An Ontology Model supporting Multiple Ontologies for Knowledge sharing

Thesis submitted in accordance with
the requirements of the University of Liverpool
for the degree of Doctor of Philosophy by

Valentina A.M. Tamma

October 2001

Dedicated to my family

Abstract

Sharing and reusing knowledge is that research area in Artificial Intelligence (AI) that aims to isolate the knowledge components which are shared across different domains and represent them in a way general enough, so that they may be reused in different applications. Knowledge to be shared and reused is of two types: *Ontologies*, knowledge about "what", and *Problem-solving methods*, knowledge about "how".

In this thesis we focus our attention on sharing ontologies, and we concentrate on a conceptual metamodel for ontologies which supports an alternative approach to knowledge sharing. These are the two main research threads followed in thesis: The primary thread concentrates on an enriched ontology model which provides a precise characterisation of the attributes used to define concepts in the ontology. This conceptual metamodel is based on a multidisciplinary theoretical background which includes the formal notions of ontological analysis (namely *identity, rigidity, unity* and *dependence*) [Welty and Guarino 2001], on the cognitive notions of prototypes and exceptions, and the notion of *modality*. The characterisation of attributes we propose has been modelled by a set of metaproperties for attributes which encompass the behaviour of concept properties in the concept definition and over time, namely: *Mutability, Mutability Frequency, Reversible Mutability, Event Mutability, Modality, Prototypicality, Exceptionality, Inheritance, Distinction*. The novelty of this extended conceptual metamodel is that it explicitly represents the behaviour of attributes over time by describing the changes in a property that are permitted for members of the concept. It also explicitly represents how concept properties are inherited by subconcepts. Finally, the metamodel does not only describe the prototypical properties holding for a concept but also the exceptional ones. No previous work on ontology has provided such a precise characterisation of attribute properties.

This conceptual metamodel has been developed to enable the assessment of semantic similarity in the structure of multiple ontologies, called ontology clustering, which is the object of the secondary research thread. Ontology clustering locates the shared knowledge in a structure of multiple ontologies which are hierarchically organised. Although this has not been a primary direction, we believe that such a structure has advantages over the others especially if considered in the context of an open environment such as the Internet.

Contents

List of Figures

List of Tables

Acknowledgments

There are a number of people to whom I wish to express my gratitude for their help during the period of my PhD and in particular, in the last period while I was writing this thesis.

Firstly I wish to thank each and everyone of those who have supervised this thesis. I would like to thank Pepijn Visser for teaching me that doing research requires a bit of method, Michael Shave and Ian Finch for their constant support and for their faith in what I could do. But I am particularly grateful to Trevor Bench-Capon, for undertaking the difficult job of supervising the final years of my PhD. I would like to thank him for introducing me to the fascinating problems of philosophy and for many stimulating discussions, but most of all for being an excellent supervisor.

Many thanks to Ray Paton for his invaluable help in providing the topic of the example and the medical data used in filling in the model in Appendix B.

This PhD has been funded by BT plc and I would like to thank in particular Zhan Cui, Paul O'Brien and Dean Jones.

I would also wish to thank Ian Dickinson and Hp Labs for their support during the period of writing up.

I would like to thank Asunción Gómez Pérez for her many comments on this thesis and for her encouragement and advice. She gave me the pleasure to enjoy a thorough but thought-provoking viva. My gratitude goes also to Floriana Esposito and Donato Malerba, for making me believe I could do research and for their support during all these years. I wish also to thank Simon Parsons and Mike Wooldridge for their constant encouragement.

A particular thank you goes to my mother Luisa, my sister Bernadette and her husband Stefano, for dealing with me when I am unbearable, for making it easier for me to take difficult decisions, and for making me feel part of the family even if I am miles away. Thanks to my father Michele for being with me always and everywhere, but especially thanks to Bernadette for being a sister, a friend and for not permitting me to be a drama queen.

A special thank you to all those great friends who have ‘endured me’ (and my phone calls) in all these years abroad: Monia Pinzaglia, for going beyond the duty of friendship and being there every time I need her, Tommaso Giannelli, for not taking me seriously, Massimo Lazazzera, for proving that I can actually give wise advice. Thanks to Francesca Tesse, Antonella Natali, Mario Laporta and Rita Marinaro for making discussions on computer science sheer fun. A special thank you to Floriana Grasso, for being a colleague, a friend and a sister in times of need, for all the endless discussions on work and life. Thanks to Michelle Usher, for her friendship which has been fundamental in my getting used to life in this country, to Duncan Neary and Graham Goulbourne, who have made the years as PhD student real fun. And finally, thanks to all those friends and relatives whose encouragement has never failed me: Marie Devlin, Michele Zito, Patricia Chiantera, Daniela Giannuzzi, Alejandra and Gianni Margutti, Pepijn Koopman, Agnes and Pepijn Visser, and Antonio Giammaria. And last but not least a special thank you to Sara Tamma.

Chapter 1

Introduction

1.1 Knowledge sharing and reuse

This thesis investigates sharing knowledge between heterogeneous knowledge sources in an open environment, where knowledge sources can join and leave at any time, in a flexible, scalable and maintainable way.

The last two decades have seen a rapid evolution of computing and communication technologies which have dramatically changed the way in which knowledge is perceived in everyday life. In computer science a new perspective, the *knowledge level* [Newell 1982] has shifted the focus of attention from *data* to *knowledge*, which has led to applications for the development and specification of knowledge based systems, and libraries of reusable components.

Technological advances in computing have made it possible to exploit the knowledge level perspective, by providing the technology to build very large intelligent systems that can use knowledge to answer complex queries. In communication technologies the advances in networks of computers and especially the Internet, have made available a large number of resources from which knowledge can be extracted and that are typically heterogeneous.

The process of building powerful intelligent systems relies on the ability of capturing and representing knowledge, also known collectively as *knowledge acquisition*. The acquisition of knowledge is complicated by the availability of many knowledge sources and by their heterogeneity; it is thus necessary to find a way to combine the knowledge provided by these sources in order to have a richer understanding of a domain. Acquiring knowledge has proved to be extremely complex and time consuming. There are some notions that are common across domains, even when these have different nature. Ideally, we would like to be able to reuse these components, but reusability is not easy to achieve, at least for the following reasons:

- the lack of standard representations, at symbolic level, that prevents the sharing of knowledge components among different software developers;
- the difficulty of locating and identifying knowledge components that are reusable from those that cannot be reused;
- the lack of suitable metrics that permit the comparison of knowledge components in order to choose the most appropriate one for a specific application;
- the lack of methodologies and tools that can support the integration of knowledge components in systems different from the ones they have been designed for;
- the lack of evaluation of most of the knowledge components that are built.

Sharing and reusing knowledge are strictly connected activities. In fact, in order to explain how a component can be reused one has often to communicate subtle issues that are more easily expressed in a formal way; these explanations (more or less formally expressed) need a shared understanding of the intended interpretations of terms.
However, sharing and reusing knowledge can be hampered by the following problems [Gómez-Pérez 1998]:

1. Heterogeneity problems;

   (a) *Heterogeneity of knowledge representation formalisms*;

   (b) *Heterogeneity of the implementation languages*;
   (c) *Lexical problems*;
   (d) *Synonymity*;

2. Background assumptions problems;
   (a) *Hidden assumptions*;
   (b) *Loss of common sense knowledge*.

These problems are discussed more in detail in Section 1.2.
The difficulties arising from sharing and reusing knowledge have been summarised quite effectively by Tom Gruber in [Gruber 1991, page 91]:

> today's knowledge systems are isolated monoliths characterised by *high internal coupling* ... and a *lack of external coupling interfaces* ...

Sharing and reusing knowledge is that research area in Artificial Intelligence (AI) that aims to isolate knowledge components which are sharable across different domains and represent them in a way general enough, so that they can be reused in different applications. Knowledge to be shared and reused is of two types: *Ontologies*, knowledge about 'what', and *Problem-solving methods*, knowledge about 'how'. In this thesis we concentrate only on the sharing of ontologies and disregard the sharing of problem-solving methods, although we acknowledge that these two knowledge components are strictly related, and that the nature of the 'knowing how' necessarily affects the way in which the 'knowing what' is represented. This problem is known in the literature as *interaction problem* [Bylander and Chandrasekaran 1988].
In this thesis we focus our attention on sharing ontologies, and we address two main research questions:

1. Is it possible to determine, only by means of a preliminary analysis of the approaches presented in the literature, whether there is an approach which

gives better prospects for achieving interoperability among heterogeneous autonomous knowledge sources?

2. What kind of knowledge should be represented in ontologies in order to facilitate interoperation among heterogeneous, autonomous knowledge sources?

1.2 Problem definition

Knowledge sharing and reuse was the aim of the ARPA Knowledge Sharing Effort [Neches *et al.* 1991] which proposed the following vision [Neches *et al.* 1991, page 37]:

> we present a vision of the future in which the idea of knowledge sharing is commonplace. If this vision is realized, building a new system will rarely involve constructing a new knowledge base from scratch. Instead, the process of building a knowledge-based system will start by assembling reusable components. Portions of existing knowledge bases would be reused in constructing the new system, and special-purpose reasoners embodying problem-solving methods would similarly be brought in. Some effort would go into connecting these pieces, creating a "custom shell" with preloaded knowledge. However, the majority of the system development effort could become focused on creating only the specialized knowledge and reasoners that are new to the specific task of the system under construction. In our vision, the new system could interoperate with existing systems and pose queries to them to perform some of its reasoning. Furthermore, extensions to existing knowledge bases could be added to shared repositories, thereby expanding and enriching them.

In this thesis we specifically address the sharing of *ontologies*, that is the sharing of formal and explicit specifications of the conceptualisations used to model the domains of interest. Therefore sharing ontologies concerns the sharing of a common, formal and explicit view on a domain.

Knowledge to be shared might be modelled and represented in many diverse ways, and so the aim is to share knowledge among heterogeneous resources that are autonomous (and so not designed to interoperate) within a general, flexible architecture. Heterogeneity of the knowledge sources can hinder the process of sharing knowledge; this does not always have to be perceived as a disadvantage but it poses problems that need to be solved in order to achieve interoperability between the knowledge sources. Neches and colleagues [Neches *et al.* 1991] have identified four types of impediment to sharing and reusing knowledge that originate from heterogeneity:

1. Heterogeneous Representations: Knowledge is represented in a formalism that cannot be easily translated into others. This type of diversity is inherent to the application for which the knowledge is represented. Indeed, there are a number of possible ways to represent knowledge and the choice of one representation over the others can affect the system's performance. There is no single knowledge representation that is the best for all problems, nor is there likely to be one.

   In most of the cases sharing and reusing knowledge involves performing a (very often manual) translation from one formalism or language into the others trying to reduce the information loss that is inevitable in these cases.

   Many efforts have been devoted to identifying and overcoming this kind of heterogeneity. For example, in [Corcho and Gómez-Pérez 2000] the authors provide a comparison of the different ontology languages on the grounds of different dimensions such as the way concepts are described, the type of taxonomies, the inference mechanisms supported by a language, etc. This kind of comparison is extremely useful for identifying the most appropriate language for the kind of application that is being designed.

   Many ontology editors such as WebODE [Arpírez *et al.* 2001] and Protege-2000 [Fridman Noy *et al.* 2000] give the possibility to write the ontology in a frame based formalism and translate it into different languages. Furthermore, some systems, such as OntoMorph [Chalupsky 2000], have dealt with the problem of providing syntactic rewriting rules to perform translations from one ontology language into another.

The problem of heterogenous representation is perceived also in more recent application areas such as the Semantic Web. Indeed, Ontoweb, the European network of excellence on ontology-based information exchange for knowledge management and electronic commerce, has a Special Interest Group working on standardisation efforts for ontology languages (more information can be found at the following URL:
http://www.cs.man.ac.uk/ horrocks/OntoWeb/SIG/);

2. Dialects within Language Families: A single family of knowledge representation formalisms can present many different variations both to syntax and semantics. Some of these are trivial whereas others can be quite substantial. However, all of them can hamper knowledge sharing and reuse;

3. Lack of Communication Conventions: Knowledge sharing can be achieved if different systems are enabled to communicate. This implies that all the knowledge sources agree to commit to some standard communication protocol allowing systems to query each others. However, only few standards have been established (at least through consensus) in the past years, among the most relevant we mention here OKBC [Chaudhri *et al.* 1998] which concerns the knowledge content, and KQML [Finin *et al.* 1997], which is the protocol for dealing with queries on the knowledge content;

4. Model Mismatches at the Knowledge Level: Even when the aforementioned problems are resolved there is still the big issue of reconciling models expressing different semantics, which poses huge difficulties. A more detailed analysis of the kind of model mismatches that can occur at the knowledge level can be found in [Visser *et al.* 1998], and in [Grosso *et al.* 1998], and we have analysed them in Section 3.5.

In this thesis we address the problem of achieving knowledge sharing between heterogeneous sources in an open environment, where sources can join and leave freely and no standard is imposed on them. This implies the use of ontologies to model the perspectives on one or more domains which are modelled in the knowledge sources, and reaching an agreement between them, that is creating one or more shared ontologies, to which the single local ontologies choose to commit in order to communicate. This is necessary in order to allow heterogeneous knowledge sources to interoperate while maintaining their autonomy.

We focus our attention on two different aspects of the problem, that is on the way in which the shared ontology (ontologies) should be structured, and the content and the modelling primitives that the shared ontology (ontologies) should present in order to achieve interoperability, while reducing the information loss which is inevitable each time knowledge is translated. In particular, in this thesis we argue the need for a structure of multiple shared ontologies, where shared knowledge is locate in smaller, multiple shared ontologies that are hierarchically organised. In this way, knowledge is modelled at different levels of abstraction, and a knowledge source can join the interoperation at any time, by committing to the shared ontology whose view on the domain is closer to the one of the knowledge source.

Building multiple shared ontologies depends on the ability to match similar concepts, by means of a matching process that takes into account also the concept's description. In order to support the matching process, this thesis provides a more precise description of the concepts in shared ontologies, which permits to better identify the possible cases of heterogeneity. For this reason, the main contribution of this thesis is a set of *meta-properties* which provides a precise characterisation of the attributes in terms of their behaviour over time (including whether the change is allowed or not, whether it happens regularly or once only in the concept's lifetime, and the reversibility of the change), their degree of applicability to subconcepts, their being prototypical or exceptional. The aim of these meta-properties is to provide a more detailed description of the concepts in terms of their characterising features (attributes) and to complement the set of meta-properties proposed by Guarino and Welty [Welty and Guarino 2001] that are used to perform a formal ontological analysis which permits the design of less tangled (and thus more sharable) ontologies.

1.3 Research aim, questions, and method

The aim of this thesis to is provide support the design of multiple shared ontologies that permit interoperability among heterogeneous knowledge sources in a way such that these sources can maintain their autonomy and their heterogeneity is not overcome but only reconciled.
This research aim is effectively summarised by the following two research questions:

1. Is it possible to determine, only by means of a preliminary analysis of the approaches presented in the literature, whether there is an approach that gives better prospects for achieving interoperability among heterogeneous autonomous knowledge sources?

2. What kind of knowledge should be represented in ontologies in order to facilitate interoperation among heterogeneous, autonomous knowledge sources?

Any answer to the research questions must satisfy the following requirements:

1. heterogeneity has to be maintained in the knowledge sources and has to be reconciled only for interoperation purposes, which means that it has to be reconciled only to the extent of permitting communication between those knowledge components which are shared by the sources willing to interoperate;

2. the commitment to the shared ontologies has to be *flexible*, as in an open environment, with knowledge sources that can interoperate freely with other sources not known in advance, only on the ground of some shared knowledge (necessary in order to have a common ground for communication).

In order to find the answers to the research questions that satisfy the requirements above, we have made the following assumptions:

- that we are restricting ourselves to the sharing of concepts and relations only, disregarding the problem of sharing axioms;
- knowledge sources provide sufficient information to permit the understanding of which knowledge components are related and thus sharable;
- ontologies modelling the knowledge sources are correct and possibly untangled, that is, they have been designed according to a lifecycle that includes a validation step;
- a knowledge source can commit only to one shared ontology at a time; different views on a domain are allowed only at different times;
- translations across ontologies might not preserve the semantics.

Two main research directions have arisen. The first investigates the way in which knowledge shared by all the sources is to be modelled in a formal and explicit way by one or more shared ontologies, and the relationships between the sources and the shared ontologies. In order to do so we have investigated a number of systems using ontologies to facilitate knowledge sharing, such as InfoSleuth [Perry *et al.* 1999], OBSERVER [Mena *et al.* 2000], and KRAFT [Preece *et al.* 2001], and we have analysed the approaches that are behind these systems.
Related to this is the question of what information should ontologies provide in order to permit interoperability which conforms to the requirements above, and

whether the ontology models that are currently used are adequate to achieve interoperability as described above in Requirement 1. The process of recognising candidate knowledge components for sharing and reuse depends heavily on semantics, and is quite demanding to perform in that it requires a deep knowledge of the domain. We indicate what kind of information on the knowledge components should be represented in the ontologies in order to facilitate the individuation of suitable candidates.

1.4 Thesis structure

The remainder of this thesis has the following structure: in Chapter 2 we review the theoretical foundations of ontologies. This term is used both in Philosophy, where it indicates the systematic explanation of Existence, and in various areas in Artificial Intelligence (AI), such as knowledge engineering, knowledge representation, qualitative modelling, database design, language engineering, information integration, information retrieval and extraction, knowledge management and organisation, agent-based system design, and, more recently, the semantic web [Guarino 1998]. In all these areas the term takes a different meaning, in some cases it denotes a set of activities performed following a standardised methodology, such as conceptual analysis and domain modelling, while in some other cases it just indicates a warehouse of vocabulary to solve lexical, semantic and synonym problems and assumptions. We review the different meanings that the term ontology takes in AI. We also present an overview of the philosophical notions on which much of the work on ontologies in AI is grounded, in particular we focus our attention on formal ontology and how it provides the tools to perform a formal ontological analysis aimed to build better conceptual models.

Chapter 3 presents an overview of knowledge sharing and reuse and focuses on the roles that ontologies play in this context. We here illustrate the diverse approaches

presented in the literature and we sketch a novel proposal for knowledge sharing, namely *ontology clustering*, which we believe is an alternative to the approaches presented in the literature for achieving interoperability in open environments while reducing the information loss.
Ontology clustering is based on the ability to assess semantic similarity between concepts. We briefly survey the measures for semantic similarity that have been presented in the literature and we argue the need for more sensitive measures, that return not only a binary value, but a degree of similarity between concepts that can be used to perform some kind of semantic matching. These measures should take into account the structure of the concept's description and the relationships holding between concepts. Based on the approach by Rodríguez and Egenhofer [Rodríguez and Egenhofer 2002] we have enriched the traditional ontology model with a set of meta properties to describe attributes, which is the object of the next chapter.

In Chapter 4 we introduce and motivate an extended conceptual model for ontologies which explicitly represents semantic information about concepts' properties. This model is grounded on the meta-properties of formal ontological analysis that we present in Chapter 2 and it results from enriching the usual conceptual model with meta-properties for attributes (that model concepts' properties) which precisely characterises the concept's properties and expected ambiguities, including which properties are prototypical of a concept and which are exceptional, the behaviour of properties over time and the degree of applicability of properties to subconcepts. The explicit treatment of time for attribute descriptions in an ontology is a novel aspect introduced by this thesis. This enriched conceptual model permits a precise characterisation of what is represented by class membership mechanisms and helps a knowledge engineer to determine, in a straightforward manner, the meta-properties holding for a concept.

Chapter 5 presents an example of modelling by using the ontology model introduced in Chapter 4. The domain chosen for the example is that of medicine, and we model a particular condition known as *Disseminated Intravascular Coagulation*. We have chosen this domain as it is extremely complex to model and because some of the properties of this condition are time and event dependent. For this reason, this example is particularly suitable to show the effectiveness of the ontology model developed in this thesis.

Finally, Chapter 6 draws conclusions, highlighting those which are deemed to be the novel contributions to the field of this thesis, and it presents future research directions that emerged from the research on this thesis.

Chapter 2

Theoretical foundations of ontologies

2.1 Introduction

The need to share diverse knowledge and/or information with other applications already built has given rise to a growing interest in research on *ontology*. This term has been originally used in Philosophy where it indicated the systematic explanation of Existence. More recently, the term has been used in various areas in Artificial Intelligence (AI) and more widely in Computer Science, such as knowledge engineering, knowledge representation, qualitative modelling, database design, language engineering, information integration, information retrieval and extraction, knowledge management and organisation, agent-based system design [Guarino 1998] and e-commerce. Ontologies play also a key role in one of the newest areas of interest, the Semantic Web, as confirmed by efforts such as OntoWeb (`http://www.ontoweb.org`) and DAML (`http://www.daml.org`).
In all these areas the term *ontology* can take a different meaning. In some cases this term denotes a set of activities performed following a standardised methodology, such as conceptual analysis and domain modelling, while in some other cases it just indicates a warehouse of vocabulary to solve lexical, semantic and synonym problems and assumptions. It is thus clear that there is no unique definition of the term *ontology*, and so this chapter introduces the different senses of the word ontology in Philosophy and in Artificial Intelligence. The meaning of the term *ontology* changes moving from philosophy to AI. The philosophical account for the term *on-*

tology is the branch of metaphysics that deals with the nature of Being, and it can be considered as a particular system of categories accounting for a certain vision of the world [Guarino 1998]. In the remainder of the thesis the word *ontology* is used as an uncountable noun (which does not have a plural form) when it refers to the philosophical notion of ontology, whereas we will use as a countable noun when referring to the engineering artifacts defined in AI.

This chapter presents an overview of theoretical foundations of ontologies. This topic is too vast to be dealt with in a chapter, so we focus our attention on those aspects which are relevant for the thesis such as the definition of *ontologies* and of the properties which hold for them, while we disregard other important features such as the languages, methodologies, and tools that can be used to build ontologies.

Many articles in the literature have been devoted to presenting different definitions and features of ontologies; for this chapter we have followed the overview by Gómez-Pérez and Benjamins [Gómez-Pérez and Benjamins 1999]. This chapter illustrates the difference between the philosophical and AI notion of ontology and presents the different definitions of ontology in AI (Section 2.2). Before describing the different types of ontologies (Section 2.4) we introduce the modelling primitives that can be used to model them (Section 2.3). Then, in Section 2.5 we present the philosophical notions on which much of the work on ontologies in AI is grounded, in particular focussing our attention on formal ontology and how it provides the tools to perform a formal ontological analysis aimed at building better conceptual models (Section 2.5.1). We end this chapter by drawing conclusions.

2.2 Ontologies: from Philosophy to Artificial Intelligence

The meaning of the term ontology has different connotations in Philosophy and in Computer Science. Guarino gave a characterisation of the philosophical account for the term *ontology* as a particular system of categories accounting for a certain

vision of the world [Guarino 1998]. In this perspective an ontology is independent from the language used to describe it. As we have already mentioned, the word *ontology* takes a different meaning in Artificial Intelligence, where it denotes an *engineering artifact* that is comprised of a specific *vocabulary* and of a set of explicit assumptions concerning the *intended meaning* of the words composing the vocabulary. Since the focus of this definition of ontology is the *vocabulary* which is used to describe a specific reality, it is clear that the Artificial Intelligence notion of *ontology* is language dependent as opposed to the philosophical one.
Although the AI community seems keen to agree on the use and on the meaning of the term "ontology", there is no a formal definition that is fully accepted and agreed upon by the community.

2.2.1 Gruber's definition of ontology

The most widely quoted definition of "ontology" was given by Tom Gruber in 1993, who defines an ontology as [Gruber 1993, page 199]:

> an explicit specification of a conceptualisation.

Gruber's definition builds on the idea that the declarative formalisation of the domain knowledge starts from the *conceptualisation* of the domain [Genesereth and Nilsson 1987], that is the identification of the objects that are hypothesised to exist in the world and the relationships between them. We use here the word *object* in its broadest meaning, so that it can denote both abstract and concrete things of the world. According to Genesereth and Nilsson a conceptualisation is [Genesereth and Nilsson 1987, page 12]:

> a triple consisting of universe of discourse, a functional basis set for the universe of discourse, and a relational basis set.

The universe of discourse is the set of objects on which the knowledge is expressed. A classic example is of conceptualisation is the Blocks World domain, where objects of the domain are five toy-blocks on a table, namely *a, b, c, d, e*. The functional basis set groups a type of basic interrelationships among objects of the universe of discourse, for example in the Blocks World case, it would make sense to include in the conceptualisation a function `hat` mapping a block into another if the second block is on top of the first one. A relational basis set is a set of a second kind of interrelationships holding among objects of the universe. We will denote a conceptualisation as $< D, \mathcal{F}, \mathcal{R} >$, where D represents the domain, that is the universe of discourse, $\mathcal{F}$ is the set of functional basis and where $\mathcal{R}$ is the relational basis set. For some purposes it might not be important to distinguish between the functional and the relational basis set, in these cases we will denote a conceptualisation as a simpler structure $< D$, **R**> where **R** is the set of all the interrelationships defined on the objects composing the universe of discourse. For example, in the Blocks World it would make sense to consider the relationship `above`, relating two blocks if one is anyway `above` the other, or the relationship `on`, holding if one block is immediately on top of another, etc. Therefore, according to Genesereth and Nilsson, a conceptualisation of the Blocks World domain can be the triple: $\{\{a, b, c, d, e\}$, $\{$`hat`$\}$,$\{$`on`, `above`, `clear`, `table`$\}\}$. In this way the conceptualisation of a domain is a set of ontological descriptions $\{C_1, C_2, \cdots, C_n\}$ where each C_i is an entity of the domain, a function or a relationship concerning one or the entities, that is $\forall\, C_i,\ i : 1, \cdots,\ n,\ C_i \in D \vee C_i \in \mathcal{F}, \vee\, C_i \in \mathcal{R}$. The explication of each symbol C_i by assigning it a meaning corresponds to describing the domain according to a particular viewpoint and this viewpoint is the ontology.

According to Gruber an ontology is a quintuple composed of *classes*, *instances*, *functions*, *relationships*, *axioms*. *Classes* correspond to entities of the domain, *instances* are the actual objects which are in the domain, *functions* and *relationships* relate entities of the domain, and finally *axioms* constrain the meaning and the use of classes and instances, functions and relationships. We have already mentioned that Gruber's definition of ontology is based on Genesereth and Nilsson's definition of conceptualisation [Genesereth and Nilsson 1987], except that Gruber refers to classes, instances and axioms used to constrain classes or instances instead of the universe of discourse.

Despite the fact that Gruber's is one of the most used definitions of ontologies it has been argued that it poses some problems. One of the possible criticisms to the use of Genesereth and Nilsson conceptualisation as cornerstone for Gruber's definition of ontology is that this sense of conceptualisation is based on the mathematical definitions of functions and relationships that are inherently extensional [Guarino and Giaretta 1995]. Thus Gruber's ontology uses something extensional to reason about the intensional meaning of the vocabulary used to describe a domain. A result of this problem is that what should be described as a different *situation*, as a kind of snapshot, of a conceptualisation, is called a *conceptualisation*. For example, let us consider the previous conceptualisation of the Blocks World: $\{\{a, b, c, d, e\}$, `hat`, $\{$`on`, `above`,`clear`, `table`$\}\}$. Genesereth and Nilsson state that the functions and relations in their definition of conceptualisation are extensional entities. Thus, in the example above, the relation `above` is just equal to the set of pairs $\{(b, c), (a, c), (a, b), (d, e)\}$. If a different arrangement in the blocks is considered, then, according to Genesereth and Nilsson we are looking at a new conceptualisation, whereas Guarino and Giaretta [Guarino and Giaretta 1995] claim it is just a new situation, a new state of affairs in the conceptualisation and not a new conceptualisation altogether. An ontology should concentrate on the *meaning* which is associated to the extensional relations in a way which is *independent from the particular state of affairs*.

It is important to note at this point that there is a distinction between a different situation in the domain to conceptualise as opposed to a different viewpoint on the domain to conceptualise. In fact there is no unique conceptualisation of a domain, but the same domain can be conceptualised differently according to a number of viewpoints from which we can consider the domain. When a conceptualisation is explicitly specified, it expresses a viewpoint on the knowledge of a domain, or an ontology. Therefore, there might be an ontology for representing the content of each different source of knowledge.

2.2.2 Guarino's definition of ontology

Alternative definitions of conceptualisation are given by Guarino in [Guarino *et al.* 1994], [Guarino 1998]. As mentioned above, Guarino's view of conceptualisation focuses on the intended meaning of the relations linking objects of the domain. He renames this type of relation (which are independent on the situation of the domain) *intensional* or *conceptual relations,* and expresses them as functions from possible worlds (in the Kripke's semantics acceptation [Kripke 1980]).
Intensional relations are defined on a *domain space*, which is the pair comprised of the domain D and W the set of the *relevant* states of affairs. Each state of affairs represents a *possible world.* A *conceptual relation* of arity n is a total function $\rho^n : W \rightarrow 2^{D^n}$, that is it associates a possible state of affairs with a n-ary (extensional) relation on the domain. Each conceptual function ρ has a number of *admissible extensions,* which are defined by the set $E_\rho = \{\rho(w)|w \in W\}$. Based on these preliminary definitions, he defines a conceptualisation as the set of conceptual relations defined on the domain space, that is a triple $< D, W, \Re >$ where $\Re$ is the set of the conceptual relations defined on $< D, W >$. Each conceptualisation corresponds to many states of affair of the kind described by Genesereth and Nilsson, that is many structures $< D$, **R**$>$ that are thus called *world structures*, and in particular a conceptualisation should have one structure for each world, which are the *intended world structures* according to the conceptualisation which is being described.
A conceptualisation, as defined above is basically an implicit process which just identifies the objects of the world and the interrelationships linking them, but it provides no means to denote them. Let us consider a logical language L, with a vocabulary V. An *interpretation* function I associates an object or a relationship of the domain with a token of the vocabulary, that is, $I : V \mapsto D \cup$ **R**. It is interesting

to note here that this definition provides only an *extensional* interpretation of the objects of the domain and the relationships linking them, in that its starting point is only a world structure. In order to have an *intensional* interpretation we have to consider an *intensional interpretation function* $\mathcal{I}$, which takes as a starting point a conceptualisation, and thus the structure $< D,\ W,\ \Re >$. $\mathcal{I}$ associates a symbol of the vocabulary V with an element of $D \cup \Re$. The pair formed by the conceptualisation and the intensional interpretation is the *ontological commitment*. Then, the role that an ontology plays according to Guarino, is provided by the following definition [Guarino 1998, page 4]:

> An ontology is a set of logical axioms designed to account for the intended meaning of a vocabulary.

From this definition arises the relationship between a *conceptualisation* and the *ontology* which specifies and formalises it. Both specification and formalisation are based on the choice of a language L (which is provided with a vocabulary V); L is associated with an ontological commitment (providing the intensional interpretation of the symbols of the vocabulary V), an ontology for L is comprised of a set of axioms such that the models derived from these axioms are the best approximation of the set of the possible interpretations of L given K (called *set of intended models* of L according to K).
The set of intended models is, according to Guarino [Guarino 1998], only a weak characterisation of a conceptualisation, one that excludes some absurd interpretations, without describing the real meaning of the vocabulary. This means that the *specification* of the conceptualisation by an ontology, as in Gruber's definition, can be obtained only in an *indirect way*. There are two main justification for this:

1. an ontology is only approximating a set of intended models,

2. the set of intended models is only a weak characterisation of a conceptualisation, because it is impossible to reconstruct the ontological commitment

of a language from a set of its intended models. In fact, an intended model can correspond to many states of affairs, with no one to one correspondence, and this implies that it is impossible, given a conceptualisation, to reconstruct the correspondence between the possible worlds and the extensional relations that are given by the conceptualisation. This is a problem which is common to all the efforts for representing in some symbolic way [Davis *et al.* 1993] what is at the *knowledge level* [Newell 1982].

Gruber's definition is thus extended by taking into account not only the conceptualisation but also the language used to describe it and the set of commitments associated with it. In this sense an ontology is language-dependent while a conceptualisation is language independent, which constitutes the main difference between ontologies in AI and ontology in Philosophy.
With the additional notions given above Gruber's definition can be reformulated as [Guarino 1998, page 5]:

> An ontology is the set of logical axioms designed to account for the intended meaning of a vocabulary, i.e. its *ontological commitment* to a particular *conceptualisation* of the world.

2.2.3 Other definitions

Gruber's and Guarino's are not the only definitions of ontologies presented in the literature, although they are the most used. In fact, each research group working in the ontological field has tried to clarify their view on ontologies and have thus ended up developing their own definition of ontology. Clearly, these definitions depend on the purposes for which they have been developed, but, despite some minor differences, all the definitions refer to the ontology as a common understanding of a domain, and that implies that it is a repository of vocabulary for the knowledge of a domain. The vocabulary contains both formal and informal definitions. Some of these definitions are not definitions in the proper sense of the term, but describe the content of an ontology. The following are of this type. Borst has extended Gruber's definition [Borst 1997, page12]:

> An ontology is a formal specification of a shared conceptualisation.

Studer and colleagues have merged Gruber's and Borst's definition, and have provided an explanation for the terms used [Studer *et al.* 1998, page 185]:

> An ontology is a formal, explicit specification of a shared conceptualisation. A 'conceptualisation' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. ... 'Formal' refers to the fact that the ontology should be machine-readable. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is it is not private to some individual, but accepted by a group.

Neches and colleagues developed the following definition about what is defined by an ontology [Neches *et al.* 1991, page 40]:

> An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary.

This definition states clearly that an ontology is not only describing the explicit knowledge about a domain, but also the knowledge that can be inferred.
The relationship between ontologies and knowledge bases has been included in the definition both by Bernaras and colleagues [Bernaras *et al.* 1996], and by Swartout and colleagues [Swartout *et al.* 1996]. Bernaras and colleagues [Bernaras *et al.* 1996, page 298] have defined what a knowledge base provides while designing an ontology:

> It [the ontology] provides the means for describing explicitly the conceptualisation behind the knowledge represented in a knowledge base.

Swartout and colleagues [Swartout *et al.* 1996], on the other hand, have defined the contribution given by an ontology to the design of a knowledge base [Swartout *et al.* 1996, page 1]:

> An ontology is a hierarchically structured set of terms describing a domain that can be used as a skeletal foundation for a knowledge base.

Finally, ontologies can be classified into *lightweight* and *heavyweight* ontologies, depending on the degree of formality used to express them. Heavyweight ontologies are those which are provided with axioms, inference mechanisms aimed to equip ontologies with deductive power (e.g., inheritance), and that are characterised by a high degree of formality (e.g., underlying formal semantics). Lightweight ontologies, on the other hand, are those ontologies that define a vocabulary of terms with some specification of their meaning [Uschold 1998].

Each of the definitions presented above highlights a specific aspect of a role played by ontologies. All definitions, however, share the idea that an ontology provides a description of a particular viewpoint about a domain and that such a description must be explicit, in that it states a vocabulary for the domain, which is expressed by a certain degree of formality, and that a group commits to use the vocabulary according to the intended meaning associated with it in order to communicate. Nevertheless, this agreement is only superficial, because the word *ontology* can be interpreted in different ways across the definitions. Some of the possible interpretations have been analysed in [Guarino and Giaretta 1995], where the authors identified seven most common interpretations, which are illustrated in Table 2.1. The definitions presented in Table 2.1 are substantially different in that some of them are defined at the semantic level (definitions 2 and 3) whereas others refer to the ontology as a concrete artifact at syntactic level, to be used for a given purpose [Guarino and Giaretta 1995]. Definition number 1 refers to the philosophical discipline of ontology and therefore differs in nature from the others and it is discussed in Section 2.5.

These definitions form a sort of continuum where at one end the main focus is on the

1. Ontology as a philosophical discipline;
2. Ontology as an informal conceptual system;
3. Ontology as a formal semantic account;
4. Ontology as a specification of a “conceptualisation”;
5. Ontology as a representation of a conceptual system via logical theory
   (a) characterised by specific formal properties;
   (b) characterised only its specific purposes;
6. Ontology as the vocabulary used by a logical theory;
7. Ontology as a (meta-level) specification of a logical theory.

