



Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
Vakgroep Computerwetenschappen
Semantics Technology and Applications Research Laboratory

Grounding Ontologies with Social Processes and Natural Language

Proefschrift ingediend met het oog op het behalen van
de graad van Doctor in de Wetenschappen

Christophe Debruyne

Promotor: prof. dr. Robert Meersman

September 2013



Abstract

The formal semantics of a computer-based system quite simply is the correspondence between this system and some real world as perceived by humans. It is usually given by a formal mapping of the symbols in the system's description to objects in that real world, such that relationships and logical statements in the specification language can be assigned a truth-value depending on whether a certain state of affairs among objects exists in the real world. As the real world is not usually directly accessible inside a computer, storing and reasoning about semantics requires the world to be replaced by an agreed specification of a conceptualization, which is often in the shape of a formal construct. This computer-based, shared, agreed formal specification of a conceptualization is what is known as an *ontology*.

The creation of ontologies for enabling semantic interoperability between two or more *autonomously developed* and *maintained* information systems is far from trivial, as it requires the necessary agreements by all stakeholders – which is called a *community* – on concepts and relations to make this interoperation possible.

As those agreements are the result of *social* interactions, appropriate ontology engineering methods should take into account the natural language used by the community during those interactions. In this thesis, a fact-oriented formalism is extended for the construction of so-called *hybrid ontologies*. In hybrid ontologies, all concepts, terms, etc. are represented not just on their own formal structures (e.g. by means of fact-orientation), but are always to be interpreted in a given context, which is the *community* that agrees on those formal structures. Agreements are made possible and are supported by *glosses* in natural language of which the shared understanding is implicit. In hybrid ontology engineering, communities consequently are promoted to “first-class citizens” by formalizing the social interactions that evolve the hybrid ontologies and by declaring the community as the context in which all processes take place.

Next to proposing a framework for hybrid ontologies, this thesis also presents a collaborative ontology engineering method called GOSPL. GOSPL – which stands for Grounding Ontologies with Social Processes and natural Language – allows prescribing how the social interactions should be orchestrated and furthermore describes how agreements on formal and informal descriptions are complementary, and how they interplay.

A part of this thesis is devoted to show how glosses can drive the hybrid ontology construction process and to show how the annotation of the aforementioned information systems are used to steer the social interactions and agreement processes.

All of the ideas presented in this work have been implemented in a tool. Both method and tool were used in two experiments, of which a discussion is presented in this thesis.

Samenvatting

De formele semantiek van een computergebaseerd systeem kan het best omschreven worden als de link die gelegd worden tussen dat systeem en een realiteit zoals deze door mensen wordt waargenomen. Deze link wordt gewoonlijk gelegd tussen een formele afbeelding van symbolen in de beschrijving van het systeem en de objecten in die realiteit. Zo wordt aan relaties en logische uitdrukkingen in de specificatietaal een waarheidswaarde toegekend, die afhangt van de stand van zaken tussen de objecten in die realiteit. Omdat de echte wereld gewoonlijk niet direct toegankelijk is vanuit een computer, vergt het opslaan en redeneren over de semantiek een abstracte benadering, m.a.w. door het vervangen van de realiteit door een overeengekomen specificatie van een conceptualisatie, vaak in de vorm van een formeel construct. Deze computergebaseerde, gedeelde en overeengekomen formele specificatie van een conceptualisatie heet een *ontologie*.

De creatie van ontologieën die semantische interoperatie tussen twee of meer *autonomo* *ontwikkeld* en *beheerde* informatiesystemen moet bewerkstelligen, is ver van triviaal. Het vereist de nodige akkoorden over de concepten en relaties tussen alle belanghebbenden – wat we een *community* noemen – om die interoperatie mogelijk te maken.

Omdat die akkoorden het resultaat zijn van *sociale* interacties, moeten gepaste methoden voor het bouwen van ontologieën de natuurlijke taal in achtig nemen, die door de community tijdens die interacties wordt gebruikt. In deze thesis wordt een feit-georiënteerd formalisme aangewend en uitgebreid voor het maken van zogenaamde *hybride ontologieën*. In hybride ontologieën worden alle concepten, termen, ... niet alleen voorgesteld door hun formele structuren (b.v. door middel van feit-oriëntatie), maar deze moeten altijd binnen een bepaalde context geïnterpreteerd worden. Deze context is de *community* die akkoorden over die formele structuren bereikt. Die akkoorden worden mogelijk gemaakt en ondersteund door glosses in een natuurlijke taal, waarvan het verstaan en begrijpen door de community impliciet is. In de constructie van hybride ontologieën worden communities bijgevolg gepromoveerd tot “first-class citizens”, enerzijds door de sociale interacties te formaliseren die de hybride ontologieën doen evolueren en anderzijds door het declareren van de community als de context waarin alle akkoorden plaats vinden.

Naast het voorstellen van een raamwerk voor hybride ontologieën, zal deze thesis ook de collaboratieve ontologie-constructiemethode GOSPL voorstellen. GOSPL staat voor “Grounding Ontologies with Social Processes and natural Language” en voorziet een kader waarin de verschillende sociale interacties kunnen worden voorgescreven. Verder beschrijft GOSPL hoe akkoorden over formele en informele beschrijvingen elkaar aanvullen en op elkaar inspelen.

Een deel van deze thesis demonstreert hoe glosses het hybride ontologie-constructie proces aandrijven en hoe de annotaties van bovengenoemde informatiesystemen worden gebruikt om de sociale interacties en akkoorden bij te sturen.

Alle ideeën voorgesteld in dit werk werden in een tool geïmplementeerd. De methode en tool werden in twee experimenten gebruikt, waarop verder in dit werk zal worden ingegaan.

Acknowledgements

I would like to thank my supervisor prof. dr. Robert Meersman for his invaluable guidance, discussions, support and patience throughout the time this work was in progress. I also would like to thank him and the whole of STARLab for allowing me to do a BSc and MSc thesis at the lab, and offering me a part time job as a developer on various projects running at the lab during my years as a MSc student. It was a privilege to become increasingly involved in STARLab's research projects while still a student.

I would also like to thank the members of my jury: prof. dr. Sven Casteleyn, prof. dr. Wolfgang De Meuter, prof. dr. Ann Nowé, prof. dr. Hans Weigand and prof. dr. Esteban Zimányi. Your valuable comments and suggestions were helpful in improving the text.

I furthermore express my gratitude towards:

- All of my (ex-)colleagues at STARLab: Evangelos Argyzoudis, Jin Berghmans, Dang Bui Bach, Stijn Christiaens, Ioana Ciuciu, Tanguy Coenen, Peter De Baer, Pieter De Leenheer, Mustafa Jarrar, Han Kang, Aggelos Liapis, Margit Mikula, Quentin Reul, Peter Spyns, Trung-Kien Tran, Damien Trog, Cristian Vasquez, Jan Vereecken, and Gang Zhao.
- INNOVIris and the Brussels Capital Region for supporting the Open Semantic Cloud for Brussels project, which provided a setting to test the ideas presented in this thesis. I would also like to thank all involved in this project.
- Guy de Bellefroid of VISITBRUSSELS asbl, Wendy Schuppen of the Centre of Fine Arts (BOZAR) and Geert van Grootel of the Department of Economy, Science and Innovation of the Flemish Government (EWI) for providing data for the case studies.
- Davor Meersman with whom I had some pleasant and fruitful collaborations in the past (and I hope in the future as well).
- Some of the students who have helped me research-wise: Nadejda Alkhaldi, Jonas Blanckaert, Wim Coosemans, Mohammed Halalo, Anu Mulmi, Niels Nijs, Sven Van Laere, and Rik Vanmechelen.
- Some of my fellow students I met here at the VUB with whom I collaborated in projects closely related to the research of STARLab: Vincent Annaert, Zakaria Arrassi, Simon Janssens, Ben Maene and Johannes Peeters.
- Bruno Neskens, Pedro Ramos, Quentin Reul, Jonathan Stevens, Sven Van Laere, Geert Van Verdegem and Nico Vannieuwenhuyse for proofreading different parts of the text.

And finally, I would like to thank some people dear to me, who lent me an ear during some of the tougher moments: Pol Bernardus, Matthieu Christiaens, Despoina Mouratidou, Alain Sieuw, Geert Van Verdegem, Nico Vannieuwenhuyse, my brother Kevin and my parents.

Contents

1	Introduction	1
1.1	Thesis Scope and Motivation	1
1.2	Structural Overview of the Thesis	7
1.3	A Note to the Reader	8
2	Background and Related Work	9
2.1	Introduction	9
2.1.1	Interoperability and the Semantic Web	9
2.1.2	Ontology- and Semantic Web Languages	11
2.1.3	Semantic Web Rule Language	15
2.1.4	Description Logics	15
2.2	Ontology Engineering	19
2.2.1	Ontology Engineering Methods	20
2.2.2	A Comparison	28
2.3	Conclusions	34
3	Hybrid Ontology Framework	35
3.1	Introduction	35
3.2	The DOGMA Ontology Framework	37
3.2.1	Application commitments	38
3.3	Towards a Hybrid Ontology Framework	44
3.3.1	The Glossary and Concept Identifiers	44
3.3.2	Community Calculus for Concepts	46
3.3.3	The Role of Glosses in Hybrid Ontologies	47
3.3.4	Glossary-consistency and Gloss-equivalences	50
3.4	Community Commitments	53
3.5	Social Processes in Ontology Engineering	55
3.6	Conclusions	65
4	Hybrid Ontology Engineering Method	67
4.1	Introduction	67
4.1.1	Running Example	68
4.2	Semantic Interoperability Requirements	68
4.3	Building the Glossary	69
4.4	The Creation of Lexons	72
4.5	Constraining Lexons	73
4.6	Committing to the Hybrid Ontology	76
4.7	Gloss-equivalences and Synonyms	78
4.8	Community and SIR Co-evolution	80
4.9	GOSPL w.r.t. other DOGMA Based Methods	81
4.9.1	Compared with DOGMA(-MESS)	81
4.9.2	Compared with Business Semantics Management	83

4.10	Conclusions	86
5	Evolving Glosses	87
5.1	Discrete Gloss Evolution	87
5.2	Gloss Evolution Modalities	89
5.2.1	Drawing Inspiration from Rhetorical Structure Theory	89
5.2.2	Other Gloss Evolution Modalities	95
5.2.3	A Grammar to Structure and Process Glosses	96
5.3	Glossary and Commitment Co-evolution	97
5.3.1	Processing Structured Glosses	103
5.4	Relation with Related Work	103
5.5	Conclusions	105
6	Using Commitments for Steering Social Interactions	107
6.1	Translating Commitments into DL	107
6.1.1	The Description Logic DL-Lite _{A,id}	108
6.1.2	Lossless Schema Transformation	111
6.1.3	“Untranslatable” Constraints	118
6.1.4	Relation with Related Work	118
6.1.5	Conclusions	122
6.2	Application Commitments in the Feedback Loop	122
6.2.1	Semantics of Constrained Lexons	122
6.2.2	Checking Constraints over Annotated Data	127
6.3	Controlled Natural Language Querying	129
6.4	Atomizing Relational Databases with Ω -RIDDL	133
6.4.1	Creating RDF out of Relational Databases	134
6.4.2	Augmenting the D2RQ Mapping File	136
6.5	Conclusions	144
7	Implementation	145
7.1	Architecture	145
7.2	The Collaborative Ontology Engineering Platform	146
7.2.1	The Community Commitment and Glossary	148
7.2.2	Social Processes in GOSPL	150
7.2.3	Managing Application Commitments	152
7.2.4	Community Activity	152
7.3	Discrete Gloss Evolution	153
7.4	Querying RDF with R-RIDDL	156
7.5	Conclusions	157
8	Case Studies	159
8.1	Assessing the User Satisfaction with PSSUQ	159
8.2	Case 1: Annotating Cultural Events in Brussels	160
8.2.1	Summative and Formative User Satisfaction	162
8.2.2	Impact of Glosses on Meaning Negotiation	163
8.2.3	Using the Ontology	164
8.3	Changes in the Prototype	164

8.4	Case 2: Research Information Systems	167
8.4.1	Summative and Formative User Satisfaction.	168
8.4.2	Recommendations for Improvement	170
8.4.3	Using the Ontology	170
8.5	Discussion	171
9	Conclusions	173
9.1	Contributions	173
9.2	Limitations and Future Work	176
A	7-step Algorithm	179
B	D2RQ Generated Mapping	183
C	R-RIDL Grammar	185
	Bibliography	187

List of Acronyms

- BSM** Business Semantics Management
DDL Data Definition Language
DL Description Logic
DOGMA Developing Ontology-Grounded Methods for Applications
(E)ER model (Enhanced) Entity-relationship model
GOSPL Grounding Ontologies with Social Processes and natural Language
IBIS Issue-Based Information System
MESS Meaning Evolution Support System
NIAM Natural Information Analysis Method
OMG Object Management Group
ORM Object Role Modeling
OWL Web Ontology Language
R2RML RDB to RDF Mapping Language
RDF Resource Description Framework
RDF(S) RDF Schema
RIDL Reference and IDea Language
RIF Rule Interchange Format
RST Rhetorical Structure Theory
SQL Structured Query Language
SPARQL SPARQL Protocol and RDF Query Language
SWRL Semantic Web Rule Language
UML Unified Modeling Language
URI Universal Resource Identifier
URL Universal Resource Locator
W3C World Wide Web Consortium
WWW World Wide Web
XML eXtended Mark-up Language

Chapter 1

Introduction

1.1 Thesis Scope and Motivation

An information system is a system containing information in a database for a given application context of a given organization. The application context defines the functionality of such a system, and is prescribed by the organizations' requirements. The development of an information system thus involves the creation of a requirements specification and an agreement on the design. Costing is an important factor here and will also influence the choice whether components will be selected for its implementation, outsourced, or even built from scratch.

