defining classes. That is, they recommend that "a class that is a subclass of several other classes should be defined by at least one property not in any of its superclasses" (Parsons & Wand, 1997). In the research on ER modeling, Wand *et al.* (1999) argue in the case of mutual properties (or interactions) that

when a thing acquires/loses a mutual property that is important for modeling purposes, this thing should be represented by a new functional schema (i.e. as a subclass or a superclass with the respective acquired relationship in case of ER modeling). They denote several advantages of this approach such as the reduction of semantic ambiguities and inconsistencies in a model, clarification of semantics of particular subclasses and of relationships between subclasses and superclasses, and clearer integrity constraints (rules/laws) governing things in the domain. In the case of UML conceptual modeling, Evermann & Wand (2001b) also recommend that if a class B is a subclass of a class A, the attributes shown for the class B should be either specialized attributes of the class A or additional attributes of the class B.

Similarly, to convey the domain information better and to clarify the semantic difference between subclasses and their superclasses in OWL ontologies of real world domains, we propose that OWL ontologies should not just declare that a certain class is a subclass of another class but in addition should explicitly model the difference between the subclass and its superclass in terms of properties (in particular, to show whether individuals of the subclass acquire new properties or specialize some properties of the superclass). We will first illustrate the idea with simple examples and then present our recommendation as an additional rule on modeling subclasses.

For example, in the case of a new property acquisition, we may have a class Person representing the class of all people, and people may have zero or more children (note that 'having a child' can be viewed as 'optional' property of the class Person and should not be a part of class definition based on shared properties according to Rule 17). However, we can define a subclass PersonWhoHasChildren by adding an additional property of having at least one child to the set of shared properties defining class Person. In OWL we can represent this explicitly by stating (in a class axiom) that the class PersonWhoHasChildren is 1) a subclass of the class Person (which will imply that all property restrictions stated in the definition of the class Person would hold for the instances of PersonWhoHasChildren), and also 2) a subclass of the anonymous class defined by an appropriate additional property restriction on the property haveChild (e.g. owl:minCardinality =1 restriction, which would imply mandatory possession of the property by all instances of this subclass, i.e. possession of at least one child).

As for the case of subclasses created by property specialization, even though OWL does not have a universal way to represent property precedence or specialization of properties, certain existing OWL mechanisms can be used to model this type of relationship

between properties. First, for some cases, the owl:subpropertyOf construct can be used to indicate that the possession of one property always implies the possession of another (i.e. preceding) property. In the earlier discussed example of flying and moving things, we could define two datatype properties (with Boolean values true or false) – CanMove and CanFly, and declare the property CanFly to be a subproperty of the property CanMove. In this case, if we have a class MovingThing and its subclass FlyingThing (which includes only those instances of the MovingThing class that can fly), then we should a) include the property CanMove in the class description for the class MovingThings, b) include the property CanFly in the class description of the subclass FlyingThing, and c) declare a subproperty relationship between these two properties. In this case, the difference between the subclass and the superclass due to property specialization will be clearly identifiable from the ontology.

Another way of representing property specialization in OWL for the purpose of defining subclasses is to constrain further for instances of the subclass certain properties possessed by instances of the superclass (using property restrictions with value or cardinality constraints). For example, let us assume that all instances of the class Car possess a property Country[59] (a manufacturer's country). We can define a class EuropeanCar of cars made in European countries ('being made in a certain European country" is preceded by the general property "being made in some country"). We can model this by declaring EuropeanCar to be a subclass of the class Car and also have a property restriction which states that all values of the property Country for the class EuropeanCar should only be European countries. Again, this way of modeling allows the explicit representation of the difference between a subclass EuropeanCar and its superclass Car in terms of properties.

Summarizing the above discussion, we propose the following rule on modeling subclasses:

Modeling Rule 18: *In OWL ontologies modeling real world domains, if classes A and B represent some ontological classes (i.e. modeled as subclasses of the Substantial_Thing class), and B is a subclass of A, then the class definition of the subclass B should reflect the semantic difference (in terms of properties) between the superclass A and its subclass B.*
This distinction can be represented in one of the following ways:

- *by including in the definition of the subclass B one or more additional property restrictions for properties that are acquired by the instances of the subclass B compared*

[59] This property can be modeled as instance valued property by defining a value class Country with one of the subclasses being EuropeanCountry and instances being specific countries. Then property restriction for the property Country could be defined in terms of owl:llValuesFrom restriction (i.e. all values from EuropeanCountry class)

to the instances of the superclass A, or

- *by including in the definition of the subclass B one or more property restrictions constraining some properties of the superclass A for the instances of the subclass B, or*
- *by including in the definition of the subclass B one or more property restrictions for subproperties of some properties of class A*

To illustrate this rule, the first way of representing the distinction could be used to differentiate the class PersonWithChildren from its superclass Person by adding a restriction on the additional property hasChild for the subclass (compared to the superclass). An example where the second type of representing the subclass-superclass distinction could be used is the earlier discussed class FemalePerson (section 4.3.4 and Appendix B), which is a subclass of the class Person and has an additional restriction (owl:hasValue="F") on the property ip_hasGender (which is possessed by both the Person and the FemalePerson classes). Finally, an example of using the third approach is when some ontology has a property hasChildren and its subpropertry hasDaughters. In this case we can define a class PersonsWhoHaveDaughters as a subclass of Persons including only those instances that possess the property hasDaughters (using a property restriction on the property hasDaughters as discussed earlier). Since hasDaughters is a subproperty of hasChildren (i.e. possessing the property hasDaughters implies possessing the property hasChildren), the class PersonsWhoHaveDaughters , by inference, would be a subclass of PersonsWhoHaveChildren (i.e. the difference between the superclass and subclass is due to the possession of a more specific property by the subclass instances, which in this case is expressed using OWL subproperties)[60].

As illustrated by the examples discussed earlier, Rule 18 requires, in particular, that an ontology should not simply show an 'IsA' (subclass-superclass) relationship between two named classes but should also include additional class axioms for the subclass which show how instances of the subclass differ in terms of their properties from the instances of its superclass (which provides added information for instances of subclasses). Following such rule would make the nature of the subclass-superclass relationships in ontology more explicit and would provide an additional ontological semantics to 'isA' relationships in a model.

Furthermore, the explicit modeling of acquisitions or specializations of properties using subclasses allows tracking states of things since the difference between a subclass and its superclass may be in terms of not only intrinsic, but also of mutual properties (resulting

[60] Note that this is an example of an ontological law – property precedence (i.e. the property 'to have children' precedes the property 'to have daughters'). OWL subproperty is one of the mechanisms that can be used to express ontological precedence (even though it does not allow representing all possible precedence cases)

from participation in interactions). For example, a person becomes a library member after participating in a library membership interaction with the library and acquires a number of mutual properties as a result (e.g. library card number or membership start date). To model this in an ontology, we can create classes Person and LibraryMember and declare that the LibraryMember class is a subclass of the Person class. In addition, in accordance with Rule 17 we would include an additional property restriction in the class description for the class LibraryMember which states that each library member must be involved in at least one library membership interaction (and thus possess additional mutual properties associated with this interaction)[61]. More detailed discussion of this example will be presented in section 6.

4.3.5.2 Note on choosing relevant classes and properties

The proposed guidelines on the representation in OWL of ontological classes and kinds (defined by a set of properties and modeled by functional schemas) are mainly applicable in cases when classes or kinds of interest and sets of the relevant ontological properties associated with them have already been identified by a modeler. Our rules and guidelines do not help in determining such properties and do not address the issue of whether a particular set of properties defines a 'good' class. They also do not provide any guidance regarding what set of classes is more appropriate or preferable for describing a domain.

The issues of which sets of properties constitute 'good' or 'useful' classes and what selection of classes is appropriate to describe a domain are beyond the scope of this thesis but have been addressed in other research. For example, some guidelines on how to select classes are proposed in the literature related to IS ontology development (McGuinness et al., 2004, Noy & McGuinness, 2001). These guidelines are driven mainly by pragmatic and implementation considerations and are not always grounded in theory.

The research by Parsons & Wand (1997) on classification builds on the ontological foundations and cognitive principles. Adopting a view of classes as sets of properties, they suggested some guidelines on choosing classes in conceptual modeling and discussed their practical implications for knowledge representation, object-oriented design, and semantic modeling. In particular, Parsons and Wand propose two principles for limiting the sets of properties that should be considered as classes – *abstraction from instances* and *maximum abstraction*. They also suggest that a set of properties should be considered a *potential class* in a relevant universe if and only if: 1) it has a non-empty extension (at some point of time), and 2) it contains all properties common to all instances in the extension.

[61] The interaction and its related mutual properties associated with it would be modeled using interaction classes as proposed in section 4.2.4.4.

In addition, they proposed two principles that apply to collections of classes describing a domain (the relevance universe of things and their properties): 1) *completeness*, which requires that all relevant properties from the relevant universe be used in classification (i.e. in a definition of at least one class in the set of classes), and 2) *nonredundancy*, which requires that no class is defined only in terms of the properties of a set of other classes; specifically, each class should have at least one property not in any of its superclasses.

To summarize, we recommend that when choosing classes and their relevant properties for modeling a domain, modelers should be guided not only by the practical considerations and the intended model usage but also by theory such as ontological and cognitive principles (for example, those proposed by Parsons & Wand, 1997). Once the appropriate sets of ontological properties that represent the classes and the overall class structure are identified, modelers can follow our guidelines to represent these classes and their properties in OWL in an ontologically consistent and clear way.

4.4 RELATIONSHIPS AMONG THINGS, PROPERTIES AND CLASSES

4.4.1 Transfer of ontological assumptions to modeling rules and guidelines

Sections 4.1-4.3 mainly focus on the analysis of specific ontological concepts (such as things, properties and classes) individually, on their mapping to OWL constructs, and on the development of modeling guidelines and rules specific to those concepts. In this section, we look at relationships between these concepts in Bunge's ontology and discuss a number of Bunge's ontological premises that specify and constraint these relationships. Through the transfer of these ontological postulates, we develop additional rules that place some restrictions on the use of OWL constructs and their combinations when modeling elements of the real world (in accordance with Bunge's ontological model) to ensure that only possible and meaningful configurations of the domain are allowed (this approach follows the methodology proposed and formalized by Evermann & Wand, 2005).

Bunge's ontology specifies certain relationships and constraints related to things, properties and classes, in particular:

- *All things have properties* (which exist whether humans are aware of them or not)
- *Properties are always attached to things* ("every property is possessed by some individual or other; there are no properties that fail to be paired to any individuals", [Bunge 1977, p.62]).

- *Classes and kinds are secondary to things and properties – they are defined in terms of properties.* A class in Bunge's ontology is defined by a set of things possessing a common property and a kind is defined by a set of properties.

OWL syntax allows a considerable degree of independence when defining individuals, classes and properties (this distinguishes OWL from many other modeling languages such as UML or ERM in which instances are tied to classes and only possess properties defined for their classes). Even though OWL provides mechanisms for 'linking' these elements in ontologies, in general, OWL permits defining classes, individuals and properties independent of each other. As an extreme example, we can create a primitive ontology with one class Class1, one property Property1, and one individual Individual1 (minimally introduced as an instance of the built-in top class owl:Thing), without specifying any other constraints or relationships between these elements. Such an ontology would be a syntactically valid ontology in OWL (even though it does not give any information about how its class, property and individual are linked to each other).

In general, no syntactic restrictions are placed in OWL regarding certain relationships between individuals, classes, and properties, specifically as to:

- *whether an OWL individual must possess at least one property*

  In OWL, an individual can simply be introduced as an instance of some class or simply as an instance of the default top class owl:Thing. The individual is not required to have any properties associated with it (either directly or by virtue of classs membership). In other words, OWL syntax allows individuals without properties.
- *whether a class must be associated with at least one property*

  As discussed earlier, properties in OWL can be associated with classes by using property restrictions (with value or cardinality constraints) in class axioms. However, OWL also allows defining classes simply as named classes without using property restrictions. Thus, OWL allows defining classes that are not associated with any properties.
- *whether a property must be associated with at least one individual*

  A property in OWL can be associated with individuals either directly (through facts – assertions about individuals possessing specific values for certain properties[62]) or through defining a non-empty class in terms of a property restriction on this property[63]

[62] For example, an ontology can state that an individual John has age 25 (value 25 for the property hasAge) though a statement of a form hasAge (John, 25).

[63] For example, we can define a class Person, for which one of the necessary conditions would be that each person has age (or, in terms of cardinality restriction, each instance of the class Person possesses exactly one value for the property hasAge). Thus, a property hasAge would be associated with the class Person (and all with

(which would imply the possession of the property by instances of this class). However, properties in OWL can be declared as separate constructs. OWL syntax does not require them to be associated with any classes or individuals, even though it may seem useless to declare a property which is not used at all. Nevertheless, properties can exist in OWL ontologies without any association to classes or individuals.

The above mentioned lack of restrictions may be considered problematic from the ontological standpoint with respect to the real-world modeling because of the potential violation of ontological assumptions. Earlier in the thesis we have already in part addressed these issues for classes and properties and proposed some rules guided by the above assumptions, such as the recommendation to define classes in terms of properties and the prohibition of "instanceless" (inconsistent) classes (Rules 13-16, section 4.3.4). In this section we focus more on the assumptions governing relationships between properties and individuals (though classes will still play a significant role). By transferring ontological assumptions about things and properties to OWL modeling, we propose additional guidelines that place restrictions on the relationship between OWL individuals and OWL properties when modeling real-world domains in OWL.

Before presenting additional rules and guidelines, we briefly review OWL mechanisms by which one can associate individuals and properties (i.e. make an individual to possess a certain property). In section 4.3.4 we already discussed how classes and properties can be associated through the use of property restrictions in class definitions. As for the linking of properties and individuals, there are two ways in OWL to express that an individual possesses a certain property:

1) *At the individual (instance) level* – by including in an ontology at least one assertion (fact) stating that the individual possesses some specific value for some property.

For example, in some ontology an assertion of a form hasAge (John, 25) associates an OWL individual John with an OWL property hasAge (i.e. the individual John is declared to possess the specific value (25) for the property hasAge).

2) *At the class level* – an individual can be associated with a property by virtue of class membership.

That is, one can declare an individual to be a member of some class that, in turn, is defined using at least one property restriction in its class axiom, where the property restriction implies the possession of at least one value of this property for each instance of

---

its instances - OWL individuals)

the class (i.e. the property is mandatory for all instances of the class).

Note that not all types of property restrictions in OWL imply the possession of at least one value for a property by an instance satisfying the restriction. For example, the restrictions such as owl:AllValuesFrom or owl:MinCardinality=0 do not necessary imply the possession of at least one value for *each* instance (rather they imply "may possess zero or more values"). However, if one needs to ensure that an individual possesses some property by virtue of class membership, the respective property restriction used in the class definition should be such that it implies the possession of at least one value for the respective property for any instance satisfying the restriction. Acceptable restriction types include constraints owl:hasValue, owl:someValuesFrom, owl:MinCardinality =N or owl:Cardinality=N (where N>=1).