Table 2.1: Interpretations of the word “ontology”

semantics of the knowledge described (and so, these are the definitions that can be referred to by the notion of *conceptualisation* as in [Guarino 1998] and illustrated in Section 2.2.2) while at the other there are more syntactic oriented notions, like logical theories or knowledge bases.

2.3 Modelling primitives

Before proceeding by illustrating the different types of ontologies we introduce here the conceptual primitives that can be used when knowledge about a domain is formalised in an ontology. The aim of this section is to help clarifying the terminology that is used in the remainder of this thesis.

In [Gruber 1993] and after [Genesereth and Nilsson 1987] ontology is defined as the quintuple:

$$(\mathcal{C}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \mathcal{A})$$

where:

- $\mathcal{C}$ is the set of the *concepts*, that is the set of the abstractions used to describe the objects of the world;
- $\mathcal{I}$ is the set of individuals, that is, the actual objects of the world. The individuals are also called *instances* of the concept;
- $\mathcal{R}$ is the set of relationships defined on the set $\mathcal{C}$, that is, each R $\in$ $\mathcal{R}$ is an ordered n-ple R=($C_1 \times C_2 \times \cdots \times C_n$). For example `subconcept-of` is the pair ($C_p,\ C_c$), where C_p is the parent concept and C_c is the child concept;
- $\mathcal{F}$ is the set of functions defined on the set of concepts and that return a concept. That is, each element F $\in$ $\mathcal{F}$ is a function F: ($C_1 \times C_2 \times \cdots \times C_{n-1} \mapsto C_n$). For example, the function `Price-of-flat` is function of the concepts `Year`, `Location` and `Number-of-square-metres`, and

returns a concept `Price`, that is `Price-of-flat`: `Year` $\times$ `Location` $\times$ `Number-of-square-metres` $\mapsto$ `Price`;

- $\mathcal{A}$: set of axioms, that is first order logic predicates that constrain the meaning of concepts, relationships and functions.

Some of these components have necessarily to be in an ontology. For example, few ontologies include also instances. In the ontologies modelled in this thesis we do not consider neither instances nor axioms (which are out of the scope of this research).

The simplest type of ontology is composed by the set of concepts $\mathcal{C}$ and the relationships $\mathcal{R}$ (lightweight ontologies), although this limits the knowledge that can be expressed about the domain. Concepts are also called *classes*. Concepts and instances and are usually hierarchically organised in an *Is-a* hierarchy, which permits, when it is a strict relationship (in the mathematical sense), *inheritance* to be exploited in the structure, that is if A is an ancestor of B (denoted by A $\rightarrow$ B) and B $\rightarrow$ C then, A $\rightarrow$ C. When ontologies include also the content concerning ground individuals and their relationships with the concepts they instantiate, then the notion of inheritance is extended to instances and it is called the *instance relation*. The *Is-a* relationship, also called the *subclass relationship*, is not the only one that can be defined on concepts.

Concepts can be defined in terms of characteristic features describing them, that are called *attributes*. If the concepts are organised in a *Is-a* hierarchy, then the inheritance is extended also to attributes. Attributes are shared by concepts either in their original form or modified in order to give the inheriting class, known also as *subclass*, a more restrictive definition than the one provided by the parent concept. Furthermore other properties can be added to form more specialised concepts.

Anomalies arising from inheritance mechanisms have been illustrated in the literature ([Brachman 1985, Touretzky 1986]), where a distinction is made between *single* and *multiple inheritance*. The former permits a concept to inherit attributes from one parent only and can cause *default conflicts*, while the latter permits a concept to inherit properties from more than one parent and can cause inconsistencies in inherited attribute values. In particular Touretzky has characterised the problems that can arise because of inheritance into a pattern of nonmonotonicity, and distinguished three main categories, which are: the so-called Tweety triangle path, the cautions monotonicity (also called the Clyde-level skip), and the path in case of multiple inheritance of positive and negative reasons (also known as Nixon diamond path) [Touretzky 1986]. In order to solve these problems, Touretzky introduced the notion of inferential distance, which proved suitable to deal with the first two types of path.

In [Carpenter 1993], default values are defined as a way to deduce information about a concept, if the information is consistent with what is already known about the concept. Reasoning about defaults can become extremely problematic when only *strict inheritance* is allowed, that is when the IS-A link amounts to logical implication or set inclusion. Then, more specific information cannot overrule information obtained from more general classes thus causing wrong conclusions to be inferred. A *defeasible* approach [Touretzky 1986] permits the more specific information to overrule the more general information thus resolving the conflict.

In some cases it might be useful to introduce the distinction between concepts and *role*. According to Sowa, [Sowa 2000, page 80]:

> a role characterises an entity by some role it plays in relationship to another entity. The type HumanBeing, for example, is a phenomenal type that depends on the internal form of an entity; but the same entity could be characterised by the role types Mother, Employee, or Pedestrian.

Roles are also defined in Description Logics [Borgida 1996], where they are interpreted as binary relations between objects.

Depending on the expressive power required from the application that needs the ontology we might apply all these conceptual primitives. However, a careful conceptual analysis is needed to understand which primitives to use and how to use them, for example to decide when an object is to be modelled as a concept or as a role.

2.4 Different types of ontologies

The ontologies presented in the literature can be classified according to different dimensions, which range from the level of generality of the concepts they describe, to the type of knowledge they model (be it related to the domain or the task). In this section we present the most commonly used classifications of ontologies. The overview presented in this section is by no means exhaustive, we have chosen those classifications that are relevant to the topic of the thesis and that are referred to in the following chapters.
The first dimension that can be used to classify ontology is the level of generality that is used in the description of a domain. It is possible to distinguish the following types of ontologies [Guarino 1998]:

- Top-level ontologies: this kind of ontology describes very general concepts or common-sense knowledge such as space, time, matter, object, event, action, etc., which are independent of a particular problem or domain.

- Domain ontologies: this kind of ontology describes the vocabulary related to a generic domain such as medicine or physics.

- Task ontologies: this kind of ontology describes the vocabulary related to a generic task or activity such as diagnosis or selling.

- Application ontologies: this kind of ontology describes concepts depending both on a particular domain and on a particular task. They are often a specialisation of both domain and task ontologies and correspond to the roles played by domain entities when they perform certain activities.

Van Heijst and colleagues propose to classify ontologies according to two dimensions, which are the *amount and the type of structure of the conceptualisation* and the *subject of the conceptualisation* [van Heijst *et al.* 1997].

- Amount and type of structure of the conceptualisation: This dimension is mainly concerned with the level of granularity of the conceptualisation and thus can be subdivided into:
  - Terminological Ontologies: these are not strictly speaking ontologies but just lexicons that specify the terminology which is used to represent knowledge in the domain of discourse. They do not represent the semantics of the terms;
  - Information Ontologies: they specify the record structure of databases (for example, database schemata). They provide means to record the basic observations concerning instances of the database, but they do not define the concepts that are instantiated by these instances;
  - Knowledge Modelling Ontologies: they specify conceptualisations of knowledge. They are structurally richer than knowledge modelling ontologies and are often specified according to a particular use of the knowledge they describe;
- Subject of the conceptualisation: This dimension concerns the type of knowledge that is modelled in the ontologies. Four categories are distinguished along this dimension:
  - Application Ontologies: specify those concepts that are necessary in order to model the knowledge required for a specific applications. Usually, application ontologies specialise terms taken from more general ontologies such as the domain and the generic ontologies described below and may extend generic and domain knowledge by representing method and

task-specific components. Application ontologies are not reusable, they reuse knowledge which may be modelled in ontology libraries by tuning it for the specific application at hand;

– Domain Ontologies: specify those concepts that are specific of a particular domain. Knowledge engineers draw a line between *domain ontologies* and *domain knowledge*, where the former has ontological nature and the latter epistemic. Domain knowledge describes factual situations in certain domains whereas domain ontologies specify the constraints to apply on the structure and the content of the domain knowledge. But the distinction between what is ontological and what is epistemological is quite subtle, and therefore such a line often cannot be drawn too neatly. This point is further discussed in Section 4.3.1;

– Generic Ontologies: specify concepts that are generic across many fields. Concepts in the domain ontologies may specialise those in the generic ontologies in order to tune them to a particular domain. Generic ontologies correspond to the top-level ontologies in Guarino's classification presented above; they typically define concepts like state, event, process, action, etc.

– Representation Ontologies: explicate conceptualisations underlying knowledge representation formalisms. They provide a representational framework without making claims about the world, because they are meant to be *neutral* with respect to the world. Domain and generic ontologies are described by means of the primitives given in the representation ontologies. A typical example of representation ontologies is the *Frame Ontology* of Ontolingua [Farquhar *et al.* 1997], which provides the primitives to use for building domain ontologies.

Ontologies differ also in the degree of formality by which the terms and their meaning are expressed in the ontology, as in [Uschold and Gruninger 1996]. Here, the knowledge expressed in the ontology might be the same, but they differ in the way in which it is expressed.

In discussing how formally ontologies are described, it makes little sense to talk about categories, because the degree of formality is better thought of as a continuum rather than a set of classes. Nevertheless, we can set four points in this continuum:

- Highly informal: are those ontologies expressed in natural language. Term definitions might be ambiguous due to the inherent ambiguity of natural language;

- Semi-informal: these ontologies are expressed in a restricted and structured form of natural language. Restricting and structuring natural language achieves improvement in clarity and reduction in ambiguity;

- Semi-formal: these are ontologies expressed in artificial languages which are formally defined, such as Ontolingua [Farquhar *et al.* 1997];

- Rigorously formal: these are ontologies whose terms are precisely defined with formal semantics, theorems and proofs of desired properties such as *soundness* and *completeness*.

Along the same line, McGuinness in [Lassila and McGuinness 2001] "classifies" ontologies on the ground of their expressiveness, that is, on the grounds of the information that the ontology needs to express. In fact, depending on the different types of interpretation which are associated with the word *ontology* (as summarised in Table 2.1), we can distinguish between less or more complex notions of ontologies, which may range from a *controlled vocabulary*, to a *glossary*, to reach, at the other end of this spectrum, ontologies which also provide general logical constraints such as disjointness, inverse, part of, etc. The points they distinguish in the spectrum are:

- Controlled vocabularies: a vocabulary is the simplest possible notion of ontology, that is a finite list of terms. A typical example of this category are catalogues. Indeed, catalogues provide terms with an unambiguous interpretation, but nothing more;

- Glossaries: they are a list of terms and their meanings. The meanings are usually expressed in natural language statements which are chiefly aimed at humans. These statements, however, are ambiguous and cannot be used by computer agents;

- Thesauri: add to glossaries the semantics emerging from the definition of the relations between terms, such as the synonym relationship. Typically, they do not provide an explicit hierarchical structure, although this can often be deduced by broader or narrower term specifications. The relationships defined in a thesaurus can often be interpreted univocally by a computer agent;

- Informal *Is-a* hierarchies: this category includes many of the ontologies on the web. These are ontologies where a general notion of generalisation and specialisation is provided although it is not strict subclass hierarchy. A typical example is *Yahoo!*, which provides a small number of top-level categories, but does not provide an explicit hierarchical structure, and its hierarchy is not a strict subclass of the *Is-a* hierarchy. In hierarchies that are not strict *Is-a* hierarchies it is not always the case that an instance of a more specific class is necessarily an instance of a more general class, therefore inheritance (with or without exceptions) cannot be assumed here;

- Formal *Is-a* hierarchies: these are ontologies where concepts are organised according to a strict subclass hierarchy. Thus, for these ontologies inheritance is always applicable because it is always the case that if a concept C is a superclass of the concept C', then any subclass of C must necessarily be a subclass of C'. These ontologies include may only class names;

- Formal instances: ontologies including formal instance relations are a natural extension of ontologies enforcing a strict hierarchical structure. Indeed, some classification schemes include only class names, as we have pointed out when discussing strict subclass hierarchies. When formal instance relations hold, the ontologies include also the content concerning ground individuals and their relationships with the concepts they instantiate;

- Frames (description of concept properties): these are ontologies whose concepts are described in terms of their characteristic properties. For example, a concept `Book` might be described in terms of features like title, author, publisher, etc. The inclusion of properties in the concept description becomes more interesting when inheritance can be applied to these properties, and thus properties can be specified for a more general concept and be inherited down the hierarchy by more specific concepts;

- Value restriction: these ontologies permit to apply restrictions on the values associated with properties. For example, in describing the concept `Book` we might restrict the value to associate with the property `Author` to be composed of maximum two names. These restrictions are usually to be inherited by the sub-concepts of the concept where they are stated for the first time, which clearly poses a problem when the type of hierarchical relation supported by the ontology is not a strict subclass relation;

- General logical constraints: these ontologies are those with the richest expressiveness. For example, properties might be based on mathematical equations which use values from other properties or properties might be expressed as logical statements. This type of ontology is usually written in very expressive ontology languages, such as Ontolingua [Farquhar *et al.* 1997] which permit the specification of first order logic constraints on concepts and their properties.

Generally speaking, we can recognise a number of properties that should be applicable to ontologies, although there is a difference in the extent to which they apply. In this thesis we take the view of Lassila and McGuinness [Lassila and McGuinness 2001] and we assume that the properties in Table 2.2 hold for an ontology apart from those concerning axioms, since this thesis does not deal with axioms.
To complete the discussion presented in the previous two sections we conclude with a quote from the SRKB (Sharable Re-usable Knowledge Bases) mailing list which is reported in [Uschold and Gruninger 1996]. This quote summarises quite nicely the nature of ontologies, the various ways they can be expressed, and the context where they can arise.

> Ontologies are agreements about shared conceptualisations. Shared conceptualisations include conceptual frameworks for modelling domain knowledge; content specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories. In the knowledge sharing context, ontologies are specified in the form of definitions of representational vocabulary. A very simple case would be a type hierarchy, specifying classes and their subsumption relationships. Relational database schemata also serve as ontologies by specifying the relations that can exist in some shared database and the integrity constraints that must hold for them.

2.5 Formal Ontologies and philosophical issues

In this section we present a brief overview of the philosophical background from which stems a particular branch of the ontological research, namely *formal ontology*.
Investigations concerning the nature of being, and thus ontology, constitutes an important issue in philosophy. The overview we present here, therefore, is far from being exhaustive, but aims only to provide the foundations for the issues investigated by formal ontology. A more thorough illustration of the philosophical background

1. Finite, controlled extensible vocabulary;

2. Unambiguous interpretation of classes and term relationships;

3. Strict hierarchical subclass relationship between classes;

Furthermore the following properties may hold and we consider them typical although not mandatory:

4. Property specification on a per-class basis;

5. Individual inclusion in the ontology;

6. Value restriction specification on a per-class basis;

Finally, the following properties are neither mandatory nor typical but might be desirable:

7. Specification of disjoint classes;

8. Specification of arbitrary logical relationships between terms;

Table 2.2: Properties applicable to ontologies according to Lassila and McGuinness

of the ontological discipline is given by John Sowa in [Sowa 2000]. Here we mention only the concepts that are further investigated in the context of this thesis.
The first notion of ontology as philosophical discipline dates back to Aristotle who defined it as the *science of being as such*, therefore, instead of concentrating on a class of beings as specialistic sciences, ontology studies [Aristotle b]:

> all species of being *qua* being and the attributes which belong to it *qua* being.

Therefore, in Aristotle's view, ontology is the discipline which tries to give an answer to questions such as *What is being?*. Such a question might seem out of the scope in an artificial intelligence context, but it becomes more relevant when reformulated as *What are the properties that can be ascribed to a thing in order for it to be considered being?* Aristotle [Aristotle a] answered the question by developing a system of categories for classifying anything that may be *predicated* about anything in the world. The emphases of this categorisation was on the physical world to be considered as the ultimate reality, and so sensory experience as reaction to external stimuli was abstracted into intangible and unchanging mathematical *forms* or *ideas*. The categorisation proposed by Aristotle had been widely accepted until Kant [Kant 1965] presented a first major challenge to such system. In devising his categorisation system, Kant emphasised that a number of concepts could be associated *a priori* with objects, and that the number of these concepts corresponds to the number of the possible logical functions. Kant developed his own system of categories which was meant as a framework for organising the Aristotelian categories. This framework was organised into four classes, each of which presenting a triadic pattern. Kant's work became a source of inspiration for the German philosophers, who tried to find an explanation for the triadic pattern of his categories.
Among those who have been influenced by Kant's work, although indirectly, there is Husserl, who introduced the notion of *formal ontology*. Husserl criticised the approach to logic as simple calculus which did not take into account the meaning

QUANTITY	QUALITY	RELATION	MODALITY
Unity	Reality	Inherence	Possibility
Plurality	Negation	Causality	Existence
Totality	Limitation	Community	Necessity

Table 2.3: Kant's system of categories

on the symbols used, and proposed a *logic of ideal content* in which he identified six topics of interest, that is: *meaning and expression*, *genus and species*, *parts and wholes*, *the role of grammar in combining meanings*, *intentional experiences and their contents*, and finally *knowledge in terms of meaning intention and meaning fulfillment*.

In particular, in [Husserl 1982] he emphasised the role of intentionality in directing attention to an object of perception and named this direction of investigation *phenomenology*.

As part of his investigation in phenomenology he distinguished the concept of *material ontology*, which roughly corresponds to the notion of *an ontology* (in the AI sense) on a specific topic (such as, for example, law as opposed to biology) from the one of *formal ontology*, which corresponds to Aristotle's ontology. Therefore, the concept of formal ontology is quite close to that of *top-level* ontology in Guarino's classification or *general ontology* in Van Heijst and colleagues' classification, that is ontologies that describe concepts so general that they are shared across all domains. The task of formal ontology is to determine under which conditions possibility applies to an object in general and what are the requirements that have to be satisfied by the way the object is constituted.

More recently Cocchiarella, in [Cocchiarella 1991], defined formal ontology as *the systematic, formal, axiomatic development of the logic of all forms and modes of being*. There is no agreement on the interpretation of the definition of formal ontology and it is still unclear how Cocchiarella's definition relates to Husserl's work on

formal ontology. However, Cocchiarella's definition nicely reconciles the possible interpretations of *formal* as both *rigorous* and *concerned with the forms of being*. According to Guarino and Giaretta, *formal ontology* is not so concerned with the existence of some objects, as with the rigorous description of their *forms of being*, that is their structural features.
In this view we can say that formal ontology is the theory of the distinctions, which can be applied independently of the state of the world, that is:

- distinction among the entities of the world (physical object, events, regions, etc.), and
- distinction among the meta-level categories which are used to model the world (concept, property, role, state, etc.).

Based on the previous notion of formal ontology, Guarino and Welty [Welty and Guarino 2001] have developed a formal ontological analysis methodology called OntoClean which we introduce in the following subsection.

2.5.1 Formal ontological analysis

OntoClean is a methodology for the evaluation of ontological decisions that makes full use of the formal properties of ontology concepts. A full description of this methodology can be found in [Welty and Guarino 2001].
OntoClean provides means to evaluate ontologies built with knowledge engineering methodologies, such as the one by Uschold and Gruninger [Uschold and Gruninger 1996] and Methontology [Gómez-Pérez 1998]. OntoClean uses formal ontological tools that can help to verify that the taxonomic structure built using one of the aforementioned methodologies does not present inconsistencies and is little or not tangled. The task of the formal ontological analysis is to establish what are the ontological properties that constrain the way in which the representational primitives are

used to model the domain [Guarino and Welty 2000c]. The formal properties proposed in OntoClean focus particularly on concepts and the properties of concepts do not explicitly contribute to determine which formal properties hold. The reason why such constraints are needed is that when modelling a domain there are a number of ways of modelling the same knowledge, and the choice of one approach over the others is left to the experience and the background knowledge of the conceptual modeller. Guarino and Welty focus on the basic notions and theories derived by the philosophical investigation to drive the modelling efforts.

In particular, Methontology and OntoClean are complementary methodologies. In fact, Methontology provides the guidelines for building or re-engineering ontologies, whereas OntoClean can be used either in the validation step (when ontologies are engineered or restructured) or simultaneously with the ontology construction (when ontologies are built from scratch). Indeed, OntoClean can assist knowledge engineers in building the classification tree which is the object of Methontology's conceptualisation phase. These two methodologies are currently undergoing an integration process [Fernández-López *et al.* 2001] as part of the activities of the OntoWeb special interest group on Enterprise-standards Ontology Environments (SIG's home page: `http://delicias.dia.fi.upm.es/ ontoweb/sig-tools/index.html`).

The formal tools of ontological analysis are some basic notions of formal ontological properties that have been an object of study in philosophy for centuries. By establishing which of these properties holds we can check whether the choices made in modelling the concepts included in the ontology and in structuring the concepts' hierarchy are sound.

The philosophical notions on which OntoClean builds are four, namely: *identity*, *essence*, *unity*, and *dependence*.

IDENTITY: Identity is the logical relation of numerical sameness, in which a thing stands only to itself. Based on the idea that everything is what it is and not anything else, philosophy has tried for a long time to identify the criteria which allow a thing to be identified for what it is even when the same thing is cognised in two different forms, by two different descriptions and/or at two different times [Wiggins 1967, Hirsch 1982]. This comprises both aspects of finding constitutive criteria (which features a thing must have in order to be what it is), and of finding re-identification criteria (which feature a thing has to have in order to be recognised as such by a cognitive agent). These are distinct, although equally important aspects of identity. Although the problem of *identifying* what features an entity should have in order to be what it is and recognised as such has been central to philosophy, it did not have the same impact in conceptual modelling and more generally AI. One of the claims of the OntoClean methodology that we subscribe to totally is that the ability of identifying individuals is central to the modelling process. We can further elaborate this point by observing that is not the mere problem of identifying an entity of the world that is central to the ontological representation of the world, but the ability of *re-identifying an entity in all its possible forms*, or more formally *re-identification in all the possible worlds*[1]. That is, the problem is related to distinguishing a specific instance of a concept from its siblings on the basis of certain *characteristic properties* which are unique and intrinsic to *that instance* in its whole.

This notion is, of course inherently time dependent, since time gives rise to a particular system of possible worlds where it is highly likely that the same instance of a concept exhibits different features [2]. This problem is known as *identity through change*, an instance of a concept may remain the same while exhibiting different properties at different instants of time. Therefore it becomes important to understand which features or properties can change and which cannot [Guarino and Welty 2000a], and we may add also the situations that can trigger such a change.

If we reformulate the identity problem as *re-identification* we realise that also re-

[1]Some philosophers, e.g. Lewis [Lewis 1993, page 39 ff], hold that there is no such thing as trans-world identity, although objects in one world can have *counterparts* in other worlds.

[2]Here the counterpart theory does not hold, and so identity through time is always accepted.

identification is affected by time; how can we re-identify the same instance at different instant of times?
We face the re-identification problem in everyday life; we are able to recognise the features that permits to distinguish an instance from the others, and when these features are not intrinsic to the instance, we ‘attach’ artificial features, that permit to establish identity. One example is the *Student ID*, which is assigned to university students, in order to identify the student univocally.

UNITY: the notion of *unity* is often included in a more generalised notion of identity, although this two notions are different. Kant [Kant 1965] included unity in his primitive categories describing *quantity* (see Section 2.5 above). While identity aims to characterise what is unique for an entity of the world when considered as a whole, the goal of unity is that of [Guarino and Welty 2000a, page 99]:

> distinguishing the *parts* of an instance from the rest of the world by means of a *unifying relation* that binds them together (not involving anything else).

For example, the question ‘Is this my car?’ represents a problem of identity, whereas the question ‘Is the steering wheel part of my car?’ is a problem of unity. Also the notion of unity is affected by the notion of time, can the parts of an instance be different at different instants of time? This problem is also known as *individuation*.

ESSENCE: The notion of *essence* is strictly related to the notion of *necessity* [Kant 1965]. An *essential property* is a property that is necessary for an object, or, in other words, a property that holds, that is true in every possible world [Lowe 1989]. Based on the notion of *essence*, Guarino and colleagues [Guarino *et al.* 1994] have introduced the notion of *rigidity*, to represent the strict connection that there exist between time and modality [Kant 1965], [Kripke 1980]. A rigid property is a property that is necessary to all instances, that is a property ϕ such that: $\forall x \phi(x) \rightarrow \Box\phi(x)$.

In [Tamma and Bench-Capon 2001a, Tamma and Bench-Capon 2001b] we have highlighted how the notion of *rigidity* depends on that of *time* and *modality*; more recently Guarino and Welty have extended the definition of rigidity by formalising this relationship, so a rigid property ϕ is such that $\Box\,(\forall x, t\phi(x,t) \rightarrow \Box\,\forall t'\phi(x,t'))$. It is important not to confuse modal necessity with temporal permanence. Modal necessity means that the property is true, in every possible world, time is undoubtably one partition of these worlds, however time permanence means that the property is true in that world (time), with no information concerning the other possible worlds, and this might happen by pure chance.

DEPENDENCE: Husserl introduced the distinction between *dependence* and *independence* [Husserl 1982]. In the OntoClean methodology the notion of dependence is considered related to properties. In this context dependence permits us to distinguish between *extrinsic* and *intrinsic* properties based on whether they depend or not on objects other than the one they are ascribed to. *Intrinsic* properties are those characterising inherently an object, and they do not depend on any other object. *Extrinsic* properties are usually assigned by some external agent, and thus are not inherent. Intrinsic properties are good candidates to became identity conditions as long as they univocally identify an object; when they do not then extrinsic properties may be assigned in order to identify the object univocally.
These four notions are further discussed in Chapter 4, where they are related to the ontology model that is the main contribution of this thesis.
In the OntoClean methodology these formal properties are used to improve the taxonomic structure of the ontology, but their importance is not confined to constraining the use of subsumption in hierarchies, as they are also extremely useful in understanding which modelling primitives are most appropriate for the domain to represent. For example, in modelling the entity of the world `student` we might have to decide whether this is a concept or a role. By applying the formal tool of ontological analysis we can realise that *being a student* is not a rigid property, as there

might exist a world where a person is not a student. One of the conditions for an object to be a role is that it is not rigid, so `student` is more correctly modelled as a role and not as a concept.

The main limitation of OntoClean is that it does not take into account concept descriptions in order to assign formal properties with concepts. Besides, it is mainly based on the knowledge engineers' understanding of a concept, and thus is not very objective.

The main goal of this thesis is to provide a set of meta-properties for attributes which provide a better characterisation of attributes and the role they play in defining a specific concept. These meta-properties, which we present and discuss more in detail in Chapter 4, are **Mutability, Mutability Frequency, Event Mutability, Reversible Mutability, Modality, Prototypicality, Exceptionality, Inheritance, Distinction**. This set of meta-properties not only provides a characterisation of attributes that provides the concept of a richer semantics but they also can be used to complement ontoclean, in that they can be used to guide knowledge engineers in determining *identity, rigidity*, and *identity conditions.* These meta-properties can also be used to disambiguate the meaning of concepts that seem similar but are actually different. Indeed, concepts have to be considered similar if they show the same properties (attributes) and these *show the same behaviour* in the concept's definition. We will examine the problem of assessing concepts similarity and the problems that can hinder such process in next Section.

Chapter 3

The role of ontologies in knowledge sharing and reuse

3.1 Introduction

In Chapter 2 (Section 2.2) we have introduced the concept of *ontology* and the different meanings that this word has taken in the literature, from the philosophical one that dates back to Aristotle to the more recent meanings associated with the word in the field of Artificial Intelligence.

In this chapter we focus our attention on the role played by ontologies in knowledge sharing. The advances on the Internet have made it available a large number of diverse knowledge sources whose content models a particular viewpoint concerning a domain. Their content needs to be reconciled and combined in order to get a richer understanding of a domain and to add value to the diverse knowledge sources.

This chapter presents an overview of knowledge sharing and reuse and focuses on the roles that ontologies play in this context. We here illustrate the diverse approaches presented in the literature and we sketch a novel proposal for knowledge sharing, namely *ontology clustering*, which motivates the ontology model that is an object of this thesis and that is presented in next chapter.

The structure of this chapter is the following: Section 3.2 illustrates the knowledge sharing problem, Section 3.3 introduces and clarifies the terminology we use throughout the chapter, while Section 3.4 discusses agent architectures for knowledge sharing. Section 3.5 presents an overview of the different types of heterogeneity that can affect knowledge sources, Section 3.6 presents the different approaches to knowledge sharing and the roles played by ontologies; the different approaches are analysed in following sub-sections. Section 3.7 presents a novel approach to knowledge sharing based on an hierarchical structure of multiple shared ontologies, each permitting the sharing of knowledge at different levels of abstraction. This novel approach is based on the ability of assessing semantic similarity among concepts belonging to different ontologies. Different means to assess semantic similarity are illustrated in Section 3.7.3. Finally, we draw conclusions.

3.2 Knowledge sharing and reuse

The last decade has seen a deep change in the way intelligent systems are built. It had already been recognised that capturing knowledge was the key to building large and powerful intelligent systems, and many tools were provided to assist knowledge engineers in the different activities involved in building such systems. Knowledge acquisition, that is the process of collecting knowledge from a human domain expert and formalising it, has proved to be extremely complex and time consuming (*knowledge acquisition bottleneck*). Acquiring and formalising knowledge can seriously hamper the building from scratch of very large knowledge bases, although it could be noticed that, when building knowledge bases, even for very diverse applications (such as medicine or electronics) there are some general notions that are common across the domains. For example both medicine and electronics share the notion of *diagnosis*, although at a very general level.

A possible approach to this problem is to try to isolate the knowledge components which are shared across different domains and represent them in a way general enough, so that they could be reused in different applications, following the similar approach in software engineering. This was the aim of the ARPA Knowledge Sharing Effort [Neches *et al.* 1991] which proposed an approach in which knowledge sharing was ubiquitous, and the process of building a new knowledge base relied on the reuse of knowledge components.

The term *sharing* denotes sharing by multiple persons, across multiple applications, and in multiple contexts. In some applications the term comprises the meanings *sharing*, *reuse*, and *exchange*, each of which may have a specific technical meaning. In particular, by *reuse* we mean the use of knowledge by one or more applications at different time; by *sharing* we intend the use of the same knowledge by multiple applications at the same time, while by *exchange* we refer to the copying of information from one application to another. Sharing, reuse and exchange are strictly intertwined. Indeed, explaining how to reuse a component often requires communicating subtle issues that are more easily expressed formally; these explanations (which might be expressed more or less formally) require shared understanding of the intended interpretations of terms. The reuse of source specifications is only feasible to the extent of their view of the domain is compatible with the intended new use. In this thesis, we disregard the differences in meaning between sharing, reuse and exchange, and we collectively refer to refer to them as *knowledge sharing and reuse*.

The literature distinguishes the type of knowledge components that can be shared and reused in *ontologies*, that is ‘knowing what’, and *problem solving methods*, that is ‘knowing how’ [Uschold *et al.* 1998]. In this chapter we focus our attention on the sharing of ontologies and we disregard the sharing of problem-solving methods, although we acknowledge that these two knowledge components are strictly intertwined and that the nature of the ‘knowing how’ necessarily affects the way in which the ‘knowing what’ is represented. This problem is known in the literature as *interaction problem* [Bylander and Chandrasekaran 1988].

Building intelligent systems in a way that saves time and is cost effective can be achieved by adapting knowledge from existing applications into the new system. This might mean reusing software and/or sharing knowledge [Gómez-Pérez 1998]. Software engineering provides us with the guidelines, methods, and techniques to apply in order to reuse software made by others. If the system to reuse is a knowledge based one then the reuse concerns not only software but also knowledge, which implies identifying the knowledge and the inference engines that can be reused. In [Gómez-Pérez 1998] the author discusses the problems that might be encountered when knowledge is to be shared and reused. We have classified these problems in two big categories:

1. Heterogeneity problems;
   (a) *Heterogeneity of knowledge representation formalism*;
   (b) *Heterogeneity of the implementation languages*;
   (c) *Lexical problem*. This problem is also known in knowledge representation as *implicit inconsistencies problem* [Morgenstern 1998];
   (d) *Synonymity*;
2. Background assumptions problems;
   (a) *Hidden assumptions*;
   (b) *Loss of common sense knowledge*.

Heterogeneity problems have been studied at length, especially in the database area [March 1990, Kim and Seo 1991]. They are further discussed in Section 3.5.
Background assumptions problems are caused either by the, often hidden, assumptions under which an intelligent system is built (for example, the type of theory used to represent time) [Lenat 1995b] or the assumptions on common sense knowledge that are taken for granted. That is, the specification of some knowledge components is not made explicit because it is assumed to be common knowledge. Ontologies can prove helpful in dealing with both heterogeneity and background assumptions problems; in the first case they provide a warehouse of vocabulary that can be used to solve heterogeneity, in the second case ontologies can provide an explicit specification of the common sense knowledge used and of the hidden assumptions made

while building the system. This point is further discussed in Chapter 4, where we present an ontology model which forces knowledge engineers to make the hidden assumptions explicit.

Finally, in building intelligent systems knowledge could be shared with other applications already built. In such an approach, knowledge sources are considered as *agents* distributed in an agent network [Gómez-Pérez 1998]. Agents provide each other with problem-solving services and carry out these tasks autonomously and independently. Interoperation between agents is obtained by the agents' commitment to use a shared vocabulary and to associate a common meaning with the terms of the vocabulary. Agents architectures for knowledge sharing are briefly introduced in Section 3.4.

In this thesis we specifically address the sharing of *ontologies*, that is the sharing of formal and explicit specifications of the conceptualisations used to model the domains of interest.

3.3 Terminology of knowledge sharing

Ontologies have moved out of the research environment and have become widely used in many expert system applications not only to support the representation of knowledge but also complex inferences and retrieval [McGuinness 2000]. More and more, ontologies are the efforts of many domain experts and are designed and maintained in distributed environments. For this reason many research efforts are now devoted to reusing and sharing the knowledge expressed in diverse ontologies [Uschold *et al.* 1998, Pinto *et al.* 1999].

However, the task of sharing ontologies is not so straightforward, as many problems arise when independently developed ontologies are used together. Research in this area has started tackling many issues, but many questions are still unanswered.

In the following sections we will introduce the problems hindering the process of sharing and reusing ontologies, but before proceeding by illustrating these problems, we need to clarify and define the terms used in the field. We have drawn from several works ([Pinto *et al.* 1999, Fridman Noy and Musen 1999, Chalupsky 2000, Klein 2001]) and we have identified the following terms, that we will use throughout the thesis, unless differently stated:

- INTEGRATING: The process of creating a new ontology from two or more existing ontologies. The domain of the integrated ontologies is different from the one of the resulting ontology, but there exist a relation among these domains. Terms from the integrated ontologies can be used as they are, specialised, adapted or augmented [Pinto *et al.* 1999].

- ALIGNING: The process of establishing links between the sources ontolologies and allowing the aligned ontologies to reuse information from one another while the source ontologies still persist [Fridman Noy and Musen 1999];

- MERGE: The process of building a single ontology that is the merged version of the source ontologies. Often, the source ontologies cover similar or overlapping domains [Fridman Noy and Musen 1999];

- COMBINING: The process of using two or more different ontologies to perform a task in which their mutual relation is important for the task at hand. Combining might mean align, merge or integrate [Klein 2001];

- ARTICULATION: The specification of the alignment, that is the set of all the shared concepts in aligned ontologies;

- MAPPING: The process of relating concepts or relations from different sources that are deemed similar according to some similarity function. It involves the definition of an equivalence relation. It corresponds to virtual integration.

- TRANSLATION: The process of changing the representation formalism of an ontology while preserving the semantics;

- TRANSFORMATION OR MORPHING: The process aiming to modify the semantics (by performing abstractions or semantic shifts), and possibly the representation formalism of concepts and relations in the ontology in order to make them suitable for purposes different from the original one [Chalupsky 2000];

- VERSIONING: The method to preserve consistency in the relations between newly created ontologies, the existing ones, and the data that instantiate them [Klein 2001];

- VERSION: The result of a change or an update of an ontology that can coexist with the original one [Klein 2001];

3.4 Agent architectures and knowledge sharing

A knowledge engineering paradigm that has proved to be useful for dealing with the integration of heterogeneous knowledge is based on a multi-agent system architecture, where human and software agents interoperate and so cooperate within common application areas. Agents in a multi-agent system are characterised by abstraction, interoperability, modularity and dynamism. These qualities are particularly useful in that they can help to promote open systems which are typically dynamic, unpredictable and highly heterogeneous [Jennings 1995], as is the Internet. In these types of application domains, the interoperability offered by the multi-agent system approach is required because the individual components that interact with agents are not known a priori. Additionally, this paradigm provides robustness and flexibility of the interfaces between both the agents that exist within the Internet and between agents and software systems, this is essential since the interfaces cannot be anticipated at design time.