One should involve end users during the requirements specification process for several reasons. Two of these reasons are the impedance mismatch between the jargon (used by end users) and the business knowledge, and the end users being experts in (their part of) the domain. The development of an information system typically involves at least four worlds [JMSV92]:

- The subject world, or the universe of discourse;
- The system world, the information system's description of the universe of discourse;
- The usage world, the organizational setting in which the system is deployed including the end users, associated activities, etc.;
- The development world, which covers the environment in which the system is developed (including developers, the adopted development methods, etc.).

As shown in Figure 1.1, domain experts and end users observe the world. Domain experts try to abstract the world, whereas the end users will interact with the world and are able to test the developed information system by comparing the instances stored in that system with objects in the real world. Both domain experts and end users will collaborate with a knowledge designer so that the latter can put the resulting agreements into a CASE¹ tool for building a conceptual schema that describes the business domain. The conceptual schema will – in turn – be used to generate parts of the processes, parts of the constraints and a data model of the information system.

The knowledge in such a conceptual schema is typically a mixture of domain-general knowledge and enterprise-specific knowledge derived from the requirements. Domain knowledge can contain shared or even generic constraints. Enterprise-specific knowledge describes the applications and constraints local to the enterprise, making use of the

¹Computer-aided software engineering.

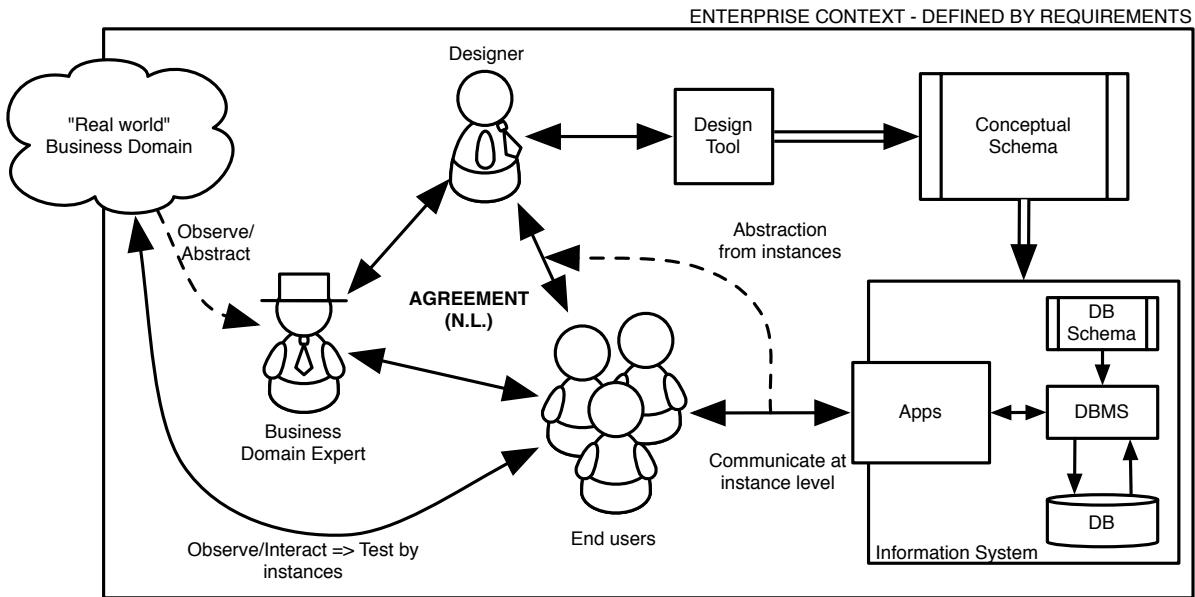


Figure 1.1: Information systems in an enterprise context.

domain knowledge. Enterprise-specific knowledge will often constrain the domain-general knowledge even further. For example, the knowledge that an ISBN identifies a book is part of the domain knowledge. In a movie rental service providing enterprise, however, the knowledge that a customer is only able to lend at most five movies at a time is part of the enterprise knowledge.

As the business domain is not accessible inside a computer, the conceptual schema – often in the shape of a formal (mathematical) construct – will actually replace the business domain. This is necessary in order to store and reason about the semantics of the business domain. Hence, the formal semantics of an information system is the correspondence between this system and the conceptual schema, which represents the business domain as perceived by the domain expert and the end users. Once the system is adequately designed and implemented, a statement output by the information system can be correctly interpreted by end users in terms of objects in the business domain if and only if such statement is derived from stored instances of concepts and relations as described in the conceptual schema. Those stored instances are mapped by the intentional semantics to agreed observed relationships among those same objects.

But what happens when two or more autonomously developed and maintained information systems need to meaningfully interoperate, but need to remain autonomous? Since the business domains of each information system overlaps with the shared domain of all humans involved, agreements on how to describe the shared domain by the representatives of these systems – which we will call a *community* – are needed. Again, as the world is not accessible inside each one of those information systems, the shared domain needs to be replaced by another formal (mathematical) construct, called an *ontology* (see Figure 1.2).

An ontology is commonly defined as: “*a [formal,] explicit specification of a [shared] con-*

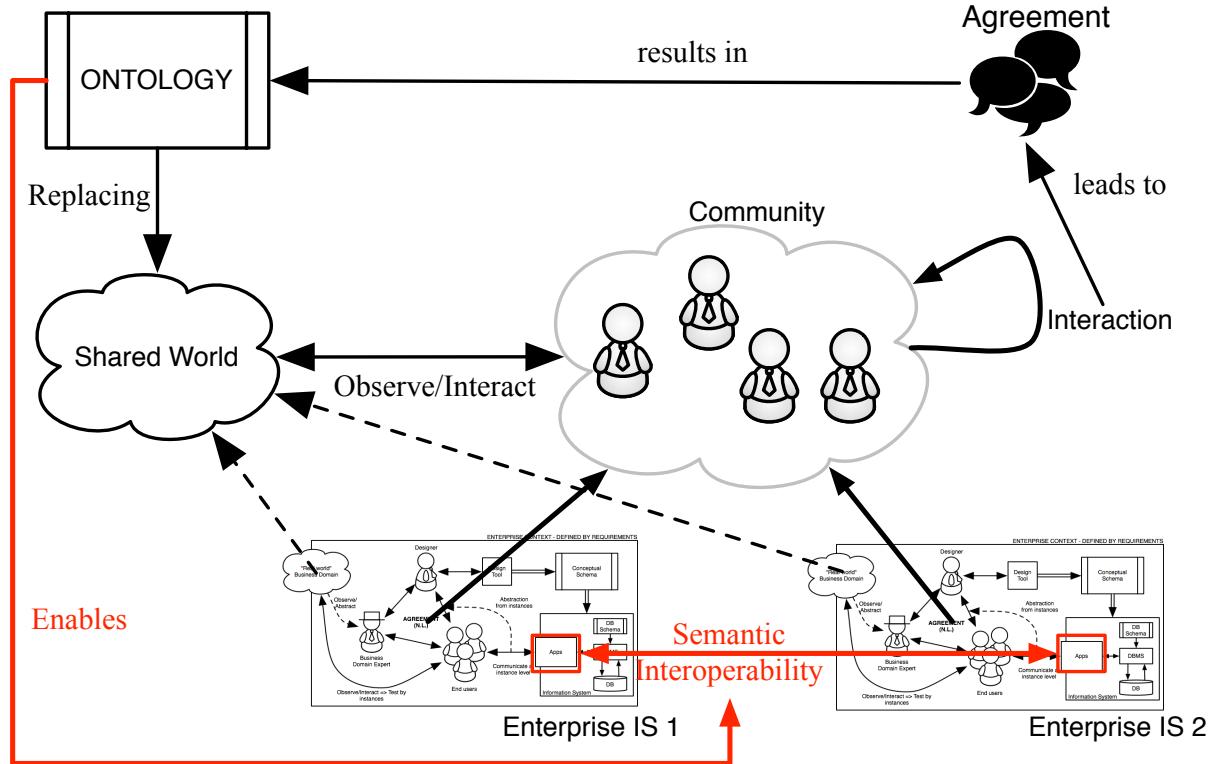


Figure 1.2: Agreements leading to ontology for enabling semantic interoperability

ceptualization" [Gru95]. Gruber's original definition is without the words "shared" and "formal", but are accepted by relevant scientific communities to describe more precisely the intention of ontologies. Ontologies constitute the key resources for realizing a Semantic Web [BLHL01]. The main difference between a conceptual schema and an ontology is that the first is intended for the development of one particular information system in one organization, and the latter for reuse and therefore general for a particular domain.

Each interaction within the community leads to agreements and these agreements are later on used to construct a new version of the ontology. The ontology will then be used to annotate the application symbols in each of the information systems for enabling semantic interoperability.

The problem, however, is not so much what ontologies in computer science are, but 1) how they become shared formal specifications of a domain, and 2) how they become operationally relevant and sustainable over longer periods of time [dMDM06]. Ontology engineering is far from trivial and requires adequate methods and tools for guiding the process [SSS09].

Suppose we consider the applications of BOZAR² and Agenda.be³. The first application is developed for maintaining and displaying cultural events, mostly taking place at the Centre for Fine Arts in Brussels. The latter is a portal for cultural events in and around

²<http://www.bozar.be/>

³<http://www.agenda.be/>

Brussels and this portal is used to promote the city. Although these applications share parts of their business domain, both were developed in a different enterprise context and for meeting different requirements. As a consequence, we observe that the different organizations have different views on the shared business domain, which can be for instance observed in the way both applications present categories of musical events to end-users, as shown in Figure 1.3. In this example, there are some simple correspondences, such as mapping the concept referred to with the label “classical music” with the concept referred to with the label “klassiek” (NL). Other mappings, however, are less obvious and make the exchange of information between the two systems a non-trivial task. For instance, the concept referred to with the label “jazz” intuitively seems to be included in the concept labeled “jazz & blues”. All instances of the latter, however, are not necessarily all instances of the first. The figure only depicts some of the taxonomic differences, however differences in the sets of attributes and the representations of these attributes can be observed in those applications as well.

Assuming that both organizations wish to annotate their data with an ontology, in order to retrieve information from both databases via that ontology. For BOZAR, it would increase the visibility for their events and Agenda.be could benefit from complementing their information on events with more detailed descriptions provided by the organizing parties. The ontology that will result of that collaboration will be the outcome of drawing inspiration from both enterprise contexts. The introduction of “classical music” in the ontology will most likely not be problematic, but the negotiation on what is considered “jazz” and “blues” will probably require more effort.

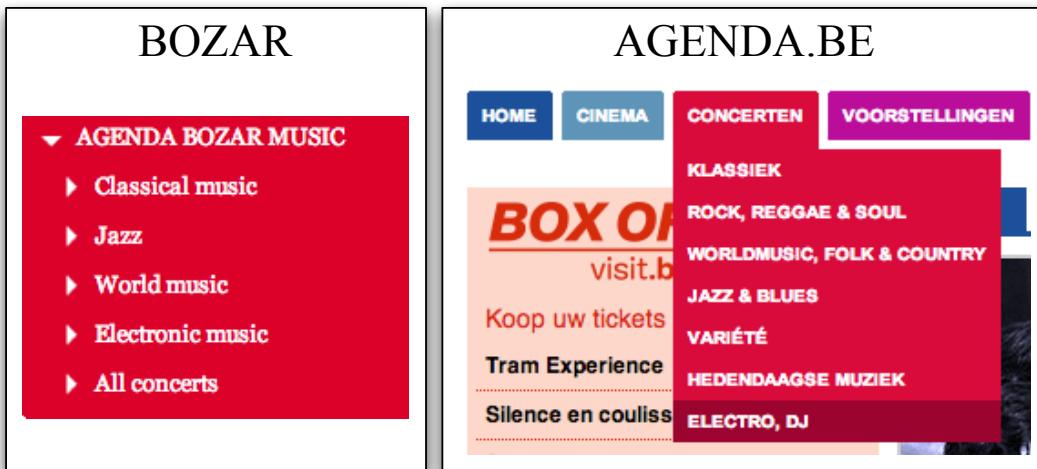


Figure 1.3: Different perspectives on the shared world by two organizations.