For example, we can define a class BlueThing as a set of things satisfying a property restriction hasValue='Blue' on the property hasColor. Such definition would imply that any individual X that is declared to be a member of the class BlueThing[64] would, by virtue of class membership, satisfy the above restriction, i.e. would possess the value 'Blue' for the property hasColor (and thus possess the property hasColor itself). In other words, we can infer some properties of an individual from its class memberships.

Returning to the ontological assumptions, the first guideline proposed in this section reflects the ontological postulate that all things possess properties:

> ***Guideline 8:*** *Every OWL individual representing a substantial individual (real world instance),[65] should possess at least one substantial property. Possession of a property can be represented by associating this individual with a property either at the instance level or at the class level (via class membership).*

The next guideline follows from another ontological postulate - that properties are always attached to things:

> ***Guideline 9:*** *Every OWL property modeling an ontological substantial property should be possessed by at least one OWL individual representing a substantial thing. This can be represented by associating this property with at least one individual either at the individual level or at the class level (through using a suitable property restriction in class definition).*

In other words, Guideline 9 requires that if some property **P** is declared in an OWL real

[64] E.g.. via a statement of the form <BlueThing rdf:ID="X">

[65] An individual which is an instance of the Substantial_Thing class (in accordance with Rule 1).

world domain ontology and P represents some substantial ontological property, then there should be at least one of the two types of statements in the ontology:

- at least one statement of the form **P(X, v),** which declares that **X** has the value **v** for the property **P** - where **X** is an OWL individual representing some substantial thing possessing that property, or
- a statement of the form **C(X),** which declares that X is an instance of some class C, where C is a class defined using a suitable property restriction on the property P, which would, in turn, imply that any instance of C necessary possesses some value v for the property P (i.e. in this case C(X) would imply that a value **v** exists such that **P(X,v)**).

Similarly, according to the Guideline 8, for each OWL individual X that represents an ontological substantial thing, there should be at least one of the two types of the above statements in the ontology with respect to at least one property P representing a substantial ontological property.

As for the relationship between classes and properties, whereas OWL allows declaring classes not associated with any properties, Bunge's ontology defines classes (of substantial things) in terms of properties, i.e. instances of a class/kind have at least one property in common. Thus, in addition to the earlier suggested rules on modeling classes (section 4.3.4) we propose one more rule that all OWL classes that represent ontological classes/kinds should be associated (through property restrictions) with at least one property. Specifically:

> ***Guideline 10:*** *Every OWL class representing an ontological class or kind (i.e. a subclass of the class* Substantial_Thing*) should have at least one property. That is, the class definition should include a class axiom that states a necessary condition for this class in terms of a suitable property restriction for at least one OWL property representing an ontological property shared by all instances of the class*

4.4.2 Note on the independence of things from classes: OWL vs. Bunge's ontology

An important assumption of Bunge's ontology is that things and properties are primary constructs and exist independent of classes. Classes/kinds are secondary to things; they represent sets of things sharing a common property (or a number of properties), i.e. they are defined in terms of properties. An important advantage of OWL compared to other conceptual languages and approaches is that it allows modeling individuals independently of

classes and that individuals in OWL are allowed to possess properties other than properties of classes they belong to. This functionality in combination with our proposed rules on modeling things, properties and classes allows supporting the above ontological assumption of 'instance independence of classes' and helps avoid a problem of "inherent classification" common to many other modeling approaches (such as OOM, ERM, UML etc). As noted by Parsons & Wand (2000), many conceptual languages, modeling approaches, and database design models have a common assumption (implicit or explicit) – termed "the assumption of inherent classification" - that specific things in the domain of interest can be referred to only as instances of classes, and that instances of classes can only possess properties that are determined by their classes. Parsons and Wand (2000) challenge this rather restrictive assumption on both theoretical and practical grounds arguing that it is inconsistent with the ontological and cognitive principles and in practice may lead to a number of problems.

OWL, on the other hand, does not have this problem of inherent classification, and the introduction of the proposed rules and guidelines also does not violate the ontological assumption about things with properties existing independent of classes. According to the proposed rules, any substantial thing can be represented as OWL individual which can be minimally introduced as an instance of the upper-level class Substantial_Thing[66] (distinguishing it from OWL individuals not representing real world things). In principle, we do not even have to declare this individual to be a member of any other classes, but can simply make direct assertions about the individual's specific properties (without referring to any classes). If an individual is declared to be a member of some classes (to take advantage of classifications' benefits), one can still make assertions about individuals' properties in addition to those implied by its class memberships. On the other hand, an individual can be inferred to be a member of some classes (defined through property restrictions) based on assertions (facts) about its properties. Such representation approach is consistent with the concept of classes in Bunge's ontological model, and is similar to some degree to the two layer model proposed by Parsons & Wand, 2000.

4.5 REPRESENTATION OF COMPOSITION RELATIONSHIP

This section discusses the ontological concept of *composition of things* and proposes some guidelines on modeling composition relationships in OWL ontologies.

Modelers often need to represent facts that things in a domain are combined in some

[66]In general, if the proposed upper level class structure is not used, OWL individuals can still be minimally introduced by declaring them as instances of the predefined top class owl:Thing and assertions about individuals' properties can be made (thus the problem of inherent classification is not present in OWL).

way to create other things. Bunge's ontology has a concept of *composite* and *component things* (which are involved in composition relationship).

According to Bunge's ontology, things can combine to form a *composite thing.* Composite things can be decomposed into *components,* which are also things. Bunge's ontology postulates that there exist simple things that can not be decomposed. Properties of composite things can be either *resultant* (*hereditary*) or *emergent* properties (Bunge, 1977). *Hereditary properties* are properties of at least one component thing of a composite, whereas *emergent properties* are those properties of a composite that are not possessed by any of its components. Emergent properties can be explained in terms of or derived from the properties of the parts but are not reducible to them, and thus, emergent properties cannot be attributed to any of the individual parts of the composite thing, but rather belong to a composite thing as a whole. Bunge's ontology postulates that every composite must possess at least one emergent property not inherited from any of the parts (Bunge, 1977); otherwise this is not a composite but simply a set of things. For example, a computer composed of memory and processor possesses a property "processing power", which is not possessed by any individual component (Evermann, 2003). Both emerging and hereditary properties can be either intrinsic or mutual.

In OWL there is no predefined construct to represent composition relationship between things. Earlier we proposed that ontological things are modeled as OWL individuals. Therefore, to represent composition relationships, we need to be able to link individuals representing composites to individuals representing components, and vice versa. For that we can use OWL object properties.

Specifically, to model composite-component relationships between substantial things, we propose defining two mutually inverse OWL object properties - isComposedOf and isComponentOf. The property isComposedOf would be used to link OWL individuals representing composite things to OWL individuals representing things which are components of the composite, and the inverse property isComponentOf would link components to the composite things of which they are components.

Modeling Rule 19*: To represent composition relationship between substantial things in OWL ontologies of real world domains, two mutually inverse object properties should be defined:* isComposedOf *and* isComponentOf. *These properties would link OWL individuals representing composite things to their component things, and OWL individuals representing component things back to their composites, respectively.*

In some cases, it may be relevant for modelers not only to model composite-component relationships between things in an ontology, but also to distinguish clearly between composite things and things that are components of some things, i.e. represent explicitly which things are composites or components. Technically, if the two properties proposed above are employed to represent composition, then one can infer whether a thing is a component of some other thing or a composite thing by looking at whether it is linked to another thing via the property isComponentOf or isComposedOf, respectively. To facilitate and explicitly model this inferred information, we can declare two upper-level classes (modeled as subclasses of the upper-level class Substantial_Thing):

- **Composite_Thing** – the class extension of this class would consist of all OWL individuals that are composite substantial things
- **Component_Thing** - the class extension of this class would consist of all OWL individuals that are substantial things and are components of at least one composite thing

Note that unlike the classes Substantial_Thing and Non_Substantial_Thing which are disjoint, the classes Composite_Thing and Component_Thing should not be declared disjoint because things may be composed of other things and at the same time be components of some other things (for example, a person may be a member of some team while the team is a member of some league).

We can define class axioms for these upper level classes stating necessary and sufficient condition for class membership in terms of possessing at least one value for the property isComposedOf or isComponentOf, respectively (using property restrictions). Specifically, in OWL terms, a class axiom for the Composite_Thing class would state that the class Composite_Thing is equivalent to the *intersection* of the class Substantial_Thing[67] and the class of all OWL individuals that possess at least one value for the property isComposedOf[68]. Similarly, a class axiom for the Component_Thing class would state that the class Component_Thing is equivalent to the *intersection* of the class Substantial_Thing and the class of all OWL individuals that possess at least one value for the property isComponentOf (using minimum cardinality or owl:someValuesFrom property restriction). Note that we can declare that the class Composite_Thing is the domain of property isComposedOf, and that the class Component_Thing is the domain of the property isComponentOf.

The relationship between the proposed classes is illustrated schematically in Figure

[67] This represents the fact that we focus on composition relationship among substantial (real world) things only.

[68] Minimum cardinality restriction (owl:minCardinality=1) or an owl:someValuesFrom property restriction (e.g. SomeValuesFrom the class Substantial_Thing) can be used; this would also imply that all instances of the Composite_Thing class possess at least one value for the isComposedOf property linking these instances to the substantial things that are their components.

5.[69] An example of representing composite-component relationship in OWL using the proposed approach can be found in section 6 (the library domain example).

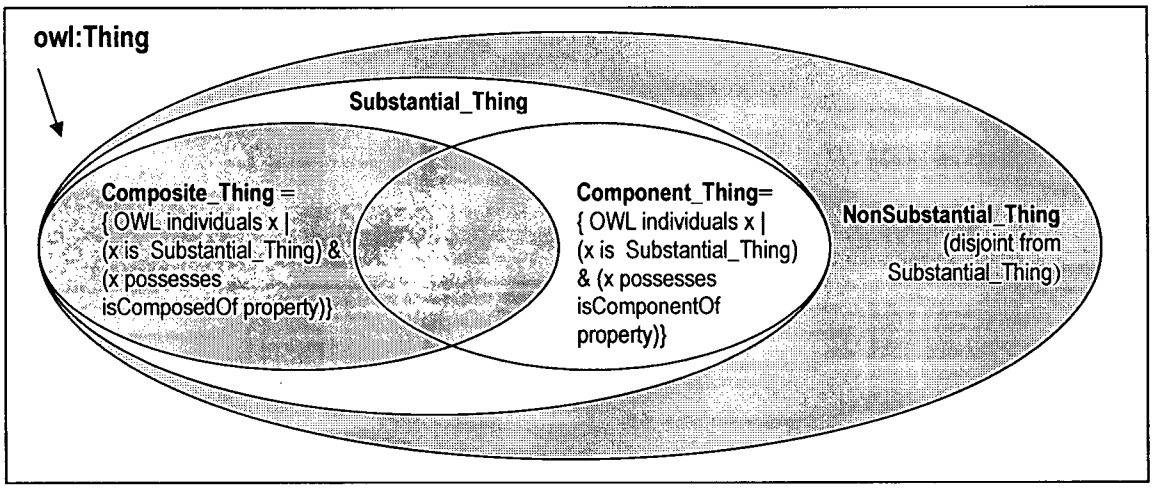

Figure 5: Upper level (meta-model) classes for representing composition

Note that if we define properties isComposedOf and isComponentOf as well as classes Composite_Thing and Component_Thing as proposed above, this would allow both humans and automatic reasoners to make certain composition-related inferences about classes or individuals representing substantial things. For example, if an ontology includes a statement of the form isComponentOf (A, B) for individuals A and B, and also states that A and B are substantial things (i.e. are instances of the Substantial_Thing class), then it can be inferred that a) A is a component thing (i.e. an instance of Component_Thing class), b) isComposedOf (B, A) holds (inverse property), and c) B is a composite thing (i.e. an instance of the Composite_Thing class). Thus, we do not have to include all such statements explicitly for individuals or classes but can only state that an individual is a component (or composite) of another individual, and other information can be inferred. On the other hand, if multiple statements related to composition are included in an ontology then reasoners can use class definitions and axioms to check the consistency of the ontology.

The proposed representation of composition relationships is summarized in the following rules:

Modelng Rule 20: *To model explicitly that substantial things are composites or components of other things, two upper level classes can be created in OWL ontologies:* Composite_Thing *and* Component_Thing

- *Both classes should be modeled as subclasses of* the Substantial_Thing *upper level class*
- *The class* Composite_Thing *can be defined (using class axioms with a cardinality or an*

[69] Please note that relative sizes of ovals representing various classes in Figure 5 are arbitrary and are not indicative of the relative number of instances in these classes (as this is not important for this discussion).

owl:SomeValuesFrom *property restriction) as the class of all OWL individuals that are instances of the class* Substantial_Thing *and possess the property* isComposedOf*;*

- *The class* Component_Thing *can be defined (using class axioms with a cardinality or an* owl:SomeValuesFrom *property restriction) as the class of all OWL individuals that are instances of the class* Substantial_Thing *and possess the property* isComponentOf.

Corollary 8: *OWL individuals representing substantial things that are components of some composite thing should be declared or inferred to be instances of the* Component_Thing *class. OWL individuals representing composite substantial things should be declared or inferred to be instances of the* Composite_Thing *class.*

Corollary 9: *Any OWL class such that all instances of that class represent substantial composite things should be declared or inferred to be a subclass of the* Composite_Thing *class. Any OWL class all instances of which represent substantial component things should be declared or inferred to be a subclass of the* Component_Thing *class.*

According to Bunge's ontology, a composite thing must possess at least one emergent property. Otherwise, it is not a composite but simply a set of things. In other words, a composite thing must be more than simply the sum of its parts (Evermann, 2003). The next rule reflects this ontological assumption:

Modeling Rule 21: *Every OWL individual representing a composite thing should be associated (at the instance or at the class level) with at least one OWL property representing an intrinsic or mutual ontological property that is an emergent property of the composite thing.*

The last rule in this section focuses on modeling emergent and hereditary properties.

Modeling Rule 22:

- *OWL properties that model hereditary properties of a composite thing can be associated with both OWL individuals (or classes, at the class level) representing the respective component thing(s) and with OWL individuals (or classes) modeling the composite;*
- *OWL properties that model emergent properties of composite things should be associated with OWL individuals (or classes, at the class level) representing the composite, but not with any individuals (classes) representing components of this composite*

In simpler words, this rule means that since emergent properties are properties of the composite as a whole but are not properties of individual parts, they should only be modeled as properties of the individual (or class) representing this composite. For example, the number of team members is a property of the team, but not of the individual players. Thus, it should not be associated with instances representing individual team members, but only with the OWL individual representing the team (or represented as a property restriction for the class Team). On the other hand, if a property is hereditary, it is possessed by a component of a composite and is also 'inherited' and exhibited by the composite. Thus, such property can be modeled as a property of both the OWL individual (class) representing a component and the individual representing the composite. For example, a property of a component - memory (RAM) size - becomes a property of a computer as a composite as well and can be modeled accordingly.

For completeness, we can also include in the meta-model another upper-level class, Simple_Thing, to represent the class of all simple ontological things, which are things that can not be further decomposed (such things exist according to Bunge's ontology). Classes Composite_Thing and Simple_Thing should be declared disjoint. The union of these classes would be equivalent to the class Substantial_Thing. Instances of the class Component_Thing, on the other hand, would be either simple things (i.e. instances of the Simple_Thing class) or composite things themselves (i.e. instances of the class Composite_Thing).