Within a multi-agent system, agents are characterised by different "views of the world" that are explicitly defined by *ontologies*, that is, views of what the agent knows to be the concepts describing the application domain which is associated with the agent, together with their relationships and constraints [Falasconi *et al.* 1996]. The interoperability typical of multi-agent systems is achieved through the reconciliation of these views of the world by a commitment to common ontologies that permit agents to interoperate and cooperate while maintaining their autonomy. In open systems, agents are associated with knowledge sources which are diverse in nature and have been developed for different purposes. Knowledge sources embedded in a dynamic environment can join and leave the system at any time. From the ontologies perspective, dealing with open systems implies that ontologies are often the efforts of many domain experts and are designed and maintained independently in distributed environments. In such a situation interoperation between agents is based on the reconciliation of their heterogeneous views, which is accomplished by merging the diverse ontologies associated with the agents composing the system [Sycara *et al.* 1998]. The merging of diverse ontologies has to be accomplished bearing in mind that since agents are highly heterogeneous, they are likely to be incapable of fully understanding each other, so that both syntactic and semantic mismatches can arise which need to be reconciled (see Section 3.5).

An agent's ability to represent domain knowledge in a consistent manner has to be complemented by some reasoning capability. According to Wooldridge and Jennings, [Wooldridge and Jennings 1995] an agent architecture is *one that contains an explicitly represented, symbolic model of the world, and in which decisions (for example about what action to perform) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation*[1]. Therefore ontologies in multi-agent systems require a high degree of expressive power to support the application of reasoning techniques that result in sophisticated inferences such as those used in negotiation, which is motivated by the requirement for agents to solve problems arising from their interdependence upon one another

[1]We are restricting the discussion to deliberative agents, as it would not make sense to define ontologies for purely reactive agents

[Parsons *et al.* 1998].

Designing multi-agent systems to deal with the sharing of heterogeneous knowledge sources gives rise to the requirement for ontologies that can be easily integrated and provide a base for applying reasoning mechanisms. In fact, the sharing of ontologies depends heavily on a precise semantic representation of the concepts and their properties [Fridman Noy and Musen 1999, McGuinness 2000, Tamma and Bench-Capon 2000].

To recap, we mention here two main advantages that can be achieved by using agent architectures for sharing knowledge [Preece *et al.* 2001]:

- Openness. Agents can dynamically join and leave a network, advertise their capabilities to one another and form alliances in order to perform a specific task, therefore they are inherently open;
- Knowledge-level communication. By means of shared ontologies which permit the commitment to a shared language and to a shared interpretation of the symbols of the language, agent communication is performed at knowledge level [Newell 1982]

3.5 Assessing heterogeneity between sources

Literature in the field has recognised a number of problems that can hinder the process of combining (that is, merging and integrating) ontologies [Visser *et al.* 1998, Visser and Tamma 1999, Chalupsky 2000, Klein 2001]. These problems can be broadly summarised by the term *heterogeneity*, which denotes the aspects in which independently developed ontologies can differ. Usually heterogeneity arises from different decisions made during the design and development of ontologies [Visser and Tamma 1999].

When dealing with heterogeneous knowledge sources/agents, one key issue is understanding what forms of heterogeneity exist between the knowledge sources and what are the mismatches caused by them. The vast amount of literature on the integration of heterogeneous information sources is sometimes confusing regarding the kinds of heterogeneity and the mismatches that can arise, especially where the knowledge engineering and data modelling fields meet. This makes it less easy to compare the different approaches [Tamma and Visser 1998, van Zyl and Corbett 2000].

An attempt to reconcile and compare the different definitions presented in the literature is given by Klein [Klein 2001] and Chalupsky [Chalupsky 2000]. We used these works and the one by Visser and colleagues [Visser *et al.* 1998] as a starting point for the definitions of the different types of heterogeneity we consider in this thesis.

Heterogeneity (and thus the mismatches it can cause) can be broadly distinguished into *non-semantic* and *semantic heterogeneity* [Kitakami *et al.* 1996]. The former type of heterogeneity is also called *syntactic* or *language heterogeneity* in [Klein 2001], while the latter is also called *ontology heterogeneity* by Visser and colleagues [Visser *et al.* 1998]. Syntactic heterogeneity denotes the differences in the language primitives that are used to specify ontologies, while semantic heterogeneity denotes differences in the way the domain is conceptualised and modelled.

3.5.1 Syntactic or Language heterogeneity

Language heterogeneity occurs when ontologies written in different ontology languages are combined. In [Klein 2001] four types of mismatch due to language heterogeneity are recognised.

- *Syntax*. Different ontology languages are often characterised by different syntaxes. Differences in the language syntax give rise to mismatches that can be resolved by means of rewrite rules.

- *Logical representations*. This kind of mismatch is caused by differences in the representation of logical notions, and more precisely, differences in the language constructs that are used to express something. So, for example, a language might permit the expression of the OR operator explicitly, for instance by using the statement `(OR, A, B)`, whereas another language may represent the same concept by the AND operator and the negation of the class declaration, *e.g.* `(AND, A, (NOT B))`.

- *Semantics of primitives*. This is, to a certain extent, a more subtle kind of mismatch deriving from non-semantic heterogeneity. Indeed, it is caused by differences in the semantics of the language statements. These differences can be sometimes quite difficult to detect, since two languages can use constructs with the same name, but slightly different interpretations, or sometimes the same interpretation might be associated with constructs with different names.

- *Language expressivity*. Mismatches due to differences in the expressivity between two languages are those which have the most impact on the problem of integrating/merging ontologies. Differences in the expressive power of the languages imply that one language can express something that the other language cannot express. For example, some languages support negation while others do not. A complete comparison of different ontology languages can be found in [Corcho and Gómez-Pérez 2000].

3.5.2 Semantic or Ontology heterogeneity

Mismatches caused by *semantic heterogeneity* occur when different ontological assumptions are made about the same domain. This kind of mismatch becomes also evident when combining ontologies which describe domains that partially overlap. In particular, mismatches due to ontology heterogeneity can occur while *conceptualising* and/or *explicating* [Visser *et al.* 1998] the domain. Visser and colleagues use

these terms to refer to the definition of ontology given by Gruber [Gruber 1993] stating that *"An ontology is the explicit specification of a conceptualisation"*. That is, the process of designing an ontology is comprised of two main stages, the *conceptualisation* of the domain and the subsequent *explication* of this conceptualisation, and the idea is that ontology heterogeneity can be introduced in both stages of the design [Visser *et al.* 1998]. These two stages can be further broken down into sub-stages [Visser and Tamma 1999], and heterogeneity can be introduced in each of these sub-stages. However, for the scope of this thesis, we will just concentrate on the conceptualisation and explication stages, without going into too much detail.
Mismatches due to ontology heterogeneity can, therefore, be subdivided into *conceptualisation* and *explication mismatches*.

Conceptualisation mismatches

Conceptualisation mismatches are semantic differences arising from different conceptualisations of the concepts and the relations in the ontology domains.

- *Model coverage and granularity*. This type of ontology heterogeneity occurs when different conceptualisations, and thus different ontologies, model the same part of domain differently both with respect of model coverage and granularity. This is usually the reason why ontologies are merged.
  For example, one ontology can model the concept `Wine` into the subconcepts `White-Wine` and `Red-Wine` and then further specify the two. Therefore, the concept `White-Wine` has two subconcepts, `Chardonnay` and `Riesling`; analogously the concept `Red-Wine` is subdivided into its subconcepts `Beaujolais` and `Chianti`. Another ontology might model the concept `Wine` into the subclasses `White-Wine` and `Red-Wine`, only [Fridman Noy and McGuinness 2001]. Here we have a problem of mismatch in the granularity of the representation, because the level of detail in which the domain is modelled differs across the conceptualisations concerning the two

ontologies. In this case it is not a matter of 'how a concept is described' (which would be a problem of attribute-type mismatch), but a problem of how the conceptualisation of the common domain is modelled.

A problem of model coverage would arise if each different ontology modelled only part of a domain. For example, one ontology could model the concept `Wine` into the subconcepts `Red-Wine` and `Rosé-Wine`, whereas another could model the concept `Wine` into the subconcepts `White-Wine` and `Champagne`. Here, the coverage of the domain is different, and merging these two ontologies would provide a richer understanding of the domain.

- *Scope*. This mismatch occurs when two concepts or relations in the ontologies seem to be the same but their extensions (that is the set of their instances) are not the same although they are not disjoint. Relations mismatches also include mismatches concerning the assignment of attributes to concepts, since those represent relations between conceptual entities [Woods 1975].

  Visser and colleagues [Visser *et al.* 1998] further distinguish between categorisation and relation heterogeneity which in turns cause the following types of mismatches:

  – *Categorisation mismatch*. This kind of mismatch occurs when two conceptualisations identify the same concept but this has different subconcepts in the ontologies. An example of categorisation mismatch can occur when one conceptualisation subdivides the concept `Person` into the subconcepts `Male` and `Female` whereas the other subdivides it into the subconcepts `Child`, `Teenager` and `Adult`. It concerns only concepts.

  – *Relation mismatch*. It occurs when different ontologies conceptualise the same domain by recognising different types of relations. It can be further broken into:

    ∗ *Structure mismatch*. A structure mismatch occurs when the conceptualisations distinguish the same set of concepts but differ in the way these concepts are structured by means of relations, whose semantics overlap, although they are not equal. For example one conceptualisation relates the concepts `Book` and `Chapter` by the relation `is-made-of` while the other relates them through the relation `has-component`.

∗ *Attribute-assignment mismatch.* An attribute-assignment mismatch occurs when two conceptualisations differ in the way they associate an attribute with other concepts. Visser and colleagues illustrate this mismatch providing the following example; let us assume to have two conceptualisations involving the concepts `vehicle`, `car` and `colour`, and that the concept `car` is subclass of the concept `vehicle`. We have an *attribute-assignment mismatch* if one conceptualisation associates the attribute `colour` with the concept `car` while the other associates it with the concept `vehicle`.

∗ *Attribute-type mismatch.* This type of mismatch occurs when different conceptualisations describe the same concept with the same set of attributes, but the values associated with the attributes have different types. For example two different conceptualisations might describe the concept `Wine` in terms of the attribute `Colour` where one conceptualisation associates with this attribute the set $\{White$, $Rosé$, $Red\}$, while the other associates the set $\{Red, White\}$. When the types are just encoded differently (for example length expressed in kilometers or in miles) it becomes a case of *encoding* mismatch (see below). While encoding mismatches can be resolved quite easily by defining mapping functions converting one value type into the other, cases of attribute-type mismatches are not so straightforward.

Explication mismatches

Explication mismatches arise because of differences in the specification of the domain conceptualisation. During the conceptualisation phase the concepts describing the domain are selected, in the explication phase these concepts are made explicit, usually labelling each of them with a *term* (which is one or more words in natural language) and associating a *definition* with each term, which could be expressed in natural language or in a formal ontology language.

We distinguish six types of mismatches, the first three of them concern the modelling choices, the following two concern the choice of terms that are used to label a concept in the ontology, while the last type of mismatch concerns the way in which concepts are encoded. This type of mismatch covers, in part, the mismatch that Visser and colleagues call *Attribute-type mismatch*.

- *Representation paradigm*. This type of mismatch depends on different representation paradigms used to model the same domain. It can became apparent with concepts such as time, actions, plans, causality, etc. Another reason for this type of mismatches could be the adoption of different knowledge representation paradigms.

- *Top-level concepts*. Top-level concepts mismatches arise because ontologies differ in the top-level ontologies they refer to (see Section 2.4).

- *Modelling conventions* (Also known as *concept description* in [Klein 2001]). Modelling convention mismatches depend on modelling decisions made while designing the ontology. For instance, it is often the case that an ontology designer has to decide whether to model a certain distinction by introducing a separate class or by introducing a qualifying attribute relation [Chalupsky 2000, Fridman Noy and McGuinness 2001]. Another example of modelling conventions might be the following, let us suppose we want to model the concept `Bird`, and its subconcepts `Robin` and `Penguin`, and their ability to fly.

We could do so by adding to the concept `Bird` the attribute `can-fly` which takes value in the set $\{Yes, No\}$. This attribute is inherited by the subconcepts, becoming a *distinguishing attribute* in that it allows us to distinguish between siblings of a same concept (see Section 3.7.1). Therefore we distinguish the ability to fly between a `Robin` and a `Penguin` because the former concept has the attribute `can-fly` associated with the value Yes while the latter has it associated with the value No. We could model the same information by creating two different subconcepts of the concept `Bird`: `Flying-Bird` (described by the attribute `can-fly` associated with the value Yes) and `Non-Flying-Bird` (described by the attribute `can-fly` associated with the value No), respectively. Then we could model the concept `Robin` as subconcept of `Flying-Bird` while `Penguin` would be modelled as subconcept of `Non-Flying-Bird`.

- *Synonym terms*. This type of mismatch is discussed in length in [Visser *et al.* 1998], where is called *term mismatch.* It occurs when the same concept, attribute, or relation is referred to by different terms and/or described by different definitions, although semantically equivalent. For example, let us suppose to have two ontologies O_1 and O_2 that describe the vehicles domain, and that both model the concept *Car*, by calling it `Automobile` in ontology O_1 and `Motor-car` in ontology O_2. Additionally, the two concepts can either be described by the same set of attributes, for example `Registration-Year`, `Maximum-Speed`, or by two sets of attributes that are semantically equivalent. For example `Automobile` in ontology O_1 could be described by the set of attributes `Mat-Year` and `Max-Speed`, while the concept `Motor-car` in ontology O_2 is described by the set of attributes `Registration-Year` and `Maximum-Speed`.

  A special case of this kind of mismatch is when ontologies are written in natural language, which gives rise to different and extremely complex mismatches which have a deep semantic nature and can require a lot of human effort to be resolved.

- *Homonym terms*. This type of mismatch occurs when a term can refer to different concepts depending on the context. It is mainly due to the existence of homonyms in natural language, such as the English word *wood* which can mean a collection of trees or the material that forms the main substance of the trunk or the branches of a tree. An example of this mismatch can be found in a situation where two ontologies O_1 and O_2 might have the same term (and even the same definition) which denotes different concepts.
  Homonym terms can appear in different ontologies *concerning the same domain* if these 'operationalise' the term in different ways. For example the concept `Year` might be described as *a period of time divided into 12 months* in two different ontologies, O_1 and O_2. If the first ontology considers a month as *a period of time of 30 days*, whereas the second ontology considers a month as a period of time that can have a number of days between 28 and 31, then the term `Year` in O_1 is an homonym of the analogous term in O_2.

- *Encoding*. This is maybe the easiest mismatch to resolve. It occurs when different ontologies encode values in different ways. In a way it includes also the *Attribute-type mismatch* introduced by Visser and colleagues [Visser *et al.* 1998], in the sense that it is arguable that one conceptualises also taking into account things such as the unity of measure used to instantiate the concept.

Heterogeneity, and especially ontology heterogeneity, can seriously hinder attempts of sharing and reusing knowledge automatically. In fact in order to recognise whether two concepts from heterogeneous knowledge sources are similar, we cannot only rely on the terms denoting them and on their descriptions, but we need to have a full understanding of the concepts in order to decide whether they are semantically related or not.

Ontology heterogeneity can also hinder the ability to reason with instantiations of shared knowledge because it can cause what in knowledge representation are called *implicit inconsistencies* [Morgenstern 1998]. Before defining implicit inconsistencies is worth noting that conflicting knowledge can and must be included in an ontology. A `Bird` can or cannot fly and the ontology should model this kind of knowledge. It is very likely that, when sharing knowledge, conflicting information has to be modelled in the shared ontology, reflecting what is possible in the different domains. However, when we need to automatically draw conclusion from the knowledge modelled in the shared ontology then inconsistencies can arise that need to be dealt with.

When dealing with heterogeneous knowledge sources, detecting inconsistencies can become even more difficult. We define *implicit inconsistencies* those for which inconsistent values are associated with concepts that are affected by either *language heterogeneity* or *ontology heterogeneity*.

3.5.3 Reconciling mismatches

In Section 3.5.1 and in Section 3.5.2 we have presented the types of heterogeneity that can affect ontologies independently built. Each heterogeneity type corresponds to a mismatch, and many heterogeneity types can affect ontologies, therefore many mismatches can occur while combining different ontologies. The mismatches have been summarised in Figure 3.1. When different ontologies are combined, the different types of heterogeneity need to be detected and reconciled. Many of the systems providing an aid to ontology integration and merging have ways to semi-automatically detect syntactic types of heterogeneity, but very few tools are able to detect and deal with semantic heterogeneity. The difficulty lies in the fact that, in order to detect heterogeneity types other than syntactic it is necessary to have a precise understanding of the the semantics of the concepts involved in the conceptualisation [Chalupsky 2000]: this involves understanding the terms used to explicate

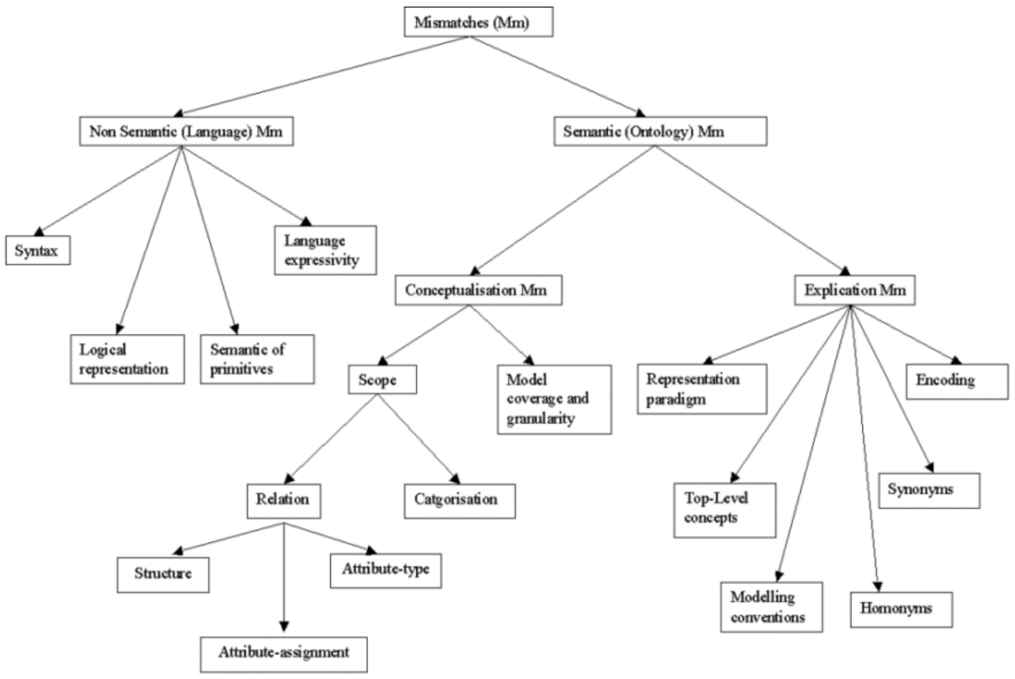

Figure 3.1: The different types of mismatches

these concepts, the structure of the concepts, and the context in which a concept is to be used.

When combining different ontologies a number of steps have to be considered, Chalupsky [Chalupsky 2000] identified five steps to ontology merging, which are:

1. Finding semantic overlap or *hypothesising alignments*;
2. Designing transformations to bring the sources into mutual agreement;
3. Editing or *morphing* the sources to carry out the transformations;
4. Taking the union of the morphed sources;
5. Checking the result for consistency, uniformity, and non-redundancy and if necessary repeating some or all of the steps above.

The first two steps are applicable also to ontology integrations (at least for what concerns the similarities among the domains of conceptualisation), and is during these two steps that the different types of heterogeneity have to be detected and reconciled. The automatic execution of these steps is supported at various degrees by the state of the art. We can relate the first two steps to the heterogeneity types (analysed in Section 3.5.1 and Section 3.5.2), and, when possible, we relate them with the state of the art.

The first step consists of identifying the concepts that semantically overlap. This step involves the assessment of the semantic similarity between concepts. Assessing semantic similarity is quite a complex task, chiefly because current similarity measures are usually binary, so they do not permit an assessment of the degree of similarity and do not usually take into account the structure of the definitions of concepts. We further investigate semantic similarity in Section 3.7.3, reviewing the literature on this topic. In current systems, semantic similarity is assessed mainly by comparing names of concepts by means of lexicons, and by comparing the structure of attributes. Non-semantic and semantic heterogeneity can make this process even more complex. Indeed, the first step should be to make the structure of the ontologies comparable, by resolving non syntactic heterogeneity. Also knowledge representation paradigm and encoding heterogeneity should be detected and reconciled at this stage.

OntoMorph [Chalupsky 2000] provides two mechanisms to describe transformations: *syntactic rewriting* via pattern-directed rewrite rules, and *semantic rewriting* modulating syntactic rewriting via partial semantic models and logical inference supported by an integrated knowledge representation systems. Thus, *syntactic rewriting* permits translations between ontology languages (and more generally knowledge representation languages) which may differ in the syntax, expressiveness, logical representation and semantics of primitives. However, in some cases syntactic rewriting might not be sufficient, and there are cases in which logical inferences have to be considered to reconcile non-semantic heterogeneity. In these cases the kind of *semantic rewriting* provided by Ontomorph might help.

Once the non-semantic heterogeneity is reconciled we can start looking for similar concepts. Systems such as Anchor-PROMPT [Fridman Noy and Musen 2001] and Chimaera [McGuinness *et al.* 2000] can perform a limited assessment of similarity between concepts, which is based on lexical similarity, but this method can be seriously affected by the presence of *synonym* and *homonym terms*. At this stage only heterogeneity concerning synonym terms can be reconciled by using some thesaurus like WordNet [Miller 1990].

The next stage while performing step one, consists in analysing the structure of the concept's definition. Analysing the structure of the definitions is necessary to detect and reconcile cases of heterogeneity causing *homonym terms mismatches*, *attribute-type mismatches*, *structure mismatches* and, on some extent, *modelling conventions mismatches* (for what concerns the structure of the concept only).

The next step to merge ontologies is to design the transformations to bring the sources into mutual agreement. In this step, those types of heterogeneity that involve the structure of ontologies have to be reconciled, because they can affect the process of bringing the source ontologies into mutual agreement. Therefore, it is while performing this step that *categorisation mismatches*, *attribute-assignment*, *model coverage and granularity*, and *top-level concepts mismatches* can be detected and reconciled. Anchor-PROPMT [Fridman Noy and Musen 2001] proposes a technique that semi-authomatically detects alignment points, whereas Chimaera [McGuinness *et al.* 2000] permits the partial browsing of the subclass/superclass hierarchy. However, the techniques currently available can help in detecting only few types of heterogeneity, for example, there is no system that can fully detect heterogeneity due to differences in the assigment of an attribute to a class (attribute-assignment mismatches), and some types of heterogeneity might not even be detected by human experts unless the context of the ontology is fully specified in the documentation.

We have summarised the relation among the different types of heterogeneity and the necessary steps of ontology combination in Figure 3.2. The following subsection presents a small example of heterogeneity mismatches and implicit inconsistencies.

3.5.4 Example

Let us consider the ontologies of two different companies producing pastries. Company C_1 produces only savoury pastries, consequently in the ontology modelling the baking products produced by company C_1 we find the concept `Pastry` described by an attribute `Savouriness` which models the salt content of the pastry and takes value in the interval $[0..20]$ where the number denotes the percentage of salt present in the pastry. Company C_2 produces only sweet pastries, and the ontology modelling the products sold by company C_2 represents the concept `Pastry` described by an attribute `Sweetness` which models the sugar content of the pastry and takes value in the interval $[0..60]$. Let us now suppose that the two companies C_1 and C_2 merge and so they have to integrate the two ontologies into one.

The first problem to take into account is that the concept `Pastry` is heterogeneous

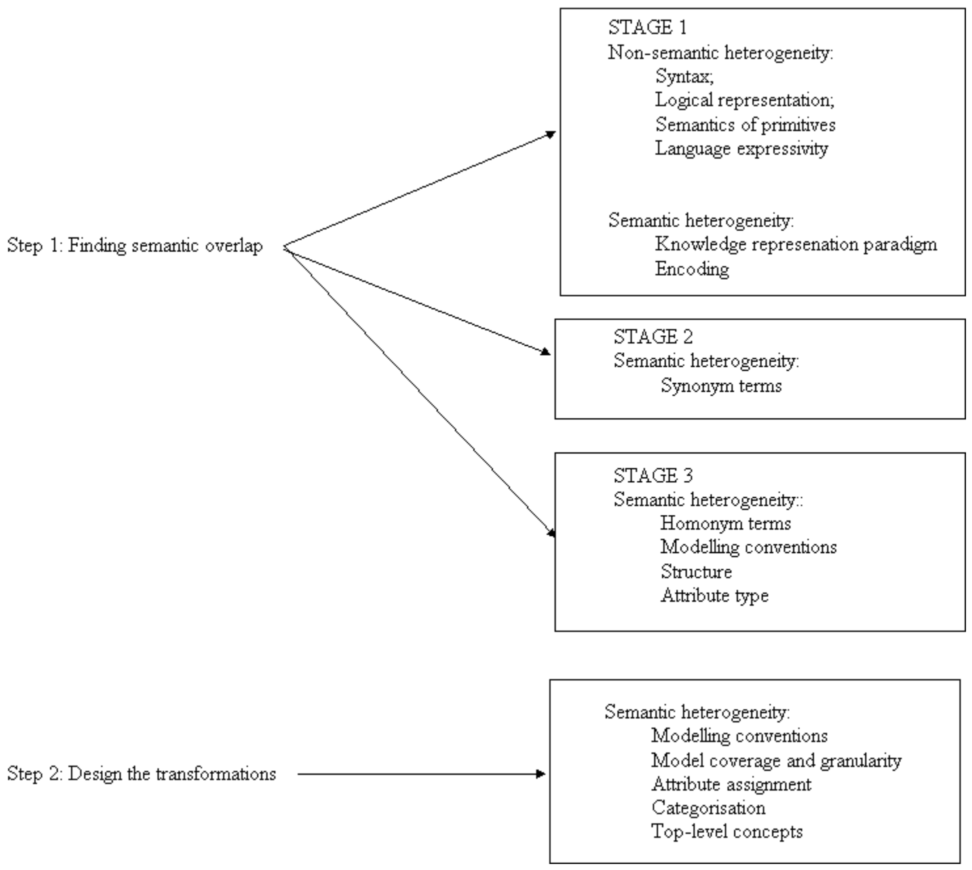

Figure 3.2: Relating the steps to combine ontologies to the heterogeneity types

across ontologies O_1 and O_2. More precisely, this is a case of *homonym mismatch*, as the same term `Pastry` has been used to model both *savoury pastries* and *sweet pastries*. Of course this would have not happened if the concept had been termed `Savoury-Pastry` in ontology O_1 and `Sweet-Pastry` in ontology O_2. However, in this case the the homonym mismatch hides a mismatch in the conceptualisation, in the sense that while company C_1 conceptualises the pastries domain by considering *only* savoury pastries, company C_2 conceptualises the same domain in terms of sweet pastries only. So, besides the homonym mismatch, which is mismatch in the explication of the concept, we have also a *scope mismatch*, which is a conceptualisation mismatch.

Let us now proceed with the attempt of merging the two ontologies, bearing in mind that we have found a concept *Pastry* which is parent of two disjoint subconcepts *Savoury-Pastry* and *Sweet-Pastry* and that these two subconcepts are actually modelled in the ontologies O_1 and O_2, though labelled with the same term `Pastry`. As design choice let us assume that the shared product ontology contains now a concept `Pastry` which is obtained by merging the concepts `Pastry-`O_1 and `Pastry-`O_2. The resulting concept `Pastry` inherits attributes from both parent concepts, so it inherits the attribute `Savouriness` and the attribute `Sweetness`. The presence of both attributes in the concept does not create any kind of ontological problem, although we could observe that the two attributes could be considered two extremes of the same spectrum, that is, from very salty, to neither salty nor sweet to very sweet. If we need to reason with the knowledge instantiating the ontology, the fact that these two attributes are representing conflicting information (indeed, they represent a distinguishing property that causes the two classes to be disjoint) can hinder the ability to draw sound conclusions. What it is worth noting here, however, is that in order to detect inconsistencies automatically, a program must be able to understand that a pastry (instance) cannot be both savoury and sweet at the same time. Therefore, an automatic reasoner should not only be able to draw conclusions from inconsistent values but in first instance should be able to recognise the inconsistency because it recognises that the two concepts have been labelled by synonym terms.

This small, and in a way ad hoc example, is not totally unrealistic: the types of mismatches described in the example are likely to occur when merging ontologies that have been independently developed. From it we could draw an important conclusion. Even in a fictitious situation like the one described above, in order to detect the mismatches and the inconsistencies we have made use of semantics that was not explicitly represented in the ontologies, which means that merging the two concepts would have not been possible *without the intervention of a human more or less expert in the domain modelled in the ontology*. Although, at the current state of the art, automatic ontology merging seems quite an unrealistic target, there are many research efforts which have concentrated on semi-automatic ontology merging, as already seen in Section 3.5.3.

In the next section we survey approaches currently available. Then we illustrate a novel proposal to tackle this problem.

3.6 Ontological approaches to knowledge sharing

In Section 3.2 we have introduced the knowledge sharing problem. The literature concerning knowledge sharing is quite rich and many systems have dealt with this problem. Among them, we mention here as relevant resources: SIMS [Arens *et al.* 1996], InfoSleuth [Perry *et al.* 1999], COIN [Goh *et al.* 1994], KRAFT [Preece *et al.* 2001], and OBSERVER [Mena *et al.* 2000]. We have decided to concentrate here on the theoretical approaches that are behind these systems, and we have reviewed three of them, InfoSleuth [Perry *et al.* 1999], OBSERVER [Mena *et al.* 2000], and KRAFT [Preece *et al.* 2001], in Appendix A.

As pointed out many approaches to knowledge sharing use ontologies for two main tasks:

- To abstract the knowledge content of a resource from the way in which this knowledge is represented in order to overcome problems of language heterogeneity, on the assumption that the semantics are the same across the different knowledge sources;
- To share the common understanding about a domain, by identifying concepts that are shared by the different resources and that represent the reference concepts on which any communication has to be grounded. Identifying the shared concepts means not only isolating those concepts which have the same or similar names, but also those concepts whose content is similar despite different names or different concept descriptions, thereby solving heterogeneity due to ontological mismatches;

In the remainder of this section attention is focused on different solutions to the integration of heterogeneous resources with different ontologies, we illustrate the approaches that are presented in the literature and we propose a novel approach.
All approaches are based on the assumption that *Concepts can be shared between different resources if an appropriate "mapping" can be found that transforms a concept understood by one resource into a concept that is understood by another resource. This is the minimal requirement for two resources to share knowledge*.
In the literature three types of approaches to knowledge sharing are mentioned [Uschold *et al.* 1999]. Here we associate a broad meaning with the term *knowledge sharing*, to mean used by multiple informations systems, by multiple persons and in multiple contexts. Following Uschold and colleagues, we use the term in this sense and thus disregard the differences between *sharing*, *reuse*, and *exchange*, each of which can have a more specific meaning, as we have illustrated in Section 3.2. In the following subsections we survey the systems and the different approaches to knowledge sharing presented in the literature on the grounds of the role ontologies play in these approaches. Furthermore in Section 3.7.1 we propose a novel approach to sharing, based on an hierarchical structure of multiple shared ontologies,

each representing knowledge at a different level of abstraction. We believe that the approach we envisaged offers advantages in scalability and maintainability.

3.6.1 One-to-one approach

In this approach knowledge sharing is achieved because each resource makes its services available to the other resources in the architecture [Uschold *et al.* 1999]. If the resources are heterogeneous then a transformation step is required as well. This approach is based on some functions (or *point-to-point translators* in [Uschold *et al.* 1999]) that *directly* map from a source format to a target format. These functions are often called in the literature *mapping functions* or simply *mappings*: Concepts can be shared between different resources if an appropriate mapping function can be found that transforms a concept understood by one resource into a concept that is understood by another resource.

An *ontology mapping* is a partial or total function that specifies mappings between terms and expressions defined in a source ontology into terms and expressions defined in a target ontology [Visser *et al.* 1998]. These functions can also be (but not necessarily are) isomorphic, that is, if a mapping function exists from a resource A to a resource B this implies that the opposite mapping from the resource B to the resource A exists.

Ontologies might be possibly used in this approach to separate the abstract knowledge from the way it is implemented in the resource, and thus to make the mapping process easier. However, these functions can be determined without explicitly referring to an intermediate ontology which plays the role of neutral interchange format [Uschold *et al.* 1999]. This is the so-called *one-to-one* approach, where for each ontology (or resource) a set of mapping functions is provided to allow the communication with the other ontologies (resources). If an ontology is used to model the local knowledge then a *wrapper* might be used to convert the local concepts into a format supported by the local ontology. It is important to note here that in such an approach the ontologies involved do not model shared knowledge.

According to [Uschold *et al.* 1999] the *one-to-one* approach can be made to work and is a relatively low-cost and practical approach in the short-term. However, this approach is only suitable when the resources are quite stable in time, that is, they do not change frequently. In fact, each time a concept is introduced in a resource or the concept is used in a way different from before, then the corresponding mappings need to be changed. Moreover, inconsistencies can be undetected until they impact on some critical application. Finally, the effort required to build and maintain the mapping functions is quite big, especially in those cases where the knowledge available is not sufficient to determine a unique mapping of a concept in one resource into another one in a different resource. Indeed, such an approach would require in the worse case, that is if the mappings are not isomorphic, the definition of $O(n^2)$ mapping functions, if n ontologies are comprised in the structure. For this reason, this approach only seems feasible if there are only a few ontologies (resources). It also would not be very scalable because if a new resource is added to the structure this approach requires the definition of n new mapping functions.

The OBSERVER system [Mena *et al.* 1996], partially overcomes the aforementioned drawbacks by performing run time transformations, where a concept might map into a synonym, hypernym or hyponym concept. In this way, a shared ontology is virtually built at run time.

3.6.2 Single centralised ontology

This approach achieves the sharing of heterogeneous sources by committing to an overarching shared ontology modelling the concepts that are shared by the knowledge sources. This is based on the assumption that in order to be shared, knowledge resources must have some common understanding that partially covers the shared domain. That is, resources modelling conceptualisations on totally different domains are hardly able to share any knowledge, whereas different conceptualisations of the same or similar domains have more to share.

However, the shared ontology in this approach can play two roles: In the first case the ontology plays the role of a *neutral interchange format*, while in the second it permits the *neutral authoring* of shared knowledge [Uschold *et al.* 1999]. The following subsections examine each of these cases.

Neutral interchange format

The application resources are modelled according to their own ontology, using an ontology language, conceptualisation and explication that are local and not shared. In the single centralised ontology approach, the integration of heterogeneous sources is performed by identifying the concepts which are shared by all the knowledge sources and locating them in a shared ontology. The shared concepts are defined in a way such that they can be easily mapped from and into the original concepts with no or little information loss. For this reason, the shared ontology is written in a sufficiently expressive neutral format, and then the local resources are provided with two-way transformations that perform the mappings from local to shared concepts and vice-versa. Many architectures to integrate resources comprise a single shared ontology, examples are given by InfoSleuth, [Bayardo *et al.* 1997] and by the KRAFT architecture [Preece *et al.* 2001].