There are several solutions to tackle this problem. One solution is to agree keeping one concept in the ontology for both jazz and blues events. Making a distinction between the two would be left to the end-user interpreting the data. Another solution would be to separate the two concepts in the ontology. This would then require more effort from Agenda.be to ensure that instances of both concepts stored in their information system are properly classified (e.g., by searching for appropriate keywords). What all solutions

have in common is being the result of negotiation and discussion processes between the stakeholders.

Ontologies evolve as the agreements within the community evolve, which means that ontologies, in our opinion, are social artifacts. The social interactions leading to those agreements implies that communities should become an integral part of the ontology. This thesis explores how communities can be promoted to first-class citizen in the ontology engineering process by grounding the ontology evolution process with the community owning the ontology. During each interaction, the community *re-internalizes* the ontology, and *externalizes* their consensus in a following version of that ontology. The terms internalization and externalization are based on Nonaka and Takeuchi's four modes of knowledge conversion [NT95]: socialization (tacit to tacit), externalization (tacit to explicit), combination (explicit to explicit) and internalization (explicit to tacit) – also known as the SECI model of knowledge dimensions. These modes have been cleverly adopted for describing meaning negotiation in collaborative ontology engineering [DdMM07, DLM08]. In [DL09], the author has defined the *community evolution process* as a special type of process cycle, which consists of the SECI sequence of knowledge conversion modes.

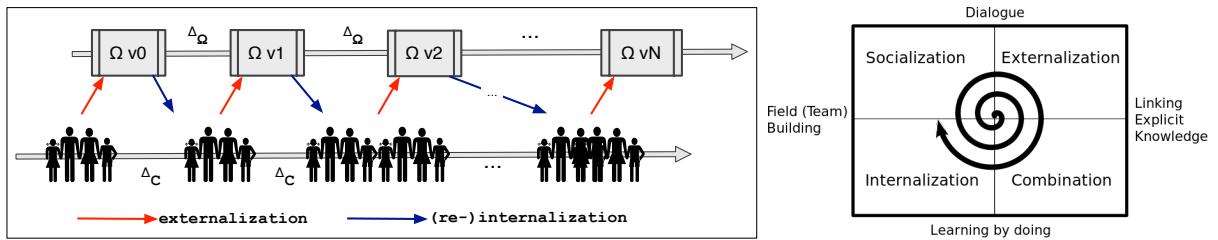


Figure 1.4: Communities evolving together with their ontologies. During each interaction, the community re-internalizes the ontology, and externalizes their consensus in a next version of that ontology. Nonaka and Takeuchi's knowledge-spiral is depicted on the right.

Interestingly, social interactions are supported by dialogue (i.e. discussions) and people reach agreements on the formal description of concepts via natural language descriptions that they exchange. We call such natural language descriptions for concepts *glosses*. By promoting communities to first-class citizen as well as their glosses, ontologies will impart a well-defined *hybrid* aspect, as they are to be resources shared among humans working in a community, as well as among networked systems such as exist in the World Wide Web. Natural language descriptions thus become important in supporting and driving community and ontology co-evolution. This brings us to the first research question:

Q1 What are hybrid ontologies?

It has already been stated that ontology development is far from trivial. For instance, some of the lessons learned in the On-To-Knowledge⁴ project are [SSS09]: 1) domain-experts need practical guidelines and 2) collaborative ontology engineering requires physical presence and advanced tool support. For the last twenty years, many methods have

⁴<http://www.ontoknowledge.org/>

been put forward developing ontologies. However, it seems that research on methodological activities has diminished in recent years [Ber10]. Bergman observed that very few discrete methods exist, and those that do are often older in nature [Ber10]. Adequate methods and tool support, however, are key in increasing the success of ontology projects. This leads us to the next two research questions:

Q2 How are hybrid ontologies constructed?

Q3 How can hybrid ontology construction be supported in a necessarily complex collaborative setting?

One of the assumptions in this thesis is that the glosses used by the community for *aligning* their thoughts evolve with the community. As a consequence, changes in glosses should be reflected as changes in the formal description of those concepts. For instance, refining a definition to include constraints or elaborate on the characteristics of the concept being described, would and should result in the addition of that constraint and extra knowledge in the formal description of that concept. The fourth research question is thus formulated as follows:

Q4 How does the evolution of a natural language definition of a concept influence the formal part of a hybrid ontology?

Finally, this thesis looks into the evolution of annotations with the ontology of the information systems in the ontology engineering process. These annotations provide valuable information on how those information systems use the concepts and relations in the ontology as well as the instances that are made accessible through the ontology via those annotations. One can, for instance, look for counterexamples for statements made by the community, or even determine to what extent one particular information system is compliant with the constraints of an ontology. The fifth question in this thesis then becomes:

Q5 How does the annotation of application symbols drive the hybrid ontology engineering process?

In summary, Q1 looks for a framework for hybrid ontology engineering, whereas Q2 and Q3 investigate a method and tool for hybrid ontology engineering respectively. Questions Q4 and Q5 look how the ontology engineering processes prescribed in the answer of Q2 can be facilitated.

This thesis addresses these questions by focusing on several research objectives:

O1 Provide an analysis of the state-of-the-art on ontology engineering with respect to the use of social processes and the use of natural language definitions to support the evolution of meaning agreements within a community.

O2 Develop the notion of hybrid ontologies to support social processes with natural language definitions for concepts.

- O3** Develop a method for hybrid ontology engineering.
- O4** Determine which parts of the method can benefit from the evolution of those natural language definitions.
- O5** Determine which parts of the method can benefit from the annotation of existing (legacy) systems.
- O6** Develop a tool that is based on this aforementioned method.
- O7** Evaluate the contributions and – by consequence – their conceptual design.

1.2 Structural Overview of the Thesis

Parts of this thesis have been reported earlier in following peer-reviewed publications. This will be mentioned when describing each chapter. This thesis is organized as follows:

Chapter 2 provides a review on ontology languages used on the Semantic Web and a comparison of the state-of-the-art on (collaborative) ontology engineering. From the latter it will become apparent that the use of natural language descriptions as a driver for the social processes and the ownership of ontologies by the community of stakeholders is not properly addressed.

The comparison of ontology engineering methods in this chapter started from the related work published in [DM12], which was then extended and compared in more detail in [DTM13].

Chapter 3 introduces a framework for ontology engineering promoting both the community and their glosses to first-class citizens. This framework starts from an existing ontology engineering framework already using natural language aspects in the formal descriptions of the concepts. Chapter 3 addresses the first research question.

Parts of this chapter stem from [MD10], [DRM10] and [DM11]. Details on the nature of some agreements were described in [DV13].

Chapter 4 proposes a collaborative ontology engineering approach built on top of the framework proposed in Chapter 3. In this method, agreements are both starting from and driven by the glosses. Chapter 4 also elaborates on the nature of meaning agreements on the labels used in the formal descriptions as well as glosses. Chapter 4 provides an answer to the second research question.

This chapter is mostly based on the work describing the hybrid ontology engineering method presented in [DM12].

Chapter 5 answers the fourth research question by proposing a method for driving the social processes within a community via their glosses. Drawing mainly inspiration from discourse theory to annotate glosses, the goal of a community would be to have the information contained in glosses to resemble the formal descriptions of the described concepts as close as possible.

Most of Chapter 5 started from [DV13], but was more thoroughly described in [DTM13].

Chapter 6 – which addresses the fifth question – proposes a method for using annotated information systems to drive social interactions within the community. Communities are not limited to re-interpreting the ontology for each iteration of the ontology engineering tasks, but also have access to the annotated data for discussing observations.

The notion of this software agent reasoning over these annotations was first mentioned in [Deb10]. [TD12] provided a first attempt at providing a translation of hybrid ontologies into a decidable fragment of first order logic, which has been refined and reported in [DTM13].

Chapter 7 and 8 respectively present the implementation of above-mentioned proposals in Chapters 3 to 6, and the application of the method and tool in the context of Linked Open Data research projects. The usability study used for Chapter 8 are partly based on [CD12] and [DC13].

Chapter 9 finally concludes this thesis, providing a summary of the contributions. This chapter also presents some of the limitations and future work.

1.3 A Note to the Reader

This thesis refers several times to Object Role Modeling (ORM) [HM08] and Natural language Information Analysis Methodology (NIAM) [Win90]. Both are methods and fact-oriented modeling languages for the development of information systems with a graphical notation that is easy to understand. ORM’s method, formalization and diagram language are directly derived from NIAM, and NIAM actually provides the basis for most ideas presented in this thesis. In 2005, Halpin introduced the next “generation” of ORM, called ORM2 [Hal05]. ORM2 is mostly concerned with the introduction of a new, more compact graphical representation of ORM diagram. The underlying formalism, however, did not change and is still based on [Hal89], a formalization of NIAM. The author of this thesis actually favors the graphical notation used in “ORM1”, as the distinction between object types and predicates are more apparent. As the graphical notation is not relevant, the thesis will therefore refer to the two “versions” of Object Role Modeling as ORM “tout court”.

Chapter 2

Background and Related Work

2.1 Introduction

This section presents an overview of what ontologies are and how they can be used. Even though the term *ontology* has its roots in philosophy [Gua98], only the notion used in computer science is taken into account. In computer science, the most cited definition of an ontology is given by Gruber [Gru95]: “an explicit specification of a conceptualization”. Since the formal definition of an ontology cannot completely specify the intended structures and semantics of each concept in the domain and can only approximate it, Guarino and Giaretta proposed to refine the definition as follows [GG95]: “an ontology is a logical theory which gives an explicit, partial account of a conceptualization.” Guarino and Giaretta thus weakened the definition given by Gruber. The main difference between a conceptual schema and an ontology is that the first is intended for the development of one particular information system in one organization and the latter for reuse and therefore general for a particular domain.

A more precise definition of ontology will be used in this thesis: *“The formal semantics of a (computer-based) system quite simply is the correspondence between this system and some real world as perceived by humans. It is usually given by a formal mapping of the symbols in the system’s description to objects in that real world, such that relationships and logical statements in the specification language can be assigned a truth-value depending on whether a certain state of affairs among objects exists in the real world. As the real world is not usually directly accessible inside a computer, storing and reasoning about semantics requires the world to be replaced by an agreed specification of a conceptualization, often in the shape of a formal (mathematical) construct. This computer-based, shared, agreed formal specification of a conceptualization is what is known as an ontology.”*

Ontologies constitute the key resources for realizing a Semantic Web [BLHL01] and help tackle the difficulty of interoperating *autonomously* developed and maintained information systems in a meaningful (i.e. semantic) way.

2.1.1 Interoperability and the Semantic Web

In general, semantic interoperability is defined as the ability of two or more information systems or their (computerized) components to exchange data, knowledge or resources and to interpret the information in those systems [DLM08].

Ontologies are not only useful for semantic interoperability. Other motivations for ontologies can be summarized as follows [UG96, RB06]: the need for communication between

humans, re-usability of shared understandings, reliability between information systems and a common vocabulary for humans to write project specifications.

In the beginning, research on ontologies was mostly driven by the idea of producing models of reality that reflect the “true” structures, independent of subjective judgment and context [Hep08]. Others, such as Fensel found that it is impossible to produce such “true” models and claimed that consensual, shared human judgments must be the core of ontologies [Fen01]. Ontologies are “social” artifacts as ontologies are the result of agreement processes within a *community*. It is difficult, if not impossible, to align and find consensus on every community member’s perceived reality. As a consequence, the compromises needed to achieve consensus result in an artifact suitable for a particular task, but not strictly comforting to reality or all of the perceived realities.

The scientific community often debates whether an ontology is a conceptual system or the specification thereof [Hep08]. Some researchers argue that an ontology is an abstraction over a domain of interest in terms of its conceptual entities and their relationships. For others, an ontology is rather the approximate specification of such an abstraction.

A common misconception is that ontologies are knowledge bases and vice versa [Hep08]. This is in part due to some formalisms such as OWL and RDF(S) allowing the creation of both in one artifact. The distinction between the two, however, is that ontologies provide vocabularies and the information in knowledge bases will be expressed in terms of these vocabularies. This means that ontologies should express things about concepts and individuals are (normally) not part of the ontology.

When creating an ontology, some aspects need to be taken into account. The following list of these aspects is based on the work of Hepp [Hep08].