5 SUMMARY – PROPOSED METAMODEL AND MODELING PROCESS GUIDELINES

Chapter 4 presented a detailed analysis of the fundamental ontological constructs and their comparison to OWL constructs. Based on this comparative analysis and ontological assumptions, we have developed a number of modeling rules and guidelines on the representation of specific ontological constructs in OWL to facilitate the development of more expressive OWL ontologies. In this section, we summarize some of the outcomes of this analysis. Specifically, we provide a summary of the proposed meta-model, which is a set of upper-level domain independent classes and properties recommended for inclusion in OWL ontologies modeling real world domains to facilitate the application of the proposed rules and guidelines and to help in development of ontologically meaningful models. In addition, in this section we propose the process steps recommended for modelers applying the approach proposed in this thesis.

The proposed meta-model (upper-level OWL classes and properties) is presented in Figure 6:

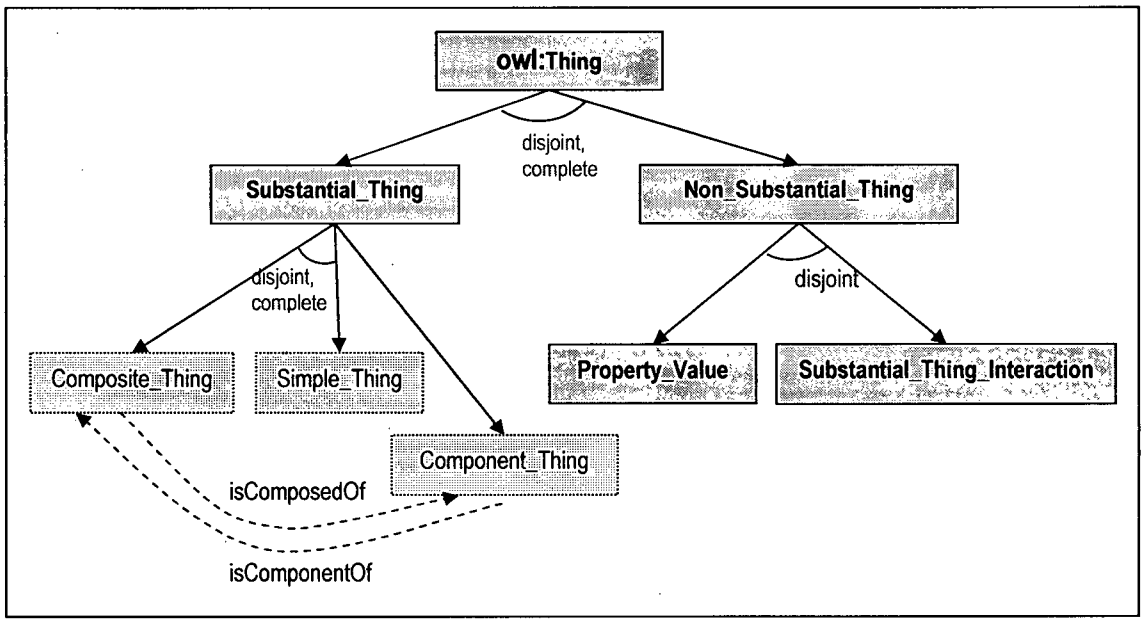

Figure 6: Proposed meta-model

The meta-model includes two upper level disjoint classes – Substantial_Thing and Non_Substantial_Thing - to distinguish between OWL classes and individuals representing ontological substantial things and OWL classes and individuals used for representing other constructs (section 4.1).

Also, the meta-model includes a special subclass of the class Non_Substantial_Thing

named Substantial_Thing_Interaction intended for distinguishing a special group of OWL classes, interaction classes, which are intended for representing sets of concurrent mutual properties arising out of interactions and shared by interaction participants (section 4.2.4.4).

Another special purpose meta-model class is the Property_Value class (also a subclass of the Non_Substantial_Thing class). It is recommended for the inclusion in OWL ontologies if modelers intend to use property value representation method using special value classes and individuals (as discussed in detail in section 4.2.3.2). If modelers do not intend to use this property value representation method and only use OWL datatype properties to represent individual properties, then this meta-model class does not have to be included. Note that the two proposed subclasses of the Non_Substantial_Thing class are declared disjoint since they are used for representing completely different concepts.

Finally, we have proposed two mutually inverse properties isComposedOf and isComponentOf for representing individual composition relationships between composite and component substantial things. We have also proposed the upper-level classes Composite_Thing, Simple_Thing, and Component_Thing (which are subclasses of the class Substantial_Thing) that may be useful if an ontology requires to show explicitly which things are composites and which things are components of some things (section 4.5). These three classes may be considered optional (since a pair of properties isComposedOf/ isComponentOf in general is sufficient for representing composite/component relationships).

The proposed upper-level meta-model elements are domain independent and help clarify the ontological semantics of domain specific elements. Domain specific classes are supposed to be modeled as subclasses of the proposed upper level classes, and thus, will have clear ontological semantics. The meta-model structure is intended to complement the proposed ontologically grounded modeling guidelines and rules and to help modelers in the application of these guidelines. The structure is intentionally kept relatively simple so that the model is easier to understand and employ while still being helpful for the development of more ontologically expressive OWL models.

Note that the meta-model can be further expanded. For example, we could not only create upper level classes but also some upper-level properties (domain specific properties could in this case be declared as subproperties of these upper-level properties using owl:subpropertyOf construct). Some candidate meta-model properties we could add are, for example, object properties isInvolvedIn and Involves that would link instances of the class Substantial_Thing to instances of the class Substantial_Thing_Interaction, and vice versa, respectively. Domain specific properties related to interactions, for example such as

isInvolvedIn_Employment or involves_Company (section 4.2.4.4), would be modeled as subproperties of these meta-model properties, respectively. Other meta-model properties can be included (for example, to distinguish between mutual and intrinsic properties). Furthermore, the proposed meta-model can be extended to include elements representing other Bunge's constructs (e.g. states and events), which are not considered in this thesis and require further research.

For this thesis, due to time and length considerations we limited ourselves to the restricted meta-model as presented above. The RDF/XML OWL syntax defining meta-model elements is presented in Appendix C. Potential extensions to this meta-model can be a topic for future research.

Next, we would like to outline the key modeling process steps that we recommend modelers to follow when applying the proposed modeling rules and guidelines on the development of OWL ontologies for real world domains:

Step 1: Create the required **meta-model elements** (classes and properties) in your OWL ontology

Step 2: Identify types of ontological **substantial things** in the modeled real world domain.

- Represent them as named OWL classes - subclasses of the Substantial_Thing meta-model class (Corollary 2, Modeling Rule 1; Rules 12-13)

Step 3: For each class of substantial things, identify **intrinsic properties**.

- Represent them as OWL properties in accordance with Guidelines 3-4 and Modeling Rules 2-4
- Associate them with the respective classes using property restrictions in class axioms (Modeling Rules 14-17, Guideline 8, 9)

Step 4: Identify relevant **interactions** between substantial things in the modeled domain and determine the **related sets of concurrent mutual properties**

- Create interaction classes corresponding to the relevant interactions, declare them subclasses of the Substantial_Thing_Interaction class (Modeling Rule 5-6, Guideline 5-6)
- Represent the individual mutual properties in each set as OWL properties (Rule 7); associate them with the respective interaction classes using property restrictions (Rule 14-17, Guideline 8, 9)
- Create OWL properties of the type 'involves_...' and 'involvedIn_...' for linking interaction classes to the participating substantial thing classes, and vice versa

(Modeling Rule 8-9); associate them with the respective interaction classes and substantial thing classes using property restrictions in class axioms (Rule 14-17, Guideline 8, 9)

- If it is necessary to represent mutual properties shared by two things only which are either non-binding or the related interaction is not relevant to the scope of the model, represent these properties using OWL object properties according to Modeling Rules 10-11.

Step 5: Identify **subclass-superclass relationships** of interest and the properties that distinguish subclass instances from superclass instances

- Represent subclass-superclass relationships using owl:subclassOf construct
- Ensure that each subclass in a model is modeled with at least one additional property (intrinsic or mutual, such as participation in some interaction) or has some property constrained or specialized compared to its superclass (Modeling Rule 18)

Step 6: If in the scope of the model, identify **composition relationships** in the domain

- Represent composition relationships using properties isComposedOf and isComponentOf (in accordance with Modeling Rules 19-22)

We would like to stress that the above proposed process steps should not be taken as absolute because OWL syntax is very flexible (for example, there are multiple ways to express the same idea in the XML/RDF syntax, and statements about ontology elements can be arranged in various order). It is acceptable to change the order of some activities involved in these steps or to combine some activities without violating the general approach. Therefore, we recommend that the proposed modeling process sequence is viewed as a general guidance to modelers as well as the checklist to ensure that all the issues involved in modeling process have been addressed. We hope that these general steps (with references to specific rules and guidelines and the related thesis sections) will facilitate the application of the rules an lead to the development of more expressive OWL models of real world domains. In the next section, we illustrate the applicability of the proposed approach using an example.

6 DEMONSTRATION OF APPLICABILITY – AN EXAMPLE

In this section we demonstrate the applicability and the process of applying the proposed rules and guidelines using a small example from a real world domain – the library domain[70]. Let us assume that there exist various libraries which carry different types of items for borrowing, such as books, magazines, or music CDs. These types of items have different borrowing conditions, such as allowed number of loan days. People come to libraries to register and become library members. Upon registration, they obtain library cards and can borrow and return items to the library.

To model this case in OWL, we follow our recommended modeling process steps. Wherever relevant, we indicate rules and guidelines that we apply. For brevity, we use abbreviations with the respective rule or guideline number: MR for Modeling Rule, G for Guideline, and C for Corollary (for example, MR1 for Modeling Rule 1, G2 for Guideline 2, or C5 for Corollary 5).

1) Upper-level (meta-model) classes

First, we create in our OWL ontology the upper level classes Substantial_Thing and Non_Substantial_Thing and declare them disjoint (MR1). We also create the class Substantial_Thing_Interaction (as a subclass of the Non_Substantial_Thing) which is used to create interaction classes with their related mutual properties (G5, MR5, MR6).

2) Substantial things

Analyzing our domain, we can identify the following substantial things: *library items* (which can be of several types), *persons,* and *libraries* (we assume that a person can be a member of more than one library). Respectively, we create the following classes of substantial things in our ontology: Library, Person, Library_Item, Book, Magazine, Music_CD. All these classes should be declared subclasses of the class Substantial_Thing (C2, MR12). In addition to defining these classes as named classes, we will later add (when defining properties) class axioms for these classes in terms of property restrictions to show which common properties the instances of these classes share (in accordance with the rules 14-15 and guidelines 8-9)

Also, we declare the classes, Book, Magazine, Music_CD to be subclasses of the class Library_Item (later we will add properties that differentiate each subclass from its superclass as per rule 18). Note that later we will need to create more classes (subclasses of the above ones) based on the analysis of interactions and of the related shared mutual properties (so as

[70] While our example is simplified, it should not affect generalizability of our approach.

to avoid 'optional properties' problem, as per rule 17).

3) Intrinsic properties

Having identified the major substantial things and classes, we proceed to identify and model properties. First, for each of the classes of substantial things we determine intrinsic properties that need to be modeled. We represent them as OWL properties in accordance with guidelines G3-4 and rules MR2-4. For example, for *libraries* we may identify such properties as *name* and *address*[71]. For *persons*, some intrinsic properties are *name*, *address* and *date of birth*. *Library items* have some common intrinsic properties, for example *title*, *barcode* (a unique inventory code assigned by the library), *subject*, and *loan days allowed*. In addition, each of the different library item types may also have properties specific to this item type. For our example, we assume that *books* have such intrinsic properties as *author*[72] and *publisher*, *magazines* have *volume* and *issue*, and *music CDs* have *artist* and *content* (e.g. list of songs). Note that these intrinsic properties differentiate subclasses from its superclass Library_Item, thus satisfying the rule MR18 on subclassification.

All these intrinsic properties are modeled as OWL properties in accordance with the guidelines G3-4 and rules MR2-4. In addition, to satisfy the guidelines 8 and 9 as well as the rules 14-15, we associate these properties with the respective classes using suitable property restrictions in class axioms (e.g. owl:Cardinality=1 or owl:minCardinality=1), thus stating that all instances of the above classes possess the respective intrinsic properties.

4) Interactions and related mutual properties

The next step is to identify and model mutual properties. To do that, we need to identify interactions and bundles of mutual properties associated with them and shared by participants (G5-6). In our example, several interactions can be identified.

First, a *person* interacts with a *library* to become a *library member* (library membership interaction). As a result of this interaction, several mutual properties are acquired by the interacting *library* and *person* things, including *library card number*, *membership start date*, and *membership status* (e.g. active or suspended). Since not all persons become library members (i.e. not all persons possess the above mutual properties) and since ontological guidelines recommend avoiding 'optional properties' in class definitions (MR17), we create a subclass Library_Member (a class of persons who are

[71] For simplicity, we treat addresses and names here as a single string values. More complex representations can be easily implemented (e.g. separate properties for first and last name, or for city, street, postal code etc.), Also, here we consider a library to be a single location (i.e. we do not consider multiple branches)

[72] By library items here we mean physical copies of books, magazines or CDs. Thus, we will not consider such entities as Author or Publisher as things here (they are out of scope of the model), but rather view them as library item's intrinsic properties (this process is termed 'unarisation of properties' in Bunge's ontology)

members of some library). To represent this interaction and the respective mutual properties we create a special interaction class i_Library_Membership as a subclass of the class Substantial_Thing_Interaction (MR6).

Individual mutual properties are represented as OWL properties with the class i_Library_Membership as their domain (MR7); they are also associated with this class using suitable property restrictions (e.g. e.g. owl:Cardinality=1) in the class definition (G8-9) to state that these properties are possessed by *all* instances of the interaction class i_Library_Membership (which would indirectly represent the possession of this bundle of properties by the respective Library and Library_Member individuals linked by instances of the interaction class). Note that one person can be a library member in different libraries and a library can have many members, therefore for each interaction (or a pair "person-library"), an ontology will have a separate instance of the interaction class representing an individual "instance" of this interaction type (which in turn will possess individual values for the bundled mutual properties).

To conclude the representation of the interaction and its related mutual properties, we need to create object properties for linking the interaction class and the involved substantial thing classes (in accordance with the rules MR8-9). For the library membership interaction, we create an object property corresponding to the interaction class - involvedIn_Library_Membership (MR8) and two object properties involves_Library, involves_Library_Member corresponding to the classes of substantial things participating in the interaction. We also add class axioms to the interaction class i_Library_Membership that state that each interaction class instance possesses one value for each of the properties involves_Library and involves_Library_Member (i.e. is linked to exactly one Library and one Library_Member instance). Similarly for the classes, Library and Library_Member we will add class axioms that state that each Library and each Library_Member instance possesses at least one value for the property involvedIn_Library_Membership (i.e. is involved in at least one library membership interaction and possesses the associated mutual properties)

Another example of an interaction in this domain is the *item loan* interaction[73] (involving a library member borrowing an item, a borrowed library item, and a library). This interaction also gives rise to certain mutual properties, shared by its participants, such as for example, *date out* and *date due.* This interaction and the related mutual properties are represented in the same manner as the above described *library membership* interaction. Note that to represent the item loan interaction, we also need to create additional subclasses of

[73] Additional examples of interactions in this domain would be item reservation, item return and item renewals. We limit ourselves to only to the above discussed interactions only for demonstration purposes.

substantial things (MR17): in addition to Library[74], the *item loan* interaction involves Borrower (a subclass of LibraryMember) and Borrowed_Item (a subclass of Library_Item).