A crucial point to be considered when analysing this approach is the strategy to follow when designing the shared ontology or, using Uschold's terminology, when designing the neutral format. This can be thought as a continuum where on one end of the spectrum we have a neutral interchange format that is very expressive (that is the expressiveness covers the expressiveness of the target languages). In other words, the concepts in the shared ontology are the union of all concepts in the local ontologies. In this way, little or no information is lost when performing the mappingss between local and shared ontologies. On the other end of the spectrum we have the opposite approach, that is the expressive power of the neutral format could be the lowest common denominator of the local formats, in other words, the concepts in the shared ontology are the intersection of all the local concepts. This makes the creation of mappings between local and shared ontologies easier, but local ontologies need to be written bearing in mind the transformations that are to be performed and the risk of losing a great amount of information in the transformation process is high. What happens in practice is that some intermediary point between these two extremes is chosen, mostly relying on the experience and the domain knowledge of the ontology designers.

The extent to which this approach is to be considered conceptually realistic has been subject of discussion [Shave 1997]. It has definitely the *potential* advantages of permitting the knowledge sources to be kept autonomous from the process of sharing knowledge and to reduce the number of mappings involved in the transformations from $O(n^2)$ to $O(n)$. But, as Uschold highlights in [Uschold *et al.* 1999], these are only *potential* advantages which may or may not be be possible to realise.

The drawbacks of dealing with a single shared ontology are similar to those of any standard (see also: [Visser and Cui 1998]). Often, standards are not very convenient to use since they have to be suitable for all potential uses. Also, the task of defining such standards is often lengthy and complicated. It is often necessary to find a trade-off concerning the expressiveness: the more expressive is the knowledge to be shared and the more difficult it is to write the translators. Moreover, committing to a standard restricts the degree of heterogeneity that may exist between those using the standards, and, last, but not least, standards - by their nature - resist changes, partly due to the aforementioned reasons.

Neutral authoring

Neutral interchange format is just one of the roles that a shared ontology can play when sharing knowledge. Another approach is to author the shared ontology in a neutral format. The difference with the previous approach is that in the neutral interchange format approach the ontologies to be shared may maintain their autonomy of language, syntax, knowledge representation paradigm, and they are transformed into neutral interchange format for sharing purposes only, whereas with the neutral authoring, a neutral format is enforced on the ontologies at design stage, so only semantically heterogeneous views are allowed [Uschold *et al.* 1999]. Both in the case of the neutral interchange format and in the case of the neutral authoring, designing and building the shared ontology is quite a time-consuming task. However, different considerations can be made in either cases. A neutral interchange format is usually meant as an aid for semi-automatic knowledge sharing. So the language used for the neutral interchange does not have to be necessarily human readable. With the neutral authoring approach, the shared knowledge is authored in a neutral format which could be human friendly (for example a restricted subset of natural language) with the aim to improve human knowledge sharing. Another design difference between the two approaches is motivated by the fact that transformations in the case of neutral authoring are in one direction only, and so designing a language which is the lowest common denominator might be preferred in such a case [Uschold *et al.* 1999].

One of the most used neutral authoring tools is Ontolingua [Farquhar *et al.* 1997]. Ontolingua was designed as a tool to support the collaborative building of ontologies by providing a neutral authoring language. It is equipped with a set of translators in the most common languages for knowledge bases such as Loom [MacGregor 1991], Clips, Prolog and many others. However, these translators can only perform syntactic transformations, and do not deal with any kind of language or ontological heterogeneity.

To summarise, this approach might prove cost-effective in limited situations while it is not suitable for general cases, due to the inherent difficulty of the mapping problem.

3.7 A novel proposal to knowledge sharing

Having discussed the various approaches for sharing heterogeneous sources, in the previous section we now focus our attention on a structure of multiple shared ontologies as presented in [Visser and Tamma 1999]. This architecture aggregates multiple shared ontologies into clusters, so as to obtain a structure that is able to reconcile different types of heterogeneity and is also intended to be more convenient to implement and give better prospects for maintenance and scaling. We discuss here the feasibility of semi-automatic ontology clustering in order to obtain such a structure. More particularly, we have investigated the different similarity measures that can be used in order to build clusters of ontologies. In contrast to an approach in which all resources share one body of knowledge here we propose to locate shared knowledge in multiple but smaller shared ontologies. This approach is referred to as ontology-based resource clustering, or shortly, ontology clustering [Shave 1997]. Resources no longer commit to one comprehensive ontology but they are clustered together on the basis of the similarities they show in the way they conceptualise the common domain. Therefore each cluster can be thought of as a micro-theory shared by all the resources to conform to that cluster. Each micro-theory is in turn generalised and they are all eventually generalised by the top-level ontology which is a standard upper ontology like the *Upper-Cyc* [Lenat 1995a]. This approach is analogous to *modularisation* in software engineering and is thought of having the same advantages, which are:

- **Modularity/separability**: Each cluster is like a module in software engineering and represents a specific aspect of the domain;
- **Composability**: Different clusters are composed by generalising the concepts that are common to them. This is the first step to permit heterogeneous resources to communicate;

- **Scalability**: The addition of a new resource to the architecture requires only the production of the mapping rules between the ontology associated to the new resource and the cluster to which this resource belongs;
- **Impact of change minimisation**: If a concept description needs to be changed only the mapping rules between the updated ontology and the cluster to which this ontology belongs need to be rewritten;
- **Division of ontology authoring efforts**: Ontologies composing a cluster do not need to be authored by the same people as long as their concepts can be mapped into the concepts of the cluster.
- **Accommodation of diverse formalisations**: A cluster can be comprised of ontologies representing different formalisations of the same domain, such as different temporal ontologies.

This approach has not been tested yet, therefore we can only foresee some disadvantages:

- There is no methodology which permits us to build the structure of ontology clusters;
- Complexity of the first order clustering problem from the machine learning viewpoint;
- Lack of a semantic-sensitive similarity measure that could be used to assess similarity among concepts;
- Lack of tools that can support the building of the ontology clusters.

3.7.1 Ontology clusters

Ontology clustering is based on the similarities between the concepts known to different resources, where each resource represents a different aspect of the domain knowledge. We assume that the ontologies modelling the resources are consistent, non-redundant, and well structured. We also assume that the ontologies have been built with a methodology that includes a formal evaluation step, such as Methontology [Fernández-López *et al.* 2001]

Since our resources need to communicate in a sensible fashion they are all supposed to be familiar with some high level concepts. We group these concepts in an ontology rooted at the top of the hierarchy of ontologies. As it describes concepts that are specific to the domain and tasks at hand we refer to this ontology as the application ontology (following Van Heijst and colleagues, [van Heijst *et al.* 1997]. These concepts are reusable within the application but not necessarily outside the application. The concept definitions in the application ontology are chosen from an existing top-level ontology, which in our case is WordNet [Miller 1990]. The application ontology thus contains a relevant subset of WordNet concepts. For each concept one or more senses are selected, depending on the domain. If some resources share concepts that are not shared by other resources then this leads to the creation of two (or more) sibling ontologies. Each sibling is a consistent extension of its parent ontology, but heterogeneous with respect to its peers. We do not pose any restriction to the types of heterogeneity that can affect the ontologies.

A cluster is referred to as a group of consistent ontologies (possibly one) in our structure and is described by an ontology which is shared by those composing the cluster. Both ontology clusters and ontologies within each cluster are organised in a hierarchical fashion where each sibling cluster specialises the concepts that are in its parent cluster. However, while multiple inheritance is permitted within the ontologies, it is not permitted between ontologies, therefore the structure of clusters is a tree. In this structure, the lower level clusters have more precise concept definitions than the higher levels, making the latter more abstract.

Clusters are linked by *restriction* or *overriding* [Visser and Cui 1998] relations, that is concepts in one parent ontology are inherited by its children cluster, but overriding is permitted. The link between the resources and the local ontologies, on the

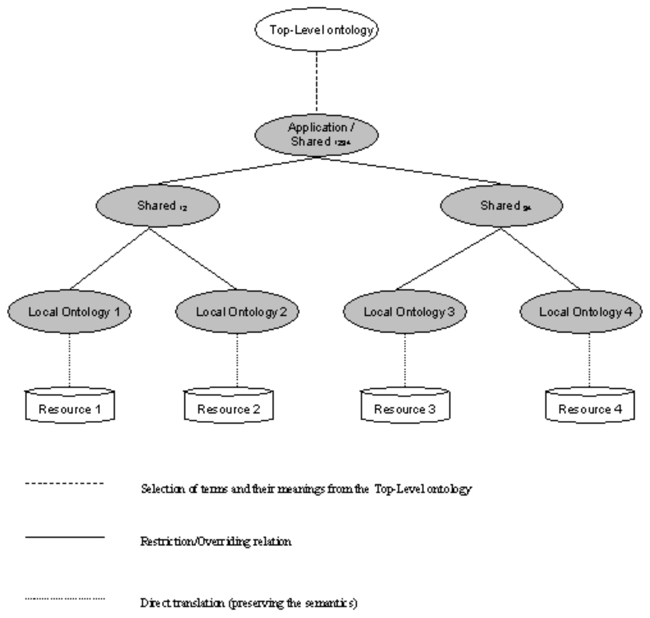
Top-Level ontology
Application / Shared 1234
Shared 12
Shared 34
Local Ontology 1
Local Ontology 2
Local Ontology 3
Local Ontology 4
Resource 1
Resource 2
Resource 3
Resource 4
Selection of terms and their meanings from the Top-Level ontology
Restriction/Overriding relation
Direct translation (preserving the semantics)

Figure 3.3: The hierarchy of multiple shared ontologies

other hand, is different, and is a *mapping relation* as defined in [Visser and Cui 1998], that is a function preserving the semantics. Finally, the relation between the top-level ontology and the application one is a simple *Subset/Superset* relation as described in [Visser and Cui 1998].

Figure 3.3 illustrates an example of this structure. Since different siblings can extend their parent cluster concepts in different ways the cluster hierarchy permits the co-existence of heterogeneous (sibling) ontologies. Figure 3.3 illustrates this particular structure, where $Local\ Ontology_1$, $Local\ Ontology_2$, $Local\ Ontology_3$, and $Local\ Ontology_4$ are the local ontologies, $Shared_{12}$ is the ontology shared by the local ontologies 1 and 2. Analogously $Shared_{34}$ is the ontology shared by the local ontologies 3 and 4. $Shared_{1234}$ indicates the ontology shared by the two below that is $Shared_{12}$ and $Shared_{34}$, and in this example is the application ontology itself, here denoted by *Application Ontology*. If some ontologies share concepts that are not shared by other ontologies then there is a reason to create a new cluster. A new ontology cluster here is a child ontology that defines certain new concepts using the concepts already contained in its parent ontology. Ultimately, ontologies are likely to have concepts that are not shared with any other ontology. In our ontology structure, we then create a separate, domain-specific ontology as sub ontology of the cluster in which the ontology resides. We refer to these ontologies as local ontologies. The local ontologies are the leaf nodes of our ontology hierarchy. In each of the ontologies in the structure, concepts are described in terms of attributes and inheritance relations holding in the ontology's structure. Concepts are hierarchically organised and the inheritance (with exceptions) allows the passing down of information through the hierarchy. Multiple inheritance is only permitted within the ontologies.

Concepts are expressed in terms of *inherited* and *distinguishing* attributes. Inherited attributes are those expressing the similarities between a parent concept and its siblings (the parent concept can be defined in the ontology itself or in a parent ontology). They describe the main characteristics of a concept that are also present in its sub-concepts. A concept that specialises a more general one inherits all the attributes from its parent concept.

To the set of inherited attributes other attributes are added to distinguish the specific concept from the more general one. These attributes describe the characteristic differences between a concept and its siblings. The distinguishing attributes are used to map concepts from a source ontology into a target ontology preserving the meaning of the concept.

3.7.2 Communication between resources

In the ontology structure presented in Section 3.7.1 communication between resources is performed via mapping functions (Section 3.3). In this structure mappings can be either partial or total functions and are not necessarily isomorphic.

The remainder of this section outlines how we envisage that communication between the resources in the ontology structure is performed. Mappings are encoded in functions mapping concepts between the ontologies composing the structure, thus transforming (because semantics might not be preserved) concepts from one ontology, possibly repeatedly, into its child or into its parent ontology. Concepts belonging to one of the local ontologies are mapped into concepts of another local ontology via one or more shared ontologies.

In the reminder of this section we will use the term source ontology to denote the ontology containing the concept that is to be mapped, whereas we use the term target ontology to denote the ontology the concept has to be mapped into.

The ontologies in the structure are hierarchically organised, and for this reason transforming concepts from the source ontology into concepts in the target ontology may generally consist of two types of mapping steps. The first type is generalisation (from the concept to its hypernym in the same or in a parent shared ontology). The second type is specialisation (from the concept in the parent shared ontology to its hyponym in the same or in another ontology). However, the mere mapping of a concept through a generalisation and a subsequent specialisation is not enough; indeed such a mapping is guaranteed to preserve the meaning only if the concept to map has a synonym in the local target ontology. If this is not the case, the concept will be mapped into a more general one, and thus it will be an approximation. This is what happens in the SIMS project [Arens *et al.* 1996] where a query is reformulated as the union of its more general concepts using the relationship holding between a class of concepts and its super-class. To preserve the meaning, however, some constraints can be added.

The mapping between local ontologies can be summarised by the following steps:

a) The concept that needs to be mapped is identified. This step requires a deep understanding of the semantics associated with the concepts and, therefore, it can only be performed in a semi-automatic way. A human expert is needed to confirm that the concepts selected are actually semantically related;

b) Once identified, the concept is mapped into the terms of the shared ontology immediately above the source ontology. If a direct mapping does not exist the first hypernym of the concept is found such that a mapping exists between the hypernym and a concept in the shared ontology immediately above. The same mapping process is applied to all the concepts in the target ontology;

c) The hypernym of the concept is then located in the shared ontology;

d) The attributes of the concept in the source ontology are compared with the attributes of the hypernym just found to select the distinguishing features;

e) Then the concept expressed in terms of the shared ontology, (that is, the relationships holding between concepts in the structure are identified) together with its distinguishing attributes is passed to the parent shared ontology;

f) If in the target local ontology there is a concept that is a specialisation of the one passed to the shared ontology, then for this local concept a mapping can be defined between the original local concept and the one just selected. If not, the procedure is recursively applied, climbing up a level to the more general shared ontology.

This kind of mapping obtained by these generalisation and specialisation steps is effective only if the source and the target concepts have a common ancestor that is not too high in the hierarchy, otherwise the generalisation steps can lead to a too general ancestor. In this latter case, the information loss due to the generalisation is too high, and the mapping obtained might be trivial.
To avoid the loss of information that is intrinsic to a generalisation, attributes and relations linking concepts play a crucial role. In fact they not only allow the identification of the hypernym of a concept (either in the same or in a shared ontology) but they also allow us to “attach” some characterising information to each concept thus giving a distinction between the concept itself and its parent. This type of information is modelled in the distinguishing attributes.

This way of mapping a concept A into a concept B is based on the idea of finding the intersection of the features (represented by the attributes) describing the two concepts. These should be the attributes describing the closest ancestor in the hierarchy. Then, the mapping is performed by transforming the source concept into the closest ancestor and passing the distinguishing attributes, and then, possibly, by specialising the ancestor in the target ontology. This specialisation step involves identifying the child concept in the target ontology, and then creating a mapping function that maps the source concept into the target by taking into account the distinguishing attributes of both concepts. This process requires a careful semantic analysis of both concepts and their attributes and therefore cannot be fully automatised.
Relationships should be mapped by hand, since they might require some kind of semantic transformation in order to be maintained in the cluster.

3.7.3 Assessing similarity between concepts to build the ontology clusters

The structure of ontology clusters introduced in Section 3.7.1 builds on the ability of identifying similar concepts in different ontologies. Identifying which concepts are similar and assessing the degree of semantic similarity between them are, thus, two essential steps in the process of building ontology clusters. However, assessing the similarity between concepts in diverse ontologies is not a trivial task because of the heterogeneity that can affect concepts and their descriptions.
The problem of assessing semantic similarity has received much attention in the artificial intelligence field [Quillian 1968], [Collins and Loftus 1975]. In these efforts, ‘semantic similarity’ refers to a form of semantic relatedness using a network representation. In particular, Rada and colleagues [Rada *et al.* 1989] suggest that similarity in semantic networks can be assessed solely on the basis of the IS-A taxonomy, without considering other types of links. One of the easiest ways to evaluate semantic similarity in taxonomies is to measure the distance between the nodes cor-

responding to the items being compared, that is the shorter the path between the nodes, the more similar they are. This way of assessing semantic similarity might be useful for semantic networks, but has the major drawback of computing the semantic distance between concepts which have a common ancestor, and thus it is not suitable for assessing the similarity of heterogeneous local ontologies that have to be clustered. Moreover, this method does not fully exploit the structure of the concept's representation, since it does not take into account the concept's description in terms of attributes, relationships, etc. thus making it more sensitive to synonym and homonym heterogeneity.

In fact, only a few efforts are addressing the problem of facilitating the (semi) automatic reconciliation of different ontologies, and they have been mainly developed for merging different ontologies. Reconciling different ontologies involves finding all the concepts in the ontologies which are similar to one another, determininig what the similarities are, and either changing the source ontologies to remove the overlaps or recording a mapping between the sources for future reference [Fridman Noy and Musen 2001]. Similarity in these efforts is mainly lexical and not semantic. Most systems for ontology merging rely on dictionaries to determine synonyms, common substrings in the names of concepts, and concepts whose documentation share many unusual words. They do not take into account the internal structure of concept representation and the structure of the ontology.

The ontology merging environment Chimaera [McGuinness *et al.* 2000] partially considers the ontology structure in that it assesses similarity between concepts also on the grounds of the subclass-superclass relationship and the attributes attached to the concept. Anchor-PROMPT [Fridman Noy and Musen 2001] reconciles ontologies by finding *matching terms*, that is, terms from different source ontologies that represent similar concepts. Anchor-PROMPT assesses both lexical and semantic matches exploiting the content and structure of the source ontologies, and the user's actions in merging the ontologies. By content and structure of the source ontologies we mean that names of classes and slots, subclasses, superclasses domains and ranges of slot values are used to assess the similarity (Anchor-PROMPT is based on the Protege [Fridman Noy *et al.* 2000] knowledge model, which is frame-based) . However, the method used in Anchor-PROMPT is based on the assumption that if the ontologies to be merged cover the same domain, the terms with the same name are likely to represent the same concepts. Such an assumption is a good rule of thumb, but does not take into account cases of heterogeneity among source ontologies. In fact, similar concepts might have different names, and be described by attributes with different names (*synonym terms heterogeneity* as defined in Section 3.5.2). Moreover, the hierarchical structure of the source ontologies might be different, thus a certain subclass-superclass relationship holding in one source ontology might not hold in the others.

In [Rodríguez and Egenhofer 2002] the authors propose a method for assessing semantic similarity which takes into account the differences in the level of explicitness and formalisation of the source ontologies specifications. Also this method does not require an *a priori* shared ontology. The similarity between concepts in different sources ontologies is assessed by a matching process over synonym sets (thus accounting for lexical similarity), semantic neighborhood, and distinguishing features. The use of distinguishing features to assess similarity enables the authors not only to handle binary similarity measures, typical of lexical similarity (two terms are either similar or not), but also to consider gradients of similarity. This is based on the assumption that, in order for concepts to be considered similar, they should present some common features. By assessing similarity on the grounds of the distinguishing and common features, this method accounts for those problem of synonym terms heterogeneity that can affect both concepts and attributes.

In [Rodríguez and Egenhofer 2002] the authors argue that from an analysis of different feature-based models for semantic similarity has emerged the necessity to account for the context dependence of the relative importance of distinguishing features and asymmetric characteristic of similarity assessments. Properties that distinguish sibling concepts from their parent are called *distinguishing properties*.

The method proposed by Rodríguez and Egenhofer is based on Tversky's [Tversky 1977] matching process, which produces a similarity value that depends on both common and different characteristic. A particular property of the matching model is that it is not a metric, therefore the usual properties for metrics (i.e. minimality, symmetry, and triangle inequality) [Esposito *et al.* 2000] do not have to be satisfied. The asymmetric evaluation of similarity is important to have similarity evaluations that are 'tuned' to people judgements. For example, cognitive studies have shown that the perceived similarity between a class and its superclass is greater than the perceived similarity between a class and it subclass.

The novelty of Rodríguez and Egenhofer's approach is that, in order to take into account common and distinguishing features it extends the usual ontology model and it includes also an explicit specification of the features. By features the authors collectively mean the set of *functions, parts* and *attributes*. *Functions* represent the intended purpose of the instances of the concept they describe. For example the function of a university is to educate. *Parts* are the structural element of a concept, and they do not necessarily coincide with those expressing the *part-of* relationship, while *attributes* correspond to additional characteristics of a concept that are considered to be neither parts nor functions.

The approach proposed by Rodríguez and Egenhofer to assess similarity between concepts is based on the enriched description of concepts they propose. Of course it could be argued that enriching the concepts' structure by distinguishing between parts, functions and attributes can give rise to the articulation of new types of mismatches associated with the classifications of features. However, the authors claim that the advantages of enriching the concept's structure, namely a matching process that compares corresponding characteristics of concepts, and the ability to distinguish different aspects of the context, modelled by the features, overweighs the possible disadvantages deriving from a higher number of mismatches.

We believe that Rodríguez and Egenhofer's approach to assess semantic similarity raises an important issue, which is that, in order to be able to have a better assessment of semantic similarity (that gives also gradients of similarity and not only a binary function) it is necessary to provide a richer description of the structure of the concepts in the source ontologies. However, we believe that the distinguishing features proposed in [Rodrı́guez and Egenhofer 2002] overlap with the semantics already modelled by some relationships, such as *part-of*.

For this reason, in Chapter 4, we describe an enriched ontology model, where the concept descriptions are enriched with metaproperties characterising the behaviour of attributes, and more precisely, the behaviour of attributes over time, the modality (i.e., the degree of applicability of the property to subconcepts, prototypical and exceptional properties). These metaproperties of attributes should better characterise a concept, so to get a better understanding of the concept *as it is used in a specific context*, and to derive the formal meta-properties holding for the concept and described in Section 2.5.1.

3.8 Chapter summary

This chapter has been devoted to presenting an overview on the problem of knowledge sharing and reuse and, particularly focussing attention on the possible approaches that use ontologies to solve the problem of sharing knowledge. We have first introduced knowledge sharing and reuse, by giving an overview of the state of the art. The terminology in this area is sometimes conflicting, and so we have devoted a section to defining the terminology that we use in this thesis.

Once having clarified the terminology, we have described multi-agents architectures. Software agents are not central to this thesis, although they cannot be ignored in the context of knowledge sharing and reuse, since many systems for knowledge sharing exploit agent architectures. The agent paradigm becomes relevant also to discuss the problems that can hinder sharing knowledge between heterogeneous resources. In order to do so we have analysed and classified the different types of heterogeneity that can affect resources developed for different purposes.

After introducing the different types of heterogeneity we have then illustrated the diverse approaches presented in the literature, with their main advantages and drawbacks. To these well-known approaches we have added a novel proposal for knowledge sharing that we have named *ontology clustering*, which motivates the ontology model that is object of this thesis and that is presented in next chapter. This approach is based on an hierarchical structure of multiple shared ontologies, that permits knowledge at different levels of abstraction to be shared. We have presented the structure of multiple shared ontologies and we have briefly illustrated how mappings are performed in this novel approach to knowledge sharing, called ontology clustering. Ontology clustering relies on the ability of assessing semantic similarity between concepts, and to assess different degrees of similarities. We have reviewed different approaches that have been presented in the literature, and we have concluded that the ones developed to assess similarity for semantic networks are not suitable, mainly because they require that a shared ontology has already been built. We have followed the approach by [Rodríguez and Egenhofer 2002], which uses the Tvarskian matching function to assess semantic similarity. In order to use such matching function, the approach proposes to extend the concept descriptions by adding so called *distinguishing features*. We have decided not to adopt the same distinguishing features proposed in [Rodríguez and Egenhofer 2002], mainly because the ones they propose seem to overlap semantics already modelled by some relationships, such as *part-of.* Our proposal is to augment the concept descriptions with attribute metaproperties (Mutability, Mutability frequency, Reversible mutability, Event mutability, Modality, Prototypicality, Exceptionality, Inheritance, Distinction) which describe the behaviour of attributes in the concept definitions, and therefore can help in detecting the different types of heterogeneity. Moreover, the proposed attribute metaproperties can be used to determine the concept metaproperties (that is, *identity, unity, rigidity* and *dependence* [Welty and Guarino 2001]) that hold for a concept. In this way the concept description we propose in the next chapter can support the OntoClean methodology presented in Section 2.5.1

The following chapter describes the novel ontology model that includes the attributes metaproperties.

Chapter 4

A conceptual metamodel to support ontology clustering

4.1 Introduction

In Chapter 3 we have introduced a structure of multiple shared ontologies that is thought to be more scalable and maintainable. One of the drawbacks of such a structure, though, is that it requires the matching of concepts from the local resource ontologies into the corresponding shared concepts. The process of recognising candidate concepts to be merged depends heavily on semantics, and is quite demanding to perform in that it requires a deep knowledge of the domain in order to determine which concepts are similar. For this reason the process is usually performed by hand, since presently there are no prospects for a full automatisation. It seems, however, potentially feasible to provide a semi-automatic process, where a computer identifies possible candidates, but the final choice is left to the domain experts. If the ontologies to be merged are built by different development teams for different purposes, as assumed in previous chapter, it is necessary to provide the knowledge engineers who are in charge of designing the shared ontologies with a deep understanding of the ontologies to be integrated. Moreover, from the perspective of semi-automatic ontology merge, providing the algorithm performing the ontology merging with enriched semantics of the concepts gives better prospects for the inclusion of concepts that are truly semantically related in the list of candidates.

In this chapter we introduce and motivate an extended conceptual model for ontologies which explicitly represents attribute metaproperties. By *conceptual model* we mean the knowledge engineer's evolving conception of the domain knowledge. It is the knowledge that actually determines the construction of a formal knowledge base. A conceptual model is an intermediate design construct, a template to begin to constrain and codify human skill, it is neither formal nor directly executable on a computer [Luger 2002]. A *conceptual metamodel* is a *meta* level on the conceptual model which describes how the elements of the conceptual model are used to describe the objects of the world.

In the following section (Section 4.2) we present our proposal to extend the conceptual metamodel for ontologies by adding a set of metaproperties for attributes. Section 4.3 and subsections analyse three kinds of problem which can benefit from the semantic information modelled by the metaproperties we add. We then describe how this enriched conceptual model can be instantiated in a frame-based knowledge model (Section 4.4) which we describe in detail in Section 4.5, then we discuss the expressive power of this model (Section4.6) and we relate the semantic information to the motivations illustrated in Section 4.3 (Section 4.7). Finally, we summarise the chapter contributions in Section 4.8.

4.2 Metaproperties for attributes: the conceptual metamodel

In the conceptual metamodel which is the object of this section concepts are described by their characterising features. We also describe the metaproperties holding for these characterising features, by describing the behaviour they show in defining a concept. We have called these metaproperties as *attribute metaproperties*, because a concept's characterising feature is usually modelled by associating a set of values with an attribute.

This conceptual metamodel is based on the metaproperties of formal ontological analysis that have been presented in Section 2.5.1 and it results from enriching the usual conceptual model (described in Section 2.3) with a set of metaproperties for attributes (namely, *Mutability, Mutability Frequency, Reversible Mutability, Event Mutability, Modality, Prototypicality, Exceptionality, Inheritance, Distinction*) which precisely characterises the concept's properties and expected ambiguities, including which properties are prototypical of a concept and which are exceptional, the behaviour of properties over time and the degree of applicability of properties to subconcepts. This enriched conceptual model permits a precise characterisation of what is represented by class membership mechanisms and helps a knowledge engineer to determine, in a straightforward manner, the metaproperties holding for a concept.

The set of metaproperties of attributes we define in this thesis might be helpful to deal with ontology heterogeneity problems in two ways. On the one hand the model complements the set of formal ontological properties proposed in [Welty and Guarino 2001], namely *Identity, Unity, Rigidity,* and *Dependence* (see Section 2.5.1). Our set of metaproperties can guide in assigning the concept metaproperties defined by Guarino and Welty to concepts, and the process of assigning the metaproperties depends on the concept definitions in terms of attributes. This might be particularly useful when knowledge engineers need to assign formal properties to ontologies that they have not designed.

On the other hand, this conceptual metamodel for ontologies facilitates a better understanding of the concepts' semantics. Currently ontology merging is performed by hand based on the expertise of the knowledge engineers and on the ontology documentation. Even in this case the ontology model we propose can prove useful by providing a characterisation of the properties, which can help to identify semantically related terms.

In the remainder of this section we briefly describe the metaproperties for attributes on which our conceptual metamodel is based.

- Behaviour of concepts' properties over time: The metaproperties which model the behaviour of the attributes over time are:

  - *Mutability*, which models the liability of a concept's property to change, a property is mutable if it can change during the concept's lifetime;
  - *Mutability Frequency*, which models the frequency with which a property can change in a concept's description;
  - *Event Mutability*, which models the reasons why a property may change;
  - *Reversible Mutability*, which models reversible changes of the property.

  These meta-properties describe the behaviour of *fluents* over time, where the term *fluent* is borrowed from situation calculus to denote a property of the world that can change over time. Modelling the behaviour of fluents corresponds to modelling the changes in properties that are permitted in a concept's description without changing the essence of the concept. Describing the behaviour over time also involves distinguishing properties whose change is *reversible* from those whose change is *irreversible*.
  Property changes over time are caused either by the natural passing of time or are triggered by specific event occurrences. We need, therefore, to use a suitable temporal framework that permits us to reason with time and events. In Section 4.7.1 we chose *Event Calculus* [Kowalski and Sergot 1986] to accommodate the representation of changes.
  We further discuss the behaviour of attributes over time in Section 4.7.1.

- Modality: The term modality is used to express the way in which a statement is true or false, which is related to establishing whether a statement constitutes a *necessary truth* and to distinguish necessity from possibility [Kripke 1980]. The term can be extended to qualitatively measure the way in which a statement is true by trying to estimate the number of possible worlds in which such a truth holds. This is the view we take in this work, by denoting the degree of confidence that we can associate with finding a certain world with the meta-property *modality*. The additional semantics encompassed by this metaproperty is important for reasoning with statements that have different degrees of credibility. Indeed there is a difference in asserting facts such as ‘Mammals give birth to live young’ and ‘Birds fly’, the former is generally more believable than the latter, for which many more counterexamples can be found. The ability to distinguish facts whose truth holds with different degrees of strength is important in order to find which facts are true in every possible world and therefore constitute *necessary truth*. We further elaborate this point in Section 4.7.3

- Prototypes and exceptions: We partially take the cognitive view of prototypes and graded structures, which is also reflected by the information modelled in the meta-property *modality*. In this view all cognitive categories show gradients of membership which describe how well a particular subclass fits people’s idea or image of the category to which the subclass belong [Rosch 1975]. Prototypes are the subconcepts which best represent a category, while exceptions are those which are considered exceptional although still belonging to the category. In other words all the sufficient conditions for class membership hold for prototypes. For example, let us consider the biological category *mammal*: a *monotreme* (a mammal who does not give birth to live young) is an example of an exception with respect to the property of giving birth to live young. We further discuss these metaproperties in Section 4.7.4.

- Inheritance and Distinction: *inherited* meta-properties regard those properties that hold because inherited from an ancestor concept, they may be overruled in the more specific concept in order to accommodate inheritance with exceptions. *Distinguishing* are those properties that permit us to distinguish among siblings of a same concept. In other words a distinguishing property ϕ is a property such that $\diamond\exists x\ \phi(x) \wedge \diamond\exists x\ \neg\phi(x)$, that is there is possibly something for which the property ϕ holds, and there is possibly something for which the property does not hold, and these are neither tautological nor vacuous [Welty and Guarino 2001]. Distinguishing properties might cause disjoint concepts in the ontology's taxonomic structure. This point is further discussed in Section 4.6.

4.3 Extending the conceptual metamodel

The interest in designing ontologies that can be easily integrated and provide a base for applying reasoning mechanisms, as pointed out in Chapter 3, has stressed the importance of suitable conceptual models for ontologies. Indeed, it has been made a point that the sharing of ontologies depends heavily on a precise semantic representation of the concepts and their properties [Fridman Noy and Musen 1999, McGuinness 2000, Tamma and Bench-Capon 2000].
The motivation for defining a conceptual metamodel, which assigns attribute metaproperties with an ontology conceptual model, draws on the following arguments which we discuss in the remainder of this section:

- To represent concepts properties and help in determining concept metaproperties, thus facilitating the ontological analysis;
- To make ontological commitments explicit by representing also the hidden assumptions made in the conceptualisation;
- To disambiguate between concepts that seem similar both when merging ontologies and when reasoning with shared knowledge;
- To better understand the concepts that are in the domain, by:
  - knowing what can sensibly be said of a thing falling under a concept,

- recognising which properties are prototypical for class membership and which are the permitted exceptions,
- distinguishing on what extent an arbitrary member of a class conforms to the prototype, understanding how and which properties change over time.

4.3.1 Nature of ontologies

The first argument is based on the nature of ontologies as views on a particular domain. Ontologies *explicitly* define the type of concepts used to describe the abstract model of a phenomenon and the constraints on their use [Studer *et al.* 1998]. An ontology is an *a priori* account of the objects that are in a domain and the relationships modelling the structure of the world seen from a particular perspective. In order to provide such an account one has to understand the concepts that are in the domain, and this involves a number of things. It involves knowing what can be sensibly said of a thing falling under a concept. This can be represented by describing concepts in terms of their properties, and by giving a full characterisation of these properties. Thus, when describing the concept *Bird* it is important to distinguish that some birds fly and others do not. A full understanding of a concept involves more than this, however: It is important to recognise which properties are *prototypical* [Rosch 1975] for the class membership and, more importantly, which are the permitted exceptions. There are, however, differences in how confident we can be that an arbitrary member of a class conforms to the prototype: it is a very rare mammal that lays eggs, whereas many types of well known birds do not fly.

Understanding a concept also involves understanding how and which properties change over time. This dynamic behaviour also forms part of the domain conceptualisation and can help to identify the *metaproperties* holding for the concept's properties.

It might be argued that this kind of of knowledge has not an *ontological* nature, but rather an *epistemic* one, and to some extent we do agree with this criticism. But ontologies should provide an actual account of a specific viewpoint on a domain. We believe that the ability of ontologies to facilitate the sharing and reuse of knowledge and reasoning with the instantiation of this knowledge can be improved if the formal metalevel of the description is complemented by a richer concept description.

Ontology is already used more broadly in computer science than philosophy: if we need also to include epistemic notions to resolve the issues that computer science ontologies are supposed to address, we should not be stopped by any consideration of ontological purity. Indeed, it has already been stated in Chapter 2 (Section 2.2) that there is a clear distinction between the *philosophical notion of ontology* and the *notion of ontology in artificial intelligence* (AI), and thus in computer science. The philosophical notion regards an ontology as a particular system of categories accounting for a certain vision of the world, and this system of categories remains always the same, independently of the language used to describe it. In artificial intelligence, an ontology is regarded as *an engineering artifact*, which is constituted by a specific *vocabulary* describing a *specific reality*, and by a set of explicit assumptions accounting for the *intended meaning* of terms in the vocabulary [Guarino 1998]. Ontologies, according to this notion, range in a spectrum where formal ontologies are at one end, while something close to knowledge bases or even simple taxonomies of terms is at the other end. For example, in some e-commerce applications, ontologies can just be taxonomies of terms, such as the one used in *Yahoo!*, which has no attributes describing the concepts and no relationships between concepts and no axioms [Lassila and McGuinness 2001].

When we consider ontologies from a pure philosophical perspective, they are an a priori description of what constitutes necessary truth in all possible worlds [Kripke 1980]. It is this formal posture on ontologies that makes it possible to add to ontologies a metalevel of description and thus to reason about metaproperties [Guarino and Welty 2000c].

Our view on ontologies, given that they make a resource conceptualisation of the domain explicit, is somewhere in the middle of this spectrum: ontologies should provide sufficient information to enable knowledge engineers to have a full understanding of a concept as it is in the domain (that is in the real world), but should also enable knowledge engineers to perform a formal ontological analysis on these concepts. If ontologies are seen in this perspective, then the boundary between what is to be considered ontological knowledge and what is epistemic knowledge becomes blurred.