- When choosing a formalism for creating the ontology, one can range from a simple vocabulary to high order logics, all depending on the purpose for which the ontology will be used. The higher the expressiveness, the more sophisticated reasoning you can perform. As the expressiveness increases, however, so will the difficulty to create ontologies and the computational costs of reasoning.
- Also the size of the relevant community (of users) is important. Hepp noted that the most important number is the number of human stakeholders that are expected to commit to (use) the ontology. The way you reach agreements within a community and how (well) you document the ontology depends on the number of users [Hep07].
- The number of changes in a time unit will influence the versioning strategy of the ontology.
- The number of concepts in the ontology. Large ontologies tend to bring problems (e.g. visualization and in-memory loading). It is observed that smaller ontologies tend to be more quickly adopted [Hep07].
- Also having an impact on the agreement mechanisms is whether the intended use of the ontology involves concepts for which a certain degree of heterogeneous subjective views are possible. This also depends on the heterogeneity of the group. Examples of these are religion and culture.
- Finally, the level of detail for describing concepts in the ontology not only influences comprehension of the ontology, but also has an impact on the agreement mechanisms.

Note that next to the number of changes in a given time unit, all the other aspects influence the agreement processes. Agreement processes need to be guided by a method. Vrandecic *et al.* reported on an experiment in which it became apparent that ontology construction without guidelines was inefficient compared to ontology construction with prescribed processes [VPST05]. *Ontology engineering methods* will be mentioned in Section 2.2. But, first the ontologies as an artifact and some popular ontology formalisms will be described.

Ontologies can provide unique identifiers to each concept. Such an identifier can be a label that unambiguously refers to a concept, a generated artificial ID, or - in popular Semantic Web languages - a URI¹. The use of unique identifiers solves – partly – the problem of homonymy and synonymy in ontologies. To reason, a software agent only needs to care about the unique identifiers and the formalism to infer new facts. In fact, a software agent sufficiently powerful to “understand” $P \rightarrow Q \rightarrow R$ can deduce that $P \rightarrow R$ then also holds. But what do P , Q and R actually mean? As it turns out, in order to avoid unwanted interpretations of ontologies, one can use “informal” semantics to clear things out. Ontologies can be documented with textual definitions, synonym sets, pictures, etc. In other words: human-readable or human-interpretable documentation. *Dog → Mammal → Animal* bears more meaning for us humans than $P \rightarrow Q \rightarrow R$.

One of the main problems Hepp noted is the interplay between ontologies and natural language [Hep08]. A tight integration with human language is crucial for ontologies to be successful. In later sections we will discuss ontology engineering methods that take this integration into account.

2.1.2 Ontology- and Semantic Web Languages

Figure 2.1 depicts the so-called Semantic Web Stack (or Semantic Web Layer Cake). It is an adaptation of Berner-Lee’s original stack in one of his talks². This thesis focuses on the following layers: (i) data interchange via the RDF (data) model, (ii) the development of concept- and role-hierarchies with RDF(S), (iii) building ontologies with the Web Ontology Language (OWL) and (iv) the support for describing business rules with SWRL. The last is important to introduce rules that cannot be modeled with the Web Ontology Language.

2.1.2.1 Resource Description Framework

A W3C Recommendation since 1999, the Resource Description Framework [BG04] (RDF) is an abstract data model that defines relationships between resources on the Web. These resources are not restricted to documents (or *information resources*), but can also be used as a surrogate for real world objects (*non-information resources*, or “things” described by documents). Information is encoded in sets of “triples”, where the sources of the relation is called the *subject*, the relation itself the *property* and the relation’s destiny the *object*.

¹Note that Semantic Web languages allow “nameless” concepts, so called blank-nodes. While a software agent processes blank nodes, it will generate an artificial node identifier on the fly.

²<http://www.w3.org/2000/Talks/1206-xml2k-tbl/slides10-0.html>

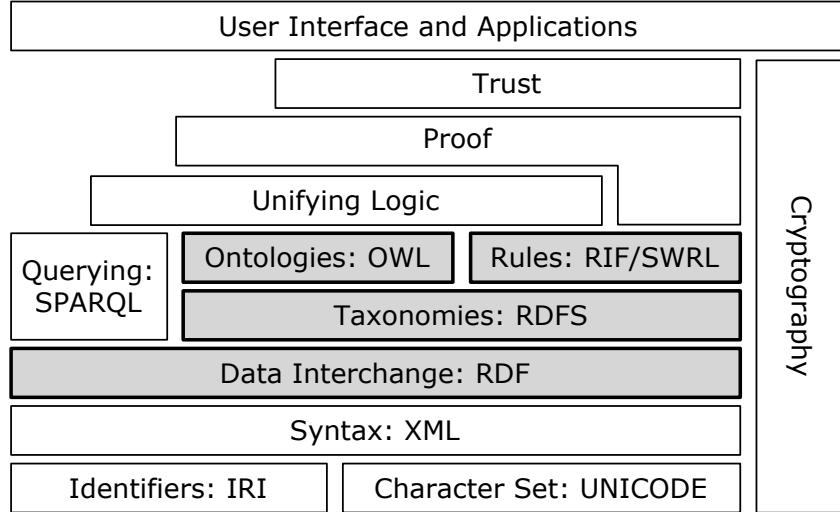


Figure 2.1: The Semantic Web Stack. Highlighted layers will be described in this thesis.

The subject of an RDF-triple is always a URI, the object either a URI or a lexical value. RDF descriptions are intended to be processed by machines and not read by humans.

Although many other serialization of RDF exists, this section only presents the XML serialization of RDF and expects the reader to be familiar with the basic concepts of XML and XML namespaces. Figure 2.2 shows an example of RDF in RDF/XML. This document describes the resource with a particular URI stating it has a property `dc:title` with value “Homepage of Christophe Debruyne” and a property `dc:creator` with value “Christophe Debruyne”. The two properties in the example come from Dublin Core³. Dublin Core is a set of labels (title, publisher, subjects, etc.) used to catalog a wide range of networked resources, such as digitized text documents, photographs, etc. Machines, however, do not understand what `dc:creator` and `dc:title` mean. Humans need to follow the descriptions of those two properties in that namespace to find more information⁴⁵. Humans need to interpret the information of those labels and write their applications in such way that data annotated with those labels are interpreted and processed in the correct way. RDF is not very expressive and therefore the possibilities for reasoning are limited. For instance, it is not possible to create concept hierarchies, role hierarchies or even declare the domain and range of properties. Such expressiveness would allow lightweight reasoning while querying. An answer to that problem came with RDF Schema, or RDF(S), which will be described in the next section.

2.1.2.2 RDF Schema

RDF Schema [BG04], a W3C recommendation since 2004, is an extension of RDF that provides a framework for describing vocabularies (classes, properties and values). It allows one to define class- and role hierarchies as well as domain- and range restrictions

³<http://www.dublincore.org/>

⁴[#title](http://dublincore.org/documents/usageguide/elements.shtml)

⁵[#creator](http://dublincore.org/documents/usageguide/elements.shtml)

```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description
    rdf:about="http://www.christophedebruyne.be/index.html">
      <dc:title>Homepage of Christophe Debruyne</dc:title>
      <dc:creator>Christophe Debruyne</dc:creator>
    </rdf:Description>
</rdf:RDF>

```

Figure 2.2: RDF/XML describing Christophe Debruyne's Web page.

on properties.

Figure 2.3 depicts an example of using RDF(S) to describe classes and properties between those classes. In this example, there are three classes: **Person**, **Teacher** and **Course** where **Teacher** is a subclass of **Person**. A property **hasTeacher** is also defined between **Course** and **Teacher** by using range- and domain restrictions. This RDF Schema can now be used in other RDF documents to create instances of those concepts or add additional statements around those concepts by referring to the physical location of that schema in a namespace. An example of creating instances of this RDF(S) is shown in 2.4.

```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
           xml:base="http://www.christophedebruyne.be/example.rdf#">
  <rdfs:Class rdf:ID="Person">
    <rdfs:comment>Person Class</rdfs:comment>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Teacher">
    <rdfs:subClassOf rdf:resource="#Person"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Course">
    <rdfs:comment>Course Class</rdfs:comment>
  </rdfs:Class>
  <rdf:Property rdf:ID="hasTeacher">
    <rdfs:comment>Teacher of a course</rdfs:comment>
    <rdfs:domain rdf:resource="#Course"/>
    <rdfs:range rdf:resource="#Teacher"/>
  </rdf:Property>
</rdf:RDF>

```

Figure 2.3: Example of RDF(S) for describing classes and properties between those classes.

Even though RDF(S) allows one to model classes, properties and their hierarchies, it is still too weak to describe resources in sufficient detail. For instance, RDF(S) does not provide localized range and domain constraints; it is, for instance, not possible to state that the range of **hasChild** are people when applied to people and armadillos

```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
           xmlns:ex="http://www.christophedebruyne.be/example.rdf#"
           xml:base="http://www.christophedebruyne.be/example2.rdf#">
  <ex:Course rdf:ID="Informationsystems">
    <ex:hasTeacher>
      <ex:Teacher rdf:ID="Robert" />
    </ex:hasTeacher>
  </ex:Course>
</rdf:RDF>

```

Figure 2.4: Example of using the RDF(S) depicted in 2.3 to create instances of those classes and properties.

when applied to armadillos. The absence of existence and cardinality constraints in RDF(S) poses another problem; e.g. one cannot describe that every person has exactly two (biological) parents. RDF(S) has also no means for defining transitive, inverse or symmetrical properties. With RDF(S) it is thus not possible to state that `isAncestorOf` is a transitive property, or that `hasPart` is the inverse of `isPartOf`, or that `touches` is symmetrical. The reasoning capabilities of RDF(S) are limited to inferring implicit knowledge for information retrieval, e.g. retrieving all the names of nurses where person have names, and nurses are people.

Why is reasoning useful? Reasoning can help one ensure that a particular knowledge base is meaningful (all named classes can have instances), correct (it captured the intuitions of domain experts) and minimally redundant, i.e. it contains no unintended synonyms. Reasoning can furthermore help one querying over ontology classes and instances, e.g. finding more general and specific classes or retrieve individuals (instances) or tuples matching a given query. The need for this expressiveness and reasoning support brings us to the Web Ontology Language (OWL).

2.1.2.3 Web Ontology Language

The Web Ontology Language (OWL) [BvHH⁺04, SHKG09], of which OWL 2.0 [MGH⁺09] became recently a W3C recommendation, envisaged an ontology language that should allow users to provide an explicit, formal conceptualization of a domain of discourse. The motivations of the development of OWL were: a well-defined syntax, with formal semantics and efficient reasoning support.

OWL first came in three different sublanguages each geared toward fulfilling different aspects of requirements: OWL Full, OWL DL and OWL Lite. OWL Full uses all the OWL language primitives and allows the combination of these primitives in arbitrary ways with RDF. Because of this, OWL Full is fully upward compatible with RDF both syntactically and semantically. The downside is that OWL Full is so powerful that it is undecidable. Undecidable means it is impossible to construct an algorithm that always leads to a correct yes-or-no answer. In OWL DL, DL stands for Description Logics, the

grounding of this particular sublanguage. Description Logics are a family of decidable fragments of first-order logic, which will be described in more detail in Section 2.1.4. OWL DL does not allow certain ways of applying constructors, losing full compatibility with RDF. Every legal OWL Document is a legal RDF document, but not every RDF document is a legal OWL DL document. OWL DL, however, permits efficient reasoning support and is the most expressive decidable OWL sublanguage. OWL Lite provides an even further restriction. Some examples of such restrictions are the exclusion of enumerated classes, disjointness statements, and arbitrary cardinality. OWL Lite was geared towards tractability, meaning that reasoning tasks – which are decision tasks – can be solved in polynomial time.

OWL 2 introduced different sublanguages [MGH⁺09], where some expressive power is traded for efficient reasoning. OWL 2 EL is particularly useful for ontologies with large numbers of classes and/or properties with a particular flavor of Description Logics with only existential quantification. OWL 2 QL stands for OWL 2 Query Language and is a flavor for, as it names implies, dealing with vast amounts of instance data. The RL in OWL 2 RL stands for Rule Language is aimed at scalable reasoning tasks such as consistency checking without loosing too much expressiveness.

2.1.3 Semantic Web Rule Language

The Semantic Web Rule Language, or SWRL is a proposal [HPSB⁺04] for a Semantic Web rules language⁶, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog).

All rules in SWRL are expressed in terms of OWL concepts (classes, properties, individuals, literals, etc.). Rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. Below is an example stating that when a person has a sibling that is male, the sibling is the brother of that person.

$$\text{Person}(\textit{?p}) \wedge \text{hasSibling}(\textit{?p}, \textit{?s}) \wedge \text{Man}(\textit{?s}) \rightarrow \text{hasBrother}(\textit{?p}, \textit{?s})$$

2.1.4 Description Logics

This section covers one of the better-known knowledge representation languages, Description Logics (DLs) [BCM⁺03, Sat03]. DLs are a family of logic based knowledge representation languages providing a set of constructors to build complex concepts and roles starting from atomic concepts and atomic roles, which can then be used to extract implicit knowledge about concepts and individuals via a reasoner. Among those tasks are, for example, subsumption and concept consistency. The first is used to check whether a given concept is a subset of another concept and the latter to see whether a concept can have instances. When the latter is not possible for a concept, this means there is a contradiction in one of the definitions of that concept.