5) Subclass-superclass relationships

We declare the classes Book, Magazine, and Music_CD to be subclasses of the class Library_Item. To ensure that subclasses differ from its superclass in terms of properties (Rule 18), we associate (using property restrictions in class axiom) the properties specific to each of these item types with the respective classes (those type-specific properties are mentioned in step 2). In addition, in step 4, we have created some other subclasses (such as Borrower – a subclass of Library_Member). Instances of these subclasses also satisfy Rule 18 since they differ from the superclass instances due to participating in a certain interaction (e.g. possessing a property involvedIn_ItemLoan and a set of related mutual properties indirectly).

6) Composition relationships

For illustration purpose, we consider one composition relationship in our domain. Various library departments or sections (for example, children book section, adult book section, or reference section) can be considered to be components of the library. To represent such composition relationship in our OWL ontology, we need to define the two mutually inverse meta-model properties: isComposedOf and isComponentOf (in accordance with MR19). We could also define meta-model classes Composite_Thing and Component_Thing (subclasses of the Substantial_Thing upper level class) if we were interested in explicitly modeling or using inference to determine which things are composites or components of some other things. However, for brevity and model readability, in this example we limit ourselves to representing composition relationships between certain substantial things, thus the pair of properties isComposedOf/ isComponentOf is sufficient. We also create a class Library_Section (a subclass of the Substantial_Thing class) to represent various library sections and state for this class (using property restrictions owl:someValuesFrom = Library on the property isComponentOf) that every Library_Section instance is a component of some library. We can also declare that every library is composed of some sections by adding a class axiom for the class Library using the property restriction owl:someValuesFrom =Library_Section on the property isComposedOf.

Finally, Rule 21 requires that every composite thing should be modeled with at least one emergent property. In our case, one emergent property of library (as a composite thing) would be the *number of sections in the library*[75]. We would model it as an OWL property

[74] We assume that every library has at least one member, lends at least one item and gets at least one item back. Therefore, we do not create subclasses for Library based on the above interactions.

[75] While this example is quite simple and may not seem very practical, it serves the purpose of illustrating key

and associate it with the class Library using a property restriction (such as owl:Cardinality=1) to state that every library has a specific number of library sections.

Figure 7 on the next page shows a graphical representation of the resulting OWL ontology developed for the library example discussed in this section. The RDF/XML OWL syntax representation of this ontology can be found in Appendix D.

steps of our modeling approach without overloading the reader with too many irrelevant details.

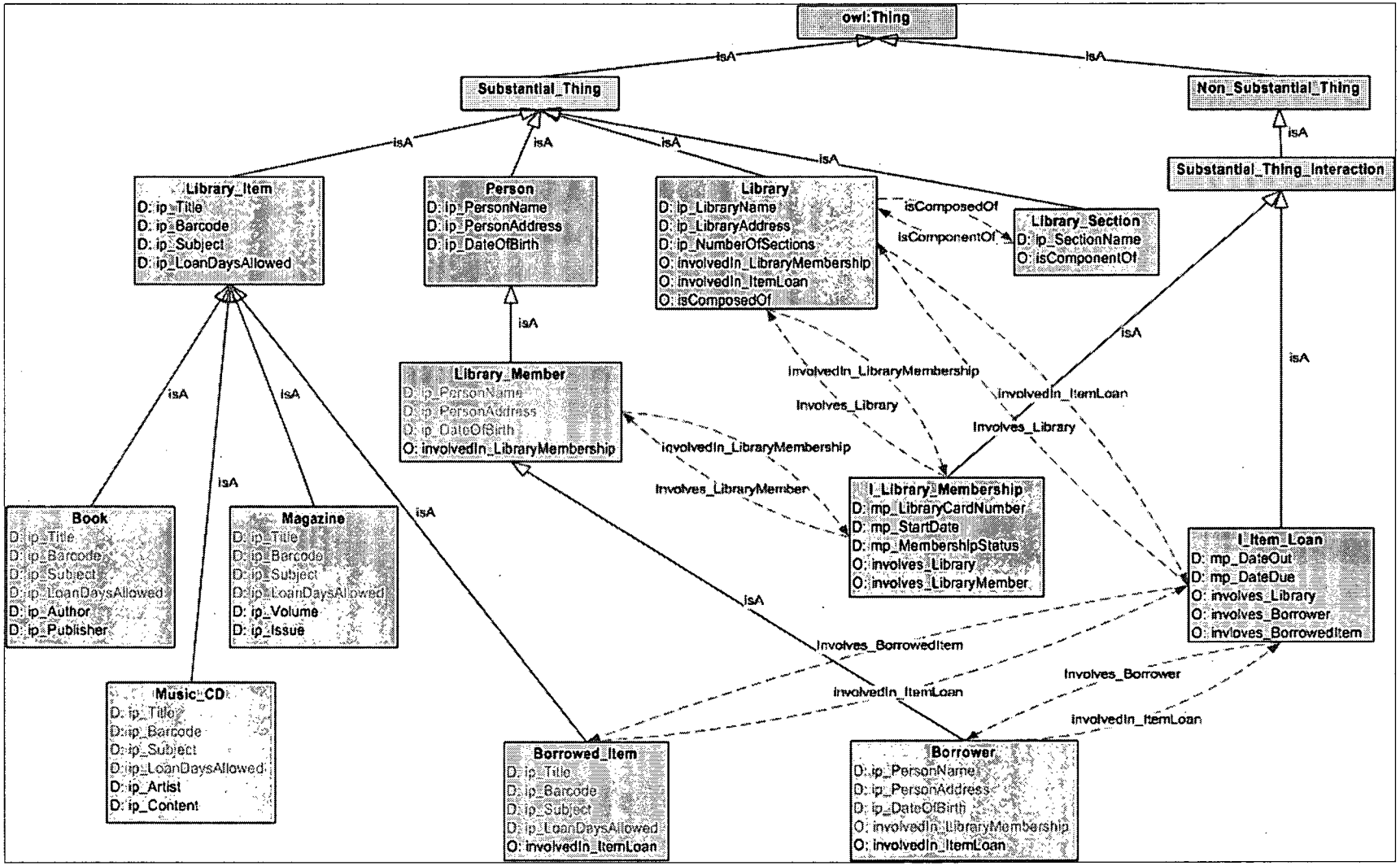

Figure 7: Graphical representation of the OWL ontology for the library example

To conclude, the example discussed in this secction demonstrates that our rules are applicable in practice and lead to the representation of useful information about the domain. The proposed modeling rules and guidelines, meta-model elements, and process guidelines help modelers to analyze and represent domain elements in a systematic way and toexplicate the assumptions about the domain, which facilitates the development of more expressive, ontologically clear and consistent models.

7 CONCLUSIONS AND FUTURE RESEARCH ISSUES

The emergence of the Semantic Web as a future of the World Wide Web has created a wide interest in IS ontologies as a means of representing - in a formal and machine-readable form - the relevant information about various domains. To effectively represent the knowledge about real-world domains, an IS ontology needs to properly convey beliefs about the real world, i.e. beliefs on what exists, might exist, or happen as perceived by a community of domain knowledge users. This thesis focuses on a recently proposed formal ontology language – the OWL Web Ontology Language, which is considered to be one of the fundamental components of the Semantic Web. It contributes to the fruitful stream of the research on the ontological analysis of conceptual modeling grammars by evaluating them against philosophical ontologies, which was initiated by Wand & Weber (1989) and continued by a number of researchers since then.

The thesis continues the work started by Bera & Wand (2004) who conducted an initial ontological analysis of OWL with the aim of evaluating OWL's ontological expressiveness, i.e. how explicitly and accurately real-world domain information can be modeled in OWL ontologies. By benchmarking OWL against a particular philosophical ontology – the Bunge-Wand-Weber ontology - they pointed out a number of issues which may undermine the expressiveness of OWL ontologies when they represent real world domains, and may also lead to ambiguity, inconsistency, and a lack of stability of representations. Bera & Wand (2004) suggested that the expressiveness of OWL can be improved by introducing ontologically grounded general representation guidelines and specific guidelines on modeling certain ontological constructs in OWL. As well, new constructs can be added to OWL to overcome the lack of expressiveness.

This thesis has been inspired by the above suggestions and continued the research in that direction. Based on the ontological foundations, it has conducted an in-depth comparative analysis of fundamental constructs of Bunge's ontology (things, intrinsic and mutual properties, interactions, classes, and composition) against OWL constructs and current OWL modeling practices in order to propose a mapping from Bunge's constructs to OWL constructs or their combinations in an ontologically clear and consistent way. Based on the proposed mapping, through a transfer of ontological assumptions, the thesis has developed a set of general guidelines and more specific modeling rules on how to represent key ontological constructs in OWL ontologies of real-world domains. In addition, this research has proposed a meta-model – a set of high-level, domain independent OWL classes

and properties - that is recommended for inclusion in OWL ontologies that model real world domains and will be developed following the proposed methodology. The proposed meta-model elements are intended to help clarify the ontological semantics of the domain specific constructs (which are supposed to be modeled as subclasses or subproperties of the respective meta-model elements) as well as to facilitate the use of the proposed rules and guidelines by OWL modelers. Furthermore, the thesis has also suggested a number of key process steps which OWL modelers should follow when developing OWL ontologies in accordance with the proposed approach. The applicability and the process of applying the proposed modeling rules and guidelines have been illustrated by an example from the library domain.

Encouraged by our initial experience with the proposed approach, by prior research applying a similar approach for other modeling languages, and by the results of some empirical studies showing support for the propositions that the use of ontologically grounded modeling guidelines leads to better models, we hope that the developed approach, which is grounded in the philosophical ontological foundations, will facilitate the development of more ontologically expressive OWL ontologies and may help alleviate or avoid the problems ambiguity, inconsistency and lack of stability of OWL models. We believe that our approach may guide modelers in analyzing their real-world domains of interest and encourage them to "think ontologically", seeking and representing domain information in more detailed and structured way. It may also help modelers focus more on domain knowledge and faithful representation of its elements, and not just on machine-readability and technical implementation issues. In addition, using the proposed approach, different modelers are more likely to come to similar representations of the same real world domains (even if they use different element names to represent domain elements), making it easier to align and merge such ontologies and making interpretation and usage in applications more consistent, which is important for effective and efficient knowledge representation and sharing.

We acknowledge several limitations of this study, which in turn provide ideas for future extensions of this research. First, this research uses a specific philosophical ontology – Bunge's ontology - as a philosophical base for analysis. However, other choices are possible and can be explored in further research. Second, this thesis only covers the analysis and representation of a limited number of Bunge ontology concepts – things, properties, interaction, classes, and composition. Representation in OWL of other constructs of Bunge's ontology (such as states, events and laws) was not discussed here and can be a subject of future research. Nevertheless, the OWL-specific representation methods used in this work

for the selected constructs provide ideas and serve as a good starting point for the analysis and development of the representation guidelines for other Bunge's constructs in the future research. A third limitation of this study is that the applicability of the approach was demonstrated only on small examples. While this initial experience is encouraging, future research may consider more extensive case studies, larger ontologies, and a variety of real-world domains to test the applicability of the proposed rules in real-world settings, and to identify issues which may need refining or revision due to usability or implementation considerations. Finally, this study is theoretical, so one can only hope (based on the intial experience and the encouraging related prior research results) that using the proposed approach will lead to better OWL ontologies. Formal empirical studies need to be conducted to evaluate whether the use of the proposed modeling approach does indeed provide any benefits for domain knowledge representation and understanding.

In addition to the above mentioned issues for further research, a number of other potentially interesting extensions of this study are possible. First, one of the important areas of research in ontological engineering is ontology alignment, merging, and evaluation. The suggested meta-model and modeling guidelines may provide some benefits in this area, and future research may explore specifically how the proposed methodology may be helpful in these aspects. Second, the methodology may also suggest how ontologies can be evaluated based on philosophical concepts. Further research may use the proposed meta-model and methodology to develop criteria for evaluating specific OWL ontologies (i.e. whether domain elements are property represented by language constructs).

Also, the proposed methodology and meta-model may be helpful for the translation of models in other languages to OWL ontologies, and vice versa. The need for mapping or translation from model in one language to another (e.g. from UML model to OWL ontology) may arise, for example, in the areas of interoperability and integration of heterogeneous information sources. It may also be useful for knowledge sharing and reuse, as it would facilitate the reuse of the available knowledge representation in other models. Specifically, as discussed in section 2 (related research), some prior studies also proposed a mapping between constructs of other languages (such as UML or ERM) and the constructs of Bunge's ontology. Thus, using Bunge's ontology as an 'intermediary' it may be possible to develop mapping between OWL elements and the constructs of another language (e.g. UML) (taking into account the ontological semantics assigned to the elements of the translated language). Such a mapping would facilitate the translation between models in another language and OWL models (assuming both models are developed in accordance with their respective

mapping and ontologically grounded guidelines), and could even be automated to some extent.

Finally, an interesting area for future research would be the development of a CASE tool to facilitate the development of OWL ontologies in accordance with the proposed methodology. This could be the development of a new tool or an extension of some existing ontology developments environment, such as Protégé-OWL (which allows creation of wizards and customized applications)[76]. Such a CASE tool would allow modelers to develop ontologies based on Bunge's ontological concepts in accordance with the proposed rules and guidelines. Rather than operating directly in terms of OWL constructs, the system could guide modelers by asking them to specify model elements in terms of Bunge's ontological concepts, i.e. modelers would specify main classes of substantial things and their intrinsic properties, interactions and related sets of mutual properties, instances of classes and their specific properties, and so on. Based on the proposed rules for representing these concepts, the CASE tool would automatically create the respective OWL elements (and their combinations) for representing the domain ontological elements specified by the modeler. The system could also automatically link elements as required by rules (for example, associate classes with properties using property restrictions, set cardinalities, or link interaction classes and participating substantial thing classes with special object properties). Then the user would be able to review and modify the resulting OWL ontology in both OWL and Bunge terms. Also, the tool could provide additional functionality including configurable automatic creation of meta-model elements (classes and properties), configurable naming conventions for the ontological elements (prefixes, etc) as well as model checking capabilities (i.e. the ability to check that an ontology created or modified by the user satisfies key ontological rules and guidelines). Finally, such tool could also provide enhanced querying capabilities allowing queries not only in terms of the standard OWL constructs (for example "List OWL individuals possessing a certain OWL property" or "Which OWL classes a particular OWL individual belongs to?") but also in terms of the ontological concepts. For example, users could ask queries such as "What intrinsic properties a certain class of substantial things has?", "In which interactions a particular substantial thing participates?", or "List all components of a particular composite thing". We believe that the development of such tool is feasible (albeit not necessarily easy) and is a promising future research direction.