The reason for adding the attribute metaproperties to the usual conceptual model, making even more demanding the process of building ontologies from scratch, is that we believe that ontologies should be compatible with an a priori account of necessary truth in all the possible worlds but also provide some information on the actual world and all the worlds accessible from it. It should state not only what is necessarily true implicitly, but also what is seen as necessarily true from a particular standpoint.

The enriched semantics that characterises the conceptual metamodel we propose in this thesis permits us to deal with mismatches that can become apparent when merging ontologies independently developed, or when reasoning with shared knowledge, as illustrated is the next two subsections.

4.3.2 Merging diverse ontologies

The second argument concerns the integration of ontologies. The ability to merge ontologies is essential to build the structure of multiple shared ontologies described in Section 3.7.1. In fact, merging ontologies involves identifying semantically overlapping ones and creating a new one, usually by generalising the overlapping concepts. This new concept inherits all the (compatible) properties of the originals and so can be easily mapped into each of them. Newly created concepts inherit properties, usually in the form of attributes, from each of the overlapping ones. However, there are cases, as highlighted in [Welty and Guarino 2001], in which recognising overlapping concepts is not sufficient to guarantee that a suitable generalising concept (expressing the shared viewpoints) can be found.

One of the key points for merging diverse ontologies is providing methodologies for building ontologies whose taxonomic structure is “clean” (that is, not very tangled) in order to facilitate the understanding, comparison and integration of concepts. Indeed, as we already mentioned in Section 3.7.1 we assume in this thesis that the individual ontologies to be merged are sound and little, if at all, no tangled. We have already described in Section 2.5.1 OntoClean, the methodology based on formal ontological analysis [Welty and Guarino 2001], which evaluates the ontological decisions taken while building the ontology. This type of evaluation is based on on a rigorous analysis of the *ontological metaproperties* of taxonomic nodes, which are based on the philosophical notions of *unity, identity, rigidity* and *dependence* [Welty and Guarino 2001] and that we have described in Section 2.5.1.

When the knowledge encompassed in ontologies built for different purposes needs to be merged mismatches (due to the different types of heterogeneity that might affect ontologies) can become evident. Many types of heterogeneity in ontologies have been defined in the literature as we have already seen in Section 3.5, and the ontology environments currently available try to deal some of the conflicts that can arise when merging ontologies, such as SMART [Fridman Noy and Musen 1999] and CHIMAERA [McGuinness *et al.* 2000]. For the scope of this discussion we broadly group heterogeneity types into two types: syntactic and semantic. Mismatches arising because of syntactic heterogeneity can be detected and resolved semi-automatically with limited intervention from the domain expert (see Section 3.5.1). Mismatches due to semantic heterogeneity require a deeper knowledge of the domain (Section 3.5.2). Examples of conflicts caused by semantic heterogeneity can be found in [McGuinness *et al.* 2000, Tamma and Bench-Capon 2000]. Adding semantics to the concept descriptions can be beneficial in solving this latter type of conflict, because a richer concept description provides more scope to resolve possible mismatches. In particular, the attribute metaproperties on which the conceptual metamodel builds can be used to disambiguate concepts that seem similar, on the assumption that *candidate similar concepts* are described by the same attributes which *show the same behaviour* in the concept's definition. Moreover, attribute metaproperties complement the concepts metaproperties defined by Welty and Guarino [Welty and Guarino 2001], and can thus help to evaluate the taxonomic structure of the individual ontologies to be merged and to make it less tangled.

4.3.3 Reasoning with shared knowledge

The last argument to support the ontology conceptual metamodel we discuss turns on the need to reason with the knowledge expressed in the ontologies. Indeed, when different ontologies are integrated, new concepts are created from the definitions of the existing ones. In such a case conflicts can arise when conflicting information is inherited from two or more general concepts and one tries to reason with these concepts. Inheriting conflicting properties in ontologies is not as problematic as inheriting conflicting rules in knowledge bases, since an ontology is only *providing the means for describing explicitly the conceptualisation behind the knowledge represented in a knowledge base* [Bernaras *et al.* 1996]. Thus, in a concept description conflicting properties can coexist. However, when one needs to reason with the knowledge in shared ontologies, conflicting properties can hinder the reasoning process. In this case extra semantic information on the properties, can be used to derive which property is more likely to apply to the situation at hand. For example, the ability to evaluate the degree of confidence in a property describing a concept can be used to resolve mismatches that can arise if a concepts inherits conflicting properties. In order to be able to reason with these conflicts some assumptions have to be made, concerning on how likely it is that a certain property holds. Of course, such sophisticated assumptions cannot be made automatically and are left to knowledge engineers who are assisted in this delicate task by a system presenting them with the most likely options.

4.4 Moving from the knowledge level to the symbolic level

The discussion in Section 4.3 has argued in favour of a conceptual metamodel, motivating the choice by arguing that more semantic information is required to be able to merge ontologies efficiently and/or effectively and to reason with the knowledge resulting from the merge process. In this section we move from the ontology level to the symbolic level [Newell 1982] by describing a possible implementation of the conceptual metamodel into a *knowledge metamodel*. By *knowledge model* we denote a precise, human-readable specification for a representation of declarative knowledge. That is, a knowledge model is a set of predicates and functions expressed in some logical kind of calculus which formally and consistently defines the meaning of every construct available in the representation [Grosso *et al.* 1998]. The knowledge metamodel we propose adds a metalevel description to a frame based knowledge model. We enrich this model by characterising attributes with respect to the role they play in the concept description and by describing their behaviour over time.

In this thesis the information encompassed in this enriched conceptual metamodel is represented at the symbolic level by using a frame-based knowledge model OKBC-like [Chaudhri *et al.* 1998]. The advantages of using such a knowledge paradigm to implement the conceptual metamodel is that we can naturally represent the metaproperties by adding to the concept description a set of additional facets representing them.

Attribute metaproperties could be added to any ontology model, but frame-based representation systems are thought to be simpler to use and easier to understand than other ontology representation systems such as first order logic, description logic, etc [Lassila and McGuinness 2001].

We have illustrated in Table 2.2 (Section 2.4) the properties that should hold in order for something to be consider an ontology. Frame-based languages for ontologies such as Ontolingua [Farquhar *et al.* 1997] and paradigms such as OKBC [Chaudhri *et al.* 1998] embed these features in the language syntax (because they are based on KIF [Genesereth *et al.* 1992], which provides a well-defined semantics) We have to note that any knowledge model for ontologies could accommodate the metaproperties for attributes on which the conceptual metamodel builds. Therefore, although we implemented our metamodel in a frame based representation, this is not mandatory. The knowledge metamodel is described in the following section and subsections.

4.5 The proposed knowledge metamodel

In this section we introduce the frame-based knowledge model we have extended to accommodate the attribute metaproperties that we have discussed in Section 4.2. The knowledge model we use is inspired by OKBC [Chaudhri *et al.* 1998], and therefore supports an object-oriented representation of knowledge. In particular, we used OKBC Lite [Karp *et al.* 1999], which is described by a subset of the slots and facets of the standard OKBC. We chose the OKBC model since it is widely accepted by the ontology community and could be easily extended to accommodate the additional features.

It is worth noting at this point that we have used the OKBC model only as a support for the proof of concepts and that there are some differences between the model we propose and the OKBC knowledge model, namely, the set of facets we define is larger than the one defined in OKBC-Lite. Our aim is not to build a new knowledge model for ontologies but to support a semantically enriched description of attributes when defining concepts in ontologies.

The OKBC knowledge model is based on the notions of *classes*, *slots*, and *facets*. *Classes* correspond to concepts and to roles (see Section 2.3) that these concepts can play, and so they are collections of objects sharing the same properties, hierarchically organised into a multiple inheritance hierarchy, linked by *IS-A* links. Classes are described in terms of *slots*, or attributes, that can either be sets or single values. A slot is described by a name, a domain, a value type and by a set of additional constraints, here called *facets*. Facets can contain the documentation for a slot, constrain the value type or the cardinality of a slot, and provide further information concerning the slot and the way in which the slot is to be inherited by the subclasses. Here we present our extension to the OKBC knowledge model, this extension mainly regards extending the set of standard slots and facets provided by OKBC in order to encompass descriptions of the attribute and its behaviour in the concept description and as it changes over time.

In the following sections we describe the main entities composing our conceptual metamodel and we describe their implementation in the OKBC like metamodel. This description is not meant to be exhaustive, but just to give an example of how the enriched semantic conceptual metamodel could be implemented in an OKBC like knowledge metamodel. For this reason, most of the definitions are taken from the OKBC protocol [Chaudhri *et al.* 1998], to which we refer throughout the section, and we describe in the detail only the suggested extensions to the protocol.

4.5.1 Classes, roles, instances and individuals

A *class* is defined as a set of entities, and each entity belonging to this set is an *instance* of the class. Such entities can be either concepts or roles, and they are distinguished and labelled accordingly. Entities that are not classes are called *individuals*. Concepts and individuals in the conceptual metamodel proposed by this thesis are defined conforming to the OKBC protocol definition for classes and individuals [Chaudhri *et al.* 1998]. Classes are related to instances by the relation *instance of* and by its inverse relation *type of*. Classes are related to other classes by the relation *superclass of* and by its inverse *subclass of*, defined as in [Chaudhri *et al.* 1998]. The relations *superclass of* and *subclass of* organise classes hierarchically thus determining the *IS-A* hierarchy.

Classes are also used to represent *roles*, which can be thought of describing the *part played* by a concept in a specific context, (a more complete discussion on roles is postponed to Section 4.7.2). So we maintain a frame-like syntax for roles as well. Concepts are distinguished from roles by adding the facet :CLASS-TYPE to the set of facets, and this can take as value either `Concept` or `Role`, and by defining the class which plays the role that is being described (see Subsection 4.5.5). It is important to note here that we are not concerned with the problem of supporting a role representation in the syntax of the conceptual metamodel. The problem we are concerned with is to provide knowledge engineers with enough semantic information in order to enable them to recognise a role and distinguish it from a concept. By representing roles as classes we enable the definition of role instances and the creation of a *IS-A* hierarchy for roles which is separate from the one for concepts (see Subsection 4.5.4).

Picture 4.1 shows the set of facets describing a generic slot in the knowledge metamodel we propose, and which is derived by OKBC Lite.

4.5.2 Frames, slots, and facets

If the frame represents a *class* then it is associated with a set of *template slots*, which describe properties of the subclasses and the instances of the class. *Own slots* are always associated with a value or a set of values.

In line with the definition of frames in OKBC, the knowledge metamodel we propose here defines a *frame* as a primitive object representing an entity of the domain we are describing. Frames can represent either classes or individuals. In the former case they are called *class frames* and in the latter *individual frames*.

A frame is associated with a set of *own slots* which describe the direct properties of the entity represented by the frame, that is own slots describe attributes whose

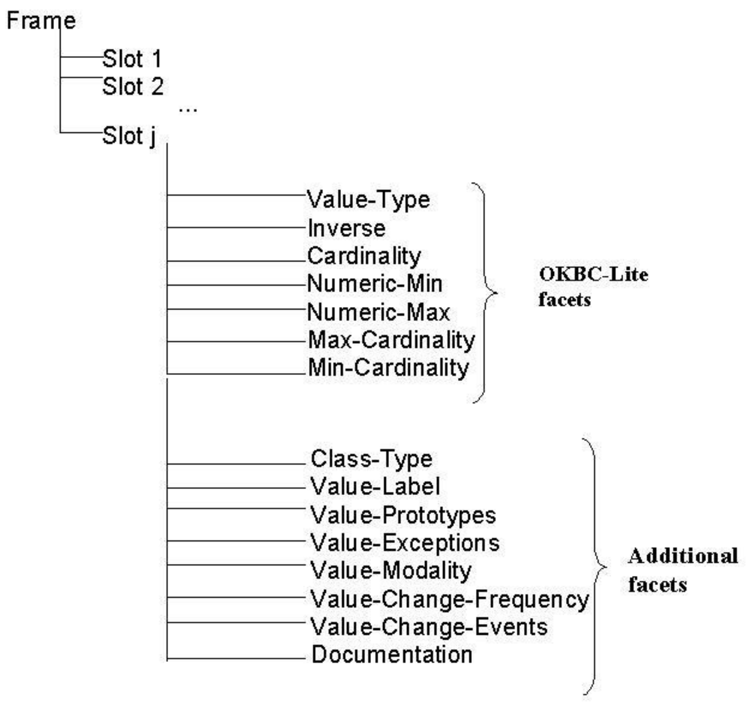

Figure 4.1: The additional and standard OKBC facets of the knowledge metamodel

value must be the same for all instances of the concept. Each own slot is associated with a set of entities called *slot values*; more formally for each value V associated with an own slot S of a frame F the own slot S is a binary relation holding between the entity represented by F and the entity represented by V. Each *own slot* has associated with it a set of *own facets* where each *own facet* of a slot S is associated with a set of *facet values*. A facet defines a ternary relation such that for each value V of own facet Fa of slot S of frame Fr, the relation Fa holds for the relation S, the entity represented by V and the entity represented by Fr.

Template slots can be associated with either a value or a class, and represent those attributes whose value may be different for each instance of a concept. The values associated with template slots are inherited to subclasses in the the class hierarchy and to instances. For each value V of a template slot S of a class C, S defines a binary relationship between the class represented by C and the entity represented by V. This relationship in turn holds for all the subclasses and all the instances of C. A *template slot* of a class frame is associated with a set of *template facets* that describe own facets for the corresponding own slots of each instance of the class. Also the values of template facets are inherited by the subclasses and the instances of the class. A facet formally defines a ternary relation Fa that holds for relation represented by the template slot S, the entity represented by value V and the class represented by the class frame C.

The knowledge metamodel we propose can also accommodate instantiated *relationships* between entities of the domain, following the OKBC knowledge model. Relations may thus be represented by frames as well, and particularly by describing a slot or a facet as a frame. This frame describes the defining properties of the relation represented by the slot or the frame. Such frames are called in OKBC *slot frame* if representing a slot and *facet frame* if representing a facet.

Just a brief note on slots and facets for roles: they are the same as those used to describe concepts, but we assume that those slots and facets for which a value would not be appropriate when describing roles would not be included into the class description.

4.5.3 Primitive and Non-Primitive Classes

As in OKBC, we also distinguish between *primitive* and *non primitive* classes. This distinction is important because for the values asssociated with the descriptions of slots and facets in our knowledge model (thus template slots values and template facet values in OKBC) usually specify a set of necessary and sufficient conditions for being an instance of the class. On the other hand, for what concerns *primitive classes* the values associated with the descriptions of slots and facets in our knowledge model specify necessary conditions only. Primitive and non primitive classes are features that OKBC inherits from description logic, which uses them to provide necessary and sufficient properties for class instances, they enable an object to be recognised as an instance of a class.

4.5.4 The pairs frame-slot and slot-facet

In our knowledge model we assume that for each frame a set of slots is defined and for each slot attached to a frame it is associated with a collection of facets, in a way such that a facet Fa is associated with a pair Fr-S (frame slot) if the facet has a value for the slot at the frame and analogously a slot S is associated with a frame Fr if the slot has a value at the frame Fr.
A facet does not always associate a single value with a slot. In fact, if the concept that is being described is a general one (thus located in the higher levels of the hierarchy) then is highly likely that one or more values are associated with a slot. In our model we assume these values to be *sets*, therefore comprising an unordered collection without multiple occurrences. The same assumption is made in OKBC to avoid the problems arising from a lack of suitable formal interpretation for:

- multiple slots or facets treated as unordered or ordered collections of objects with possibly multiple occurrences of the same value in the collection;
- the ordering of values in those collections of values which are ordered and result form multiple inheritance;
- the multiple occurrence of values in those unordered collections that result from multiple inheritance;

4.5.5 Standard classes, slots and facets

We assume here that our knowledge model includes only a subset of the collections of classes, slots and facets with pre-specified name and semantics, that are provided by the OKBC standard. We have selected the most commonly used slots and facets as in OKBC-Lite [Karp *et al.* 1999], a simplified version of OKBC. The standard classes are not considered mandatory for representing a concept, but if they are used, then they have to satisfy the semantics specified here.

Classes

The following standard classes are defined in OKBC-Lite and the model guarantees that their names are valid values for the :VALUE-TYPE facet described in Section 4.5.5:

Thing
:THING is assumed to be the root of the class hierarchy for any ontology, that is the superclass of every class in every ontology.

Class
:CLASS is the class of all classes. In other words any entity that is a class is an

instance of CLASS.

Individual

:INDIVIDUAL is the class of entities that are not classes. In other words any entity that is not a class is an instance of INDIVIDUAL.

Number

:NUMBER is the class of all numbers, and is a subclass of INDIVIDUAL. We do not make any specific assumption on the precision.

Integer

:INTEGER is the class of all integers. It is a subclass of :NUMBER, and so no assumption is made on the precision also for :INTEGER

String

:STRING is the class of text strings. It is a subclass of INDIVIDUAL

Standard facets

This subsection illustrates the standard facets that can be attached to a slot. The facets we use in our model are taken from OKBC-Lite, and a full specification of these facets can be found in the OKBC reference manual [Chaudhri *et al.* 1998]. We only disregard the :COLLECTION-TYPE facet, as we assume that our knowledge metamodel deals with the set type alone.

Value type

The :VALUE-TYPE facet defines a type restriction on the values of a slot of a frame. If the facet :CLASS-TYPE is associated with the value `Role` then :VALUE-TYPE facet describes the concept playing the role described in the frame. For example if the frame is describing the role *Student* then the :CLASS-TYPE has value `Role` and the facet :VALUE-TYPE has value `Person` to indicate that a person plays the role of a student depending on some situations, for instance when they enroll to the university. The values associated with a :VALUE-TYPE facet must be classes. A value C for the facet :VALUE-TYPE of the pair slot S-frame F means that every value associated to the slot S describing the frame F must be an instance of the class C. If the :VALUE-TYPE facet has multiple values for a slot S describing a frame F, then the values of S must be an instance of *every* class denoted by the values of :value-type.

Inverse

The :INVERSE facet of a slot defines inverses for that slot for the values associated with the slot of the frame. Values of this facets are slots themselves. If we have a slot S_1 of frame F whose inverse slot is S_2 (defined by associating S_2 with the facet :INVERSE), this means that if V is a value of slot S_1 of F, then F must be a value of S_2 of V.

Cardinality

The :CARDINALITY facet is associated to a nonnegative integer which defines the exact number of values that can be asserted for a slot on a frame. That is if the facet CARDINALITY is associated with the value N on slot S on frame F, then S has N values on F.

For example if we modelled the concept *Mother* in the frame `Mother`, then the slot `Parent-of` will have :CARDINALITY greater than or equal to 1 to indicate the fact that a parent is such if they have 1 or more children.

The values to associate to a slot S of frame F need not to be known in advance, the only information provided by the facet :CARDINALITY is that *when* the values for slot S are known the number of values for slot S *must* be exactly N, the value associated with :CARDINALITY.

Maximum Cardinality

For some slots one might need to assert not the exact number of values but a range of number of values, and this is done by associating values with the facets :MAXIMUM-CARDINALITY and :MINIMUM-CARDINALITY (see below). The facet :MAXIMUM-CARDINALITY defines the maximum number of values that the slot S of frame F can take. It is always a nonnegative integer N, which denotes that the slot S of frame F has *at most* N values.

Minimum Cardinality

The facet :MINIMUM-CARDINALITY asserts the minimum number of values that the slot S of frame F can take. This value is a nonnegative integer N, which denotes that the slot S of frame F has *at least* N values.

Numeric minimum

The :NUMERIC-MINIMUM facet specifies a lower bound on the slot S of frame F, whose values are numbers. The filler associates a number with the facet :NUMERIC-MINIMUM.

Numeric maximum

An upper bound on the values of slot S of frame F can be specified by associating a numeric value with the facet :NUMERIC-MAXIMUM of slot S. As for the NUMERIC-MINIMUM facet, the slot S has to be associated with values which are numbers.

The standard OKBC slots

In this section we introduce the standard slots of the OKBC-Lite [Karp *et al.* 1999] knowledge model and those defined on slot frames. These latter type of slots are those used in defining slot frames, they describe properties of a slot which hold at any frame that can have a value for the slot [Chaudhri *et al.* 1998].

Slots usually specify attributes of a class or a relationship between classes.
We do adopt all the slots on slot frames of OKBC-Lite but :SLOT-COLLECTION-TYPE, since we only consider sets in the knowledge metamodel we propose.

Documentation

The :DOCUMENTATION slot is associated with values in a frame that are text strings providing the documentation for that frame. Note that the documentation that describes a class is a value of the own slot :DOCUMENTATION on the class. This slot should give an account of information such as why the ranking has been set to a specific value or what is the context associated with a prototype (see below the discussion concerning prototypes). It should permit keeping track of the process leading to the modelling decisions.

Slots on Slot Frames

Domain

The :DOMAIN slot specifies the domain of the binary relation which is modeled by the slot frame. Each value associated with this slot has to be a class. If a slot frame S associates a value C to the *own* slot :DOMAIN then every frame that has a value for own slot S must be an instance of C, and every frame that has a value for template slot S must be C or own of its subclasses.
The :DOMAIN slot of a slot frame S can be associated with multiple values C_1, C_2, $\cdots$, C_n, and in such a case the domain of slot S is constrained to be the intersection of classes C_1, C_2, $\cdots$, C_n. Moreover, every slot is considered to have :THING as a value for its :DOMAIN slot.

Slot value type

The :SLOT-VALUE-TYPE slot defines the range of the binary relationship represented by the slot, that is the classes of which the value of a slot must be an instance.

The value associated with :SLOT-VALUE-TYPE can also be a set of keywords defined using the *set-of* constructor provided by OKBC.

If the additional facet :CLASS-TYPE (see Section 4.5.5) is associated with the value `Role` then :SLOT-VALUE-TYPE describes the concept playing the role described in the frame. For example if the frame is describing the role *Student* then the :CLASS-TYPE has value `Role` and the slot :SLOT-VALUE-TYPE has value `Person` to indicate that a person plays the role of a student depending on some situations, for instance when they enrol to university. The values associated with a :SLOT-VALUE-TYPE slot must be classes. A slot frame S that has an own slot :SLOT-VALUE-TYPE associated with a value V must have the own facet :value-type associated with the value V for the slot S of any frame (i.e., entity) that is in the domain of S.

Slot inverse

The :SLOT-INVERSE slot allows the specification of an inverse relation for a slot. The values associated with :SLOT-INVERSE are slots themselves. If we have a slot S with own slot :SLOT-INVERSE associated with value V, then the own facet :INVERSE has value V for slot S of any frame that is in the domain of S.

Slot cardinality

The :SLOT-CARDINALITY slot is a nonnegative integer which defines the exact number of values that can be asserted for a slot for those entities which are in the slot domain. If the own slot :SLOT-CARDINALITY of a slot frame S has value V, then the own facet :CARDINALITY for the slot S of any frame that is in the domain of S must be associated with V.

Slot maximum cardinality

The slot :SLOT-MAXIMUM-CARDINALITY is a non-negative integer defining the maximum number of values that can be asserted for a slot of those entities that are in the slot domain. If the own slot :SLOT-MAXIMUM-CARDINALITY of a slot frame

S has value V then the own facet :MAXIMUM-CARDINALITY must have value V for slot the S of any frame that is in the domain of S.

Slot minimum cardinality

The slot :SLOT-MINIMUM-CARDINALITY is a non-negative integer defining the minimum number of values that can be asserted for a slot of those entities that are in the slot domain. If the own slot :SLOT-MINIMUM-CARDINALITY of a slot frame S has value V then the own facet :MINIMUM-CARDINALITY must have value V for slot the S of any frame that is in the domain of S.

Slot numeric minimum

Analogously to the :NUMERIC-MINIMUM facet, the slot :SLOT-NUMERIC-MINIMUM defines a lower bound on the values that might be associated with a slot for those entities which are in the slot domain. If the slot frame S associates a value V with the own slot :SLOT-NUMERIC-MINIMUM then the own facet :NUMERIC-MINIMUM must associates the value V with the slot S of any frame which is in the domain of S.

Slot numeric maximum

The slot :SLOT-NUMERIC-MAXIMUM defines an upper bound on the values associated with a slot for those entities which are in the slot domain. If the slot frame S associates a value V to the own slot :SLOT-NUMERIC-MAXIMUM then the own facet :NUMERIC-MAXIMUM must associates the value V with the slot S of any frame which is in the domain of S.

Extensions proposed in this thesis

In the metamodel we propose in this thesis we do not really add any *modelling* slot to the ones of OKBC Lite described above [Karp *et al.* 1999], which adopt with no modifications to their meaning. We only disregard those facets and slots concerning collections, since we only consider sets. The only addition we make is a *documentation* slot, which documents the frame.

Documentation in frame

The facet :DOCUMENTATION-IN-FRAME associates with a slot of a frame text strings with the documentation for that slot on that frame.

The additional facets

The additional facets

We have extended the OKBC-Lite model to accommodate the attribute metaproperties *Mutability, Mutability Frequency, Event Mutability, Reversible Mutability, Modality, Prototypicality, Exceptionality, Inheritance,* and *Distinction*.
However, these additional facets are not mapped by corresponding slots into slot frames since the information encompassed in the additional facets makes sense only when a slot is associated with a frame, whereas it is undefined when the slot is considered on its own.

Class type

The facet :CLASS-TYPE has been added to the OKBC-Lite ones to specify whether the class that is being defined is a concept or a role. This facet can take two possible values: `concept` and `role` which are used to change the meaning of some of the frame facets.

Value label

This facet models the metaproperties *Inheritance* and *Distinction*, that distinguish those concepts properties that permit the distinction between siblings of a same parent concept from those concepts properties that are inherited from some parent concepts. The value associated with the facet :VALUE-LABEL of slot S of frame F is one or more elements from the set of keywords:{`Inherited`, `Inherited with ex-`

`ceptions`, `Distinguishing`, `Value`}. If the value associated with the slot is `Inherited` this means that the value associated with S has been inherited from some super class, whereas if it is `Inherited with exceptions` then the slot inherits the value from its parent frame, however some modifications (for example restriction on the domain) is made on the set of values associated with the slot. If the slot value is labelled through the facet :VALUE-LABEL as `Distinguishing` this means that it is a value that differentiates among siblings with a common super class. If the slot value is labelled as `Value` it means that the value is neither prototypical, nor inherited or distinguishing.

It should be noted that inherited and distinguishing values are incompatible in the same concept description, that is a value is either inherited or distinguishing, but cannot be both. On the other hand a value can be prototypical (see next facet) and inherited. Distinguishing values become inherited for subclasses of the class.

Of course also for distinguishing values it can be that inheritance does not concern the whole range of values, but only a subrange.

Value prototypes

This facet models *Prototypicality*, which permits us to identify a prototypical concepts property (modelled by associating a set of prototypical values to an attribute). The facet :VALUE-PROTOTYPES of slot S of frame F specifies which values of slot S are considered *prototypical* in the context specified by the frame F, that is those values that are normally (in the conception of the ontology designers) associated with the slot S when this is describing the concept at frame F. This enables the ontology designers to express what is believed to be normal from their perspective.

Therefore, the values associated with the slot S at frame F are those true for any prototypical instance of the class, but exceptions are permitted with a degree of credibility expressed by the slot :MODALITY (see also the facet :VALUE-EXCEPTIONS). For example, in the concept `Blood Pressure` the prototypical values (that is the values of blood pressure for an healthy individual over 18) are between 90 and

130 for systolic pressure and between 60 and 85 for diastolic pressure. This notion of prototypical values is related to the analogous one in cognitive science [Rosch 1975] and is discussed, together with the notion of exception in Section 4.7.4.

Value exceptions

The facet :VALUE-EXCEPTIONS of slot S of frame F specifies which values of those associated with slot S are to be considered as exceptional, that is those values that are permitted in the concept description because they are in the domain, but deemed exceptional from a common sense viewpoint. It models the metaproperty *exceptionality*. Exceptional values are not only those which differ from the prototypical ones but also any value which is possible but highly unlikely. The value that this facet can take is therefore a value or a subset of the values associated with the slot S. Let us consider again the blood pressure example. Exceptions are those values registered for people affected by conditions such as hypertension or hypotension and are therefore those in the range of values for the slot `blood pressure` but outside the range determined by the prototypical values. That is, exceptional values for systolic pressure are those in the range of the slot that are smaller than 90 and greater than 130, whereas for diastolic pressure, the exceptional values are those smaller than 60 and greater than 85.

Value modality

The facet :VALUE-MODALITY of slot S of frame F models the *Modality* metaproperty and denotes the degree of confidence of the fact that the slot is associated with one or more specified values. It describes the class membership conditions. The value associated with this facet is a nonnegative integer between 1 and 7. Each of these numbers is associated with a specific meaning. The possible values associated with this slots are reported below together with an example showing cases in which each of the values apply:

1. `All`. Let us assume we have a frame `Person` which is described by the

property *has fingerprints* modelled by associating the value `Yes` with the slot :HAS-FINGERPRINTS. The property of having fingerprints (as opposed to a specific instance of it, for example John Doe's fingerprints) is inherited by all subclasses, that is all the subclasses of `Person` (such as `Child` , `Teenager`, `Adult` , and so on) have fingerprints. This kind of information is described by associating the filler `All` with the facet :VALUE-MODALITY when describing the slot :HAS-FINGERPRINTS;

2. `Almost all`. A typical example of a property which holds for *almost all* the subconcepts of the concept which is being described is the mammals' ability to give birth to live young. In fact almost all species of mammals give birth to live young with the exceptions of a particular family, called *monotremes* who does not. If we were to model this situation, the slot :ABILITY-TO-GIVE-BIRTH-TO-LIVE-YOUNG would be described by value `Almost all` associated with the facet :VALUE-MODALITY;

3. `Most`. The filler `Most` is to be used in those cases where the majority of subclasses inherit the property. For example, let us suppose to consider the concept *Cat*. The majority of cats have short hair, although there is a considerable number of cat species who have long hair. If we had to model the concept by associating with it the property *has short hair*, then such a slot would be described by associating the filler `Most` to the facet :VALUE-MODALITY;

4. `Possible`. In some cases, however, we might not have any information concerning the degree of applicability of a property to the properties subconcepts. For example, let us consider the concept *university professor*. In some countries, like Italy, for instance, it is not always the case that in order to be a professor one has to be awarded a PhD. On the other hand in some other countries, like the UK or the United States, it is often the case that a professor has a PhD. If we had to model the concept *professor* the property *has a phd* would be described by associating the value `Possible` to the modality facet,

in that it is possible that the property holds for some of the subclasses of the concept, but we do not have information concerning the specific subclasses for which the property holds neither we know for how many of them;

5. `A Few`. It has the opposite semantics of `Most`. For example, let us suppose that we are modelling the concept *Penguin*. Most penguins live in cold environments, however there are few penguins who have adapted to live on a beach in Cape Town, where the temperature can rise up to 40 C. If we modelled the concept *Penguin*, it would be described by the property *living in warm environments*, but with the modality set to `A few`.

6. `Almost none`. It has the opposite semantics of `Almost all`. For example, let us suppose to model the concept *graduates*. Usually graduates are those who have been awarded with a degree from a university. However, there are degrees awarded *honoris causae* which are awarded also to people who have not attended any university. So, the property of having a second degree for the class of people who received a degree *honoris causae* would be modelled by associating the value `Almost none` to the facet :VALUE-MODALITY;

7. `None`. It has the opposite semantics of `All`. It models the absence of a certain property. For example, if we model the concept *Bird*, then its ability to fly would be described by associating the modality `Possible` with the property *can fly*. However, this property is not inherited by all subconcepts, therefore, if we model the concept *Penguin*, the property describing the ability to fly may be characterised by associating with it the modality `None`, which is equivalent to say that the property does not hold for any instance of that concept.

The last three fillers, `A Few`, `Almost none` and `None`, they can be thought as the counterparts of the values `Most`, `Almost all`, and `All`. In particular the value

`None` associated with this facet is tantamount to negation. The value `None` as a possible filler for the slot VALUE-MODALITY makes sense especially in the context of conflicts resolution in case of inheritance, and has been added to the model in the hypothesis that such model is used to support semi-automatic conflict resolution. It would make little sense for a knowledge engineer to include in the concept description a property whose degree of applicability to subclasses is none.

Value change frequency

The facet :VALUE-CHANGE-FREQUENCY of slot S of frame F specifies whether and how often the value of slot S changes during the lifetime of the concept which is represented by the frame F. It models the properties *Mutability* and *Mutability Frequency*. The value associated with this slot is an element of the set: {`Regular`, `Once only`, `Volatile`, `Never`}.

If the value of the slot is `Regular` it denotes that the change process is continuous, for instance the age of a person can be modelled as changing regularly. If the facet value is to `Once only` it means that only one change over time for the value of slot S is possible, while if the value of the slot is `Never` it specifies that the value of the slot S is set only once and then it cannot change again, for example a person's date of birth once set cannot change again, and finally `Volatile` means that the change process is discrete and can be repeated at irregular intervals, that is the attribute's value can change more than once; for example people can change job more than once

---

Value-change-events

This facet models the metaproperty *Mutability Event*, and it identifies those events that can cause an attribute to change its value. The :VALUE-CHANGE-EVENTS facet specifies the conditions under which the values associated with slot S change. It is associated with one or more quadruples

$$\{((E_j, S_j, V_j), R_j) | j = 1, \cdots, m\}$$

where E_j is an event, S_j is the state of the pair attribute-value associated with a property, V_j defines the event validity and R_j denotes whether the change is reversible or not, thus modelling the property *Reversible Mutability* (which denotes a concept's property that can change in time, but whose change is reversible). The semantics of this facet is explained in Section 4.7.1.

If the class describes a role, that is the facet :CLASS-TYPE is associated with the value `role`, then the facet :VALUE-CHANGE-EVENTS defines the conditions regulating the acquisition and the relinquishment of a role. This point is further discussed in Section 4.7.2.

4.6 Expressive power of the conceptual metamodel

The conceptual metamodel whose implementation has been presented in the previous section can accommodate almost all the modelling primitives which are considered necessary to write ontologies. We do not have axioms as they were out of the scope of this thesis. However, axioms will be considered in future developments.

As we have already mentioned, concepts are represented by classes, which are described in terms of attributes or properties, described by pairs *slots-values*. In this knowledge model slots are used to describe both *intrinsic* and *extrinsic* concept properties. According to Guarino and Welty [Guarino and Welty 2000a, page 100]:

> An intrinsic property is typically something inherent to an individual, not dependent on other individuals, such as having a heart or having a fingerprint. Extrinsic properties are not inherent, and they have relational nature, like "being a friend of John". Among these, there are some that are typically assigned by external agents or agencies such as having a specific social security number, having a specific customer i.d., even having a specific name.

Here we take the same view by considering intrinsic properties as those inherent to individuals, and which are not determined by other individuals, such as having a particular set of fingerprints. Intrinsic properties represent descriptive features of the class and thus correspond to the modelling primitive *attributes*.

Extrinsic properties are those not inherent to an individual and which have a relational nature. They represent relations between classes, thus corresponding to the modelling primitive *relationship*.

So slot-value(s) pairs are used to describe properties holding for a class, in turn properties are described by a set of pairs *facet-value* that characterise the properties. The original OKBC facets describe the syntax of properties, for example by defining whether they are represented by strings or numbers and, in the latter case, by defining their minimum and maximum values. The additional facets that extend the OKBC model are more concerned with the semantics of the properties when they are used in a specific context determined by the frame, describing, for example, the behaviour over time or the degree of credibility with which the property holds.

The knowledge model presented in Section 4.2 is motivated by the the discussion illustrated in Section 4.3. It is based on an enriched semantics that aims to provide a better understanding of the concepts and their properties by characterising their behaviour.

Depending on the object they characterise, properties can be defined for *instances* and *classes*, but we can also define *metaproperties*. *Instance properties* are those exhibited by all the instances of a concept. They might specialise *class properties*, which instead describe properties holding for the class. *Concept metaproperties* have been mainly described in philosophy (see Section 2.5.1 for a full account), and include *identity, unity, rigidity* and *dependency*. In Section 4.2 we have described the set of attribute metaproperties which defines the metalevel on our conceptual model (Section 2.3).