⁶A specification can be found here: <http://www.daml.org/2003/11/swrl/>

In a DL, the basic notions are atomic concepts and atomic roles. Atomic concepts are used to group objects that share common features and atomic roles refer to a binary relation between concepts and concepts, or concept and data values. DLs also offer constructs to create complex concepts and roles. The set of concepts is denoted with \mathbf{C} and the set of roles with \mathbf{D} .

There are many DL dialects that are named following a naming convention that states which operators are allowed starting from a base language. The base language \mathcal{AL} , which stands for attributive language, allows (i) the negation of concepts that do not appear on the left hand side of axioms (atomic negation), (ii) concept intersection, (iii) universal restrictions and (iv) limited existential quantification. Limited existential quantification means that one can look for the existence of a role played by an instance of concept with another instance, but cannot specify the specific concept to which the second instance has to belong to. The letters used in the naming convention and the extension they refer to are given below⁷:

- \mathcal{F} for functional properties;
- \mathcal{E} for full existential qualification⁸;
- \mathcal{C} for complex concept negation;
- \mathcal{U} for concept union;
- \mathcal{H} for role hierarchies;
- \mathcal{R} for role inclusion axioms, reflexivity, irreflexivity and role disjointness;
- \mathcal{O} for nominal, which are enumerated classes of object value restrictions;
- \mathcal{I} for inverse properties;
- \mathcal{N} for cardinality restriction⁹;
- \mathcal{Q} for qualified cardinality restrictions;
- (\mathcal{D}) for using data type properties, data types and data values.

A popular DL is \mathcal{ALC} , the attributive language extended with complex concept negations. The semantics of this DL is given in terms of first-order logic interpretations. An interpretation I is a pair $(\Delta^I, .^I)$, where Δ^I is the interpretation domain and $.^I$ is the interpretation function. The domain is a non-empty set of objects, and the interpretation function maps each atomic concept $A \in \mathbf{C}$ to a subset $A^I \subseteq \Delta^I$ and each atomic role $r \in \mathbf{D}$ to a binary relation $r^I \subseteq \Delta^I \times \Delta^I$. The extension of $.^I$ to arbitrary concept descriptions for the language \mathcal{ALC} is defined as shown in the semantic columns of Table 2.1. Note that \mathcal{ALC} is equivalent with \mathcal{ALCEU} , and therefore existential restriction and disjunction will be described.

A knowledge base (KB) generally contains two distinct components, the Terminological Box (TBox) and the Assertional Box (ABox). The TBox provides the vocabulary for the universe of discourse (i.e. the ontology), whereas the ABox defines named individuals in terms of this vocabulary.

⁷This list is based on http://en.wikipedia.org/wiki/Description_logic.

⁸The type of the second instance can thus be specified.

⁹Not qualified, thus the type of the instances are not specified.

Table 2.1: The syntax and semantics of DL \mathcal{ALC} .

Description	Syntax	Semantics
Top (all concept names)	\top	$\Delta^{\mathcal{I}}$
Bottom (empty concept)	\perp	\emptyset
Conjunction of concepts	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction of concepts	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation (complement)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Universal restriction	$\forall r.C$	$\{x x \in \Delta^{\mathcal{I}} \wedge \forall y(r(x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$
Existential restriction	$\exists r.C$	$\{x x \in \Delta^{\mathcal{I}} \wedge \exists y(r(x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$

Terminological Box Statements in the TBox are called terminological axioms and are of the form $A \sqsubseteq B$ or $A \equiv B$. The first are called concept inclusions and state the necessary conditions for an instance of A to be considered also an instance of B . The latter is called a concept equivalence, which is basically a shorthand for stating both $A \sqsubseteq B$ and $B \sqsubseteq A$. With a concept equivalence, one describes the necessary and sufficient conditions for an instance of a concept to be considered an instance of the other and vice versa. The interpretation of both concept inclusion and concept equivalence are given in Table 2.2. A TBox \mathcal{T} is said to be coherent if and only if the interpretation \mathcal{I} satisfies all terminological axioms in \mathcal{T} . Coherence is written as $\mathcal{I} \models \mathcal{T}$.

Table 2.2: The syntax and semantics of concept inclusions and concept equivalence.

Description	Syntax	Semantics
Concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$

\mathcal{ALC} does not provide role axioms as more complex DLs do. One DL that provides a set of statements to describe the characteristics of roles is $\mathcal{EL+}$ [BBL05]. A role inclusion is an axiom of the form $r_1 \circ \dots \circ r_n \sqsubseteq s$, where r_1, \dots, r_n and s are role names and \circ denotes the composition of binary relations. These axioms can be used to define more complex role definitions, e.g. transitive roles of the form $r \circ r \sqsubseteq r$. Similar to concept equivalence, a role equivalence is merely an abbreviation of both $r \sqsubseteq s$ and $s \sqsubseteq r$. The interpretation of both role inclusion and role equivalence are given in Table 2.3.

Table 2.3: The syntax and semantics of role inclusions and role equivalence.

Description	Syntax	Semantics
Role inclusion	$r_1 \circ \dots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
Role equivalence	$r \equiv s$	$r^{\mathcal{I}} = s^{\mathcal{I}}$

Assertional Box The ABox provides information that is specific for an application domain in terms of concept and roles by (i) defining individuals and giving individuals names, and (ii) fill in the roles with relation individuals such that the information is appropriate. The interpretation \mathcal{I} is then extended to map each individual a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Given two named individuals a and b , a concept C and a role r ; an interpretation \mathcal{I} is said to satisfy a concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and a role assertion $r(a, b)$ if $r(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. An ABox \mathcal{A} is said to be satisfied with respect to an interpretation \mathcal{I} if and only if every assertion in \mathcal{A} is satisfied. This is written as $\mathcal{I} \models \mathcal{A}$.

There are two common assumptions about ABoxes:

1. The unique name assumption (UNA) infers that individual names in \mathcal{I} are distinct. In other words, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.
2. The open world assumption (OWA) presumes that the information in the ABox is incomplete. In other words, assertions in the ABox expresses that these assertions are true in all models, but not that all unknown information not occurring in the models are not true.

Reasoning DL systems allows one not only to describe a universe of discourse, but also to reason about a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. A knowledge base \mathcal{K} is said to be satisfied by an interpretation \mathcal{I} if and only if \mathcal{I} satisfies both \mathcal{T} and \mathcal{A} . Testing the satisfiability of a knowledge base can be done by checking whether $\mathcal{K} \not\models \perp$ holds. This thus means checking whether \mathcal{K} under \mathcal{I} has a model. A concept C is unsatisfiable with respect to \mathcal{K} if there exists no model of \mathcal{I} of \mathcal{K} for which $C^{\mathcal{I}} \neq \emptyset$. Subsumption checking, written as $\mathcal{K} \models C \subseteq D$ can be defined in terms of concept satisfiability checking, as checking whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ is true for every model of \mathcal{K} under \mathcal{I} can be reduced to checking whether the concept $C \sqcap \neg D$ is concept satisfiable in \mathcal{K} under \mathcal{I} . Once the TBox is coherent and consistent, one can populate the ABox with assertions, which in turn will need to be assessed. This is done by checking whether the knowledge base models each assertion in every model of \mathcal{K} under \mathcal{I} .

Relationship with First-order Logic Description logics are decidable fragments of first-order logic. Concept names are unary predicates and role names are binary predicates. Concept descriptions correspond to first-order formulas with one free variable, which will be bound when used in a concept inclusion statement [BHS08]. The translation of assertions in a DL into first-order formulas are provided by [BHS08]. The translation of concept description C into a first-order formula with one free variable $\tau_x(C)$ is defined as follows:

1. $\tau_x(A) := A(x)$ for atomic concepts A
2. $\tau_x(C \sqcap D) := \tau_x(C) \wedge \tau_x(D)$
3. $\tau_x(C \sqcup D) := \tau_x(C) \vee \tau_x(D)$
4. $\tau_x(\neg C) := \neg \tau_x(C)$
5. $\tau_x(\forall r.C) := \forall y(r(x, y) \rightarrow \tau_y(C))$, where variable y is different from x
6. $\tau_x(\exists r.C) := \exists y(r(x, y) \wedge \tau_y(C))$, where variable y is different from x

Given a TBox \mathcal{T} with concept-inclusions, the translation $\tau(\mathcal{T})$ of \mathcal{T} is given by:

$$\tau(\mathcal{T}) := \bigwedge_{C \sqsubseteq D \in \mathcal{T}} \forall x(\tau_x(C) \rightarrow \tau_x(D))$$

This section described what DLs are and provided more details on the fairly common DL \mathcal{ALC} .

So far, this chapter explained what ontologies in computer science are and described ontology languages used in the Semantic Web. The problem of how ontologies come to be, however, has not yet been described. The next section will therefore provide a literature review on (collaborative) ontology engineering methods.

2.2 Ontology Engineering

The problem is not so much what ontologies in computer science are, but how they come to be. This section will describe the state of the art on ontology engineering and ontology engineering methods in particular.

Definition 1 (Ontology Engineering [GPFLC03])

The set of activities that concern the ontology development process, the ontology life cycle, the principles, methods and methodologies for building ontologies, and the tool suites and languages that support them, can be referred to as ontology engineering.

For the last twenty years, many methods have been put forward for how to develop ontologies. It seems, however, that research on methodological activities has diminished in recent years [Ber10]. Bergman observed that very few discrete methods exist and those that do are often older in nature [Ber10]. He furthermore noted that most methods shared several logic steps from assessment to deployment, from testing to refinement.

Quite a few surveys on the state of the art on ontology engineering methods exist. Recent surveys include [SS10], [ST06] and [GPFLC03]. Corcho *et al.* observed in [CFLGP03] that there is no correspondence between ontology building methods and tools, except for - at that time - METHONTOLOGY [FLGPJ97] (which has the WebODE [CFLGPV02] tool¹⁰) and On-To-Knowledge (with their OntoEdit [SEA⁺02] tool suit that later became the KAON¹¹ project). The DOGMA [JM09] project developed several tools that supported their method [DD08].

In general, one can identify two phases in an ontology engineering method: elicitation and application [DdMM07]. In the elicitation phase, knowledge is extracted from various resources such as documents of any kind or the experience of domain experts within a

¹⁰WebODE was discontinued in 2006, see <http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/technologies/60-webode>

¹¹<http://kaon.semanticweb.org/>

specific context. In the subsequent application phase, an ontology is used in an application context.

In ontology engineering, different stakeholders sharing a common goal constitute a *community* and that community tries to create a shared conceptualization of their domain that enables communication and semantic interoperability between their autonomously developed and maintained information systems. These stakeholders need to collaborate in order to achieve this shared agreement, rendering ontology engineering a collaborative process.

The requirements and needs of the community of stakeholders constantly evolve, and thus the common ontology they are creating constantly co-evolves. Scalable ontology engineering is quite difficult when the requirements rapidly evolve, since multiple stakeholders might have multiple views for a particular part of the world. These different views need to be interpreted, aligned and negotiated about.

Ontology engineering can be related to collective intelligence. Collective intelligence is defined as the degree to which the agents in a system collectively can make good decisions as to their future course of action; in particular, the degree to which the agents collectively can make better decisions than any of them individually [Hey99]. Collective intelligence typically emerges by combining knowledge from different agents, as the knowledge of one agent differs from the knowledge of another agent, the combined knowledge will constitute a larger pool of more diverse knowledge [Hey11]. Key is the coordination of activity. Coordination is the arrangement or mutual alignment of actions so as to maximize synergy and minimize friction in their overall pattern on activity [Hey11]. The coordination of the ontology engineering processes can be facilitated by the use of *glosses*. Glosses are natural language definitions of concepts and will turn out to be key in the alignment of every stakeholder's thoughts before working together towards an ontology. Indeed, if something is not clear between human agents, they will ask questions such as "What do you mean with ...?", "Could you explain ...?" and discuss the definitions which will then be used as a baseline for subsequent interactions.

As a consequence, a complex socio-technical process of ontology alignment and meaning negotiation is required. For a collaborative ontology engineering method to be successful, the method should thus support domain experts in gradually building and managing increasingly complex versions of an ontology in all its aspects [dMDM06, DdMM07].

2.2.1 Ontology Engineering Methods

This section now proceeds with a closer look into the existing methods and analyze to what extent they mention or treat glosses or social processes as a first-class citizen in the method. The methods are ordered alphabetically.

- The Cyc [GL90, Len95] project attempted to assemble a comprehensive ontology and knowledge base of everyday common sense knowledge to support human-like reasoning by machines. The Cyc system includes a variety of interface tools that permit the user to browse, edit, and extend the knowledge base, send a query to the inference engine and to interact with the natural language and database integration

modules. Statements are made with a special syntax. For instance, the following statement asserts that Christophe Debruyne is an instance of Person.

```
(#$isa #$ChristopheDebruyne #$Person)
```

In Cyc, the concept names are known as constants and are used for individuals, collections, truth functions (applied to one or more concepts and return either true or false), and functions (which produce new terms from given ones). Every constant in Cyc should be well-documented and ontology engineers are thus required to give every constant a comment upon creation. This is the most common form of documentation in the knowledge base for which guidelines are given to the users¹². Note that the comments reside at the same level of the ontology and that the collection of descriptions are not treated as a separate resource.