[76] The Protégé Ontology Editor and Knowledge Acquisition System is a free open-source Java tool providing an extensible architecture for the creation of customized knowledge-based applications and ontologies. The Protégé-OWL is an extension of Protégé (http://protege.stanford.edu/overview/protege-owl.html)

To conclude, we believe that this research is an encouraging step towards improving - based on the ontological philosophical foundations - the ontology development process and the expressiveness, clarity, and consistency of the resulting OWL ontologies. This work is theoretically grounded and contributes to the existing research on the ontological analysis of conceptual languages as well as to the area of ontological engineering. It also gives some promising results for practical OWL developers by providing specific rules and guidelines which can be relatively easily applied in the existing OWL development environments and ontologies. Finally, even though this work has considered the OWL language in its current form and has not dealt with the issue of adding new constructs to OWL as a way to improve its expressiveness, the results of this work suggest some ideas and guidance for possible future extensions and modifications of the OWL language as well as for improvement and enhancement of the existing ontology development environments.

BIBLIOGRAPHY

Angeles, P. (1981). *Dictionary of Philosophy*. Harper Perennial, New York, NY

Antoniou, G. and van Harmelen, F. (2004). Web Ontology Language: OWL. In Staab, S. and Studer, R., eds., *Handbook on Ontologies in Information Systems,* 67-92.

Bechhofer, S. (2003). OWL reasoning examples (draft paper), December 3, 2003, retrived September 15, 2004, from http://owl.man.ac.uk/2003/why/20031203.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A. (2004). OWL Web Ontology Language Reference, W3C Recommendation, Dean, M., Schreiber, G. (eds.), February 10, 2004, http://www.w3.org/TR/owl-ref/

Bera, P. and Wand, Y. (2004). Analyzing OWL using a Philosophy-based Ontology. In *Proceedings of the Third International Conference on Formal Ontology in Information Systems (FOIS-2004)*. Torino, Italy, 353-362.

Bera, P., Krasnoperova, A., and Wand, Y. (2005). Improving Real-World semantics in OWL. Paper submitted to IEEE Transactions on Knowledge and Data Engineering.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *ScientificAmerican,* 284(5), May 2001, 34-43.

Bodart, F., Patel, A., Sim, M. and Weber, R. (2001). Should Optional Properties Be Used In Conceptual Modelling? A Theory And Three Empirical Tests. *Information Systems Research,* 12 (4): 2001, 384-405.

Bunge, M. (1977). *Ontology 1: The Furniture of the World*, vol. 3. New York: D. Reidel Publishing, 1977.

Bunge, M. (1979). *Ontology II: A World of Systems*, vol. 4. New York: D. Reidel Publishing, 1979.

Burton-Jones, A. and Weber, R. (1999). Understanding Relationships with Attributes in Entity Relationship Diagrams. In P. De and J. DeGross (Eds.). *Proceedings of the Twentieth International Conference on Information Systems*, Charlotte, NC, 214-228.

Burton-Jones, A. and Weber, R. (2003). Properties do not have properties: Investigating a questionable conceptual modeling practice. In D. Batra, J. Parsons, and V. Ramesh (Eds.), *Proceedings of the Second Annual Symposium on Research in Systems Analysis and Design*, Miami, 14 pp.

Carroll, J.J., De Roo, J. (2004). OWL Web Ontology Language Test Cases. W3CWorld Wide Web Consortium, Recommendation, REC-owl-test-20040210, February, 2004 http://www.w3.org/TR/2004/REC-owl-test-20040210

Chandrasekaran, B., Josephson, J. R. and Benjamins, V. R. (1999). What Are Ontologies, and Why Do We Need Them?. IEEE Intelligent Systems 14 (1), 20-26.

Evermann, J. (2003). Using design languages for conceptual modeling: The UML case. Unpublished PhD dissertation, Sauder School of Business, University of British Columbia.

Evermann, J. and Wand, Y. (2001a). An ontological examination of object interaction in conceptual modeling. In *Proceedings of the Workshop on Information Technologies and Systems WITS'01, New Orleans, December 15-16, 2001.*

Evermann, J. and Wand, Y. (2001b). Towards ontologically based semantics for UML constructs." In H. Kunii, S. Jajodia, and A. Solvberg, eds., *Proceedings of the 20th International Conference on Conceptual Modeling, Yokohama, Japan, Nov. 27-30, 2001.*

Evermann J, Wand Y (2005) Toward formalizing domain modeling semantics in language syntax *IEEE Transactions On Software Engineering* 31 (1): 21-37

Fensel, D. (2001). *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Springer, Berlin.

Fikes, R., Hayes, P., and Horrocks, I. (2004). OWL-QL - a language for deductive query answering on the Semantic Web. *J. of Web Semantics,* 2(1), 19-29.

Gangemi, A., Guarino, N., Masolo, C., and Oltramari, A. (2001). Understanding top-level ontological distinctions. *Proceedings of the 2001 IJCAI Workshop on Ontologies and Information Sharing.*

Gemino, A. (1998). To be or may to be: An empirical comparison of mandatory and optional properties in conceptual modelling. *Proc. Ann. Conf. Admin. Sci. Assoc. of Canada, Information Systems Division*. Saskatoon, Saskatchewan, 33–44.

Gemino, A. and Wand, Y. (2000). Comparing Mandatory and Optional Properties in Conceptual Data Modeling. In P. Bowen and V. Mookerjee (Eds.), *Proceedings of the Tenth Annual Workshop on Information Technologies and Systems*, Brisbane, 97-102.

Gomez-Perez, A. 1998. Knowledge Sharing and Reuse. In *The Handbook of Applied Expert Systems*, ed. J. Liebowitz, 10-1-10-36. Northwest Boca Raton, Fla.: CRC.

Gomez-Perez, A., Fernandez-Lopez, M., and Corcho, O. (2004) *Ontological Engineering: with examples for the area of knowledge management, e-commerce and the Semantic Web*: Springer-Verlag London Limited Guarino, N. and Welty, C. (2000) A Formal Ontology of Properties. In, Dieng, R., and Corby, O., eds, *Proceedings of EKAW-2000: The 12th International Conference on Knowledge Engineering and Knowledge Management.* Spring-Verlag LNCS Vol. 1937:97-112.

Gruber, T. R. (1993) A Translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2): 199-220.

Gruninger, M., and Fox, M. S. 1995. Methodology for the Design and Evaluation of Ontologies. Paper presented at the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, 19-21 August, Montreal, Quebec, Canada.

Guarino, N. (ed.) (1998) *Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998.* Amsterdam, IOS Press, pp. 3-15.

Guarino, N. and Giaretta, P. (1995). Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Sharing, N. Mars (eds.), *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge,* IOS Press, Amsterdam, 25-32.

Haarslev, V. and Möller, R. (2003). Racer: An OWL Reasoning Agent for the Semantic Web. In *Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with the 2003 IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, October 13, 91–95.

Haarslev, V., Möller, R. , and Wessel, M. (2004). Querying the Semantic Web with Racer + nRQL. In *Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04),* Ulm, Germany, September 24, 2004.

Heflin, J., ed. (2004). OWL Web Ontology Language Use Cases and Requirements. W3C Recommendation, February, 10, 2004, http://www.w3.org/TR/2004/REC-webont-req-20040210/

Horridge, M. (2004). *A practical guide to building OWL ontologies with the Protégé-OWL Plugin.* Edition 1.0, June 13, 2004, retrived July 30, 2004, from http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

Horrocks, I. and Patel-Schneider, P.F. (2004). A Proposal for an OWL Rules Language. In *Proc. Int. WWW Conference*, May 17-22, 2004, New York, USA.

Horrocks, I., Patel-Schneider, P.F., and van Harmelen, F. (2003). From *SHIQ* and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1), 7-26.

Horrocks, I., Patel-Schneider, P.F., Bechhofer, S., and Tsarkov, D.(2005). OWL rules: A proposal and prototype implementation. *J. of Web Semantics*, 3(1), 23-40.

Lenat, D. B., and Guha, R. V. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project.* Reading, Mass.: Addison-Wesley.

McGuinness, D. L. (2002). Ontologies Come of Age. In D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, eds. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential.* MIT Press.

McGuinness, D. L., Smith, K. M., and Welty, C., eds. (2004). OWL Web Ontology Language Guide. W3C Recommendation, February, 10 2004, http://www.w3.org/TR/2004/REC-owl-guide-20040210/

Noy, N., ed. (2004). Representing Classes as Property Values on the Semantic Web. W3C Working Draft, July 21, 2004, http://www.w3.org/TR/2004/WD-swbp-classes-as-values-20040721/

Noy, N.F. and Hafner, C.D. (1997). The State of the Art in Ontology Design – A Survey and Comparative Review. *AI Magazine,* 36(3), 53-74.

Noy, N. F. and McGuinness D.L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory, Technical Report KSL-01-05.

Noy, N. and Rector, A., eds. (2004). Defining N-ary Relations on the Semantic Web: Use with Individuals. W3C Working Draft, July 21, 2004, http://www.w3.org/TR/2004/WD-swbp-n-aryRelations-20040721/

Patel-Schneider, P.F., Patrick Hayes, P., Ian Horrocks, I. (eds) (2003). Web Ontology Language (OWL) Abstract Syntax and Semantics. W3C Working Draft, February 3, 2003, http://www.w3.org/TR/owl-semantics/

Parsons, J. and Cole, L. (2004). An Experimental Evaluation of Property Precedence in Conceptual Modelling. *APCCM 2004*: 101-110

Parsons, J. and Wand, Y. (1997). Choosing Classes in Conceptual Modeling. *Communications of the ACM, 40 (6)*, 63-69

Parsons, J. and Wand Y. (2000). Emancipating Instances from the Tyranny of Classes, *ACM Transactions on Database Systems*, 25: 228-268.

Parsons, J. and Wand, Y. (2003). Attribute-Based Semantic Reconciliation of Multiple Data Sources, *Journal on Data Semantics*, Vol. 1.1, October 2003.

Pease, A., Niles, I., and Li, J., (2002). The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications, in *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web.*

Rector, A., ed. (2004). Representing Specified Values in OWL: "value partitions" and "value sets". W3C Working Draft, August, 3, 2004, http://www.w3.org/TR/2004/WD-swbp-specified-values-20040803/

Rector, A. and Welty, C., eds. (2005). Simple part-whole relationships in OWL ontologies. W3C Editor's Draft, August 11, http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/

Smith B. (2003). Ontology; in: Floridi, L. (ed.): *Blackwell Guide to the Philosophy of Computing and Information*, Blackwell, Oxford, 2003, 155-166. http://ontology.buffalo.edu/smith/articles/ontology_pic.pdf

Sowa, J. F. (1997). *Knowledge Representation: Logical, Philosophical and Computational Foundations.* Boston: PWS.

Uschold, M., and Gruninger, M. (1996). Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review* 11(2), 93-155.

Wand, Y. (1989). A Proposal for a Formal Model of Objects. In *Object-Oriented Concepts, Databases, and Applications*. Ed. Kim, W. & Lochovsky, F. New York, ACM Press, 537-559.

Wand, Y. and Wang, R.Y. (1996) Anchoring data quality dimensions in ontological foundations. *Communications of the ACM* 39(11) , 86-95.

Wand, Y. and Weber, R. (1989).An ontological evaluation of systems analysis and design methods. In: E.D. Falkenberg, P. Lindgreen (Eds.), *Information System Concepts: An In-depth Analysis*, North-Holland, Amsterdam, pp. 79-107.

Wand, Y. and Weber, R. (1990a) An Ontological Model of an Information System, IEEE Transactions on Software Engineering, v.16 n.11, p.1282-1292, November 1990.

Wand, Y. and Weber, R. (1990b). Mario Bunge's ontology as a formal foundation for information systems concepts. In *Studies on Mario Bunge's Treatise*, vol. 18, *Poznan Studies in Philosophy of Sciences and Humanities*, G. W. D. Dorn, Ed..

Wand, Y. and Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems*, 3: 217-237.

Wand, Y. and Weber, R. (1995). On the deep structure of information systems. *Journal of Information Systems*, 5: 203-223.

Wand, Y., Monarchi, D. E., Parsons, J., and Woo, C. C. (1995). Theoretical foundations for conceptual modelling in information systems development. *Decision Support Systems*, 15: 285-304.

Wand, Y., Storey, V. C., and Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems*, 24: 494-528.

Weber, R. (1997). Ontological Foundations of Information Systems: Coopers and Lybrand Research Methodology. *Coopers and Lybrand Research Methodology*, Melbourne.

W3C press release (2004). Wide Web Consortium Issues RDF and OWL Recommendations. February 10, 2004, retrieved from http://www.w3.org/2004/01/sws-pressrelease