Properties can also be divided into prototypical, necessary, distinguishing, inherited and simple value assignments. Concepts in the knowledge model are hierarchically organised according to an *Is-a* relationship that permits property inheritance from ancestors to descendants. The properties that are inherited from an ancestor are labelled as inherited. However, inherited properties can be overruled in the more specific concept in order to accommodate inheritance with exceptions. The properties that have been overruled are labelled as *distinguishing* (according to the definition given in Section 3.7.1), since they allow us to distinguish between siblings of a same parent concept. We give to the term distinguishing a broader meaning and we decide to associate the label distinguishing with any case where a value assignment permits to disambiguate among siblings. A property is to be considered necessary if it is essential to all instances of the concept, while is prototypical if it holds for the prototypical instances of the concept only. The notion of essential property relates to the idea of necessary condition while prototypical properties permit to identify prototypes, discussed in Section 4.7.4. Finally, a property labelled as value assignment associates a value to an attribute in order to describe a specific feature of the instances of the concept, such as hair colour = brown.

Roles, already defined in Section 2.3, are also supported in this knowledge model; they are represented as concepts but the facet :CLASS-TYPE is set to `role`, so that we are able to distinguish them from a concept definition. As for the rest, a role has exactly the same definition as a concept since roles are described in terms of attributes that are typical of a role and are organised into a is-a hierarchy totally analogous to the one defined for concepts, where the inheritance of properties through the role hierarchy is permitted in order to represent properties that are typical of roles. Most of the consideration we made for concepts hold for roles as well, therefore we can consider prototypical properties for roles, distinguishing properties and so on and so forth. What distinguishes a role from a concept is that the role holds during a specific span of time. Roles and their properties are discussed below in Section 4.7.2

4.7 Relating the extended knowledge model to the motivations

In order to use the enriched conceptual model knowledge engineers have to provide more details concerning the concepts than if they were using a traditional OKBC-like knowledge model; they are thus guided in performing the ontological analysis which is usually demanding to perform.

Furthermore, the enriched knowledge model forces knowledge engineers to make ontological commitments explicit. Indeed, real situations are information-rich events, whose context is so rich that, as it has been argued by Searle [Searle 1983], it can never be fully specified. Many assumptions about meaning and context are usually made when dealing with real situations [Rosch 1999]. These assumptions are rarely formalised when real situations are represented in natural language but they have to be formalised in an ontology since they are part of the ontological commitments that have to be made explicit. Enriching the semantics of the attribute descriptions with things such as the behaviour of attributes over time or how properties are shared by the subclasses makes some of the more important assumptions explicit.

The enriched semantics is essential to solve the inconsistencies that arise either while integrating diverse ontologies or while reasoning with the integrated knowledge. By adding information on the attributes we are able to measure better the similarity between concepts, to disambiguate between concepts that *seem* similar while they are not, and we have means to infer which property is likely to hold for a concept that inherits conflicting properties.

A possible disadvantage of such a semantically enriched knowledge model is the high number of facets that need to be filled when building ontologies. We realise that this can make building an ontology from scratch even more time consuming but we believe that the outcomes balance the increased complexity of the task. Indeed, in order to fill the additional facets knowledge engineers need to have a full understanding not only of the concept they are describing, but also of the context in which the concept is used. Arguably, they need such knowledge if they are to perform the modelling task thoroughly.

The remainder of this section describes the additional facets and relates them to the discussion in Section 4.3.

4.7.1 Attribute behaviour over time and characterisation of identity

The metaproperties *Mutability, Mutability Frequency, Event Mutability*, and *Reversible Mutability* are modelled in the knowledge metamodel by the facets :VALUE-CHANGE-FREQUENCY and :VALUE-CHANGE-EVENT which describe the behaviour of *fluents* over time. The behaviour over time is closely related to establishing the *identity* of concept descriptions [Guarino and Welty 2000b], in that some properties can change without affecting the identity of the changing individual. Describing the behaviour over time involves also distinguishing properties whose change is *reversible* from those whose change is *irreversible*.

Property changes over time are caused either by the natural passage of time or are triggered by specific event occurrences. We need, therefore, to use a suitable temporal framework that permits us to reason with time and events. The model chosen to accommodate the representation of the changes is the *Event Calculus* [Kowalski and Sergot 1986]. Event calculus deals with local event and time periods and provides the ability to reason about change in properties caused by a specific event and also the ability to reason with incomplete information.

Changes in concept properties (which correspond to changes in the values associated with attributes) can be modelled as *processes* [Sowa 2000]. Processes can be described in terms of their starting and ending points and of the changes that happen in between. We can distinguish between *continuous* and *discrete changes*, the former describing incremental changes that take place continuously while the latter describe changes occurring in discrete steps called *events*. Analogously we can define *continuous properties* as those changing regularly over time, such as the age of a person, versus *discrete properties* which are characterised by an event which causes the property to change. If the value associated with change frequency is `Regular` then the process is continuous, if it is `Volatile` the process is discrete and if it is `Once only` the process is considered discrete and the triggering event is set equal to *time-point=T*.

Any regular occurrence of time can be, however, expressed in form of an event, since most of the forms of reasoning for continuous properties require discrete approximations. Therefore in the knowledge metamodel presented in Section 4.5 and subsections, continuous properties are modelled as discrete properties where the event triggering the change in property is the passing of time from the instant t to the instant t'. Each change of property is represented by a set of quadruples $\{((E_j, S_j, V_j), R_j)|j = 1, \cdots, m\}$ where E_j is an event, S_j is the state of the pair attribute-value associated with a property, V_j defines the event validity while R_j indicates whether the change in properties triggered by the event E_j is reversible or not. The model used to accommodate this representation of the changes adds reversibility to *Event Calculus*, where each triple (E_j, S_j, V_j) is interpreted either as *the concept is in the state S_j before the event E_j happens* or *the concept is in the state S_j after the event E_j happens* depending on the value associated with V_j. The interpretation is obtained from the semantics of the event calculus, where the former expression is represented as *Hold(before(E_j, S_j))* while the latter as *Hold(after(E_j, S_j))*.

The idea of modelling the permitted changes for a property is strictly related to the philosophical notion of *identity*. In particular, the knowledge model addresses the problem of modelling identity when time is involved, namely *identity through change*, which is based on the common sense notion that an individual may remain the same while showing different properties at different times [Kant 1965]. The knowledge model we propose explicitly distinguishes the properties that can change from those which cannot, and describes the changes in properties that an individual can be subjected to, while still being recognised as an instance of a certain concept.

Events in this representation are always *point events*, and we consider *durational events* (events which have a duration) as being a collection of *point events* in which the state of the pair attribute-value as determined by the value of V_j, holds as long as the event lasts. The duration is determined by the definition of an *event* in *Event Calculus*, where for each event is given an initial and a final time point. We realise that this representation oversimplifies the dynamic of process changes and we aim to investigate a more sophisticated change representation as future work.

The notion of changes through time is also important to establish whether a property is *rigid*. A *rigid property* is defined in [Guarino *et al.* 1994] as:

> a property that is essential to *all* its instances, i.e. $\forall x \phi(x) \rightarrow \Box \phi(x)$.

that is, if for every x the property ϕ holds in x, then ϕ is necessary for x. By *essential property* we mean a property holding for an individual in every possible circumstance in which the individual exists. The interpretation that is usually given to *rigidity* is that if x is an instance of a concept C, then x has to be an instance of C in every possible world [Kripke 1980]. Here we specifically concentrate on one of these systems of possible worlds, that is time.

In [Tamma and Bench-Capon 2001a, Tamma and Bench-Capon 2001b] we have related the notion of *rigidity* to those of *time* and *modality* and in Section 4.7.3 we show that, by using the information represented in the slot :VALUE-MODALITY and that concerning the behaviour over time, we can precisely identify rigidity in the subset of the set of possible worlds.

More recently Guarino and Welty have re-formulated the definition of rigidity which now takes in explicit account the relationship between time and modality. A rigid property ϕ is thus a property such that $\Box\ (\forall x, t \phi(x,t) \rightarrow \Box\ \forall t' \phi(x,t'))$.

That is, for every x and for every instant of time t, if ϕ holds for x in t, then ϕ is necessary for x in every instant t'.

By characterising the rigidity of a property in this subset of possible worlds we aim to provide knowledge engineers with the means to reach a better understanding of the *necessary* and *sufficient* conditions for the class membership. However, this does not mean that the rigidity of a property depends on any account on whether the property is used to determine class membership or not. That is, the final aim is to try to separate the properties constitutive of identity from those that permit re-identification.

4.7.2 The need for identity and rigidity: Roles

Establishing whether rigidity holds for a property is not only central in order to distinguish necessary truth but also to recognise *roles* from concepts. The notion of *role* is as central to any modelling activity as those of *objects* and *relations*.
A definition of role that makes use of the formal metaproperties and includes also the definition given by Sowa [Sowa 1984] is provided by Guarino and Welty. In [Guarino and Welty 2000a] they define a role as:

> properties expressing the *part played* by one entity in an event, often exemplifying a particular relationship between two or more entities. All roles are *anti-rigid* and *dependent*... A property ϕ is said to be anti-rigid if it is not essential to *all* its instances, i.e. $\forall x \phi(x) \rightarrow \neg \Box \phi(x)$... A property ϕ is *(externally) dependent* on a property ψ if, for all its instances x, necessarily some instance of ψ must exist, which is not a part nor a constituent of x, i.e. $\forall x \Box(\phi(x) \rightarrow \exists y \psi(y) \wedge \neg P(y,x) \wedge \neg C(y,x))$.

In other words a concept is a role if its individuals stand in relation to other individuals, and they can enter or leave the extent of the concept without losing their identity. From this definition it emerges that the ability of recognising whether rigidity holds for some property ϕ is essential in order to distinguish whether ϕ is a role.
In [Steimann 2000] the author compares the different characteristics that have been associated in the literature with roles. From this comparison it emerges that the

notion of role is inherently temporal, indeed roles are acquired and relinquished in dependence either of time or of a specific event. For example the object *person* acquires the role *teenager* if the person is between 13 and 18 years old, whereas a person becomes *student* when they enrol for a degree course. Moreover, from the list of features in [Steimann 2000] it emerges that many of the characteristics of roles are time or event related, such as: an object may acquire and abandon roles dynamically, may play different roles simultaneously, or may play the same role several time, simultaneously, and the sequence in which roles may be acquired and relinquished can be subjected to restrictions.

Roles may be "naturally" determined when social context is taken into account, and the social context determines the way in which a role is acquired and relinquished. For example, the role of `President of the country` is relinquished differently depending on the context provided by the country. So, for example, in Italy the role may be acquired and relinquished only once in the lifetime of an individual, whereas if the country is the United Sates, the role can be acquired and relinquished twice, because a president can be re-elected. Social conventions may also determine that once a role is acquired it cannot be relinquished at all. For example, the role `Priest` in a catholic context is relinquished only with the death of the person playing the role.

For the aforementioned reasons ways of representing roles must be supported by some kind of explicit representation of time and events. The knowledge model we have presented provides sufficient semantics to model the dynamic features of roles. Indeed the model provides a way to explicitly represent time intervals which can be used to used to model roles as fluents; moreover, the facet :VALUE-CHANGE-EVENT gives knowledge engineers the ability to model events, which describe the events that constrain the acquisition or the relinquishment of a role.

The ability to distinguish roles gives also a deeper understanding of the possible contexts in which a concept can be used. Recognising a role can be equivalent to defining a context, and the notion of context is the basis on which prototypes and exceptions are defined.

4.7.3 Modality: Weighing the validity of attribute properties

The metaproperty *Modality* is used to express the way in which a statement is true or false. In this thesis, we denote with the facet :VALUE-MODALITY of slot S at frame F the degree of confidence that we can associate with finding a certain world. The notion of *Modality* is quite similar to the one of *rankings* as defined by Goldszdmidt and Pearl [Goldszmidt and Pearl 1996, page 60]:

> Each world is ranked by a non-negative integer κ representing the degree of surprise associated with finding such a world.

Here we use the term modality to denote the degree of surprise in finding a world where the property P holding for a concept C does not hold for one of its subconcepts C'. The attribute metaproperties modelled by this facet is important to reason with statements that have different degrees of credibility. Indeed there is a difference in asserting facts such as “Mammals give birth to live young” and “Birds fly”, the former is generally more believable than the latter, for which many more counterexamples can be found. The ability to distinguish facts whose truth holds with different degrees of strength is important in order to find which facts are true in every possible world and therefore constitute *necessary truth*. The concept of necessary truth brings us back to the discussion about *rigidity*. *Rigidity* is one of the concepts metaproperties on which the OntoClean methodology builds [Welty and Guarino 2001]. OntoClean does not provide any means to assign such property to concepts, since it does not focus on concept descriptions. The metaproperties for attributes that are the base for the conceptual metamodel proposed in this thesis can support the assesment of *rigidity*. In the knowledge metamodel described above, the value associated with the :VALUE-MODALITY facet together with the temporal information on the changes permitted for the property can determine whether the property described by the slot is a rigid property. In particular, we can exactly determine rigidity in a subset of all possible worlds. Indeed, since an ontology defines a vocabulary, we can restrict ourselves to the set of possible worlds which is defined as the set of maximum descriptions obtainable using the vocabulary defined by the ontology [Plantiga 1989]. Then, under the assumption of restricting the discourse to this set of possible worlds, *rigid properties* are those whose :VALUE-MODALITY facet is equal to `All` and that cannot change in time, that is whose :VALUE-CHANGE-FREQUENCY facet is set to `Never`.

The ability to evaluate the degree of confidence in a property describing a concept is also related to the problem of reasoning with ontologies obtained by merging. In such a case, as mentioned in Section 4.3.2 conflicting properties may be inherited if heterogeneous ontologies are merged. In order to reason with these conflicts it is necessary to make some assumptions on how likely it is that a certain property holds; the facet :VALUE-MODALITY models this information by modelling a qualitative evaluation of how subclasses inherit the property. This estimate represents the common sense knowledge expressed by linguistic quantifiers such as `All`, `Almost all`, `Few`, etc. It is important to note at this point that, although we have implemented the *Modality* metaproperty in the :VALUE-MODALITY facet whose values are which takes value in the set {*All, Almost all, Most, Possible, A Few, Almost none, None*}, the choice of such a set is totally arbitrary, and it was meant to be such. Knowledge engineers should be able to associate with this meta-property either a probability value, if they know the probability with which the property is inherited by subconcepts, or a degree of belief (such as a κ-value, as in [Goldszmidt and Pearl 1996], which depends on an ϵ whose value can be changed according to the knowledge available, thus causing the κ function to change), if the probability function is not available.

In case of conflict the property's degree of credibility can be used to rank the possible alternatives following an approach similar to the non-monotonic reasoning approach developed by [Goldszmidt and Pearl 1996]: in case of more conflicting properties holding for a concept description, properties are ordered according to the degree of credibility, that is a property holding for all the subclasses is considered to have a higher rank than one holding for few of the concept subclasses. This ordering of the conflicting properties needs to be validated by the knowledge engineer, although, it reflects the common sense assumption that, when no specific information is known, people assume that the most likely property holds for a concept.

4.7.4 Prototypes, exceptions, and concepts

In order to get a full understanding of a concept it is not sufficient to list the set of properties generally recognised as describing a typical instance of the concept but we need to consider the expected exceptions. Here we denote by *prototype* those values that are prototypical for the concept that is being defined, while we denote *exceptions* those that differs from what is normally thought to be a feature of the cognitive category and not only what differs from the prototype. In this way, we partially take the cognitive view of prototypes and graded structures, which is also reflected by the information modelled in the facet :VALUE-MODALITY. In this view all cognitive categories show gradients of membership which describe how well a particular subclass fits people's idea or image of the category to which the subclass belong [Rosch 1975]. Prototypes are the subconcepts which best represent a category, while exceptions are those which are considered exceptional although still belonging to the category. In other words all the sufficient conditions for class membership hold for prototypes. For example, let us consider the biological category *mammal*: a *monotreme* (a mammal who does not give birth to live young) is an example of an exception with respect to this attribute. Prototypes depend on the context; there is no universal prototype but there are several prototypes depending on the context, and so a prototype for the category *mammal* could be *cat* if the context taken is that of *animals that can play the role of pets* but it is *lion* if the assumed context is *animals that can play the role of circus animals*. In the knowledge model presented above we explicitly describe the context in natural language in the *Documentation* facet, however, the context can be also described by the roles that the concept which is being described is able to play.

Ontologies typically presuppose context and this feature is a major source of difficulty when merging them.

Prototypes are also quite important in that they provide a frame of reference linguistic quantifiers such as *tall*, *short*, *old*, etc. These quantifiers are usually defined or at least related to the prototypical instance of the class whose members they are describing, and indeed their definition changes if we change the point of reference. For example, if we are defining the concept *tall* using as frame of reference the class :PERSON then *tall* means over 2 metres, whereas if we define *tall* with respect to the class :BUILDING it means over 300 metres. And again, depending of the level of granularity chosen for the description the linguistic quantifiers can have more specific meanings. For example, if we subdivide the class :BUILDING into two subclasses, :COTTAGE and :SKYSCRAPER, then an adjective such as *tall* related to the prototypical instances of the two classes takes the meaning of over 10 metres in the first case and over 300 metres in the latter case.

Therefore including the notions of prototypes and exceptions permits us to provide a frame of reference for defining these qualifiers with respect to *a specific class*. For the purpose of building ontologies, distinguishing the prototypical properties from those describing exceptions increases the expressive power of the description. Such distinctions do not aim at establishing default values but rather to guarantee the ability to reason with incomplete or conflicting concept descriptions.

The ability to distinguish between prototypes and exceptions helps to determine which properties are necessary and sufficient conditions for concept membership. In fact a property which is prototypical and that is also inherited by all the subconcepts (that is it has the facet :VALUE-MODALITY set to `All`) becomes a natural candidate for a necessary condition. Prototypes, therefore, describe the subconcepts that best fit the cognitive category represented by the concept *in the specific con-*

text given by the ontology. On the other hand, by describing which properties are exceptional, we provide a better description of the class membership criteria in that it determines what are the properties that, although rarely hold for that concept, are still possible properties describing the cognitive category.

Also the information on prototype and exceptions can prove useful in dealing with inconsistencies arising from ontology integration. When no specific information is made available on a concept and it inherits conflicting properties, then we can assume that the prototypical properties hold for it.

The inclusion of prototypes in the knowledge model provides the grounds for the semi-automatic maintenance and evolution of ontologies by applying techniques developed in other fields such as machine learning.

4.8 Chapter summary

This chapter has presented a knowledge metamodel that extends the usual ontology frame-based model such as OKBC by explicitly representing additional information on the slot properties. In order to represent the metaproperties *Mutability, Mutability Frequency, Reversible Mutability, Event Mutability, Modality, Prototypicality, Exceptionality*, *Inheritance* and *Distinction* we have added to an OKBC like knowledge model a set of extra facets modelling these metaproperties, namely the facets :VALUE-LABEL, :VALUE-PROTOTYPES, :VALUE-EXCEPTIONS, :VALUE-CHANGE-FREQUENCY, and :VALUE-CHANGE-EVENTS. We also added the facet :CLASS-TYPE to model the distinction between roles and concepts.

This knowledge metamodel is driven by the formal ontological analysis by Guarino and Welty [Guarino and Welty 2000b] which permits ontologies that have a cleaner taxonomic structure to be built and so gives better prospects for maintenance and integration. Such a formal ontological analysis is usually difficult to perform and we believe our knowledge model can help knowledge engineers to determine the metaproperties holding for the concept by forcing them to make the ontological commitments explicit.

The knowledge metamodel we propose results from a conceptual metamodel which encompasses attribute metaproperties aiming to characterise the behaviour of properties in the concept description. We have motivated this enriched conceptual model by identifying three main categories of problems that require additional semantics in order to be solved.

The extention of the conceptual model with a metalevel constitutes the novel contribution of this thesis. This extension explicitly represents the behaviour of attributes over time by describing the changes in a property that are permitted for members of the concept. It also explicitly represents the class membership mechanism by associating with each slot a qualitative quantifier representing how properties are inherited by subconcepts. Finally, the model describes not only the prototypical properties holding for a concept but also the exceptional ones.

Chapter 5

Modelling a domain

5.1 The chosen domain

In this chapter we provide an example of how the instantiation of the conceptual model which we propose in this thesis, and which is presented in Chapter 4, can be used to model a domain of interest. The purpose of this chapter is not only to illustrate by means of a practical example how the proposed model can be used; we also intend to provide a proof of concept for the claims we made in the previous chapter that this enriched knowledge model gives a better representation of the semantics and permits to identify the metaproperties associated with concept properties (see Section 4.7) more easily. The domain of interest we have chosen is a medical one and we are focusing our attention on a particular condition called *Disseminated Intravascular Coagulation* or *DIC*. We have chosen this particular domain for two reasons: the first is mainly concerned with modelling something as complex as a disease, while the second is related to the specific disease we have decided to model.

The literature presents many examples of medical taxonomies, such as SNOMED [Rothwell 1995] UMLS [Lindberg *et al.* 1993], and GALEN [Zanastra *et al.* 1995]. Some of them like SNOMED or UMLS are just terminology hierarchies whereas others, such as GALEN have a more detailed concept definition. Those medical ontologies which provide a formal information model have to face the inherent problems related with the attempt to define a disease. Indeed, in the field it is still unclear how disorders should be described as the functioning of the human body and of pathogens have not been well understood. This lack of knowledge is reflected by the fact that disorders have been modelled according to the ways in which they can be defined, some defined at genetic level, others at chemical level, some others in terms of their association with physical factors and lastly as syndromes or collections of observations.

Another real source of complexity is the qualitative nature of medical expertise, which is very difficult to model in a knowledge model. Indeed, practitioners tend to reason with concepts such as *low number of platelets*, where the definition of *low* is not only dependent on the context but also on the conceptualisation used by a specific practitioner.

When DIC is described from a physiological viewpoint, it is necessary to study a complex interaction of the chemical substances involved in the coagulation process. In particular, some of these substances contribute to the creation of *platelets* which are mainly responsible for the formation of blood clots and, at the same time other chemical substances in the blood start a process of clot destruction, which eventually results in haemorrhage. It is because of these contradictory symptoms that a second source of complexity arises: in fact DIC could become manifest taking the form of haemorrhage, or of blood clot or a combination of the two. In other words DIC is a disorder in which systemic activation of the coagulation system *simultaneously* leads to intravascular thrombus formation (which compromises blood supply to organs) and exhaustion of platelets and coagulation factors (which results in haemorrhage). Thus, on a first sight it seems that DIC is characterised by conflicting symptoms. [Levi and deJonge 2000]

The diagnosis of DIC is made even more difficult because DIC is not a so called *primary condition*: that is, DIC is always the consequence of some primary disease which needs to be treated in order to treat the DIC symptoms. Following is a list of clinical primary disorders that can lead to DIC [Levi and deJonge 2000]:

- Malignancy in solid tumors, such as myeloproliferative, lymphoproliferative. DIC can further complicate both solid tumors and hematologic malignancies;
- Obstetric complications such as amniotic fluid embolism, *abruptio placentae*. The most common obstetric emergency associated with activation of coagulation is pre eclampsia;
- Organ destruction such as severe pancreatitis;
- Sepsis and severe infection caused by any microorganism. In particular, DIC is associated with septicemia but it might also be caused by systemic infections with other microorganisms such as viruses and parasites;
- Severe hepatic failure;
- Severe toxic or immunologic reactions, for example snake bites, recreational drugs, transfusion reactions, and transplants rejection;
- Trauma, such as polytrauma, neurotrauma, trauma resulting in fat embolism. In particular head trauma is associated with DIC;
- Vascular abnormalities, for example giant hemangiomas (Kasabach-Merrit syndrome), large vascular aneurysms.

This also implies that this disorder manifests with a number of different symptoms and thus several clinical tests are needed in order to detect the disease. Diagnosing DIC is therefore quite difficult, and it is even more difficult to be able to detect the clues that the condition is developing before it is fully manifested.
We distinguish here two subtypes of DIC, *acute DIC* and *chronic or subacute DIC*. They are both described by the same attributes, that is they are both revealed from the same clinical tests, although the findings might be different. The physical findings associated with DIC permit to distinguish between this subtypes of DIC and they do affect the findings of the clinical tests [Schmaier 2001]:

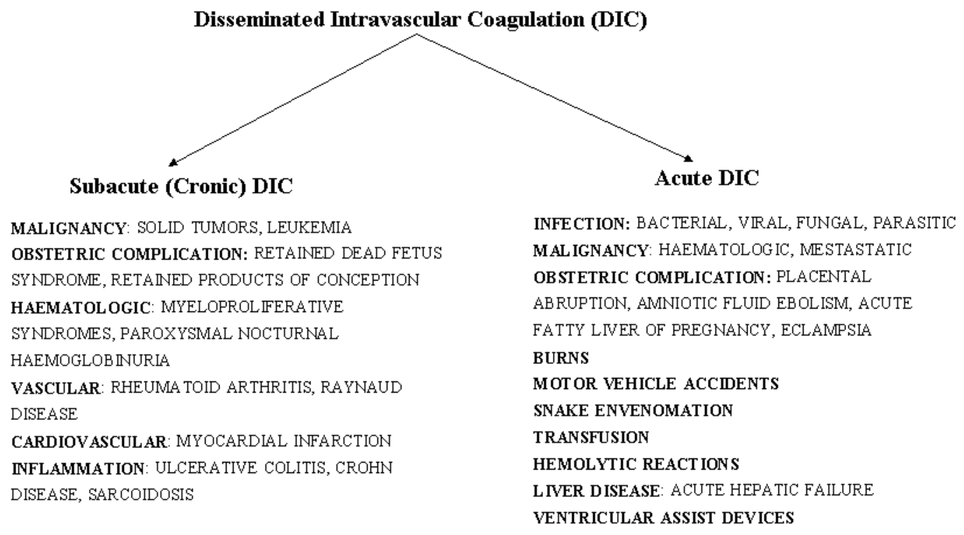
Disseminated Intravascular Coagulation (DIC)
Subacute (Cronic) DIC
Acute DIC
MALIGNANCY: SOLID TUMORS, LEUKEMIA
OBSTETRIC COMPLICATION: RETAINED DEAD FETUS SYNDROME, RETAINED PRODUCTS OF CONCEPTION
HAEMATOLOGIC: MYELOPROLIFERATIVE SYNDROMES, PAROXYSMAL NOCTURNAL HAEMOGLOBINURIA
VASCULAR: RHEUMATOID ARTHRITIS, RAYNAUD DISEASE
CARDIOVASCULAR: MYOCARDIAL INFARCTION
INFLAMMATION: ULCERATIVE COLITIS, CROHN DISEASE, SARCOIDOSIS
INFECTION: BACTERIAL, VIRAL, FUNGAL, PARASITIC
MALIGNANCY: HAEMATOLOGIC, MESTASTATIC
OBSTETRIC COMPLICATION: PLACENTAL ABRUPTION, AMNIOTIC FLUID EBOLISM, ACUTE FATTY LIVER OF PREGNANCY, ECLAMPSIA
BURNS
MOTOR VEHICLE ACCIDENTS
SNAKE ENVENOMATION
TRANSFUSION
HEMOLYTIC REACTIONS
LIVER DISEASE: ACUTE HEPATIC FAILURE
VENTRICULAR ASSIST DEVICES

Figure 5.1: The primary conditions associated with acute and subacute (or chronic) DIC

- **Acute DIC**: This is an acute haemorrhagic disorder which is associated with excess plasmin formation. Patients with acute DIC have petechiae on the soft palate and legs from persistent decrease in the number of platelets (*thrombocytopenia*) and ecchymosis at venipuncture sites. These patients also present with ecchymosis in traumatised areas;

- **Chronic or subacute DIC**: It is an indolent chronic disorder that is not associated with bleeding and presents as thrombosis as result of excess thrombin formation. It manifests with symptoms and signs of venous thrombosis.

The primary disorders associated with acute and subacute DIC are shown in Figure 5.1. These are important because they determine the events that can modify the attribute behaviour over time in the modelling example in the Section 5.3.

5.2 The model of DIC

In order to model the concept DIC we have partially reused part of the hierarchy of concepts present in MeSH [Nelson *et al.* Forthcoming]. MeSH is a taxonomy of medical terms, and so no attributes were defined for the concepts. We have partially modified the hierarchical structure and have associated attributes with the concepts. In particular, we have described the disorders from the viewpoint of the medical tests necessary to diagnose the disorder. The hierarchy of concepts and the properties used to describe it is shown Figure 5.2. The hierarchy shows the *subclass* relationships holding between the concepts. In the picture the properties characterising the concepts are described by associating a value with an attribute which is one of the clinical tests that can be used to diagnose haematologic diseases. The values associated with the attributes are those that are usually found in individuals who are affected by a kind of haemathologic disease, but other values are admissible as well.

This portion of hierarchy is modelled in Appendix B using the knowledge model presented in the previous chapter. In the modelling example in Appendix B the hierarchy above has been enriched by attaching to the properties characterising the concepts the additional information concerning the properties' behaviour over time, their degree of applicability to subconcepts, their prototypical and exceptional values.

We assume that the concept `Blood-Coagulation-Disorder` is considered to inherit all the attributes from its ancestor `Haemathologic-Disease`. Decimal measurements are supposed to be with two decimal figures. An event that can cause the platelet aggregation time to change is the external temperature, but since we have not associated a specific value to the slot, the state component of the :VALUE-CHANGE-EVENT is set equal to the set of all the possible values associated with the slot. When an event has a duration, such as a disease, we always use the start of the event. Therefore when we write (inherited-proteinC-deficiency, [0, 59], after, I) we

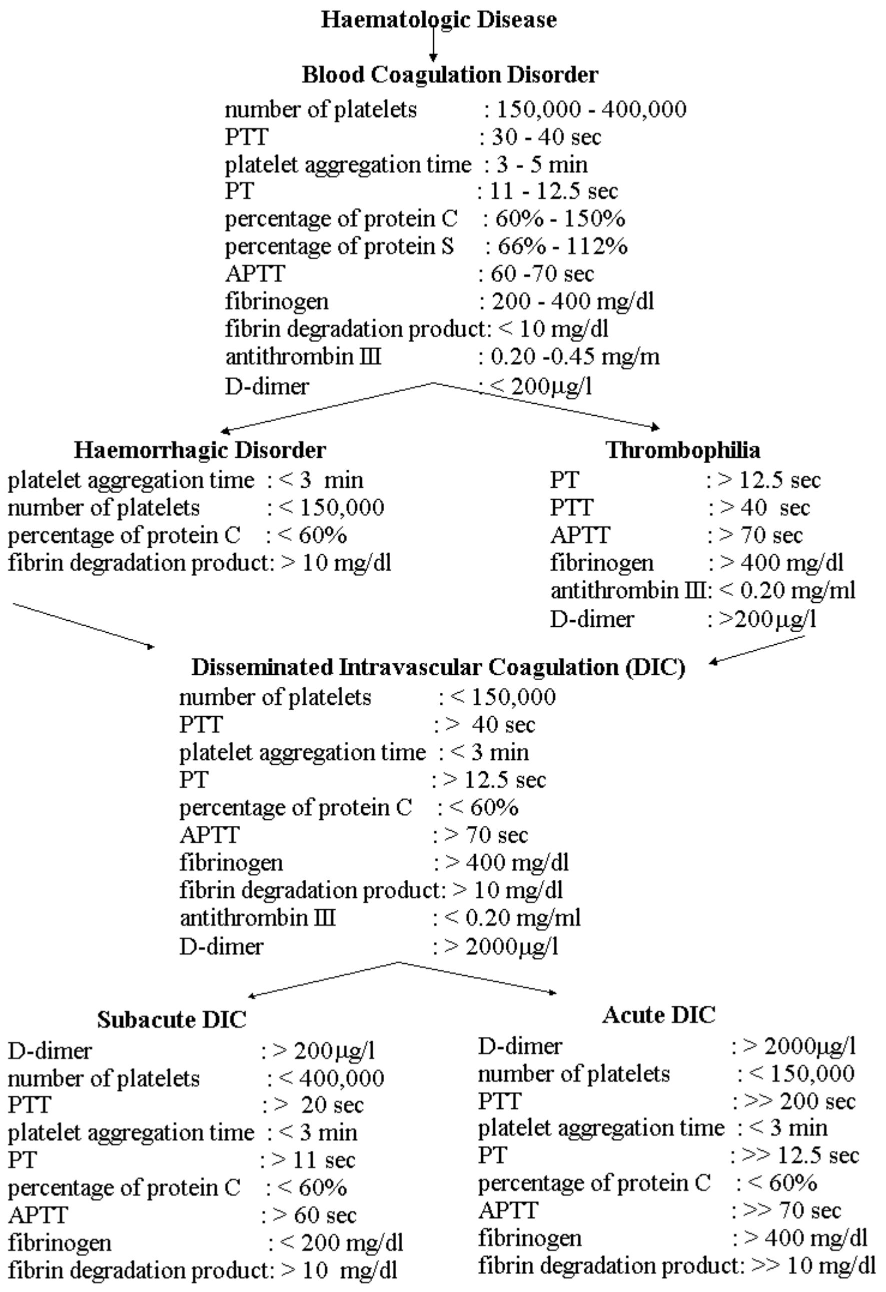
Haematologic Disease
Blood Coagulation Disorder
number of platelets : 150,000 - 400,000
PTT : 30 - 40 sec
platelet aggregation time : 3 - 5 min
PT : 11 - 12.5 sec
percentage of protein C : 60% - 150%
percentage of protein S : 66% - 112%
APTT : 60 -70 sec
fibrinogen : 200 - 400 mg/dl
fibrin degradation product: < 10 mg/dl
antithrombin III : 0.20 -0.45 mg/m
D-dimer : < 200μg/l
Haemorrhagic Disorder
platelet aggregation time : < 3 min
number of platelets : < 150,000
percentage of protein C : < 60%
fibrin degradation product: > 10 mg/dl
Thrombophilia
PT : > 12.5 sec
PTT : > 40 sec
APTT : > 70 sec
fibrinogen : > 400 mg/dl
antithrombin III: < 0.20 mg/ml
D-dimer : >200μg/l
Disseminated Intravascular Coagulation (DIC)
number of platelets : < 150,000
PTT : > 40 sec
platelet aggregation time : < 3 min
PT : > 12.5 sec
percentage of protein C : < 60%
APTT : > 70 sec
fibrinogen : > 400 mg/dl
fibrin degradation product: > 10 mg/dl
antithrombin III : < 0.20 mg/ml
D-dimer : > 2000μg/l
Subacute DIC
D-dimer : > 200μg/l
number of platelets : < 400,000
PTT : > 20 sec
platelet aggregation time : < 3 min
PT : > 11 sec
percentage of protein C : < 60%
APTT : > 60 sec
fibrinogen : < 200 mg/dl
fibrin degradation product: > 10 mg/dl
Acute DIC
D-dimer : > 2000μg/l
number of platelets : < 150,000
PTT : >> 200 sec
platelet aggregation time : < 3 min
PT : >> 12.5 sec
percentage of protein C : < 60%
APTT : >> 70 sec
fibrinogen : > 400 mg/dl
fibrin degradation product: >> 10 mg/dl

Figure 5.2: The hierarchy of concepts described in the modelling example

mean that a condition such as *inherited protein C deficiency* causes a low level of percentage of protein C, and this happens just before the start of the deficiency.

5.3 Discussing the model

The concepts in the hierarchy shown above are described by a number of attributes which permit the characterisation of the concept from a specific viewpoint. We are assuming here that the ontology we are modelling is the result of the integration of two different ontologies, one concerning *thrombophilia* and the other *haemorrhagic disorders*, that all the problems of syntactic heterogeneity have been reconciled and that we are in the phase where inconsistencies due to ontological heterogeneity are detected and resolved.