“Knowledge enterers” communicate with Cyc in an expressive language. Their input is then converted into the heuristic language, which is efficient for dealing with many sorts of frequently recurring inference problems. Cyc users simultaneously enter the Cyc data and information is checked before it actually enters the system. The platform, however, does not support collaboration. The knowledge enterers furthermore need proper training in entering and commenting the information.

OpenCyc is the open source version of the Cyc technology¹³.

- DILIGENT [PST04, VPST05, TPS06, PTS09], which stands for *distributed, loosely controlled and evolving engineering of ontologies*, supports domain experts in a distributed setting to engineer and evolve ontologies with the help of a fine-grained methodological approach based on Rhetorical Structure Theory [MT88] to structure and analyze the discussions in ontology engineering processes. The ability to present the reasons and arguments for a modeling decision to the new entrants could speed up the design process. A similar problem arises, when the ontology is revised and the ontology engineers need to recall the reasons for the previous design. The users of the ontology can as well profit from a well-documented ontology for a better understanding. In this method, the users discuss issues, which are usually specified at the ontology level (e.g. how should a particular classification be structured). The users present their arguments, suggest alternatives, agree and disagree with one another, and vote on the resolution. The editing environment explicitly supports these steps, by extending an existing tool. A version is locally adopted by the users after which a board of ontology stakeholders analyze those local changes and try to converge the differences into a new version of the ontology. After that, the whole cycle starts again.

DILIGENT also provides an argumentation ontology [TSPSS05]. The main notions in this ontology are issues, ideas and arguments, which are represented as classes. The central motivation for this is to keep track of the change arguments.

An existing ontology engineering tool, OntoEdit [SEA⁺02, SAS03] and developed by the same group, was extended to support DILIGENT [PSST04]. OntoEdit is

¹²See <http://www.cyc.com/doc/handbook/oe/04-documenting-the-KB.html>

¹³See <http://www.opencyc.org/>

an ontology engineering-environment for inspecting, browsing, implementing and modifying ontologies. Modeling ontologies using OntoEdit is done at modeling at a conceptual level. It is done so in a formalism-agnostic manner and with graphical user interfaces to represent views on conceptual structures.

In [SEA⁺02], Sure *et al.* referred to a lexicon component. The authors did not describe agreement processes for natural language definitions. The description of the method and tool (including screenshots) suggests that only the formal descriptions of the concepts were taken into account.

- A DOGMA [JM09] – which stands for *Developing-Ontology Grounded Methods for Applications* – inspired ontology is based on the classical model-theoretic perspective [Rei82] and decomposes an ontology into a lexon base and a layer of ontological commitments [Mee99b, Mee01a], called the principle of double articulation [SMJ02]. A lexon base holds (multiple) intuitive conceptualization(s) of a particular domain. Each conceptualization is simplified to a “representation-less” set of context-specific binary fact types¹⁴. The commitment layer mediates between the lexon base and its applications. Each such ontological commitment defines a partial semantic account of an intended conceptualization [GG95]. It consists of a finite set of axioms that specify which lexons of the lexon base are interpreted and how they are visible in the committing application, and (domain) rules that semantically constrain this interpretation.

A collaborative ontology engineering method and a meaning evolution support system (MESS) were defined. The whole was called DOGMA-MESS [dMDM06]. A Concept Definition Server (CDS) [DBSM04, DdM05, Jar06] was defined for keeping natural language descriptions - called glosses - for concepts referred to by the terms in the lexon base. As different terms (in different contexts) can refer to the same concept, it also provided a way for keeping track of synonyms.

DOGMA also served as the basis for Business Semantics Management [DCM10], which encompasses the technology, method, organization, and culture that brings business stakeholders together to collaboratively realize the reconciliation of their heterogeneous metadata.

Business Semantics Management (BSM) [DCM10] draws from best practices in ontology management [DLM08] and ontology evolution [HDdS08]. The representation of business semantics is based on the DOGMA approach. BSM consists of two complementary cycles: *semantic reconciliation* and *semantic application* that each groups several activities.

Semantic Reconciliation is the first cycle of the method. In this phase, business semantics are modeled by extracting, refining, articulating and consolidating lexons from existing sources such as natural language descriptions, existing metadata, etc. Ultimately, this results several consolidated language-neutral semantic patterns that are articulated with glosses (e.g. WordNet [Fel98] word senses). These patterns are reusable for constructing various semantic applications. This process is supported

¹⁴A fact type is a generalization of a fact, e.g. "Person is born on Date" is a fact type and "Christophe is born on 8 August 1984" a fact. Facts are thus instances of fact types. Fact types are elementary when they cannot be simplified without loss of meaning.

by the Business Semantics Glossary. Semantic Application is the second cycle. During this cycle, existing information sources and services are committed to a selection of lexons, as explained earlier. In other words, a commitment creates a bidirectional link between the existing data sources and services and the business semantics that describe the information assets of an organization. The existing data itself is not moved nor touched.

The tool supporting BSM is called Business Semantics Glossary (BSG)¹⁵ [DDM11]. BSG is a web-application aimed at both business as well as technical users. It lets people collaboratively manage their business semantics and is based on the Wiki paradigm that is a proven technique for stakeholder. Governance models are built-in and user roles (e.g. steward, stakeholder) can be applied to distribute responsibilities and increase participation. The software takes care of the audit trails who changed what, when and why.

The BSG is the vehicle that serves the reconciliation of the newly scoped concepts. The BSM cycle is repeated until an acceptable balance of differences and agreements is reached between the stakeholders that meets the requirements of the semantic community. Gradually, closed divergent metadata sources are replaced with metadata sources that follow an open standard, and are kept coherent via BSG. After a consensus has been obtained using BSM with the glossary, the terms and relations in the ontology can be implemented in other formalisms such as OWL and RDF(S). BSG has, for instance, been applied in the Flemish public administration [DDS⁺11].

- HCOME (Human-Centered ONtology engineering MEthodology) [KV03, KVA04, KV06] supports the development of ontologies in a decentralized fashion. They introduce three different spaces in which ontologies can be stored. The first one is the Personal Space. In this space users can create and merge ontologies, control ontology versions, map terms and word senses to concepts and consult the top ontology. The evolving personal ontologies can be shared in the Shared Space. All participants can access the shared space. In the shared space users can discuss ontological decisions. After a discussion and agreement the ontology is moved to the Agreed space. HCOME aimed at (i) supporting the active and decisive involvement of knowledge workers in all stages of the ontology life cycle, and (ii) further empowering people to engineer their ontologies shaping their information space in ways that are seamless to their day-to-day working activities.

The HCOME prototype – called HCONE [KV03] (Human Centered ONtology engineering Environment) – explicitly groups members for a given ontology. While working on the ontology, the facts are verbalized in natural language for validation by the user to ensure what they have entered is what they meant to enter. The prototype furthermore supports aligning the concepts in the ontology with external sources providing definitions such as WordNet [Fel98]. From the papers, however, it seems that users are not able to enter their own definitions.

Through an argumentation dialogue, any group member can raise issues, propose solutions and post arguments concerning an ontology that has been shared. The concepts in those dialogues are: (i) issue, representing a problem to be solved; (ii)

¹⁵<http://collibra.com/products-and-solutions/products/business-semantics-glossary>

position, either a solutions to an issue raised, or a new version of the discussed conceptualization; and (iii) argument. Arguments are either supporting arguments speaking in favor of a position or objecting argument speaking against a position.

In later work, wiki's were adopted for the HCOME method [VKCL07, Kot08].

- Two approaches for collaborative ontology design were proposed in [HJ02] and [KA06]. The two approaches consist of four phases: preparation, anchoring, iterative improvement and application. Holsapple and Joshi emphasize the design criteria for the ontology and how the ontology should be evaluated in the preparation phase [HJ02]. The anchoring phase is used to specify the initial ontology or ontologies to seed the collaborative effort.

In [HJ02], Holsapple and Joshi uses an adaptation of the Delphi method for collecting and integrating the views of multiple people about the same topic. The Delphi method is a research method which aims to reach consensus on a subject by: 1) gathering the opinions from several experts, 2) presenting to the experts the opinions of others (rendered anonymous, of course), 3) allowing the experts to refine their opinion based on the other opinions they have received. With this method, one tries to reach a consensus (one opinion) in several rounds.

Karapiperis and Apostolou adopted the phases of Holsapple and Joshi, but replaced the Delphi method with voting in a Nominal Group Technique manner [KA06]. Nominal Group Technique is a formal technique to make pooled judgments and decisions in groups that meet face-to-face. Unlike traditional voting, this technique takes into account all opinions by following five stages: 1) introduction and explanation of the problem, 2) silent generation of ideas, 3) sharing ideas, 4) group discussion, 5) voting and ranking.

Both approaches introduced existing techniques to achieve consensus. These techniques can also be used to elicit knowledge, as that knowledge will be part of the different solutions in the different opinions. Both approaches did not mention any specific tool support.

- METHONTOLOGY [FLGPJ97] adopts a development process for ontologies where the tasks and their order are defined and were based on the main activities of software development and database engineering methods. METHONTOLOGY was an early attempt to reuse components from existing database modeling method. Ontologies in this method also follow a life-cycle, where the different stages that an ontology passes are identified as well. Terms and axioms can be described in natural language, but those are only intended for documentation purposes.

WebODE [CFLGPV02, CFLGPLC03, AVCFLGP03] is an ontology engineering tool suite based on an application server, whose development started in 1999. The server provides services upon which clients could be developed. It provides support for the METHONTOLOGY method, but can be used with other methods as well. Development and support for this tool stopped in 2006. Even though the platform allows several people to work on the same ontology, the platform does not support social collaboration processes.

Indeed, the goal of the authors was to prescribe the processes rather than focus on the social aspects of ontology engineering.

- The NeOn Methodology [GPdCSF09] – a result of the NeOn project¹⁶ – for building ontologies is a scenario-based method that supports the collaborative aspects of ontology development and reuse as well as the dynamic evolution of ontology networks in distributed environments. The scenarios emphasize the reuse of ontological and non-ontological resources, the reengineering and merging, and taking into account the collaborative and dynamic aspects of ontologies. The NeOn Glossary [dCSFGP08] identify and define the processes and activities carried out when ontology networks are collaboratively built by teams. Guidelines were given for the different processes and activities of the ontology; including template forms, workflows and examples.

The NeOn Toolkit¹⁷ provides ontology engineering tools developed during the NeOn project. The tools are published as open source. After the project, the NeOn Foundation¹⁸ was raised to support the toolkit and other technologies developed in the project. The toolkit is built around the Eclipse framework and provides the ontology engineering tools as a set of plugins.

Both the Cicero [DEMB⁺08] wiki and the corresponding NeOn Toolkit plugin¹⁹ support ontology developers and users to keep track of discussions on a given ontology. The actual discussions are held in the Cicero-Wiki on a central server. The plugin allows for establishing links between elements in an ontology and discussions that influenced their design. Knowledge from the DILIGENT method was used to structure the discussions. Agreements on this platform do not automatically lead to ontology evolution, but knowledge engineers then have to change the ontology [DEMB⁺08]. One of the objectives of Cicero was also to enhance documentation by providing links with elements in an ontology and the discussions [DEMB⁺08].

- The On-To-Knowledge project²⁰ resulted in a generic method and tool suite [SSS03, SSSS01, SSS09]. The method was based on CommonKADS²¹. The goal of this project was to build an ontology-based tool environment to improve knowledge management dealing with large numbers of heterogeneous, distributed and semi-structured documents. The processes in the On-To-Knowledge method are: 1) feasibility study, 2) kickoff, 3) refinement, 4) evaluation and 5) application and evolutions, with cycles (iterations) over steps 3) to 5).

OntoEdit [SAS03] is the ontology engineering environment developed for On-To-Knowledge and is used for inspecting, browsing, implementing and modifying ontologies. Modeling ontologies using OntoEdit is done at modeling at a conceptual level. It is done so in a formalism-agnostic manner and with graphical user interfaces to represent views on conceptual structures. To facilitate collaboration

¹⁶<http://www.neon-project.org/>

¹⁷<http://www.neon-toolkit.org/>

¹⁸<http://www.neon-foundation.org/>

¹⁹<http://neon-toolkit.org/wiki/Cicero>

²⁰<http://www.ontoknowledge.org/>

²¹<http://www.commonkads.uva.nl/>

between domain experts and knowledge engineers, several plugins for OntoEdit were developed.

The OntoKick plugin was built to support the collaborative creation of the requirement specification document and the extraction of relevant structures building a semi-formal ontology description. OntoKick elicits information about requirements by formulating competency questions [UK95] that need to be answered by the appropriate stakeholders. Competency questions just state which queries the ontology should support.