APPENDICES

APPENDIX A Employment example (section 4.2.4.4) – diagrams and RDF/XML syntax

Graphical representation of the employment example ontology

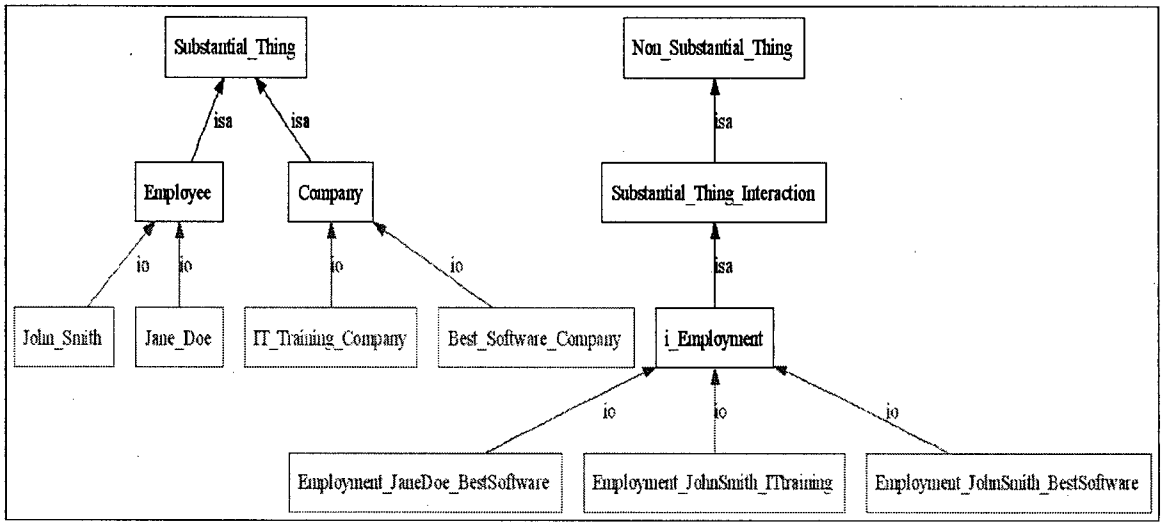

Figure 8: Employment example ontology - hierarchy of classes and instances

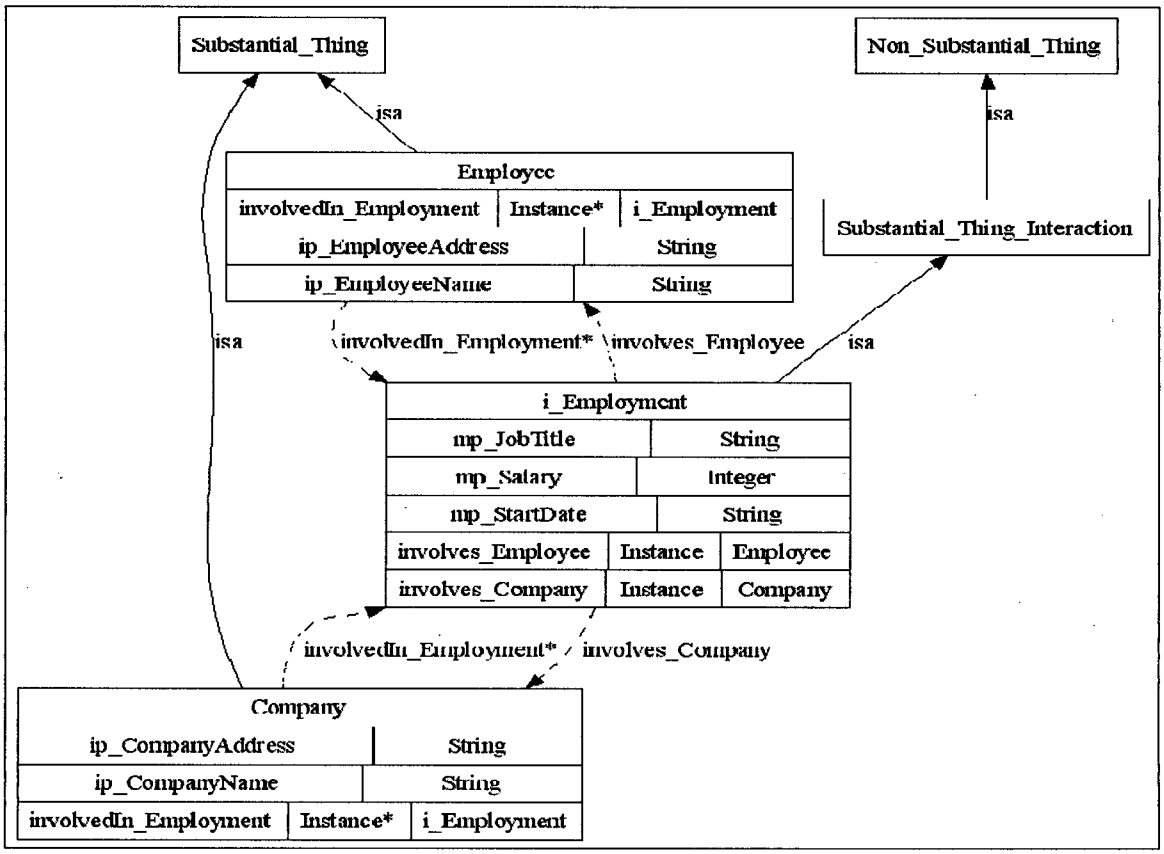

Figure 9: Employment example ontology – classes and properties

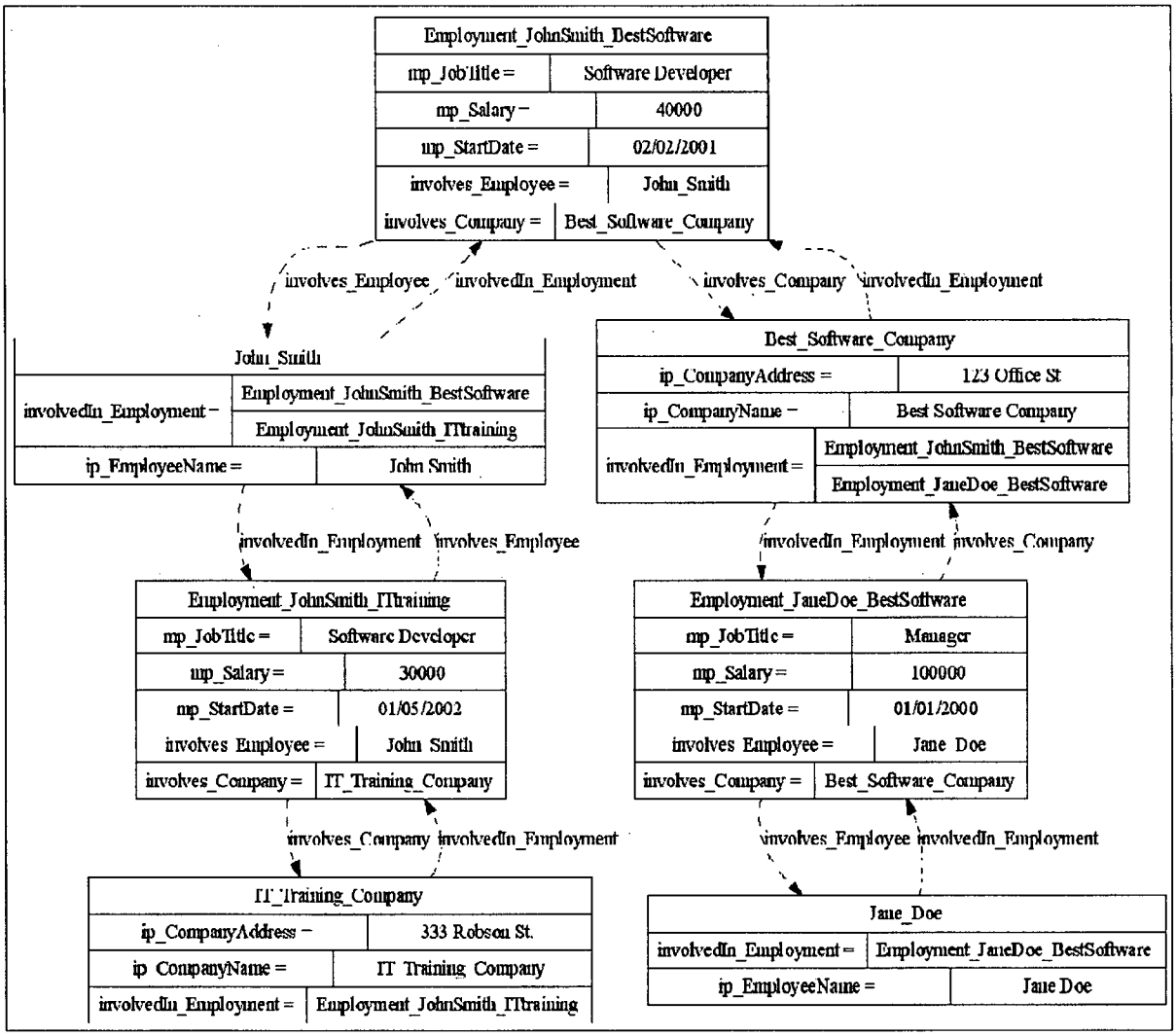

Figure 10: Employment example ontology – instances and their properties

RDF/XML OWL representation of the employment example ontology

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
   xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>

<!- Declaring meta-model classes: Substantial_Thing, Non_Substantial_Thing, Substantial_Thing_Interaction-->

 </owl:Class>
  <owl:Class rdf:ID="Substantial_Thing">
   <owl:disjointWith>
    <owl:Class rdf:ID="Non_Substantial_Thing"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Non_Substantial_Thing">
   <owl:disjointWith rdf:resource="#Substantial_Thing"/>
  </owl:Class>
  <owl:Class rdf:about="#Substantial_Thing_Interaction">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Non_Substantial_Thing"/>
   </rdfs:subClassOf>
  </owl:Class>

<!-- Declaring classes of substantial things (Employee and Company) and their intrinsic properties -->

<owl:Class rdf:ID="Company">
   <rdfs:subClassOf rdf:resource="#Substantial_Thing"/>
   <rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="involvedIn_Employment"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="ip_CompanyAddress"/>
     </owl:onProperty>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="ip_CompanyName"/>
     </owl:onProperty>
    </owl:Restriction>
```

```
  </rdfs:subClassOf>
 </owl:Class>

<owl:DatatypeProperty rdf:about="#ip_CompanyName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Company"/>
 </owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#ip_CompanyAddress">
  <rdfs:domain rdf:resource="#Company"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>

<owl:Class rdf:ID="Employee">
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Substantial_Thing"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_EmployeeAddress"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_EmployeeName"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:about="#involvedIn_Employment"/>
    </owl:onProperty>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
   </owl:Restriction>
 </owl:Class>

<owl:DatatypeProperty rdf:about="#ip_EmployeeName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Employee"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_EmployeeAddress">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
  <rdfs:domain rdf:resource="#Employee"/>
 </owl:DatatypeProperty>

<!—Declaring the interaction class (i_Employment), mutual properties associated with it, and special (linking)
object properties involvedIn_Employment, Involves_Company, involves_Employee -->

<owl:Class rdf:ID="i_Employment">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Substantial_Thing_Interaction"/>
  </rdfs:subClassOf>
```

```
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="involves_Company"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="involves_Employee"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="mp_JobTitle"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="mp_StartDate"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="mp_Salary"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
 <owl:DatatypeProperty rdf:about="#mp_JobTitle">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#i_Employment"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#mp_Salary">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#positiveInteger"/>
  <rdfs:domain rdf:resource="#I_Employment"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#mp_StartDate">
  <rdfs:domain rdf:resource="#I_Employment"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>

 <owl:ObjectProperty rdf:about="#involvedIn_Employment">
  <rdfs:domain>
   <owl:Class>
```

```
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Company"/>
        <owl:Class rdf:about="#Employee"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#i_Employment"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#involves_Employee">
  <rdfs:domain rdf:resource="#i_Employment"/>
  <rdfs:range rdf:resource="#Employee"/>
 </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#involves_Company">
  <rdfs:domain rdf:resource="#i_Employment"/>
  <rdfs:range rdf:resource="#Company"/>
 </owl:ObjectProperty>
```

<!-- Example declaration of instances (specific companies and employees), values for their intrinsic properties and instances of interactions linking specific employee and company in an employment interaction and its related values of mutual properties[77] *-->*

```
 <i_Employment rdf:ID="Employment_JaneDoe_BestSoftware">
   <mp_Salary rdf:datatype="http://www.w3.org/2001/XMLSchema#int">100000</mp_Salary>
   <mp_JobTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Manager</mp_JobTitle>
   <involves_Employee>
    <Employee rdf:ID="Jane_Doe">
     <ip_EmployeeAddress rdf:datatype="http://www.w3.org/2001/XMLSchema#date">999 Main St
     </ip_EmployeeAddress>
     <ip_EmployeeName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Jane Doe
     </ip_EmployeeName>
     <involvedIn_Employment rdf:resource="#Employment_JaneDoe_BestSoftware"/>
    </Employee>
   </involves_Employee>
   <involves_Company>
    <Company rdf:ID="Best_Software_Company">
     <involvedIn_Employment>
      <i_Employment rdf:ID="Employment_JohnSmith_BestSoftware">
       <involves_Employee>
        <Employee rdf:ID="John_Smith">
         <involvedIn_Employment>
          <i_Employment rdf:ID="Employment_JohnSmith_ITtraining">
           <mp_Salary rdf:datatype="http://www.w3.org/2001/XMLSchema#int">30000</mp_Salary>
           <mp_StartDate rdf:datatype="http://www.w3.org/2001/XMLSchema#string">01/05/2002
           </mp_StartDate>
           <involves_Company>
            <Company rdf:ID="IT_Training_Company">
             <involvedIn_Employment rdf:resource="#Employment_JohnSmith_ITtraining"/>
             <ip_CompanyName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >IT_Training_Company </ip_CompanyName>
             <ip_CompanyAddress rdf:datatype="http://www.w3.org/2001/XMLSchema#string">333 Robson
             St.</ip_CompanyAddress>
```

[77] OWL syntax allows multiple ways of representing the same statements and sequence of statements about classes, properties and instances is not important. For example there are several ways to declare instances and make assertions their properties. The above code was generated automatically in Protégé, so it is not as well arranged as manually written code could be (but it still represents the same intended information)

```
            </Company>
          </involves_Company>
          <mp_JobTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Software Developer
          </mp_JobTitle>
          <involves_Employee rdf:resource="#John_Smith"/>
        </i_Employment>
      </involvedIn_Employment>
      <involvedIn_Employment rdf:resource="#Employment_JohnSmith_BestSoftware"/>
      <ip_EmployeeName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >John Smith</ip_EmployeeName>
      <ip_EmployeeAddress rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
      >123 Oak St</ip_EmployeeAddress>
    </Employee>
  </involves_Employee>
  <involves_Company rdf:resource="#Best_Software_Company"/>
  <mp_Salary rdf:datatype="http://www.w3.org/2001/XMLSchema#int">40000</mp_Salary>
  <mp_StartDate rdf:datatype="http://www.w3.org/2001/XMLSchema#string">02/02/2001</mp_StartDate>
  <mp_JobTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Software Developer
  </mp_JobTitle>
 </i_Employment>
</involvedIn_Employment>
<involvedIn_Employment rdf:resource="#Employment_JaneDoe_BestSoftware"/>
<ip_CompanyName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Best Software Company
</ip_CompanyName>
<ip_CompanyAddress rdf:datatype="http://www.w3.org/2001/XMLSchema#string">123 Office St
</ip_CompanyAddress>
</Company>
</involves_Company>
<mp_StartDate rdf:datatype="http://www.w3.org/2001/XMLSchema#string">01/01/2000</mp_StartDate>
</i_Employment>
</rdf:RDF>

<!-- Created with Protégé (with OWL Plugin 1.3, Build 225.1)  http://protege.stanford.edu -->
```

APPENDIX B Person class example (section 4.3.3) - RDF/XML syntax

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about="An example of declaring a class in terms of properties"/>

<!-- Declaring meta-model classes: Substantial_Thing, Non_Substantial_Thing -->

<owl:Class rdf:about="#Substantial_Thing">
   <owl:disjointWith>
     <owl:Class rdf:ID="Non_Substantial_Thing"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Non_Substantial_Thing">
   <owl:disjointWith rdf:resource="#Substantial_Thing"/>
  </owl:Class>

<!-- Declaring the class  Person and its intrinsic properties -->

<owl:Class rdf:ID="Person">
   <rdfs:subClassOf>
     <owl:Class rdf:ID="Substantial_Thing"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:DatatypeProperty rdf:ID="ip_Name"/>
       </owl:onProperty>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
       <owl:onProperty>
         <owl:DatatypeProperty rdf:ID="ip_DateOfBirth"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
       <owl:onProperty>
         <owl:DatatypeProperty rdf:ID="ip_Gender"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:DatatypeProperty rdf:about="#ip_Name">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>

 <owl:DatatypeProperty rdf:about="#ip_Gender">
  <rdfs:range>
   <owl:DataRange>
    <owl:oneOf rdf:parseType="Resource">
     <rdf:rest rdf:parseType="Resource">
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">F</rdf:first>
     </rdf:rest>
     <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">M</rdf:first>
    </owl:oneOf>
   </owl:DataRange>
  </rdfs:range>
  <rdfs:domain rdf:resource="#Person"/>
 </owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#ip_DateOfBirth">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
 </owl:DatatypeProperty>

<!-- Declaring a class of substantial things Female_Person (an example of using owl:hasValue restriction); this
class inherits all the constraints from the class Person and has an extra constraint of having a specific value (F)
for the ip_Gender property -->

<owl:Class rdf:ID="Female_Person">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:about="#ip_Gender"/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string">F</owl:hasValue>
   </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>

<!-- Declaring a specific instance of the class Person and its properties (i.e. property values for the generic
properties -->

<Person rdf:ID="JohnSmith">
  <ip_Name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">John W. Smith</ip_Name>
  <ip_Gender rdf:datatype="http://www.w3.org/2001/XMLSchema#string">M</ip_Gender>
  <ip_DateOfBirth rdf:datatype="http://www.w3.org/2001/XMLSchema#date">05/04/1960</ip_DateOfBirth>
 </Person>

</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 1.3, Build 225.1) http://protege.stanford.edu -->
```