In this particular example, the conditions are described from the viewpoint of the symptoms which underlie them and are shown by a specific clinical test. It is important to note that the values here are usually indicated as ranges because we are oversimplifying the problem. Indeed, when practitioners describe the symptoms of a condition they tend to use qualitative rather than quantitative measures. Here we have tried to translate qualitative measures into numerical values, and these translations might not reflect the viewpoint of a medical expert. So, for example, we have translated *low number of platelets* as a range between 0 and 150,000, however the boundaries of this range are not fixed. So, the representation of this range as a closed interval is just an over simplification.

A condition is revealed by a combination of tests which give a positive result. Concepts are described in terms of medical tests: for each test we provide the prototypical values that the test might show *in the context determined by the condition which is being described*, the exceptional values that can be associated with the test (which in this case might represent the tests results typical for an individual who is *not* affected by the condition), and we describe the degree of applicability of the attribute to the subconcepts, and the attribute behaviour over time.

It is important to note that we assume *inheritance with exceptions* to hold in this structure, which is shown by inheriting both the slots and the pairs *slot-attributes*. Thus, we assume that all slots are inherited down the hierarchy, although, since inheritance with exception holds, a frame describing a subconcept might override the set of slots describing it, by adding or removing some slots. In the same way, the values associated with the facets describing the slot may be overridden when inherited by a more specific concept on the hierarchy. For example, the facet `Haematologic-Disease` is described by the slot `percentage-of-protein-c` whose prototypical value is a percentage of inhibition in the range [60, 150] . These are the prototypical values for an healthy individual, because the concept `Haematologic-Disease` is so high in the hierarchy that it is not possible to associate a specific range of values with the slots describing it. The percentage of protein C is a volatile type of attribute in that it can change more than once during the patient history and can, for example, decrease noticeably if the patient is affected by inherited protein C deficiency. If we consider its direct descendant `Haemhorrhagic-Disorder`, this inherits most of the slots from its parent frame, but with exceptions. For example, the percentage of protein C in this case is known to be low, so its prototypical values are those in the range [0, 59]. Exceptional values are those outside this range. The slot `percentage-of-protein-c` does not inherit the behaviour of being modified by inherited protein C deficiency, because this event is a type of haemhorragic disorder.

It is also important to note that all slots describing the frame `Haematologic-Disease` are characterised by associating the value `possible` with the facet :VALUE-MODALITY. This is because the concept described at the frame is quite high in the hierarchy of the disorders. Going down the hierarchy the filler associated with this facet changes, showing that refining the concept description we discover properties that are necessary for class membership. Indeed, if we consider both subconcepts of `Haematologic Disease`, that is, `Haemorrhagic-Disorder` and `Thrombophilia`, the slots associated with these concepts are described by the filler `most` associated with the facet `:value-modality`. This is because it is likely (but not certain) that a specific type of either *haemorrhagic disorder* or *thrombophilia* is characterised by the same values of clinical tests which are associated with the parent concept, although exceptions are possible. Different disorders are distinguished on the basis of the different combinations of slots they inherit from the parent concept, that is, they might be revealed by different combination of clinical tests.

Some of the values associated with a slot are the findings of clinical tests which are true only for most of the patients affected by the condition. This is the case for the values of `PT` and `APTT` whose values are prolonged for 50% to 70% of the patients affected by *acute DIC*. This is reflected in the model by associating the filler `most` with the facet :VALUE-MODALITY, to indicate that most of the instances of the concept will take the prototypical value.

When we consider the concept `Disseminated Intravascular Coagulation` the degree of applicability of property to subconcepts is mainly described by associating the filler `most` to the filler :VALUE-MODALITY, because the combination of clinical tests describing the concept `Disseminated Intravascular Coagulation` is highly likely to be inherited by the subconcept ACUTE-DIC and SUBACUTE-DIC, with some restrictions on the value. This is true for all slots but two, NUMBER-OF-PLATELETS, whose value is in the range [0, 150,000] in all cases of *acute DIC*, and PERCENTAGE-OF-PROTEIN-C, whose value remains low both for `Acute-DIC` and `Subacute-DIC`. In both concepts, the property of having a low platelet number is *rigid*, because their modality is set to `All` and the platelet count cannot change over time in these two concepts.

It is worth noting that in this example there are no necessary conditions and no distinguishing attributes. The reason for this is that DIC is an extremely complex condition to model and there is no ultimate test that can clearly indicate whether the patient is affected by the condition or not. This is reflected in the model by labelling most of the properties as possible, or, in some cases as inherited by most of the subclasses.

Finally, the concept DIC inherits from both its parents (that is thrombophilia and haemorrhagic disorder). Multiple inheritance is dealt with by assuming that the child concepts inherits from the frames describing the parents, the slot values that are more specific. Of course, here we are assuming that the parents are disjoint. This is the case with thrombophilia and haemorrhagic disorder: these must have disjoint slot values, and so DIC inherits from the parent with the most specific description. The role of events is quite important because it shows how the attributes can change and the effects that these changes can have on the concept such as DIC. In the modelling example above we have also tried to associates with volatile attributes those events that can cause the attribute to change value *in the specific context of diagnosing DIC*. In particular, we have simplified the events by grouping them in categories such as *obstetric complications* or *trauma*, instead of listing each of them singularly. These events are not always inherited down the hierarchy, but only when they are relevant. Events are particularly valuable in describing this condition because DIC is a *secondary* condition, that appears only depending on a primary condition. Therefore, by giving an explicit list of the events that can cause the characterising properties (modelled in the slots) to change we are also providing a way to list explicitly the primary conditions that can be aggravated by DIC. This can be seen with the values associated with the slot `fibrinogen`. In fact, in DIC usually the level of *fibrinogen* is decreased, but this is an acute phase reactant and so its values may be initially elevated secondary to the primary disease. This is modeled by associating with the prototypical value facet of the slot `fibrinogen` the normal values (that is those between 200 and 400) and by stating that a high level of fibrinogen is registered *after* the beginning of one of the events listed in the `:value-change-events` filler. By listing the events we can distinguish the concepts of acute and subacute DIC also based on the fact that the properties of the former can change because of trauma and burns, whereas this is not true for the latter.

Associating the events with the attributes provides a better characterisation of the properties, and not only of their behaviour over time. Indeed, by associating some events with the pairs *slot-frame* we are actually providing a more thorough definition of the concept described in the frame. This can be seen in the two frames *Acute DIC* and *Subacute DIC*. Although these conditions are described by the same slots, the slots can take different values (for example the number of platelets may be normal for subacute DIC while is always decreased for acute DIC) and their behaviour over time is modified by different events, which shows the fact that *trauma* is an important event in *Acute DIC* in that can cause a change in the values of the slots associated with this frame, whereas this event is less important when describing the *Subacute DIC*.

5.4 Chapter summary

In this chapter we have provided an example in which we have modelled a complex domain such as the one of Disseminated Intravascular Coagulation. The conceptual model that is the object of this thesis has proven to be particularly valuable to model the interaction of different factors that contribute to the definition of this disease, such as the primary conditions on which DIC depends. The richness of the model permits us to have a more complete snapshot of the semantics of the concept while it is defined, and shows clearly which properties are transmitted down the hierarchy, whether the inheritance is strict or with exceptions, what are the properties that are permitted to change over time and why that change.

One of the main drawbacks that we have noticed in preparing this example is that in order to use the conceptual model it is necessary to have a deep and thorough knowledge of the domain that is being modelled, including also the interaction between events and attributes. This in turns requires a richer top-level ontology that includes concepts such as processes and a rich temporal ontology. These are needed if we want to give a more precise representation of the events.

Chapter 6

Conclusion

6.1 Thesis summary

The research presented in this thesis has focused on an enriched ontology metamodel to support an alternative approach to knowledge sharing. Two research threads have been followed in this thesis, a primary and a secondary one. The primary research thread has concentrated on an enriched ontology model which provides a precise characterisation of the attributes used to define concepts in the ontology. This characterisation is based on a multidisciplinary theoretical background which includes the formal tools of ontological analysis (namely *identity, rigidity*, *unity* and *dependence*), on the cognitive notions of prototypes and exceptions, the notion of *modality*, and on the notion of inherited and distinguishing concept properties. This ontology conceptual metamodel has been developed to support the assessment of semantic similarity in the structure of multiple ontologies which is the object of the secondary research thread.

We have analysed the approaches to knowledge sharing, and we have reached the conclusion that current approaches (one to one and single shared ontology approaches, as reviewed in Section 3.6) present weaknesses, especially when knowledge sharing has to be achieved in an open environment, which was one of the initial requirements of this thesis. This has led us to the devise an alternative approach based on locating the shared knowledge in a structure of multiple shared ontologies, which are hierarchically organised and can represent knowledge at different levels of abstraction. Knowledge sources do not have to commit to a single overarching shared ontology, but a group of knowledge sources (or agents) which share a specific understanding of the domain commits to the shared ontology expressing a view which is closer to the way knowledge sources conceptualise the domain.

In pursuing these two research threads we have first reviewed the theoretical foundation of ontologies, in Chapter 2.
In this chapter we have presented the philosophical discipline of *ontology* and the AI discipline of *ontologies*, relating the second to the first, we have given an account of the different connotations that ontologies take in AI and of the different types of ontologies. We have also reviewed the contribution that philosophy has given to the ontological field, particularly concentrating on the notion of formal ontology.

The motivation for devising a novel approach to knowledge sharing emerged by the analysis of the current approaches. The result of such analysis is presented in Chapter 3, where we presented an overview on the problem of knowledge sharing and reuse, particularly focussing the attention on the possible approaches that use ontologies to solve this problem. In this chapter we have also analysed the problems caused by different types of heterogeneity that can hamper the sharing and reuse of knowledge.
After presenting here the different approaches to knowledge sharing we introduced the novel approach based on multiple shared ontologies, and we discussed some of the issues arising from this approach.

The ability to build this structure of shared ontologies semi-automatically is based on the ability to group together "similar concepts", which in turn depends on the ability to assess semantic similarity among the concepts in the different ontologies. Current similarity measures are usually binary, that is, they typically return a Boolean value which indicates whether the two assessed concepts are similar or not. Furthermore, these similarity measures do not usually take into account the concept's definition in terms of attributes, but only some lexical similarity among concepts' names. For these reasons they are not suitable to be used in the process of building ontology clusters. A more recent approach to assess semantic similarity for building shared ontologies [Rodríguez and Egenhofer 2002] is based on Tverskian similarity functions [Tversky 1977], which depends on the notion of *features* and *distinguishing features*. Such an approach, however, needs an extended the concept's definition in order to accommodate features (usually modelled by attributes) and some extra semantics that provides the ability to distinguish between features.

In this thesis we have decided to take an approach analogous to the one in [Rodríguez and Egenhofer 2002], and thus we have developed an ontology conceptual metamodel, which encompasses a metalevel modelling the behaviour of concept properties in the concept definition and over time into *attribute metaproperties* (namely, *Mutability, Mutability Frequency, Reversible Mutability, Event Mutability, Modality, Prototypicality, Exceptionality, Inheritance, Distinction*). This conceptual metamodel has become the primary research thread of this thesis, and it has been implemented in a knowledge metamodel that extends the usual ontology frame-based models, such as OKBC, by explicitly representing additional on the slot properties. The set of attribute metaproperties we define in this thesis may help to deal with ontology heterogeneity problems in two ways. On the one hand the model complements the set of formal ontological properties proposed in [Welty and Guarino 2001], namely *Identity, Unity, Rigidity,* and *Dependence*. Our set of metamproperties can assist in assigning to concepts the concept metaproperties defined by Guarino and Welty. This might result particularly useful when knowledge engineers need to assign formal properties to ontologies that they have not designed.

On the other hand, the extra semantics for concept descriptions which is provided by the conceptual metamodel can be used to distinguish among features and to implement some kind of Tverskian similarity function, although the actual implementation of the function is out of the scope of this thesis. Furthermore, this conceptual metamodel for ontologies facilitates a better understanding of the semantics of concepts. Currently ontology merging is performed by hand based on the expertise of knowledge engineers and on the ontology documentation. Even in this case the ontology metamodel we propose can prove useful by providing a characterisation of the properties, which can help to identify semantically related terms or to disambiguate among concepts that only "seem similar", on the assumptions that two concepts are similar if they present a similar concept description (that is, if they are described by similar attributes) and these attributes show the same pattern of behaviour in the concept definition and over time.

The novelty of this extended knowledge metamodel is that it explicitly represents the behaviour of attributes over time by describing the changes in a property that are permitted for members of the concept. It also explicitly represents the class membership mechanism by associating with each slot a qualitative quantifier representing how properties are inherited by subconcepts. Finally, the model does not only describe the prototypical properties holding for a concept but also the exceptional ones. No previous work on ontology has addressed the problem of providing a precise characterisation of attribute properties.
The ontology model presented in Chapter 4 has been used to have modelled a complex domain in medicine, that is Disseminated Intravascular Coagulation (DIC). The conceptual model that is the object of this thesis has proven to be particularly valuable in modelling the interaction of different factors that contribute to the definition of this disease, such as the primary conditions on which DIC depends. The richness of the model permits to have a more complete description of the semantics of the concept while it is defined, and shows clearly which properties are transmit-

ted down the hierarchy, whether the inheritance is strict or with exceptions, what are the properties that are permitted to change over time and why.

One of the main criticisms to the use approach that we have noticed in preparing this example is that in order to use the conceptual model it is necessary to have a deep and thorough knowledge of the domain that is being modelled, deeper than the knowledge usually modelled in ontologies. We do not perceive this as a serious drawback, in fact building ontologies is a very demanding task which is not going to be affected in a major way by the extensions we have suggested in this thesis.

Knowledge concerning a domain includes also the interaction between events and attributes. This in turns requires richer top-level ontologies that include concepts such as processes and a rich temporal ontology. These are needed if we want to give a more precise representation of the events.

6.2 Results

We believe this thesis gives two contributions to the field of ontologies and knowledge sharing. The first is, certainly the enriched ontology conceptual metamodel. Up to this moment ontology models have concentrated their attention on concepts and on their interrelationships in the ontology. Properties of the concepts are described by associating specific values with the attributes defining the concept. The main contribution of this thesis is that it provides a precise characterisation of the attributes in terms of their behaviour over time, their degree of applicability to subconcepts, their being prototypical or exceptional, inherited or distinguishing. This characterisation is modelled in a set of metaproperties (*Mutability, Mutability Frequency, Reversible Mutability, Event Mutability, Modality, Prototypicality, Exceptionality, Inheritance, Distinction*) that constitues the metalevel on the traditional conceptual model.

We have argued that such a precise characterisation might help to disambiguate among concepts that only seem similar, and in turn can support mappings across the structure of multiple shared ontologies that we have devised as alternative to the current approaches to knowledge sharing. We claim that this characterisation of the concept's properties is also very important in order to provide a precise specification of the semantics of the concepts. Such characterisation is essential if we want to perform a formal ontological analysis, in which knowledge engineers can precisely determine which formal tools they can use in order to build an ontology which has a taxonomy that is clean and not very tangled.

The novelty of this characterisation is that it explicitly represents the behaviour of attributes over time by describing the permitted changes in a property that describe a concept. It also explicitly represents the class membership mechanism by associating with each attribute (represented in a slot) a qualitative quantifier representing how properties are inherited by subconcepts. Finally, the model does not only describe the prototypical properties holding for a concept but also the exceptional ones. By providing this explicit characterisation, we are asking knowledge engineers to make more hidden assumptions explicit, thus providing a better understanding not only of the domain in general, but also of the role a concept plays in describing a specific domain.

The second result we have achieved in this thesis is the structure of multiple shared ontologies for knowledge sharing. Although this has not been a primary direction of research during this thesis, we believe that such a structure has advantages over the others especially if considered in the context of an open environment such as the Internet. We believe that this kind of modularisation is the key to applications where intelligent agents (whose knowledge is represented by ontologies) interoperate dynamically, by agreeing on the vocabulary (and shared knowledge) which is closer to the conceptualisations of *only those agents which are involved in the interoparation* and not of all agents that can be potentially involved. We realise that we have not investigated in sufficient detail the issues related to building such a structure in an efficient and cost effective manner, and the relationships existing within and between the ontologies composing the structure (both topics are future research directions that we will consider, see next section); but we think that we have laid the basis for future research.

6.3 Future research

There are several issues that stem from the research presented in this thesis. As mentioned in the previous section one of the issues to investigate further concerns the relationships between and within ontologies, which need to be clarified with respect to previous work presented in the literature. Two candidate sets of relations have been identified, these are Borst's *ontology projections*: include and extend, include and specialise, include and map [Borst 1997]; and Visser and Cui's *ontology relations*: subset/superset, extension, restriction, mapping [Visser and Cui 1998].

Another issue emerging from this research is how knowledge sources (or agents), reach consensus on which cluster in the structure of multiple shared ontologies they have to join in order to achieve interoperation. This kind of consensus should be based on suitable similarity measure, that takes into account the semantics of the concepts involved, and the semantics of their properties. There are no similarity functions of this type, that we are aware of, and it would be interesting to investigate complex similarity measures, such as those for symbolic objects [Esposito *et al.* 2000]. We are particularly interested in investigating similarity functions that make use of the extra semantics provided by the conceptual metamodel, in a way analogous to the similarity measure presented in [Rodríguez and Egenhofer 2002]. These kind of similarity functions usually provide a measure of the degree of similarity among different concepts, and not just a binary measure that indicates whether two concepts are similar or not.

Similarity measures are also important to determine the way in which knowledge sources (or agents) are grouped together to form a cluster. We are exploring the machine learning direction, which seems the more promising one, but more has to be done. One of the obstacles is the lack of efficient clustering techniques, and also the techniques currently available have to be adapted in order to include the notion of semantic proximity.

Another challenge is the idea than an ontology could actually 'learn' a new concept when it encounters it. We have assumed throughout this thesis that ontologies are static, and that any inclusion of new concepts has to be done manually and off-line. But this assumption is not realistic if ontologies have to be used in applications such e-commerce, where a dynamic response to the changes of the environment is needed. Therefore, learning techniques are needed to incorporate a new concept in an ontology when this concept is recognised to be relevant.

From the viewpoint of the ontology conceptual metamodel, future work includes a more formal characterisation of the attribute's behaviour over time, and particularly a formal characterisation of the dynamic of process changes. We aim to investigate a more sophisticated and formal representation of changes, which permit to apply some form of temporal reasoning to reason about the events that can modify the values associated with an attribute.
The reasoning mechanisms that are supported by the additional semantics included in the ontology metamodel should be explored as well, to understand the kind of inferences supported by this model. In order to support complex reasoning inferences, we will consider the implementation of the metamodel in some description logic based language, which should provide the capabilities to perform the inferences.

This model is also quite demanding to use, future work should concentrate also on identifying the kinds of applications that can benefit from the expressive power provided by this model.

In order to test the effectiveness of the conceptual metamodel, we are planning to include the metaproperties in tools to build ontologies such as WebOde [Arpírez *et al.* 2001] or Protégé [Fridman Noy *et al.* 2000].

Appendices

Appendix A

Knowledge sharing in InfoSleuth, KRAFT, and OBSERVER

There are many projects that can be discussed to illustrate the framework, here we focus on three of them: InfoSleuth, KRAFT, and OBSERVER.

A.1 InfoSleuth

InfoSleuth [Perry *et al.* 1999]. is a system for the integration of heterogeneous sources developed by MCC (Microelectronics and Computer Technology Corporation, Austin, Texas, USA). The purpose of the InfoSleuth project is to retrieve and process information in a network of heterogeneous information sources (also called resources). In InfoSleuth, the heterogeneity concerns three issues: the paradigms used to represent the knowledge (also referred to as schema heterogeneity); the languages used to represent the knowledge and the conceptualisation underlying the schema. The different sources are integrated in a dynamic way and this is made possible by using a network of co-operating agents that form the InfoSleuth architecture. The InfoSleuth architecture includes both core and application dependent components. A core application provides fundamental services, they are:

- User Agent: This agent allows the user to access the InfoSleuth system. It obtains information about the ontologies known to the system and it uses them to prompt its user in selecting an ontology that will be used to formulate queries. Each of these is sent to the most appropriate task execution agent (fee below) that will send the obtained results to the user agent.

- Resource Agent: This agent allows the InfoSleuth architecture to access the information sources and executes the requests concerning a specific resource. The resource agent answers queries translating them from the common query languages into a language understood by the resources. This translation comprises both the mapping of the shared ontology into database schema, and the mapping of the query language into the native language.

- Ontology Agent: This agent is a specialised Resource Agent whose main task is to answer questions about ontologies. It answers queries about the ontologies available, such as the source of an ontology and searches the ontologies for concepts.

- Broker Agent: This agent aims at finding the resources required to solve a user query. All InfoSleuth agents advertise their capabilities to the broker agent that semantically matches agents looking for a particular service with agents providing that particular service (information brokering technique). At least, an agent has to advertise its name, its location and its language, but it can also advertise meta-information and domain constraints. The advertisement is expressed in terms of one or more ontologies thus enabling the dynamic matching.

- Task Execution Agent: This agent routes requests to the appropriate Resource Agents. It decomposes user queries into sub-queries and reassembles the answers, thus co-ordinating the executions of high-level information gathering sub-tasks. The strategy followed is based on task plan with procedural attachments.

The application dependent components of the InfoSleuth architecture contribute only to some applications. They are:

- Data Analysis Agent: This agent performs data analysis/mining operations.

- Monitor Agent: This agent stores records of the agent interactions and of the task execution steps. The co-operation between multiple agents is obtained by using the information brokerage technique that routes all the requests only on to the relevant resources. Information brokerage and ontologies are two aspects of the InfoSleuth approach strictly intertwined. Agent communications take advantage of the use of ontologies as they are used to the agent infrastructure (this is done by specifying the information and the relationships between the various agents). This (facilitates) aids the routing of the requests to a specific agent. InfoSleuth allows different formats and representations of ontologies by the use of an ontology meta-model that provides a unified view on the way ontologies are specified. In this way agents might reason about ontologies using different languages depending on the type of inference to be made.

A.2 KRAFT

KRAFT (Knowledge Reuse and Fusion / Transformation) [Preece *et al.* 2001] is a multi-site research project conducted at the universities of Aberdeen, Cardiff and Liverpool in collaboration with BT (British Telecommunications PLC) in the UK. The overall aim of this project is to enable the sharing and reuse of constraints embedded in heterogeneous databases and knowledge systems. In the KRAFT approach to the integration problem there are three types of heterogeneity: ontological assumptions (conceptualisations and organisations of the data), paradigm and language. KRAFT recognises a small number of shared ontologies. Moreover each resource has its own local ontology, and provides a translation to at least one shared ontology; in this way local ontologies allow the communication between heterogeneous resources that can maintain their intrinsic heterogeneity. The KRAFT network has the following components:

- User Agent: is the interface between users and services provided by KRAFT domain;
- Resource: is the knowledge source to integrate. It provides services to the KRAFT domain. Examples of KRAFT resources are databases, knowledge bases and constraint solvers.
- Wrapper: is the interface between the domain and the user agent or the resources. Wrappers provide communication services, both at high and at low level. At high level they support the mechanisms linking the resources to mediators and facilitators (see below). At low level they provide a translation service between the internal data formats of users agent and resources and the internal data format supported by the KRAFT domain. They co-operate with the ontology agent (see below) to perform translations.
- Mediator: is the component that retrieves information about a domain. In achieving this purpose it uses domain knowledge to transform data. It performs operations on queries to implement a certain task and can process queries by decomposing, combining them and transforming their content.
- Ontology Agent: is the component that translates knowledge expressed against a source ontology into the knowledge expressed against a target ontology. If a mediator or a wrapper requires an ontology translation it passes the expression and references to both source and target ontologies to the ontology agent who will translate and return the expression.
- Facilitator: is the KRAFT component performing the internal routing services for messages within the KRAFT domain. Its main functions are to maintain records of the location and of the capabilities of the resources, and to accept and route messages from other KRAFT resources.

A.3 OBSERVER

OBSERVER (Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution) [Mena *et al.* 2000] is a project which was conducted at the University of Zaragoza, Spain. The aim of the OBSERVER project is to retrieve and process information stored in heterogeneous knowledge sources (called repositories). The heterogeneity in this project concerns paradigms and ontological assumptions. To overcome the differences in the formats and in the languages OBSERVER relates repositories to domain ontologies; these are pre-existing ontologies defining a set of terms in a specific domain. The OBSERVER architecture comprises four main components:

- Query processor: This component has as input a user query expressed in a chosen user ontology. The query processor accesses the data repositories to answer the query. If the user is not satisfied with the answer, the query processor translates (partially or totally) the query into another user-selected ontology using predefined inter-ontology relationships. The query processor generates a list of translation plans, where each plan has an associated loss of information.

- Ontology server: This component provides the user processor with mappings that link each term in an ontology with structures in data repositories and it translates queries for the retrieval of data from the repositories. In the access the ontology server is assisted by the wrapper (see below) of the corresponding data repository.

- Interontology Relationships Manager (IRM): This component deals with inter-ontology relationships that relate terms in different ontologies. OBSERVER considers three kinds of possible relationships: synonym, hypernym and hyponym.

- Wrapper: This component has knowledge of the data organisation in the repositories. The wrapper actually accesses the data repository using the mapped information provided by the ontology server.

The processing is performed according to the following steps: First users choose one domain ontology whose term will be used to build the query. Once the query is formulated, the ontology server verifies its syntax, then it performs ontological transformations of the query, and decomposes it. After the decomposition, the ontology server uses relevant mappings rules to relate terms in the ontology to the data structure in the underlying repositories. In accessing the repository to retrieve a queried data, the ontology server is assisted by the wrapper. Once the data is retrieved, the ontology server returns the user with the answers obtained. If the user is not satisfied with the answer, the query processor reformulates the query using another user chosen ontology.

Appendix B

The knowledge model applied to the concept of DIC

B.1 Introduction

In this Appendix we have modelled the portion of hierarchy in Figure 5.2 according to the knowledge model given in Section 4.5. It is worth pointing out that the aim of this example is not to provide an exercise of modelling, but a proof of concepts for the conceptual model that is the object of this thesis. In order to do so, we have over simplified the domain and some of the information below might not be completely accurate from the medical viewpoint.

In defining the concepts we follow some conventions: all words used to name concepts and slots are separated by dashes, class names are capitalised whereas we use low-case letters for slot and facet names. Other assumptions were made while modelling, for example, we have determined the upper bound of those slots with numeric values as being 20 times the upper bound of the so-called normal values. So, for example, the normal *platelet aggregation time* is between 3 and 5 minutes, so we have considered the facet :NUMERIC-MAXIMUM describing the slot `platelet-aggregation-time` to be 100. The numeric minimum is usually 0, because negative values are impossible for those tests that measure the time of some phenomenon or the quantity of some substance. In some cases the fillers associated with the facets were either infinite or some finite set. In these cases the values were denoted using a matematical notation (either a set notation or an interval notation when possible).

We have provided a precise characterisation of attributes, including their behaviour over time, for the concepts `Disseminated-Intravascular-Coagulation`, `Acute-DIC` and `Subacute DIC` only. In most of the other cases we have indicated by none the absence of any relevant event that might modify the value associated with a slot.

B.2 Frame Descriptions

Frame `Blood-Coagulation-Disorder`

:SUBCLASS-OF `Hemathologic-Disease`

Template-Slot `number-of-platelets`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1,000,000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [150,000, 400,000]

Template-Facet :VALUE-EXCEPTIONS [0, 149,999] $\cup$ [400,001, 1,000,000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `ptt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 800

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [30, 40]

Template-Facet :VALUE-EXCEPTIONS [0, 29]∪[41,800]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `platelet-aggregation-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [3.01, 4.99]

Template-Facet :VALUE-EXCEPTIONS [0, 3] ∪ [5, 100]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS
(change-external-temperature, [0, 100], after, R),

Template-Facet :VALUE-CHANGE-EVENTS
(use-of-aggregation-stimulator-drugs, [0, 100], after, R)

Template-Slot `percentage-of-protein-C`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 3000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [60, 150]

Template-Facet :VALUE-EXCEPTIONS [0, 59] ∪ [151, 3000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(inherited-proteinC-deficiency, [0, 59], after, I)

(excess-of-weight, [0-59], after, R)

(Vitamin-K-Deficiency, [0, 59], after, R)

(Cancer, [0, 59], after, I)

(Infection, [0, 59], after, R)

(Vascular-Disorders, [0, 59], after, R)

Template-Slot `pt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :SLOT-NUMERIC-MAXIMUM 250

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES[11, 12.5]

Template-Facet :VALUE-EXCEPTIONS [0, 11.5] ∪ [13, 250]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTSnone

Template-Slot `percentage-of-protein-S`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :SLOT-NUMERIC-MAXIMUM 2240

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [66, 112]

Template-Facet :VALUE-EXCEPTIONS [0, 65] $\cup$ [113, 2240]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS
(inherited-proteinS-deficiency, [0, 59], after, I)

Template-Slot `activated-partial-thromboplastin-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :SLOT-NUMERIC-MAXIMUM 1400

Template-Facet :VALUE-LABEL inherited

Template-facet :VALUE-PROTOTYPES [60, 70]

Template-Facet :VALUE-EXCEPTIONS [0, 59] $\cup$ [71, 1400]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS
(proteinK-deficiency, [70, 1400], after, I)

Template-Slot `fibrinogen`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :SLOT-NUMERIC-MAXIMUM 8000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [200,400]

Template-Facet :VALUE-EXCEPTIONS [0, 199] ∪ [401, 8000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS
(bacterial-infection, [0, 19], after, R)
(eclampsia, [41, 8000], after, R)

Template-Slot `fibrin-degradation-products`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 10]

Template-Facet :VALUE-EXCEPTIONS [10.1, 100]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `antithrombinIII`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 90

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0.20, 0.45]

Template-Facet :VALUE-EXCEPTIONS [0, 0.19] ∪ [0.46, 90]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `D-dimer`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 5000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 200]

Template-Facet :VALUE-EXCEPTIONS [0, 199] ∪ [201, 5000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [200, 5000], after, R)

(Head-Trauma, [200, 5000], after, R)

(Cancer, [200, 5000], after, I)

(Obstetric-emergency, [200, 5000], after, R)

(Vascular-Disorders, [200, 5000], after, R)

Template-Facet :DOCUMENTATION-AT-FRAME

Own-Facet :FRAME-TYPE `Concept`

Frame `Haemorrhagic-Disorder`

:SUBCLASS-OF `Blood-Coagulation-Disorder`

Template-Slot `percentage-of-protein-C`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 3000

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPES [0, 59]

Template-Facet :VALUE-EXCEPTIONS [60, 3000]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `number-of-platelets`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1,000,000

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [0, 150,000]

Template-Facet :VALUE-EXCEPTIONS [150,001, 1,000,000]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `platelet-aggregation-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [0, 3]

Template-Facet :VALUE-EXCEPTIONS [3.1, 100]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(change-external-temperature, [0, 100], after, R),

(use-of-aggregation-stimulator-drugs, [0, 100], after, R)

Template-Slot `fibrin-degradation-products`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [10.1, 100]

Template-Facet :VALUE-EXCEPTIONS [0, 10]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Facet :DOCUMENTATION-AT-FRAME

Own-Facet :FRAME-TYPE `Concept`

Frame `Thrombophilia`

:SUBCLASS-OF `Blood-Coagulation-Disorder`

Template-Slot `pt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-PROTOTYPE [13, 100]

Template-Facet :VALUE-EXCEPTION [0, 12.99]

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-EXCEPTIONS none

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `ptt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 800

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [41, 800]

Template-Facet :VALUE-EXCEPTIONS [0, 40]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `activated-partial-thromboplastin-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1400

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [71, 1400]

Template-Facet :VALUE-EXCEPTIONS [0, 70]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `fibrinogen`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 8000

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [0, 400]

Template-Facet :VALUE-EXCEPTION [401, 8000]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `antithrombin-III`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 90

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPES [0, 0.20]

Template-Facet :VALUE-EXCEPTIONS [0.21, 90]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Slot `D-dimer`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 5000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 200]

Template-Facet :VALUE-EXCEPTIONS [0, 199] ∪ [201, 5000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS none

Template-Facet :DOCUMENTATION-AT-FRAME

Own-Facet :FRAME-TYPE `Concept`

Frame `Disseminated-Intravascular-Coagulation`
:SUBCLASS-OF `Haemorrhage-Disorder`, `Thombophilia`

Template-Slot `percentage-of-protein-C`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 3000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [0, 59]

Template-Facet :VALUE-EXCEPTIONS [60, 3000]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [20, 60], after, R)

(Trauma, [100, 3000], after, R)

(Cancer, [500, 3000], after, I)

(Obstetric-emergency, [10, 40], after, R)

(Vascular-Disorders, [40, 70], after, R)

Template-Slot `number-of-platelets`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1,000,000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 400,000]

Template-Facet :VALUE-EXCEPTIONS [400,001, 1,000,000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [0, 150,000], after, R)

(Trauma, [0, 150,000], after, R)

(Cancer, [0, 150,000], after, I)

(Obstetric-emergency, [0, 150,000], after, R)

(Vascular-Disorders, [0, 150,000], after, R)

Template-Slot `platelet-aggregation-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [0, 3]

Template-Facet :VALUE-EXCEPTIONS [3.1, 100]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(use-of-aggregation-stimulator-drugs, [0, 100], after, R)

(Infection, [0, 3], after, R)

(Trauma, [6.1, 17.0], after, R)

(Cancer, [3.1, 5.0], after, I)

(Obstetric-emergency, [7.1, 20.0], after, R)

(Vascular-Disorders, [0, 3.0], after, R)

Template-Slot `fibrin-degradation-products`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [10.1, 100]

Template-Facet :VALUE-EXCEPTIONS [0, 10]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [0, 10.0], after, R)

(Trauma, [10.1, 40], after, R)

(Cancer, [30.1, 60], after, I)

(Obstetric-emergency, [40.1, 80], after, R)

(Vascular-Disorders, [3.1, 15], after, R)

Template-Slot `pt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-PROTOTYPE [20,100]

Template-Facet :VALUE-EXCEPTION [0, 19.99]

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-EXCEPTIONS none

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [23, 40], after, R)

(Trauma, [5, 19.99], after, R)

(Cancer, [0, 15.99], after, I)

(Obstetric-emergency, [20.1, 40], after, R)

(Vascular-Disorders, [20.1, 70], after, R)

Template-Slot `ptt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 800

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [41,800]

Template-Facet :VALUE-EXCEPTIONS [0, 40]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [60, 120]after, R)

(Trauma, [10.1, 40], after, R)

(Cancer, [3.1, 60], after, I)

(Obstetric-emergency, [40, 240], after, R)

(Vascular-Disorders, [40, 210], after, R)

Template-Slot `activated-partial-thromboplastine-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1400

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [71, 1400]

Template-Facet :VALUE-EXCEPTIONS [0, 70]

Template-Facet :VALUE-MODALITY possilbe

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [120, 500], after, R)

(Trauma, [80.5, 125.5], after, R)

(Cancer, [20, 40.5], after, I)

(Obstetric-emergency, [40.1, 700], after, R)

(Vascular-Disorders, [60, 150], after, R)

Template-Slot `fibrinogen`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :SLOT-NUMERIC-MAXIMUM 8000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 400]

Template-Facet :VALUE-EXCEPTIONS [401, 8000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [401,8000], after, R)

(Trauma, [401, 8000], after, R)

(Cancer, [401, 8000], after, I)

(Obstetric-emergency, [401, 8000], after, R)

(Vascular-Disorders, [401, 8000], after, R)

Template-Slot `antithrombin-III`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 90

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 0.20]

Template-Facet :VALUE-EXCEPTIONS [0.21, 90]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Templ-Facet :VALUE-CHANGE-EVENTS

(Infection, [0.05, 0.18], after, R)

(Trauma, [1.0, 10], after, R)

(Cancer, [0.7, 3.5], after, I)

(Obstetric-emergency, [0.1, 30], after, R)

(Vascular-Disorders, [0.1, 15.0], after, R)

Template-Slot `D-dimer`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 5000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [2000, 5000]

Template-Facet :VALUE-EXCEPTIONS [0, 1999]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [2000, 4000], after, R)

(Trauma, [200, 1500], after, R)

(Cancer, [1900, 4800], after, I)