The Mind2Onto plugin supports the integration of brainstorming sessions to build relevant structures of the semi-formal ontology description. This plugin is built around a commercial solution for the collaborative construction of mind maps and discussions. The output of this mind map is then imported into the ontology via XML.

After this project, some of the lessons learned are [SSS09]: 1) domain-experts need practical guidelines and 2) collaborative ontology engineering requires physical presence and advanced tool support.

- Al-Debei and Fitzgerald proposed a systematic design method for ontology engineering in information systems called OntoEng [ADF09]. OntoEng is inspired by design science research and is based on the lessons learnt from existing ontology development methods and from the experience of building an ontology in the domain of telecommunications. The method seems only to prescribe processes. To the best of our knowledge, no tool was developed for this method. It is implied that existing tools could be reused (e.g. for modeling UML as mentioned in the paper). Also the community aspects of ontology engineering were not proposed, and thus no descriptions for social processes for agreement. Guidelines for the social processes for elicitation stem from the work on the Unified Method described later in this section.
- Ontology 101 [NM01] includes simple guidelines based on iterative design to help stakeholders in creating ontologies using Protégé [NGM00]²² and Ontolingua²³ tools.

Interesting to note is that within the same group, a collaborative ontology engineering tool called collaborative Protégé was developed [TNTM08] as well as a light-weight Web-based version called Web Protégé [TVN08]. Tudorache *et al.* explicitly mention the requirement to store discussions for future reference, and thus support the ontology engineering processes. Both collaborative Protégé and Web Protégé, however, do not refer to a method that can be supported with this tool.

Recently, the group proposed to structure some of the negotiation processes for collaborative ontology engineering [ANT⁺11]. Those requests mainly concern the introduction of terms, concepts and relations. Natural language definitions are not negotiated as a separate request, but can be included in a term.

²²<http://protege.stanford.edu/>

²³<http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/index.html>

- The Unified Method proposed in [UG96] and [Usc96] is drawn from experiences in the developing the Enterprise Ontology method [UK95] and the TOVE (Toronto Virtual Enterprise) project ontology [GF95]. The method is language agnostic (i.e. not developed for a particular formalism) and has no specific tool support. An important aspect of is the use of a middle-out approach to produce the ontology (instead of a bottom-up or top-down). The approach furthermore gives a higher priority to highly connected concepts, since they are more difficult to define correctly and accurately.

The stages in this method are: 1) identify the purpose and the scope; 2) build the ontology, which consists of capturing the knowledge, implementing the ontology and integrating existing ontologies; 3) evaluate the result; and 4) document the ontology.

An important social process is brainstorming. The brainstorm sessions are used for identifying all relevant terms and facts. If the group does not have sufficient domain knowledge, external resources can be consulted to fill gaps. While terms are grouped in different categories, the decisions why certain terms belong to a certain group are noted. The method furthermore gave guidelines that support meaning agreement processes.

While knowledge is captured, the participants are asked to produce natural language definitions. This is, in fact, one of the guidelines for reaching meaning agreements. If necessary, examples are recorded as well. Those descriptions have to be as informal as possible to be understandable by all. Equivalent, more technical descriptions may be kept next to those.

- UPON [DNMN05, DNMN09] or the United Process for ONtologies is an incremental and iterative approach to building ontologies using use cases based on UML. It is based on the Unified Software Development Process or Unified Process (UP), a popular iterative and incremental software development process framework. This process divides the project into four phases: inception, elaboration, construction and transition. The elaboration, construction and transition phases are divided into a series of time-boxed iterations. The inception phase may be divided into iterations if the project is deemed large.

In UPON, domain experts mainly focus on the construction of the lexicon and glossary in the earlier phases of the project, whereas the knowledge engineering play a more important role in the development of semantic networks and implementations of the ontology (e.g. in OWL) at later stages. UPON thus uses two special linguistic resources: a lexicon that is merely a set of terms and a glossary, which is a set of term-definition pairs. Definitions are given in natural language. UPON furthermore defines a reference lexicon and a reference glossary, both of which are subsets of the aforementioned linguistic resources validated and approved by the community. UPON thus implicitly defined agreement processes.

The social processes in this method have not been explicitly mentioned except for interview and brainstorming sessions with domain experts and end users to elicit terms and requirements. However, the authors did not mention how these processes should be supported by the method (or a tool). Specific tool support was

not mentioned in [DNMN09], but as the method is based on the unified process and UML, existing tool support can be adopted. The authors did mention an existing ontology management platform – called ATHOS²⁴ – developed by the same group in which they will consider which parts to integrate for supporting the UPON method. Any report on such an integration, however, was not found.

It claimed to be the only iterative and incremental ontology building method at that time. This, however, is not true. Methods such as DOGMA-MESS (and later on Business Semantics Management) already took into account the incremental building of ontologies.

2.2.2 A Comparison

Table 2.4 provides a comparison of the aforementioned methods (and tools). The rows in this table represent the aspects that have been compared. Each aspect will now be discussed in the following paragraphs. In this Table, one can see that Ontology 101 has been considered twice. The first column refers to the method developed for creating ontologies with the Protégé tool. The latter – even though no method is mentioned – refers to the use of Collaborative Protégé, which has been developed by the same group.

Table 2.4: Comparison of different ontology engineering methods.

	CYC	Business Semantics Management	DILIGENT	DOGMA(-MESS)	HCOME	Holsapple et al. 2002	Karapiperis et al. 2006	METHONTOLOGY	NeOn	On-To-Knowledge	OntoEng	Ontology 101 (pre-collab)	Ontology 101 (collab)	Unified Method	UPON
Explicitly intended for distributed & collaborative construction?	N	Y	Y	Y	Y	C	C	N	Y	C	C	N	Y	C	C
Natural language descriptions of concepts?	D	Y	N	Y	A	N	N	Y	Y	D	Y	D	Y	Y	Y
Special linguistic resource as software artifact?	N	Y	N	Y	W	N	N	N	N	N	N*	N	N	N	N*
Tool support?	Y	Y	A	Y	Y	N	O	Y	Y	Y	O	Y	Y	N	O
Tool support for dialogue?	N	Y	Y	N	Y	N	N	N	E	E	N	N	Y	N	N
Social processes for agreements on formal descriptions?	N	I	I	Y	I	Y	Y	N	I	I	Y	N	Y	Y	Y
Tool support for social processes on formal descriptions?	N	I	I	P	I	-	-	N	I	I	-	N	Y	-	-
Social processes for agreements on informal descriptions?	N	I	N	N	I	N	N	N	I	N	Y	N	P	N	N
Tool support for social processes on informal descriptions?	N	I	N	N	I	-	-	N	I	N	-	N	I	-	-
Agreement leads to ontology evolution?	-	N	M	M	M	M	M	-	M	M	M	-	Y	M	M
"Owner" of the ontology?	-	C	H	H	H	K	K	-	K	K	N	-	K	K	K

²⁴<http://leks-pub.iasi.cnr.it/Athos>

Explicitly intended for distributed and collaborative construction? The values for this aspect are 'Y', 'C', and 'N'. The 'Y' stands for an "yes", the 'N' for "no" or "not proposed" and finally 'C' is used that collaborative aspects are touched upon, but not explicitly mentioned. A 'C' is put when some tasks or processes are described which imply group activities, e.g. brainstorming sessions to elicit knowledge.

METHONTOLOGY and Ontology 101 (pre-collab) all prescribed tasks, but provided no information on how to collaborate. CYC allowed multiple users to enter information through the tools, but collaboration processes were not described nor supported.

Natural language descriptions of concepts? The possible values for this value are: 'Y' for "yes", 'N' for "no", 'D' for support for documentation in which natural language definitions can be provided and 'A' for the adoption of existing resources to align with concepts of the ontologies.

It seems from the papers of HCOME that users are not able to provide their own natural language definitions. The authors did mention that concepts in the ontology can be aligned with natural language definitions from WordNet. CYC, On-To-Knowledge and Ontology 101 (pre-collab) allow for documentation. In CYC, they explicitly mention that concepts would be appropriately documented, providing definitions in natural language in comments. For the latter two, the tools showed the possibility to enter documentation, which can of course contain such a definition.

Special linguistic resource as software artifact? This aspect investigates whether there is a special linguistic resource next to the formal descriptions of the ontology. This aspect indicates a 'N' when the method stores the natural language definitions as documentation or as annotations in the ontology (e.g. `rdfs:comment`). 'Y' is used when the natural language definitions are stored in a separate artifact (in other words, materialized and treated as first-class citizen). 'W' stands for the use of existing artifacts. Both OntoEng and UPON do explicitly refer to a glossary in their papers, but it is not stored as a software artifact.

The verbalization of the formal description of a concept to generate a natural language definition of that same concept was not considered as an aspect in this table. The methods that do take the verbalization of formal descriptions into natural language are: DOGMA-MESS, Business Semantics Management and HCOME. DOGMA-MESS and Business Semantics Management have already a part of formal descriptions verbalized thanks to their grounding in fact-oriented modeling based on NIAM/ORM.

Ontologies should be considered as evolving entities. Argumentation and negotiation processes to discuss the evolution of ontologies are thus critical. A negotiation process is defined as a specification conversation about a concept (e.g. a process model) between selected domain experts from the stakeholders (community of organizations) [DM07]. In order to substantiate their perspectives, domain experts must formulate arguments. The following aspects look to what extent the methods have support for specific social processes and whether their tools provide support for those.

Tool support? A fairly straightforward aspect. The values used here are:

Y	Specific tool support for this method
N	No specific tool support for this method proposed
A	Adopting/extending existing ontology engineering tools for the method
O	Refers to other existing (type of) tools for some tasks

Tool support for dialogue? The values for these aspects are:

Y	Yes
N	No or not proposed
E	Via external (integrated) service or tool

The most accepted argumentation model is IBIS²⁵ [KR70], which provides a simple and abstract infrastructure for non-trivial problems that cost a lot to solve (in terms of time, money, etc.). IBIS was the model that DILIGENT, HCOME, and NeOn adopted. DILIGENT processes are still under control of knowledge engineers (the completion of some activities is partially dependent on the decisions of a board of experts), whereas HCOME aims at empowering users to be completely autonomous in their actions and decisions. The first is also true for NeOn. The NeOn toolkit provides - via a plugin - support for discussing issues (based on DILIGENT). On-To-Knowledge proposed a plugin that is based on existing commercial software for the brainstorming and elicitation of competency questions. Both NeOn and On-To-Knowledge are therefore regarded as using external (integrated) services and tools.

Social processes for agreements on formal descriptions (A)? This aspect investigates to what extent methods explicitly describe or prescribe special social processes for agreeing on formal descriptions of concepts. The values are:

Y	Yes
N	No or not proposed
I	“Implied” by the (tool) support for dialogue

Methods with a tool supporting dialogue rely on the dialogue support to support the social processes. DOGMA-MESS proposed a meaning evolution system. The two methods described in [HJ02] and [KA06] defined processes for achieving consensus and both OntoEng and the Unified Method mentioned the use of - amongst others - brainstorming sessions to elicit knowledge for the formal descriptions. On-To-Knowledge does not provide such support as the dialogue framework is only used for the elicitation of competency questions. In other words, dialogue is used to agree what questions should be supported by the ontology, but not how the ontology would look like.

Tool support for social processes on formal descriptions (B)? This aspect compares to what extent the method provides tool support for some of the processes described for (A). The values are:

²⁵IBIS stands for Issue-Based Information System.

Y	Yes
N	No or not proposed
P	Partial
I	“Implied” by the (tool) support for dialogue
-	Not applicable as there is no tool support or the authors referred to other existing (type of) tools

DILIGENT, HCOME and NeOn have taken argumentation frameworks into account, which are reflected in the tool support and therefore these processes can be considered implied. Web Protégé offers support for dialogue in a forum-like manner, and recently provided special requests for formal changes that are fairly frequent in the medical domain [ANT⁺11]. Also the Business Semantics Glossary for Business Semantics Management supports dialogue via their wiki technology. DOGMA-MESS is considered to provide specific tool support for their Meaning-Evolution-Support-System module. In DOGMA-MESS, so-called “tickets” are sent around to the stakeholders for rendering their perspectives [DD08]. The stakeholders receive the assignment to provide their perspective, which are then stored on the server. The meaning negotiation processes for evolving the ontology, however, were only described and not implemented in a tool. Only some support for analyzing conflicts between different perspectives was proposed for someone leading the MESS process, usually a knowledge engineer or core domain expert [DD08].

Social processes for agreements on informal descriptions (C)? Here, we investigate to what extent methods explicitly describe or prescribe special social processes for agreeing on informal descriptions of concepts. Informal here means that concepts are described by means of natural language descriptions rather than a formalism. The values are:

Y	Yes
N	No or not proposed
P	Partial
I	“Implied” by the (tool) support for dialogue

Again here, the methods with a tool supporting dialogue rely on the dialogue system to support the social processes. Only OntoEng proposed social processes for the construction of natural language definitions of concepts, albeit as keywords such as “brainstorming”. Unfortunately, OntoEng does not propose tool support for these processes.