APPENDIX C Meta-model (Chapter 5) - RDF/XML syntax

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
 <owl:Ontology rdf:about="Meta-model"/>

<owl:Class rdf:ID="Substantial_Thing">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Meta-model class of all OWL
things representing ontological substantial things (in Bunge's ontology sense)</rdfs:comment>
  <owl:disjointWith>
   <owl:Class rdf:about="#Non_Substantial_Thing"/>
  </owl:disjointWith>
 </owl:Class>

<owl:Class rdf:about="#Non_Substantial_Thing">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Meta-model class of all OWL
things that are not representing ontological substantial things</rdfs:comment
  <owl:disjointWith rdf:resource="#Substantial_Thing"/>
</owl:Class>

<owl:Class rdf:about="#Substantial_Thing_Interaction">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Meta-model class used to
represent interaction classes (to models sets of mutual properties) </rdfs:comment
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Non_Substantial_Thing"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Property_Value"/>
 </owl:Class>

<owl:Class rdf:ID="Property_Value">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Meta-model class (optional)
used to represent special classes and instances modeling enumerated property values</rdfs:comment
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Non_Substantial_Thing"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
   <owl:Class rdf:ID="Substantial_Thing_Interaction"/>
  </owl:disjointWith>
 </owl:Class>

<owl:Class rdf:ID="Composite_Thing">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A class of all substantial things
that are composite things</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Substantial_Thing"/>
  <owl:disjointWith>
   <owl:Class rdf:about="#Simple_Thing"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="isComposedOf"/>
```

```
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isComposedOf"/>
     </owl:onProperty>
     <owl:allValuesFrom>
      <owl:Class rdf:ID="Component_Thing"/>
     </owl:allValuesFrom>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>

<owl:Class rdf:about="#Component_Thing">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A class of all substantial things
that are components of some composite substantial thing</rdfs:comment>
   <rdfs:subClassOf rdf:resource="#Substantial_Thing"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="isComponentOf"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isComponentOf"/>
     </owl:onProperty>
     <owl:allValuesFrom rdf:resource="#Composite_Thing"/>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>

 <owl:Class rdf:about="#Simple_Thing">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Meta-model class of all
substantial things  which are not composed of any other ontological substantial things</rdfs:comment
    <rdfs:subClassOf rdf:resource="#Substantial_Thing"/>
    <owl:disjointWith rdf:resource="#Composite_Thing"/>
  </owl:Class>

 <owl:ObjectProperty rdf:about="#isComposedOf">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Object property used to
represent that a certain substantial thing has another substantial thing as a component. It links an
instance of a Composite_Thing class to an instanse of a Component_Thing class. Inverse property -
isComponentOf.
  </rdfs:comment>
   <rdfs:domain rdf:resource="#Composite_Thing"/>
   <rdfs:range rdf:resource="#Component_Thing"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="#isComponentOf">
```

```
    <rdfs:domain rdf:resource="#Component_Thing"/>
    <rdfs:range rdf:resource="#Composite_Thing"/>
  </owl:ObjectProperty>

</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 1.3, Build 225.1)  http://protege.stanford.edu -->
```

APPENDIX D Library example (Chapter 6) - RDF/XML syntax

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>

<!-- Declaring meta-model classes and properties -->

<owl:Class rdf:about="#Substantial_Thing">
   <owl:disjointWith>
     <owl:Class rdf:ID="Non_Substantial_Thing"/>
   </owl:disjointWith>
  </owl:Class>
<owl:Class rdf:about="#Non_Substantial_Thing">
   <owl:disjointWith rdf:resource="#Substantial_Thing"/>
 </owl:Class>
 <owl:Class rdf:about="#Substantial_Thing_Interaction">
   <rdfs:subClassOf>
     <owl:Class rdf:about="#Non_Substantial_Thing"/>
   </rdfs:subClassOf>
  </owl:Class>

<owl:ObjectProperty rdf:about="#isComponentOf">
   <owl:inverseOf rdf:resource="#isComposedOf"/>
  </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isComposedOf">
   <owl:inverseOf>
     <owl:ObjectProperty rdf:about="#isComponentOf"/>
   </owl:inverseOf>
  </owl:ObjectProperty>

<!-- Declaring the classes of substantial things and associating them with properties using property
restrictions-->

<owl:Class rdf:ID="Person">
   <rdfs:subClassOf>
     <owl:Class rdf:ID="Substantial_Thing"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:DatatypeProperty rdf:ID="ip_DateOfBirth"/>
       </owl:onProperty>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
```

```
        <owl:DatatypeProperty rdf:ID="ip_PersonAddress"/>
       </owl:onProperty>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
      </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
      <owl:Restriction>
       <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="ip_PersonName"/>
       </owl:onProperty>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
      </owl:Restriction>
     </rdfs:subClassOf>
    </owl:Class>

<owl:Class rdf:about="#Library_Member">
   <rdfs:subClassOf rdf:resource="#Person"/>
   <rdfs:subClassOf>
    <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
       <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#involvedIn_LibraryMembership"/>
       </owl:onProperty>
      </owl:Restriction>
      <owl:Restriction>
       <owl:allValuesFrom>
        <owl:Class rdf:about="#I_Library_Membership"/>
       </owl:allValuesFrom>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#involvedIn_LibraryMembership"/>
       </owl:onProperty>
      </owl:Restriction>
     </owl:intersectionOf>
    </owl:Class>
   </rdfs:subClassOf>
  </owl:Class>

<owl:Class rdf:about="#Borrower">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Library_Member"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:ID="involvedIn_ItemLoan"/>
       </owl:onProperty>
       <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
      </owl:Restriction>
      <owl:Restriction>
       <owl:allValuesFrom rdf:resource="#I_ItemLoan"/>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#involvedIn_ItemLoan"/>
```

```
        </owl:onProperty>
       </owl:Restriction>
      </owl:intersectionOf>
     </owl:Class>
    </rdfs:subClassOf>
   </owl:Class>

<owl:Class rdf:about="#Library">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Substantial_Thing"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_LibraryName"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_LibraryAddress"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_NumberOfSections"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:ID="involvedIn_LibraryMembership"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
     </owl:Restriction>
     <owl:Restriction>
      <owl:allValuesFrom><owl:Class rdf:ID="I_Library_Membership"/> </owl:allValuesFrom>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#involvedIn_LibraryMembership"/>
      </owl:onProperty>
     </owl:Restriction>
    </owl:intersectionOf>
   </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
```

```
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:ID="involvedIn_ItemLoan"/>
       </owl:onProperty>
       <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
      </owl:Restriction>
      <owl:Restriction>
       <owl:allValuesFrom><owl:Class rdf:ID="I_ItemLoan"/> </owl:allValuesFrom>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#involvedIn_ItemLoan"/>
       </owl:onProperty>
      </owl:Restriction>
     </owl:intersectionOf>
    </owl:Class>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="isComposedOf"/>
     </owl:onProperty>
     <owl:someValuesFrom><owl:Class rdf:ID="Library_Section"/></owl:someValuesFrom>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>

 <owl:Class rdf:about="#Library_Section">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Substantial_Thing"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="ip_SectionName"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="isComponentOf"/>
     </owl:onProperty>
     <owl:someValuesFrom rdf:resource="#Library"/>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>

 <owl:Class rdf:about="#Library_Item">
   <rdfs:subClassOf rdf:resource="#Substantial_Thing"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="ip_Subject"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_Title"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_Barcode"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="ip_LoanDaysAllowed"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>

<owl:Class rdf:about="#Borrowed_Item">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Library_Item"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#involvedIn_ItemLoan"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
     </owl:Restriction>
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#involvedIn_ItemLoan"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#I_ItemLoan"/>
     </owl:Restriction>
    </owl:intersectionOf>
   </owl:Class>
  </rdfs:subClassOf>
 </owl:Class>

<owl:Class rdf:ID="Book">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Library_Item"/>
  </rdfs:subClassOf>
```

```
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:ID="ip_Publisher"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:ID="ip_Author"/>
      </owl:onProperty>
     </owl:Restriction>
    </rdfs:subClassOf>
   </owl:Class>

  <owl:Class rdf:ID="Magazine">
    <rdfs:subClassOf>
     <owl:Class rdf:about="#Library_Item"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:ID="ip_Volume"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:ID="ip_Issue"/>
      </owl:onProperty>
     </owl:Restriction>
    </rdfs:subClassOf>
   </owl:Class>

  <owl:Class rdf:ID="Music_CD">
    <rdfs:subClassOf>
     <owl:Class rdf:about="#Library_Item"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality>
      <owl:onProperty>
       <owl:DatatypeProperty rdf:ID="ip_Artist"/>
      </owl:onProperty>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
      <owl:onProperty>
```

```
      <owl:DatatypeProperty rdf:ID="ip_Content"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>

<!-- Declaring the interaction classes and associating them with properties using property restrictions --
>

<owl:Class rdf:about="#I_Library_Membership">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Substantial_Thing_Interaction"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="mp_LibraryCardNumber"/>
     </owl:onProperty>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="mp_StartDate"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="mp_MembershipStatus"/>
     </owl:onProperty>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:ID="involves_LibraryMember"/>
       </owl:onProperty>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
      </owl:Restriction>
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#involves_LibraryMember"/>
       </owl:onProperty>
       <owl:allValuesFrom rdf:resource="#Library_Member"/>
      </owl:Restriction>
     </owl:intersectionOf>
    </owl:Class>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
```

```
    <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
       <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#involves_Library"/>
       </owl:onProperty>
      </owl:Restriction>
      <owl:Restriction>
       <owl:onProperty>
        <owl:ObjectProperty rdf:about="#involves_Library"/>
       </owl:onProperty>
       <owl:allValuesFrom rdf:resource="#Library"/>
      </owl:Restriction>
     </owl:intersectionOf>
    </owl:Class>
   </rdfs:subClassOf>
  </owl:Class>

<owl:Class rdf:ID="I_ItemLoan">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Substantial_Thing_Interaction"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="mp_dateOut"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    <owl:onProperty>
     <owl:DatatypeProperty rdf:ID="mp_dateDue"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:ID="involves_Borrower"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     </owl:Restriction>
     <owl:Restriction>
      <owl:allValuesFrom>
       <owl:Class rdf:ID="Borrower"/>
      </owl:allValuesFrom>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#involves_Borrower"/>
      </owl:onProperty>
     </owl:Restriction>
```

```
    </owl:intersectionOf>
   </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:ID="involves_Library"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     </owl:Restriction>
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#involves_Library"/>
      </owl:onProperty>
      <owl:allValuesFrom>
       <owl:Class rdf:ID="Library"/>
      </owl:allValuesFrom>
     </owl:Restriction>
    </owl:intersectionOf>
   </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty rdf:ID="involves_BorrowedItem"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
     </owl:Restriction>
     <owl:Restriction>
      <owl:allValuesFrom>
       <owl:Class rdf:ID="Borrowed_Item"/>
      </owl:allValuesFrom>
      <owl:onProperty>
       <owl:ObjectProperty rdf:about="#involves_BorrowedItem"/>
      </owl:onProperty>
     </owl:Restriction>
    </owl:intersectionOf>
   </owl:Class>
  </rdfs:subClassOf>
 </owl:Class>

 <!-- Declaring substantial properties (intrinsic and mutual) -->

 <owl:DatatypeProperty rdf:about="#ip_PersonName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Person"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_PersonAddress">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Person"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_DateOfBirth">
```

```
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_LibraryName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Library"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_LibraryAddress">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Library"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_NumberOfSections">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonPositiveInteger"/>
  <rdfs:domain rdf:resource="#Library"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_SectionName">
  <rdfs:domain rdf:resource="#Library_Section"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>

 <owl:DatatypeProperty rdf:about="#ip_Title">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Library_Item"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_Subject">
  <rdfs:domain rdf:resource="#Library_Item"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_Barcode">
  <rdfs:domain rdf:resource="#Library_Item"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_LoanDaysAllowed">
  <rdfs:domain rdf:resource="#Library_Item"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
 </owl:DatatypeProperty>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_Author">
  <rdfs:domain rdf:resource="#Book"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_Publisher">
  <rdfs:domain rdf:resource="#Book"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_Volume">
  <rdfs:domain rdf:resource="#Magazine"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_Issue">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Magazine"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:about="#ip_Artist">
  <rdfs:domain rdf:resource="#Music_CD"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#ip_Content">
 <rdfs:domain rdf:resource="#Music_CD"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#mp_LibraryCardNumber">
 <rdfs:domain rdf:resource="#I_Library_Membership"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#mp_StartDate">
 <rdfs:domain rdf:resource="#I_Library_Membership"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
<owl:FunctionalProperty rdf:about="#mp_MembershipStatus">
 <rdfs:domain rdf:resource="#I_Library_Membership"/>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
 <rdfs:range>
  <owl:DataRange>
   <owl:oneOf rdf:parseType="Resource">
    <rdf:rest rdf:parseType="Resource">
     <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SUSPENDED</rdf:first>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </rdf:rest>
    <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">ACTIVE</rdf:first>
   </owl:oneOf>
  </owl:DataRange>
 </rdfs:range>
</owl:FunctionalProperty>
<owl:DatatypeProperty rdf:about="#mp_dateOut">
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
 <rdfs:domain rdf:resource="#I_ItemLoan"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#mp_dateDue">
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
 <rdfs:domain rdf:resource="#I_ItemLoan"/>
</owl:DatatypeProperty>

<!--Declaring special linking object properties (for linking substantial thing and interaction classes) -->

<owl:ObjectProperty rdf:about="#involvedIn_LibraryMembership">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Library_Member"/>
    <owl:Class rdf:about="#Library"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdfs:range rdf:resource="#I_Library_Membership"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#involvedIn_ItemLoan">
 <rdfs:range rdf:resource="#I_ItemLoan"/>
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
```

```
        <owl:Class rdf:about="#Borrowed_Item"/>
        <owl:Class rdf:about="#Borrower"/>
        <owl:Class rdf:about="#Library"/>
       </owl:unionOf>
     </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#involves_Library">
    <rdfs:range rdf:resource="#Library"/>
    <rdfs:domain>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#I_Library_Membership"/>
        <owl:Class rdf:about="#I_ItemLoan"/>
      </owl:unionOf>
     </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#involves_LibraryMember">
    <rdfs:range rdf:resource="#Library_Member"/>
    <rdfs:domain rdf:resource="#I_Library_Membership"/>
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#involves_Borrower">
    <rdfs:domain rdf:resource="#I_ItemLoan"/>
    <rdfs:range rdf:resource="#Borrower"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="#involves_BorrowedItem">
    <rdfs:range rdf:resource="#Borrowed_Item"/>
    <rdfs:domain rdf:resource="#I_ItemLoan"/>
  </owl:ObjectProperty>

</rdf:RDF>

 <!-- Created with Protege (with OWL Plugin 1.3, Build 225.1)  http://protege.stanford.edu -->
```

APPENDIX E List of Guidelines and Rules

Table 2 below provides a summarized list of modeling guidelines, rules, and corollaries developed in the thesis. The rules are listed in the order they appeared in the thesis and are grouped based on the representation issues they address (with reference to the respective thesis sections). For additional convenience, the table includes for each rule or guideline the number of the page where the rule or guideline is proposed in the thesis.