(Obstetric-emergency, [2400, 3500], after, R)

(Vascular-Disorders, [1800, 5000], after, R)

Template-Facet :DOCUMENTATION-AT-FRAME

Own-Facet :FRAME-TYPE `Concept`

Frame `Subacute-DIC`

:SUBCLASS-OF `Disseminated-Intravascular-Coagulation`

Template-Slot `percentage-of-protein-C`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 3000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [0, 59]

Template-Facet :VALUE-EXCEPTIONS [60, 3000]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [10, 59], after, R)

(Cancer, [0, 30], after, I)

(Obstetric-emergency, [30, 59], after, R)

(Vascular-Disorders, [20, 80], after, R)

Template-Slot `number-of-platelets`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1,000,000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 400,000]

Template-Facet :VALUE-EXCEPTIONS [400,001, 1,000,000]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY never

Template-Facet :VALUE-CHANGE-EVENTS

Template-Slot `platelet-aggregation-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPES [0, 3]

Template-Facet :VALUE-EXCEPTIONS [3.1, 100]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [0, 5], after, R)

(Cancer, [3, 10], after, I)

(Obstetric-emergency, [0, 3], after, R)

(Vascular-Disorders, [0, 1], after, R)

Template-Slot `fibrin-degradation-products`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited -with-exception

Template-Facet :VALUE-PROTOTYPE [10.1, 40]

Template-Facet :VALUE-EXCEPTIONS [0, 10] ∪ [41.1, 100]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [10.1, 30], after, R)

(Cancer, [5, 15], after, I)

(Obstetric-emergency, [30, 70], after, R)

(Vascular-Disorders, [20, 35], after, R)

Template-Slot `pt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [11, 25]

Template-Facet :VALUE-EXCEPTION [0, 10.9] ∪ [25.1-100]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [7, 20], after, R)

(Cancer, [0, 50], after, I)

(Obstetric-emergency, [25, 80], after, R)

(Vascular-Disorders, [1, 11], after, R)

Template-Slot `ptt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 800

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [20, 30]

Template-Facet :VALUE-EXCEPTIONS [0, 19] ∪ [31, 800]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [15, 25], after, R)

(Cancer, [19, 40], after, I)

(Obstetric-emergency, [5, 20], after, R)

(Vascular-Disorders, [10, 45], after, R)

Template-Slot `activated-partial-thromboplastin-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1400

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [60, 100]

Template-Facet :VALUE-EXCEPTIONS [0, 59] ∪ [101, 1400]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [20, 59], after, R)

(Cancer, [65, 80], after, I)

(Obstetric-emergency, [40, 70], after, R)

(Vascular-Disorders, [80, 90], after, R)

Template-Slot `fibrinogen`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 8000

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [0, 400]

Template-Facet :VALUE-EXCEPTIONS [401, 8000]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [401, 8000], after, R)

(Cancer, [401, 8000], after, I)

(Obstetric-emergency, [401, 8000], after, R)

(Vascular-Disorders, [401, 8000], after, R)

Template-Slot `antithrombin-III`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 90

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [0, 0.20]

Template-Facet :VALUE-EXCEPTIONS [0.21, 90]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [0, 0.1], after, R)

(Cancer, [0.8, 0.35], after, I)

(Obstetric-emergency, [0.05, 0.1], after, R)

(Vascular-Disorders, [0.18, 9], after, R)

Template-Facet :DOCUMENTATION-AT-FRAME

Own-Facet :FRAME-TYPE `Concept`

Template-Slot `D-dimer`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 5000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [200, 2000]

Template-Facet :VALUE-EXCEPTIONS [0, 199] ∪ [2001, 5000]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [200, 400], after, R)

(Cancer, [300, 800], after, I)

(Obstetric-emergency, [10, 100], after, R)

(Vascular-Disorders, [500, 2000], after, R)

Frame `Acute-DIC`

:SUBCLASS-OF `Disseminated-Intravascular-Coagulation`

Template-Slot `percentage-of-protein-C`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 3000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [0, 59]

Template-Facet :VALUE-EXCEPTIONS [60, 3000]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [0, 59], after, R)

(Cancer, [40, 70], after, I)

(Trauma, [0, 30], after, R)

(Burns, [80, 1500], after, I)

(Obstetric-emergency, [40, 160], after, R)

(Vascular-Disorders, [0, 59], after, R)

Template-Slot `number-of-platelets`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 1,000,000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [0, 150,000]

Template-Facet :VALUE-EXCEPTIONS [150,001, 1,000,000]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY never

Template-Facet :VALUE-CHANGE-EVENTS

Template-Slot `platelet-aggregation-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPES [0, 3]

Template-Facet :VALUE-EXCEPTIONS [3.1, 100]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [2.5, 4], after, R)

(Cancer, [0, 3], after, I)

(Trauma, [0, 1.5], after, R)

(Burns, [0, 5], after, I)

(Obstetric-emergency, [0, 5], after, R)

(Vascular-Disorders, [3, 7], after, R)

Template-Slot `fibrin-degradation-products`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [40, 100]

Template-Facet :VALUE-EXCEPTIONS [0, 39]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [10, 39], after, R)

(Cancer, [60, 100], after, I)

(Trauma, [29, 45], after, R)

(Burns, [40, 60], after, I)

(Obstetric-emergency, [70, 100], after, R)

(Vascular-Disorders, [40, 60], after, R)

Template-Slot `pt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 100

Template-Facet :VALUE-PROTOTYPE [20.1, 100]

Template-Facet :VALUE-EXCEPTION [0, 20]

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-EXCEPTIONS none

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [20, 40], after, R)

(Cancer, [80, 100], after, I)

(Trauma, [10.5, 29], after, R)

(Burns, [5, 15], after, I)

(Obstetric-emergency, [20.1, 50], after, R)

(Vascular-Disorders, [2, 12], after, R)

Template-Slot `ptt`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 800

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [100, 1400]

Template-Facet :VALUE-EXCEPTIONS [0, 99]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [200, 400], after, R)

(Cancer, [0, 80], after, I)

(Trauma, [300, 500], after, R)

(Burns, [100, 150], after, I)

(Obstetric-emergency, [30, 700], after, R)

(Vascular-Disorders, [400, 1000], after, R)

Template-Slot `activated-partial-thromboplastin-time`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :SLOT-NUMERIC-MAXIMUM 1400

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [200, 1400]

Template-Facet :VALUE-EXCEPTIONS [0, 199]

Template-Facet :VALUE-MODALITY most

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [200, 400], after, R)

(Cancer, [300, 800], after, I)

(Trauma, [20, 150], after, R)

(Burns, [80, 250], after, I)

(Obstetric-emergency, [400, 1000], after, R)

(Vascular-Disorders, [200, 350], after, R)

Template-Slot `fibrinogen`

Template-Facet :VALUE-TYPE Integer

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 8000

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [0, 200]

Template-Facet :VALUE-EXCEPTIONS [201, 8000]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [200, 400], after, R)

(Cancer, [300, 800], after, I)

(Trauma, [], after, R)

(Burns, [], after, I)

(Obstetric-emergency, [10, 100], after, R)

(Vascular-Disorders, [500, 2000], after, R)

Template-Slot `antithrombin-III`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 90

Template-Facet :VALUE-LABEL inherited-with-exception

Template-Facet :VALUE-PROTOTYPE [10, 90]

Template-Facet :VALUE-EXCEPTIONS [0, 9.9]

Template-Facet :VALUE-MODALITY all

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [0.5, 7.9], after, R)

(Cancer, [30, 60], after, I)

(Trauma, [9, 35], after, R)

(Burns, [10, 70], after, I)

(Obstetric-emergency, [10, 30], after, R)

(Vascular-Disorders, [4.5, 8], after, R)

Template-Facet :DOCUMENTATION-AT-FRAME

Own-Facet :FRAME-TYPE `Concept`

Template-Slot `D-dimer`

Template-Facet :VALUE-TYPE Number

Template-Facet :CARDINALITY 1

Template-Facet :NUMERIC-MINIMUM 0

Template-Facet :NUMERIC-MAXIMUM 5000

Template-Facet :VALUE-LABEL inherited

Template-Facet :VALUE-PROTOTYPE [2000, 5000]

Template-Facet :VALUE-EXCEPTIONS [0, 1999]

Template-Facet :VALUE-MODALITY possible

Template-Facet :VALUE-CHANGE-FREQUENCY volatile

Template-Facet :VALUE-CHANGE-EVENTS

(Infection, [2000, 5000], after, R)

(Trauma, [1000, 3000], after, R)

(Burns, [1500, 2100], after, I)

(Cancer, [200, 1000], after, I)

(Obstetric-emergency, [3500, 5000], after, R)

(Vascular-Disorders, [1800, 3000], after, R)

Slot-Frame descriptions

Slot `Platelet-Count`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Integer

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0,

:SLOT-NUMERIC-MAXIMUM 1,000,000

:DOCUMENTATION Counts the number of platelets per millimiter cubed

Slot `PTT`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Number

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0,

:SLOT-NUMERIC-MAXIMUM 800

:DOCUMENTATION Partial thromboplastine time in seconds

Slot `Platelet-Aggregation-Time`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Number

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0,

:SLOT-NUMERIC-MAXIMUM 100

:DOCUMENTATION Time in minutes of aggregation

Slot `Percentage-of-Protein-C`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPEInteger

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0

:SLOT-NUMERIC-MAXIMUM 3000

:DOCUMENTATION Measures the percentage of inhibition

Slot `PT`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Number

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0

:SLOT-NUMERIC-MAXIMUM 250

:DOCUMENTATION prothrombin time in seconds, normal values may vary depending on labs

Slot `Percentage-of-protein-S`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Integer

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0

:SLOT-NUMERIC-MAXIMUM 2240

:DOCUMENTATION Measures the percentage of inhibition

Slot `APTT`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Number

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0

:SLOT-NUMERIC-MAXIMUM 1400

:DOCUMENTATION Activated partial thromboplastin time in seconds

Slot `Fibrinogen`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Integer

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0

:SLOT-NUMERIC-MAXIMUM 8000

:DOCUMENTATION Amount of fibrinogen in blood mweasured in mg/dl. It is initially in relation to the primary condition and then decreases.

Slot `Fibrin-Degradation-Product`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Number

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0

:SLOT-NUMERIC-MAXIMUM 100

:DOCUMENTATION blood contains less than 10mcg/ml fibrin split products (FSP)

Slot `D-dimer`

:DOMAIN Blood-Coagulation-Disorder

:SLOT-VALUE-TYPE Number

:SLOT-CARDINALITY 1

:SLOT-NUMERIC-MINIMUM 0

:SLOT-NUMERIC-MAXIMUM 5000

:DOCUMENTATION blood contains less than 200 μg/l.

References

[Arens *et al.* 1996]

Y. Arens, C.A. Knoblock, and W. Shen. “Query reformulation for dynamic information integration”. *Journal of Intelligent Information Systems*, volume 6, number 2/3, pages 99–130, 1996.

[Aristotle a]

Aristotle. *The categories, On Interpretation, Prior Analytics*. Harvard University Press, Cambridge, MA.

[Aristotle b]

Aristotle. *Metaphysics*. Harvard University Press, Cambridge, MA.

[Arpírez *et al.* 2001]

J.C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. “WebODE: A scalable workbench for ontological engineering”. In *Proceedings of the First International Conference on Knowledge Capture, K-CAP 2001*, ACM-Sigmod, 2001.

[Bayardo *et al.* 1997]

R.J. Bayardo, B. Bohrer, R.S. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M.H. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. “InfoSleuth: Agent-based semantic integration of information in open and dynamic environments”. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 195–206, ACM Press, New York, 1997.

[Bernaras *et al.* 1996]

A. Bernaras, I. Laresgoiti, and J. Corera. “Building and reusing ontologies for electrical network applications”. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI)*, pages 298–302, John Wiley & Sons, Ltd, Chichester, England, 1996.

[Borgida 1996]

A. Borgida. “On the relative expressiveness of description logics and predicate logics”. *Artificial Intelligence*, volume 82, number 1-2, pages 353–367, 1996.

[Borst 1997]

P. Borst. *Construction of Engineering ontologies for knowledge sharing and reuse*. PhD thesis, Centre for Telematica and Information Technology, University of Twente, 1997.

[Brachman 1985]

R.J. Brachman. “On the epistemological status of semantic networks”. In R.J. Brachman and H.J. Levesque, editors, *Readings in Knowledge Representation*, pages 191–215, Morgan Kaufmann, Los Altos, CA, 1985.

[Bylander and Chandrasekaran 1988]

T. Bylander and B. Chandrasekaran. “Generic tasks in knowledge-based reasoning: The right level of abstraction for knowledge acquisition”. In B. gaines and J. Boose, editors, *Knowledge acquisition for knowledge bases*, pages 65–77, Academic Press, London, 1988.

[Carpenter 1993]

B. Carpenter. “Skeptical and credulous default unification with application to templates and inheritance”. In T. Briscoe, A. Copestake, and V. de Paiva, editors, *Default reasoning and Lexical Organization*, Cambridge University Press, 1993.

[Chalupsky 2000]

H. Chalupsky. “Ontomorph: A translation system for symbolic knowledge”.

In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning. Proceedings of the seventh international conference (KR'2000)*, pages 471–482, Morgan Kaufmann, San Francisco, CA, 2000.

[Chaudhri *et al.* 1998]

V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp, and J.P. Rice. "OKBC: A programmatic foundation for knowledge base interoperability". In *Proceedings of the Fifteenth American Conference on Artificial Intelligence (AAAI-98)*, pages 600–607, AAAI Press/The MIT Press, Madison, Wisconsin, 1998.

[Cocchiarella 1991]

N.B. Cocchiarella. "Formal ontology". In H. Burkhardt and B. Smith, editors, *Handbook of metaphysics and ontology*, pages 640–647, Philosophia Verlag, Munich, 1991.

[Collins and Loftus 1975]

A.M. Collins and E.F. Loftus. "A spreading-activation theory of semantic processing". *Psychological Review*, volume 82, pages 407–425, 1975.

[Corcho and Gómez-Pérez 2000]

O. Corcho and A. Gómez-Pérez. "A roadmap to ontology specification languages". In R. Dieng, editor, *Proceedings of the 12th EKAW Conference*, pages 80–96, Springer Verlag, Berlin, 2000.

[Davis *et al.* 1993]

R. Davis, H. Shrobe, and P. Szolovits. "What is a knowledge representation?". *AI Magazine*, volume 14, number 1, pages 17–33, 1993.

[Esposito *et al.* 2000]

F. Esposito, D. Malerba, V. Tamma, and H.-H. Bock. "Classical resemblance measures". In H.-H. Bock and E. Diday, editors, *Analysis of Symbolic data.*

Exploratory methods for extracting statistical information from complex data, pages 139–152, Springer-Verlag, Berlin, 2000.

[Falasconi *et al.* 1996]

S. Falasconi, G. Lanzola, and M. Stefanelli. "Using ontologies in multi-agent systems". In *Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW)*, University of Calgary, Banff, Alberta, Canada, 1996.

[Farquhar *et al.* 1997]

A. Farquhar, R. Fikes, and J. Rice. "The ontolingua server: a tool for collaborative ontology construction". *International Journal of Human-Computer Studies*, volume 46, pages 707–728, 1997.

[Fernández-López *et al.* 2001]

M. Fernández-López, A. Gómez-Pérez, and N. Guarino. *The Methontology & OntoClean merge*. Technical Report, OntoWeb special interest group on Enterprise-standards Ontology Environments, 2001.

[Finin *et al.* 1997]

T. Finin, Y. Labrou, and J. Mayfield. "KQML as an agent communication language". In *Software Agents*, AAAI/MIT press, 1997.

[Fridman Noy *et al.* 2000]

N. Fridman Noy, R. W. Fergerson, and M. A. Musen. "The knowledge model of protege-2000: Combining interoperability and flexibility". In R. Dieng, editor, *Proceedings of the 12th EKAW Conference*, pages 17–32, Springer Verlag, Berlin, 2000.

[Fridman Noy and McGuinness 2001]

N. Fridman Noy and D.L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Technical Report SMI-2001-0880, Stanford Medical Informatics (SMI), Department of

Medicine, Stanford University School of Medicine, 2001. http://smi-web.stanford.edu/pubs/SMI$_{A}bstracts/SMI-2001-0880.html.$

[Fridman Noy and Musen 2001]

N. Fridman Noy and M.A. Musen. “Anchor-PROMPT: Using non-local context for semantic matching”. In A. Gómez-Pérez, M. Gruninger, H. Stuchenschmidt, and M. Uschold, editors, *Proceedings of the IJCAI’01 workshop on ontologies and information sharing*, 2001. http://www.semantic-translation.com/IJCAIwp/.

[Fridman Noy and Musen 1999]

N. Fridman Noy and M.A. Musen. “SMART: Automated support for ontology merging and alignment”. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW)*, University of Calgary, Banff, Alberta, Canada, 1999.

[Genesereth *et al.* 1992]

M.R. Genesereth, R. Fikes, D. Bobrow, R. Brachman, T. Gruber, P. Hayes, R. Letsinger, V. Lifschitz, R. MacGregor, J. McCarthy, P. Norvig, R. Patil, and L. Schubert. *Knowledge Interchange Format. Version 3.0. Reference manual*. Technical Report, Stanford University, 1992. http://www-ksl.stanford.edu/knowledge-sharing/kif/.

[Genesereth and Nilsson 1987]

M.R. Genesereth and N.J. Nilsson. *Logical foundations of Artificial Intelligence*. Morgan Kauffman, 1987.

[Goh *et al.* 1994]

C.H. Goh, S.E. Madnick, and M.D. Siegel. “Context interchange: Overcoming the challenges of large-scale interoperable database systems in a dynamic envoronment”. In *Proc of the Third International Conference on Information and Knowledge Management*, pages 337–346, 1994.

[Goldszmidt and Pearl 1996]

M. Goldszmidt and J. Pearl. “Qualitative probabilisties for default reasoning, belief revision, and causal modelling”. *Artificial Intelligence*, volume 84, number 1-2, pages 57–112, 1996.

[Gómez-Pérez 1998]

A. Gómez-Pérez. “Knowledge sharing and reuse”. In J. Liebowitz, editor, *The Handbook of Applied Expert Systems*, pages 10.1–10.36, CRC Press LLC, Boca Raton, FL, 1998.

[Gómez-Pérez and Benjamins 1999]

A. Gómez-Pérez and V.R. Benjamins. “Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods”. In V.R. Benjamins and A. Gómez-Pérez, editors, *Proceedings of the IJCAI’99 Workshop on Ontology and Problem-Solving Methods: Lesson learned and Future Trends*, pages 1.1–1.15, CEUR Publications, Amsterdam, 1999.

[Grosso *et al.* 1998]

W.E. Grosso, J.H. Gennari, R.W. Ferguson, and M.A. Musen. “When knowledge models collide (how it happens and what to do)”. In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW’98), Banff, Alberta, Canada*, University of Calgary, Banff, Alberta, Canada, 1998.

[Gruber 1991]

T. Gruber. “The role of common ontology in achieving sharable, reusable knowledge bases”. In J. Allen, R. Fikes, and E. Sandewall, editors, *Principles of knowledge representation and reasoning, KR’91, Proceedings of the second conference*, pages 601–602, Cambridge, Massachusets, USA, 1991.

[Gruber 1993]

T. R. Gruber. “A translation approach to portable ontology specifications”. *Knowledge Acquisition*, volume 5, number 2, pages 199–220, 1993.

[Guarino 1998]

N. Guarino. “Formal ontologies and information systems”. In N. Guarino, editor, *Proceedings of FOIS’98*, page , IOS Press, Amsterdam, 1998.

[Guarino *et al.* 1994]

N. Guarino, M. Carrara, and P. Giaretta. “An ontology of meta-level-categories”. In *Principles of Knowledge representation and reasoning: Proceedings of the fourth international conference (KR94)*, pages 270–280, Morgan Kaufmann, San Mateo, CA, 1994.

[Guarino and Giaretta 1995]

N. Guarino and P. Giaretta. “Ontologies and knowledge bases. towards a terminological clarification”. In N. Mars, editor, *Towards very large knowledge bases: knowledge building and knowledge sharing*, pages 25–32, IOS Press, Amsterdam, 1995.

[Guarino and Welty 2000a]

N. Guarino and C. Welty. “A formal ontology of properties”. In R. Dieng, editor, *Proceedings of the 12th EKAW Conference*, pages 97–112, Springer Verlag, Berlin, 2000.

[Guarino and Welty 2000b]

N. Guarino and C. Welty. “Identity, unity and individuality: Towards a formal toolkit for ontological analysis”. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, pages 219–233, IOS Press, Amsterdam, 2000.

[Guarino and Welty 2000c]

N. Guarino and C. Welty. “Ontological analysis of taxonomic relationships”. In A. Laender and V. Storey, editors, *Proceedings of the International conference on conceptual modelling (ER-2000)*, pages 210–224, Springer-Verlag, Berlin, 2000.

[Hirsch 1982]

E. Hirsch. *The concept of identity*. Oxford University Press, New York, 1982.

[Husserl 1982]

E. Husserl. *Ideas pertaining to a pure phenomenology and to a phenomenological philosophy. First book. General introduction to a pure phenomenology*. M. Nijhoff Publishers, The Hague, Holland, 1982. Translation by F. Kersten from *Ideen zu einer reinen Phänomenologie und phänomenologischen Philosophie. Erstes Buch: Allgemeine Einführung in die reine Phänomenologie,* 1913.

[Jennings 1995]

N.R. Jennings. “Agent software”. In *Proc. UNICOM Seminar on Agent Software*, pages 12–27, 1995.

[Kant 1965]

I. Kant. *Critique of pure reason*. St. Martin’s press, New York, 1965. Translation by N. Kemp Smith from *Kritik der reinen Vernunft,* 1787.

[Karp *et al.* 1999]

P. Karp, V. Chaudhri, and J. Thomere. “XOL: An XML-based ontology exchange language”. In *Bio-Ontologies’99 Meeting*, Heidelberg, Germany, 1999.

[Kim and Seo 1991]

W. Kim and J.J. Seo. “Classifying schematic and data heterogeneity in multi-database systems”. *IEEE computers*, volume 24, number 12-18, December 1991.

[Kitakami *et al.* 1996]

H. Kitakami, Y. Mori, and M. Arikawa. “An intelligent system for integrating autonomous nomenclature databases in semantic heterogeneity”. In R.R. Wagner and H. Thoma, editors, *Proceedings of Database and expert system applications (DEXA’96)*, pages 187–196, Springer, 1996.

[Klein 2001]

M. Klein. “Combining and relating ontologies: an analysis of problems and solutions”. In A. Gómez-Pérez, M. Gruninger, H. Stuckenschmidt, and M. Uschold, editors, *Proceedings of the IJCAI’01 workshop on Ontologies and Information Sharing*, pages 53–62, 2001.

[Kowalski and Sergot 1986]

R. Kowalski and M. Sergot. “A logic-based calculus of events”. *New Generation Computing*, volume 4, pages 67–95, 1986.

[Kripke 1980]

S.A. Kripke. *Naming and necessity*. Harvard University Press, Cambridge, Massachusetts, USA, 1980.

[Lassila and McGuinness 2001]

O. Lassila and D. McGuinness. “The role of frame-based representation on the semantic web”. *Electronic Transactions on Artificial Intelligence (ETAI) Journal: area The Semantic Web*, volume To appear, 2001.

[Lenat 1995a]

D.B. Lenat. “Cyc: a large-scale investment in knowledge infrastructure”. *Communications of the ACM*, volume 38, number 11, pages 33–38, November 1995.

[Lenat 1995b]

D.B. Lenat. “Steps to sharing knowledge”. In N. Mars, editor, *Towards very large knowledge bases: knowledge building and knowledge sharing*, pages 3–6, IOS Press, Amsterdam, 1995.

[Levi and deJonge 2000]

M. Levi and E. deJonge. “Current management of disseminated intravascular coagulation”. *Hospital Practice Electronic Journal*, volume August, 2000, 2000.

[Lewis 1993]

D.K. Lewis. *Counterfactuals*. Blackwell, Oxford, 1993.

[Lindberg *et al.* 1993]

D. Lindberg, B. Humphrey, and A. McCray. "The unified medical language system". *Methods of Information in Medicine*, volume 32, pages 281–289, 1993.

[Lowe 1989]

E.J. Lowe. *Kinds of being. A study of individuation, identity and the logic of sortal terms*. Basil Blackwell, Oxford, UK, 1989.

[Luger 2002]

G.F. Luger. *Artificial intelligence. Structures and strategies for complex problem solving*. Addison Wesley-Pearson Education, Harlow, England, fourth edition, 2002.

[MacGregor 1991]

R. MacGregor. "Inside the LOOM classifier". *SIGART bulletin*, volume 2, number 3, pages 70–76, June 1991.

[March 1990]

S.T. March. "Special issue on heterogeneous databases". *ACM Computing Surveys*, volume 22, number 3, 1990.

[McGuinness 2000]

D.L. McGuinness. "Conceptual modelling for distributed ontology environments". In B. Ganter and G.W. Mineau, editors, *Proceedings of the Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues (ICCS 2000)*, 2000.

[McGuinness *et al.* 2000]

D.L. McGuinness, R.E. Fikes, J. Rice, and S. Wilder. "An environment for merging and testing large ontologies". In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning. Proceedings of the seventh international conference (KR'2000)*, pages 483–493, Morgan Kaufmann, San Francisco, CA, 2000.

[Mena *et al.* 2000]

E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. "OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies". *Distributed and Parallel databases. An international journal*, volume 8, number 2, pages 223–271, April 2000.

[Mena *et al.* 1996]

E. Mena, V. Kashyap, A.P. Sheth, and A. Illarramendi. "OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies". In *Conference on Cooperative Information Systems*, pages 14–25, 1996.

[Miller 1990]

G.A. Miller. "Nouns in WordNet: a lexical inheritance system". *International Journal of Lexicography*, volume 3, number 4, pages 245–264, 1990.

[Morgenstern 1998]

L. Morgenstern. "Inheritance comes of age: Applying nonmonotonic techniques to problems in industry". *Artificial Intelligence*, volume 103, pages 1–34, 1998.

[Neches *et al.* 1991]

R. Neches, R.E. Fikes, T. Finin, T.R. Gruber, R. Patil, T. Senator, and W.R. Swartout. "Enabling technology for knowledge sharing". *AI Magazine*, volume 12, number 3, pages 36–56, 1991.

[Nelson *et al.* Forthcoming]

S.J. Nelson, D. Johnston, and B.L. Humphreys. "Relationship in medical subjects headings MeSH". In C.A. Bean and R. Green, editors, *Relationship in the organisation of knowledge*, Kluwer Academics Publishers, New York, Forthcoming.

[Newell 1982]

A. Newell. "The knowledge level". *Artificial intelligence*, volume 18, number 1, pages 87–127, 1982.

[Parsons *et al.* 1998]

S. Parsons, C. Sierra, and N.R. Jennings. "Agents that reason and negotiate by arguing". *Journal of Logic and Computation*, volume 8, number 3, pages 261–292, 1998.

[Perry *et al.* 1999]

B. Perry, M. Taylor, and A. Unruh. "Information aggregation and agent interaction patterns in infosleuth(tm)". In *Conference on Cooperative Information Systems*, pages 314–324, 1999.

[Pinto *et al.* 1999]

H.S. Pinto, A. Gómez-Pérez, and J.P. Martins. "Some issues on ontology integration". In V.R. Benjamins and A. Gómez-Pérez, editors, *Proceedings of the IJCAI'99 Workshop on Ontology and Problem-Solving Methods: Lesson learned and Future Trends*, pages 7.1–7.11, CEUR Publications, Amsterdam, 1999.

[Plantiga 1989]

A. Plantiga. *The nature of necessity*. Clarendon Library of logic and philosophy. Clarendon Press, New York, 1989.

[Preece *et al.* 2001]

A. Preece, K. Hui, A. Gray, P.Marti, T. Bench-Capon, Z. Cui, and D. Jones. "KRAFT: an agent architecture for knowledge fusion". *International journal of cooperative information systems*, volume 10, number 1/2, pages 171–196, March & June 2001.

[Quillian 1968]

M.R. Quillian. "Semantic memory". In Marvin Minsky, editor, *Semantic Information Processing*, pages 227–270, MIT Press, Cambridge, Massachusetts, 1968.

[Rada *et al.* 1989]

R. Rada, H. Mili, E. Bicknell, and M. Blettner. "Development and application of a metric on semantic nets". *IEEE Transactions on Systems, Man, and Cybernetics*, volume 19, number 1, pages 17–30, 1989.

[Rodríguez and Egenhofer 2002]

M.A. Rodríguez and M.J. Egenhofer. "Determining semantic similarity among entity classes from different ontologies". *IEEE transactions on knowledge and data engineering*, 2002. in press.

[Rosch 1975]

E.H. Rosch. "Cognitive representations of semantic categories". *Journal of Experimental Psychology: General*, volume 104, pages 192–233, 1975.

[Rosch 1999]

E.H. Rosch. "Reclaiming concepts". *Journal of Consciousness Studies*, volume 6, number 11-12, pages 61–77, 1999.

[Rothwell 1995]

D.J. Rothwell. "SNOMED-based knowledge representation". *Methods of Information in Medicine*, volume 34, pages 209–214, 1995.

[Schmaier 2001]

A.H. Schmaier. "Disseminated intravascular coagulation (dic)". *eMedicine Journal*, volume 2, number 5, May 2001.

[Searle 1983]

J.R. Searle. *Intentionality*. Cambridge University Press, Cambridge, 1983.

[Shave 1997]

M.J.R. Shave. “Ontological structures for knowledge sharing”. *The new review of information networking*, volume 3, pages 125–133, 1997.

[Sowa 1984]

J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984.

[Sowa 2000]

J.F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[Steimann 2000]

F. Steimann. “On the representation of roles in object-oriented and conceptual modelling”. *Data and Knowledge Engineering*, volume 35, pages 83–106, 2000.

[Studer *et al.* 1998]

R. Studer, V.R. Benjamins, and D. Fensel. “Knowledge engineering, principles and methods”. *Data and Knowledge Engineering*, volume 25, number 1-2, pages 161–197, 1998.

[Swartout *et al.* 1996]

B. Swartout, R. Patil, K. Knight, and T. Russ. “Towards distributed use of large-scale ontologies”. In *Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW)*, University of Calgary, Banff, Alberta, Canada, 1996.

[Sycara *et al.* 1998]

K. Sycara, M. Klusch, S. Widoff, and J. Lu. “Dynamic service matchmaking among agents in open information systems”. *ACM SIGMOD Record. Special Issue on semantic interoperability in global information systems*, 1998.

[Tamma and Bench-Capon 2001a]

V.A.M. Tamma and T.J.M Bench-Capon. “A conceptual model to facilitate knowledge sharing in multi-agent systems”. In S. Cranefield, T. Finin, and S. Willmott, editors, *Proceedings of the Autonomous Agents’01 workshop on ontologies in agent systems (OAS 2001), Montreal, Quebec, Canada*, CEUR Publications, Amsterdam, 2001.

[Tamma and Bench-Capon 2001b]

V.A.M. Tamma and T.J.M. Bench-Capon. “An enriched knowledge model for formal ontological analysis”. In C. Welty and B. Smith, editors, *Proceedings of the international conference on formal ontology and information systems (FOIS’01)*, ACM press, New York, 2001.

[Tamma and Bench-Capon 2000]

V.A.M. Tamma and T.J.M Bench-Capon. “Supporting inheritance mechanisms in ontology representation”. In R. Dieng, editor, *Proceedings of the 12th EKAW Conference*, pages 140–155, Springer Verlag, Berlin, 2000.

[Tamma and Visser 1998]

V.A.M. Tamma and P.R.S. Visser. “Integration of heterogeneous sources: Towards a framework for comparing techniques”. In N. Guarino, editor, *Proceedings of the 6 Convegno dell’Associazione Italiana per l’Intelligenza Artificiale (AI*IA), Workshop su Strumenti di organizzazione ed accesso intelligente per informazioni eterogenee,* pages 89–93, Edizioni Progetto Padova, 1998.

[Touretzky 1986]

D.S. Touretzky. *The Mathematics of Inheritance Systems*. Morgan Kaufmann, San Mateo, CA, 1986.

[Tversky 1977]

A. Tversky. “Features of similarity”. *Psychological Review*, volume 84, number 4, pages 327–372, 1977.

[Uschold 1998]

M. Uschold. “Knowledge level modelling: concepts and terminology”. *Knowledge Engineering Review*, 1998.

[Uschold *et al.* 1998]

M. Uschold, P. Clark, M. Healy, K. Williamson, and S. Woods. “An experiment in ontology reuse”. In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW’98), Banff, Alberta, Canada*, University of Calgary, Banff, Alberta, Canada, 1998.

[Uschold and Gruninger 1996]

M. Uschold and M. Gruninger. “Ontologies: principles, methods, and applications”. *Knowledge Engineering Review*, volume 11, number 2, pages 93–155, 1996.

[Uschold *et al.* 1999]

M. Uschold, R. Jasper, and P. Clark. “Three approaches for knowledge sharing: A comparative analysis”. In *Proceedings of the Twelfth Workshzp on Knowledge Acquisition, Modeling and Management (KAW’99). October 1999, Banff, Alberta, Canada*, University of Calgary, Banff, Alberta, Canada, 1999.

[van Heijst *et al.* 1997]

G. van Heijst, A.Th. Schreiber, and B.J. Wielinga. “Using explicit ontologies in kbs development”. *International Journal of Human-Computer Studies*, volume 45, pages 184–292, 1997.

[van Zyl and Corbett 2000]

J. van Zyl and D. Corbett. “Population of a framework for understanding and classifying ontology applications”. In V.R. Benjamins, A. Gómez-Pérez, N. Guarino, and M. Uschold, editors, *Proceedings of the ECAI00 workshop on Applications of Ontologies and Problem-Solving Methods*, pages 10.1–10.12, 2000.

[Visser and Cui 1998]

P. Visser and Z. Cui. “On accepting heterogeneous ontologies in distributed architectures". In P. Borst, V.R. Benjamins, A. Farquhar, and A. Gómez-Pérez, editors, *Proceedings of the ECAI'98 workshop on Applications of Ontologies and Problem-solving methods, Brighton, UK*, pages 112–119, 1998.

[Visser *et al.* 1998]

P.R.S. Visser, D.M. Jones, T.J.M. Bench-Capon, and M.J.R. Shave. "Assessing heterogeneity by classifying ontology mismatches". In N. Guarino, editor, *Formal Ontology in Information Systems. Proceedings FOIS'98, Trento, Italy*, pages 148–182, IOS Press, 1998.

[Visser and Tamma 1999]

P.R.S. Visser and V.A.M. Tamma. "An experience with ontology-based agent clustering". In V.R. Benjamins and A. Gómez-Pérez, editors, *Proceedings of the IJCAI'99 Workshop on Ontology and Problem-Solving Methods: Lesson learned and Future Trends*, pages 12.1–12.19, CEUR Publications, Amsterdam, 1999.

[Welty and Guarino 2001]

C. Welty and N. Guarino. "Supporting ontological analysis of taxonomical relationships". *Data and knowledge engineering*, volume 39, number 1, pages 51–74, 2001.

[Wiggins 1967]

D. Wiggins. *Identity and Spatio-Temporal continuity*. Basil Blackwell, Oxford, 1967.

[Woods 1975]

W.A. Woods. "What's in a link: Foundations for semantic networks". In D. G. Bobrow and A. Collins, editors, *Representation and Understanding*, pages 35–82, Academic Press, New York, 1975.

[Wooldridge and Jennings 1995]

M. Wooldridge and N.R. Jennings. "Intelligent agents: Theory and practice". *Knowledge engineering review*, volume 10, number 2, pages 115–152, 1995.

[Zanastra *et al.* 1995]

P.E. Zanastra, A.L. Rector, W.D. Solomon, T. Rush, W.A. Nowlan, and S. Bechofer. “A terminology server for integrating clinical information systems: the GALEN approach”. In *Proceedings of current Perspectives in Healthcare Computing*, pages 256–262, BJHC Books, Weybridge, UK, 1995.