Table 2: List of proposed modeling guidelines, rules, and corollaries

Guidelines, rules and corollaries	Section/ Page #
Representation of substantial things and classes	***4.1.1***
Guideline 1: *Substantial things (Bunge-things) in a domain should be modeled in OWL ontologies as OWL individuals.*	p.33
Guideline 2: *OWL ontologies intended to model real world domains should clearly distinguish between OWL individuals representing substantial things (in the ontological sense) and OWL individuals representing other concepts (i.e. non-substantial, or conceptual, things).*	p.33
Modeling Rule 1: *In order to distinguish between OWL individuals representing substantial things and OWL individuals used for other purposes, an OWL ontology intended to model a real world domain should include two disjoint upper-level classes:* • Substantial_Thing *class – the extension of this class would consist of all OWL individuals that represent substantial things* • Non_Substantial_Thing *class – the extension of this class would consist of all OWL individuals that are used to represent anything other than substantial things.*	p.34
Corollary 1: *Substantial things should be modeled as OWL individuals that are instances of the class* Substantial_Thing *or its subclasses; OWL individuals used for other purposes should be made instances of the* Non_Substantial_Thing *class or its subclasses.*	p.34
Corollary 2: *Any OWL class, all instances of which are intended to represent substantial things, should be made a subclass of the* Substantial_Thing *class. OWL classes used for other purposes should be made subclasses of the* Non_Substantial_Thing *class*	p.34
Corollary 3: *No OWL individual in an ontology can represent both a substantial thing and non-substantial thing at the same time*	p.34
Corollary 4: *No OWL class can (other than built-in top class* owl:Thing*) can include both OWL individuals representing substantial things and OWL individuals representing non-substantial things*	p.34
Corollary 5: *Other OWL constructs (such as OWL properties) should not be used to represent substantial things*	p.35

Guidelines, rules and corollaries	Section/ Page #
Representation of properties – general recommendations	***4.2.2***
Guideline 3: • *In OWL ontologies modeling real world domains, ontological properties in general should be modeled as OWL properties, and ontological properties in particular should be modeled as property values of those OWL properties that represent the corresponding properties in general;* • *Depending on a property type (e.g. intrinsic or mutual) and model usage and reasoning requirements, ontological properties in particular (property values) may be represented either as XML datatype values or as special OWL classes and their individuals;* • *If ontological properties in particular (property values) are modeled using OWL classes/ individuals, then such classes and individuals should be clearly distinguished from OWL classes and individuals that represent substantial things.*	p.42
Corollary 6: *OWL classes (individuals) representing property values should be subclasses (instances) of the upper-level class* Non_Substantial_Thing	p.42
Guideline 4: *OWL ontologies modeling real world domains should distinguish among OWL properties that are used to represent the following groups of properties:* • *Ontological intrinsic properties of substantial things* • *Ontological mutual properties of substantial things* • *Other OWL properties, i.e. properties that are not intended to represent substantial properties but are used for other purposes in the ontology*	p.43
Modeling Rule 2: *If an OWL property is intended to represent an ontological (substantial) property, then the domain of such property should be either the* Substantial_Thing *class or its subclasses.*	p.44
Representation of intrinsic properties	***4.2.3***
Modeling Rule 3: *If an OWL class (and its instances) is used to represent a collection of property values for some OWL property representing an intrinsic generic property of substantial things, then* • *This intrinsic generic property should be represented as an OWL object property (rather than an OWL datatype property)* • *The domain of this property should be the class* Substantial_Thing *or some of its subclasses* • *The range of this property should be defined as the OWL class that is used to represent property value collection* • *The OWL class representing the collection of property values should be declared a subclass of the* Non_Substantial_Thing *upper-level class (to distinguish it from substantial thing classes)*	p.51
Modeling Rule 4: *If OWL classes and individuals are used in an OWL ontology modeling a real world domain, then a special upper-level class* Property_Value *should be included in the ontology as follows:* • *This upper level class* Property_Value *should be declared a subclass of the*	p.52

Guidelines, rules and corollaries	Section/ Page #
upper-level class Non_Substantial_Thing • *Any OWL class used to represent a collection of property values for some ontological property should be made a subclass of the upper level class* Property_Value *(and thus also a subclass of the upper-level class* Non_Substantial_Thing*)*	
Representation of bundles of mutual properties based on interactions	***4.2.4.4***
Guideline 5: *In OWL ontologies modeling real world domains, a set of mutual properties of substantial things arising out of the same interaction should be represented as OWL properties associated with the specially defined OWL class – an interaction class*	p.59
Guideline 6: *Each interaction class represents a set of related concurrent mutual properties (usually arising out of the same interaction). Different interaction classes should be used if sets of properties are not concurrent and/or pertain to different interactions.*	p.59
Modeling Rule 5: *Interaction classes should be modeled as subclasses of the upper level class* Non_Substantial_Thing *(since they do not represent substantial ontological things)*	p.59
Modeling Rule 6: *To further distinguish interaction classes from other types of classes in OWL ontologies, additional methods can be employed:* • *A special upper-level class,* Substantial_Thing_Interaction, *can be created as a subclass of the upper-level class* Non_Substantial_Thing. *All interaction classes then would be modeled as subclasses of this class* Substantial_Thing_Interaction *(which would also automatically make them subclasses of the* Non_Substantial_Thing class); • *Naming conventions can be used in naming interaction classes and instances for easier identification (e.g. a prefix I_ or R_ (which stands for 'interaction' or 'relation')*	p.59
Modeling Rule 7: *Each individual mutual property in a bundle of concurrent properties (represented by some interaction class) should be modeled as an OWL property in accordance with the following rules:* • *The domain of each property should be the interaction class representing the bundle* • *Use of a prefix (e.g.* mp_*) is recommended in mutual property name to distinguish it from other types of properties (to conform to Guideline 4)*	p.60
Modeling Rule 8: *A special OWL object property should be defined to link OWL classes (and their instances) that represent substantial things sharing a set of mutual properties to the interaction class that represents this set of shared mutual properties:* • *This OWL object property represents the ontological mutual property of having the relationship of interest (or participating in the respective interaction that gives rise to that set of mutual properties);* • *The domain of this OWL object property should be defined as a union of the*	p.60

Guidelines, rules and corollaries	Section/ Page #
classes that represent interacting substantial things related by the respective interaction class; the range of this object property should be the respective interaction class; • *Use of naming conventions (e.g. a prefix* 'InvolvedIn_' *combined with the interaction class name) is recommended for such object property to explicitly show that the substantial things possessing it are involved in a particular interaction (to which this object property links them).*	
Modeling Rule 9: *For each class of substantial things involved in an interaction, a special OWL object property should be defined to link the OWL interaction class representing the shared set of mutual properties back to the OWL classes (and their instances) representing the involved substantial things sharing this set of mutual properties:* • *The domain of this object property should be the respective interaction class; the range of this property should be defined as a union of the classes that represent interacting substantial things that participate in this interaction;* • *Use of naming conventions (e.g. a prefix* 'Involves_' *combined with the respective substantial thing class name) is recommended for such object properties to show explicitly that this property links the interaction class to a specific class of things involved in the interaction.*	p.61
Representation of mutual properties as OWL object properties directly linking two substantial things	**4.2.4.5**
Modeling Rule 10: *In OWL ontologies modeling real world domains, if an OWL object property is used to represent the existence of a relationship between* ***two*** *substantial things (i.e. a mutual property of having this relationship), then* • *Two mutually inverse OWL object properties should be defined to link pairs of individuals - instances of classes A and B respectively, where A and B are classes of instances representing substantial things having a relationship* • *One of these two properties should have the class A as its domain and the class B as its range, while the other property should have the class B as its domain and the class A as its range* • *Use of naming conventions (such as a prefix 'mp_') is recommended for both these object properties to indicate that they represent a mutual property (existence of the relationship) shared by the two things*	p.65
Modeling Rule 11: *A non-binding mutual property shared by two substantial things can be represented using an OWL object property to link the two OWL individuals representing those two things.*	p.67

Guidelines, rules and corollaries	Section/ Page #
Representation of ontological classes/kinds based on properties	***4.3***
Guideline 7: *Ontological (Bunge's) classes, kinds, or natural kinds as sets of substantial things correspond to OWL class extensions (i.e. sets of OWL individuals representing substantial things).*	p.75
Modeling Rule 12: *If the class extension of some OWL class is intended to represent an ontological class, kind, or natural kind, then such class should be declared a subclass of the upper-level class* Substantial_Thing.	p.75
Modeling Rule 13: *In OWL ontologies representing real world domains, if an OWL class is intended to model an ontological class or kind, then it should not be defined only by class name or only by direct enumeration of instances (i.e. without representing any information about common properties of class instances)*	p.76
Corollary 7: *In OWL ontologies representing real world domains, subclasses of the class* Substantial_Thing *should not be defined simply by class name or by enumeration of instances (i.e. without representing any information about common properties of class instances)*	p.77
Modeling Rule 14: *An ontological class or kind C, modeled by a functional schema with the state functions modeling some common properties P_1, ..., Pn of this class/kind C, can be represented in OWL as the class extension of an OWL class defined as follows:* • *A named OWL class* (e.g. ClassC) *should be created;* • *Each of the properties P_1, ..., Pn should be modeled by a suitable OWL property (in accordance with Guidelines 3-7 and Modeling Rules 2-11 on property representation);* • *Class axiom(s) for the ClassC should be included that state (or imply) that all instances of the ClassC necessarily possess each property Pi;* • *To achieve that, such axioms should state that the classC is a subclass of the anonymous class defined by suitable property restriction on the property Pi, for each Pi. Or, alternatively, the ClassC can be declared to be a subclass of the intersection of the anonymous classes defined by suitable property restrictions for each of the properties Pi.*	p.81
Modeling Rule 15: *If a set of ontological properties P_1, ..., P_k is a subset of common properties of a class or kind C that is sufficient to classify a thing as an instance of the class C (i.e. P_1, ..., P_k are class identifying properties), then this information can be represented in OWL in the following way:* • *A class axiom for the OWL class representing the class C should be defined to represent the fact that possessing properties P_1, ..., P_k is a necessary and sufficient condition for individuals to be members of the class C* • *This class axiom should state that the class C is equivalent to the intersection of the anonymous classes defined by suitable property restrictions for each of the properties P_1, ..., P_k (where each property restriction should imply the possession of the respective property Pi by all the instances of the class C).*	p.82

Guidelines, rules and corollaries	Section/ Page #
Modeling Rule 16: *Every OWL class representing an ontological class or kind (and thus, modeled as a subclass of the class* Substantial_Thing*) should have or imply non-empty class extension.*	p.83
Modeling Rule 17: *If an OWL class represents an ontological class or kind (defined by a set of common properties) then the property restrictions used in the class description for the respective OWL properties (which model common class properties) should not imply optional possession of a property (e.g. a zero cardinality constraint or zero minimum cardinality constraint). Instead, subclassification with property restrictions implying 'mandatory' possession of properties by all instances of the subclass is preferable.*	p.84
Modeling Rule 18: *In OWL ontologies modeling real world domains, if classes A and B represent some ontological classes (i.e. modeled as subclasses of the* Substantial_Thing *class), and B is a subclass of A, then the class definition of the subclass B should reflect the semantic difference (in terms of properties) between the superclass A and its subclass B.* *This distinction can be represented in one of the following ways:* • *by including in the definition of the subclass B one or more additional property restrictions for properties that are acquired by the instances of the subclass B compared to the instances of the superclass A, or* • *by including in the definition of the subclass B one or more property restrictions constraining some properties of the superclass A for the instances of the subclass B, or* • *by including in the definition of the subclass B one or more property restrictions for subproperties of some properties of class A*	p.88
Relationships governing classes, individuals and properties (integrity rules and guidelines)	***4.4.1***
Guideline 8: *Every OWL individual representing a substantial individual (real world instance) should possess at least one substantial property. Possession of a property can be represented by associating this individual with a property either at the instance level or at the class level (via class membership).*	p.94
Guideline 9: *Every OWL property modeling an ontological substantial property should be possessed by at least one OWL individual representing a substantial thing. This can be represented by associating the property with at least one individual either at individual level or at class level (through using a suitable property restriction in class definition).*	p.94
Guideline 10: *Every OWL class representing an ontological class or kind (i.e. a subclass of the class* Substantial_Thing*) should have at least one property. That is, the class definition should include a class axiom that states a necessary condition for this class in terms of a suitable property restriction for at least one OWL property representing an ontological property shared by all instances of the class.*	p.95

Guidelines, rules and corollaries	Section/ Page #
Representation of composition relationships	***4.5***
Modeling Rule 19: *To represent composition relationship between substantial things in OWL ontologies of real world domains, two mutually inverse object properties should be defined*: isComposedOf and isComponentOf. *These properties would link OWL individuals representing composite things to their component things, and OWL individuals representing component things back to their composites, respectively.*	p.97
Modelng Rule 20: *To model explicitly that substantial things are composites or components of other things, two upper level classes can be created in OWL ontologies:* Composite_Thing and Component_Thing: • *Both classes should be modeled as subclasses of the* Substantial_Thing *upper level class*; • *The class* Composite_Thing *can be defined (using class axioms with a cardinality or* an owl:SomeValuesFrom *property restriction) as the class of all OWL individuals that are instances of the class* Substantial_Thing *and possess the property* isComposedOf • The class Component_Thing *can be defined (using class axioms with a cardinality or* an owl:SomeValuesFrom *property restriction) as the class of all OWL individuals that are instances of the class* Substantial_Thing *and possess the property* isComponentOf	p.99
Corollary 8: *OWL individuals representing substantial things that are components of some composite thing should be declared or inferred to be instances of the* Component_Thing *class. OWL individuals representing composite substantial things should be declared or inferred to be instances of the* Composite_Thing *class.*	p.100
Corollary 9: *Any OWL class such that all instances of that class represent substantial composite things should be declared or inferred to be a subclass of the* Composite_Thing *class. Any OWL class all instances of which represent substantial component things should be declared or inferred to be a subclass of the* Component_Thing *class.*	p.100
Modeling Rule 21: *Every OWL individual representing a composite thing should be associated (at the instance or at the class level) with at least one OWL property representing an intrinsic or mutual ontological property that is an emergent property of the composite thing.*	p.100
Modeling Rule 22: • *OWL properties that model hereditary properties of a composite thing can be associated both with OWL individuals (or classes, at the class level) representing the respective component thing(s) and with OWL individuals (or classes) modeling the composites* • *OWL properties that model emergent properties of a composite thing should be associated with OWL individuals (or classes, at the class level) representing the composite but not with any individuals (classes) representing components of this composite*	p.100