This aspect for Ontology 101 (collab) is considered partial as the authors proposed special interactions for formal changes, but not for informal descriptions. They did, however, provide a request to introduce terms in which they foresaw a field for a natural language definition.

Tool support for social processes on informal descriptions (D)? Here, we compare to what extent the method provides tool support for some of the processes described for (C). The values are:

Y	Yes
N	No or not proposed
I	“Implied” by the (tool) support for dialogue
-	Not applicable as there is no tool support or the authors referred to other existing (type of) tools

Agreement leads to ontology evolution? This aspect looks to what extent agreements lead to ontology evolution. Ideally, agreements should automatically lead to ontology evolution. The values for this aspect are:

A	Automatically
M	Manually
N	Not
-	Not applicable as method is not explicitly intended for distributed collaboration

It is interesting to see that most methods that take the collaborative aspects of ontology engineering into account manually evolve the ontology after agreement has reached. Only Ontology 101 (collab) proposed some special requests which – after agreement – automatically evolves the ontology [ANT⁺11]. The types of requests are made after the types of changes most occurring in the medical domain. The requests are stored as annotations in the ontology.

The only method not really taking this aspect into account is Business Semantics Management. The Business Semantics Glossary allows anyone with sufficient rights to add new knowledge, without discussion. The formal parts of the ontology, however, do have a status attribute stating whether some part is a candidate, accepted, etc. The informal parts of the ontology do not have such properties. The Business Semantics Glossary actually follows a wiki paradigm where anyone can change the ontology, and discussions can happen afterwards. This approach has several problems:

- The ontology in the Business Semantics Glossary is not guaranteed to be stable at any time. The authors actually state that a stable version of the ontology has to be then “compiled” for use for the specific interoperability requirements (e.g. into UML, XSD, etc.)
- Community members could already commit to the knowledge they entered, even if that knowledge has not been accepted by all yet. This would hamper the possibility of finding compromises.

The reason most methods require the manual evolution of ontology is that either someone elicits knowledge and agreements without tool support and then enters the results or the argumentation frameworks allow users to discuss issues, solutions, etc. rather than discuss changes. If the latter would have been adopted, motivating and discussing the change, then ontology evolution could be automated.

“Owner” of the ontology? This aspect compares who the “owners” of an ontology are for a particular method. Here, the word “owner” refers to the users who can change the ontology. The values for this aspect are:

C	The community of stakeholders (possibly including knowledge engineers) are the owner
H	Stakeholders are the owner of their ontology, knowledge engineering ensure the evolution of the shared space
K	Knowledge engineers have ownership
N	Not proposed
-	Not applicable as method is not explicitly intended for distributed collaboration

It is interesting to see that Business Semantics Management is the only method that allows a community to develop and maintain their ontologies. In most methods, the knowledge engineers are the owners of the ontology.

DOGMA-MESS, HCOME and DILIGENT all allow individual stakeholders to maintain their view on matters, but the shared perspective is managed by the knowledge engineers.. DOGMA-MESS has the notion of organizational ontologies, HCOME refers to it as personal spaces and DILIGENT as local ontologies. A board of stakeholders with sufficient rights will then try to find a consensus or compromise from the different perspectives to evolve the ontology. The stakeholders remain thus owner of their ontology.

The problem with this method is that even though a consensus is sought, people describe their perspective on matters in a formal way and could thus already commit to their own descriptions. Not only that, they could as well already annotate their existing systems with their predicates. Rather than discussing changes in the ontology, changes are performed locally and then discussed upon. And one would indeed benefit from keeping as much as possible their desired changes. This could thus hamper or delay reaching a consensus as it is possible that stakeholders need to revert and commit to the new version of the ontology as decided upon by the board (with the involvement of all stakeholders, of course).

OntoEng did not explicitly state who the owners of an ontology are. The owner of the ontology in Ontology 101 (collab) is presumed to be the knowledge engineer as the authors did not explicitly refer to a method in their papers describing Collaborative Protégé, but a case study in the medical domain hinted at the use of knowledge engineers [NTDCM08].

We can conclude from above that the methods and tools described in the state-of-the-art do not take into account the social processes for constructing natural language definitions of concepts. Most of the methods that provide support for social interactions rely on this without specifying any specific processes or considering the natural language definitions as an equal import artifact next to the ontology. Definitions are often seen as annotations to the ontology (e.g. comments).

Collaborative Protégé is not a method, but a tool. This tool was taken into consideration as Ontology 101 (collab) as it was developed by the same group that proposed Ontology 101 and developed Protégé. It is interesting to note that in this comparison table, only the authors of Collaborative Protégé propose the formalization of specific requests to evolve the ontology. This is an important step towards agreement evolving ontologies as changes are discussed, and not issues.

Also apparent is that most methods rely on knowledge engineers to be the owners of the ontology (either immediately, or via a setting in which they own the shared part). Ontologies, however, should belong to the community and the role of knowledge engineers should be reduced to a minimum or even removed. As Heylighen noted in [Hey13]: “If the process were directed by a single individual (say, the group leader), who imposes a consensus view on the others, then that perspective would not be more powerful than the perspective of the leading individual. In other words, the collective would not be in any way more intelligent than its leader.”

2.3 Conclusions

This chapter provided 1) a definition of ontology in computer science, 2) a survey of ontology languages on the Semantic Web and 3) the state-of-the-art and comparison on method for ontology construction.

In this thesis, the problem tackled is not so much what ontologies are, but how they become social artifacts enabling interoperability between stakeholders representing their autonomously developed and maintained information systems. Those social artifacts are the results of social agreement processes, in other words ontologies evolve together with the communities “owning” the ontologies. From the state-of-the-art, however, one can see that the community aspects of ontology engineering are often neglected.

An important driver for the agreements are the natural language definitions used by the community for aligning their thoughts. In the state-of-the-art, however, these natural language definitions are often considered a second-class citizen: used as documentation in the ontology and not treated as a special artifact that co-evolves with the community and the formal definitions in the ontology. Not all methods (and their tools) cover all aspects. Lacking is thus a method that takes into account: 1) the social interactions between the community, 2) promoting the natural language definitions to first-class citizen and use those as a driver for the engineering processes, 3) empower the community in having their agreements automatically leading to ontology evolution.

Chapter 3

Hybrid Ontology Framework

3.1 Introduction

Ontologies are keystone technologies used for meaningful and efficient interoperation of information systems as such systems on the Web are in general developed and maintained *autonomously*. This calls for agreements between the stakeholders on the semantics of the shared concepts involved. Those agreements are captured and subsequently stored in an ontology. The process of reaching an agreement is defined as “*a social process, a dialogue between multiple human stakeholders in a community containing: (i) a subject on which needs to be agreed upon, (ii) a series of “utterances” from the stakeholders to each other and (iii) a conclusion*”. As a consequence, ontologies will *evolve* while such agreements are developed and finally put in place.

These ontologies are approximations of a real world [Fen01]; in fact, to the Web services involved, ontologies *are* the world. Ontologies represent an *externalization* of the semantics outside of the information system. The basic techniques and architecture for semantic interoperation is based on annotation (of an application system) and reasoning (about the concepts involved, in terms of the ontology).

From above it follows that modeling ontologies within a community of stakeholders is a critical activity for the eventual success of semantic interoperability. Fundamental to the approach that will be presented is the involvement of structured natural language as a vehicle to elicit useful and relevant concepts from community communication, and the mapping of these social processes to evolutionary processes in the emerging ontology. The formalism and language presented here are therefore “upstream” from the usual ontology languages such as RDF(S) and OWL and should not be confused with those; in fact it is relatively straightforward to compile the resulting/emerging ontologies into, for example, RDF(S) and OWL at any time. The process of “implementing” the hybrid ontology in one of those ontology languages will be dubbed the “downstream usage” of the ontology.

First, we give a more precise definition of an ontology as we need it in the sequel: “*The formal semantics of a (computer-based) system quite simply is the correspondence between this system and some real world as perceived by humans. It is usually given by a formal mapping of the symbols in the system’s description to objects in that real world, such that relationships and logical statements in the specification language can be assigned a truth-value depending on whether a certain state of affairs among objects exists in the real world. As the real world is not usually directly accessible inside a computer, storing and reasoning about semantics requires the world to be replaced by an agreed specification of a conceptualization, often in the shape of a formal (mathematical) construct. This*

computer-based, shared, agreed formal specification of a conceptualization is what is known as an ontology.”

One fundamental principle of all large system design is the so-called separation of concerns resulting in architectures that delegate respective functionalities to the stakeholders responsible for them. Examples are modules and databases. Modules are provided by the (generic) architecture of information systems driven by a database and largely separate the concern of basic data management from that of application development, the famous paradigm of data independence.

The approach we will use in this thesis reapplyes this principle by the rigorous separation in conceptualizations of “fact modeling” from all enterprise-specific interpretations. It is this interpretation process (formally, of statements shared in the application system in terms of ontology concepts) that is usually called “reasoning” in the Semantic Web literature. However, there is little or no attention to such separation of concerns in the usual reasoning formalisms of Semantic Web in terms of Description Logics and its syntactical manifestations such as OWL and its dialects. In this approach, this interpretation is exclusively delegated to the mapping between an application system and the “lexon base” of the ontology. The lexon base is a possibly vast set of plausible binary fact types to be interpreted in a certain context and will be described in more detail in subsequent sections. These mappings are called *ontological commitments*¹ after [Gua98], but these shall be reified in a well-defined manner suited to the formalism. Intuitively, commitments select the fact types needed, map application symbols to ontology concepts, and contain the rules and constraints – expressed in ontology terms – under which application symbols, relationships and business rules must be interpreted when they are to be shared with other autonomous systems. Those systems will share the concepts, but of course will have their own symbols, business rules, etc.

This separation of concerns allows a natural introduction of formalized social processes in goal-oriented communities such as exist in enterprises, professional networks, standardization groups, etc. In fact, this is true in any “human agent” context for which agreement about fact types is more efficient than reasoning from axioms. Note that nearly all data models for databases and business information systems were arrived at in this manner for the last 50 or so years.

In [Mee01b] and [Mee99a] a formalism and method for ontology development called DOGMA² was defined that illustrated and implemented these principles, now lifted to domain level from the mere enterprise system level. As indicated above, such descriptions must be seen as different from their eventual implementations, e.g. using RDF(S) and/or OWL. In the method and life cycle of semantic systems, the creation of DOGMA ontology descriptions belongs upstream from such implementation - although of course in many cases one will have to “mine” or elicit the required knowledge from existing information systems and their enterprise environments.

The next section will define what DOGMA Ontology Descriptions are before extending it for a hybrid ontology engineering framework.

¹Often referred to as commitments in this thesis.

²DOGMA originally stood for “Developing Ontology-Guided Mediation by Agents” [Mee01b] and stood later for “Developing Ontology-Grounded Methods and Applications”.

3.2 The DOGMA Ontology Framework

Definition 2 (DOGMA Ontology Descriptions)

A DOGMA Ontology Description Ω is an ordered triple $\langle \Lambda, ci, K \rangle$ where Λ is a lexon base, i.e. a finite set of lexons. A lexon is an ordered 5-tuple $\langle \gamma, t_1, r_1, r_2, t_2 \rangle$ where $\gamma \in \Gamma$ is a context identifier, $t_1, t_2 \in T$ are terms (or term labels), and $r_1, r_2 \in R$ are role labels. A lexon is a binary fact type that can be read in two directions: t_1 playing the role of r_1 on t_2 and t_2 playing the role of r_2 on t_1 . Here, the usual alphabets for constructing the elements of $T \cup R$ are omitted for simplicity. $ci : \Gamma \times T \rightarrow C$ is a function mapping pairs of context identifiers and terms to unique elements of C , called *concepts*. K is a finite set of ontological commitments. Each commitment is an ordered triple $\langle \sigma, \alpha, c \rangle$ where $\sigma \subset \Lambda$ is a selection of lexons from the DOGMA ontology description, $\alpha : \Sigma \rightarrow T$ is a mapping called an annotation from the set Σ of application (information, system, database) symbols to terms occurring in that selection, and c is a predicate over $T \cup R$ of that same selection expressed in a suitable first-order language.

Note that concepts are not described further than elements of a set. This is deliberate as we posit that concepts can only and exclusively be “known” through the interpretation of their properties and behavior as observed in communities. Context-identifiers are pointers to the origin of a lexon, and helps with the disambiguation of term- and role labels. Within a context $\gamma \in \Gamma$ and $t \in T$, $ci(\gamma, t)$ is the definition itself of the concept agreed by all users. To emphasize this explicit agreement, we shall avoid labeling concepts as such in our formalism, and assuming they are “computed” by the community from the term labels. Indeed, in this formalism, a *constructivist* [FHL⁺96] approach is adopted [DL09]. A constructivist is *“somebody who believes that “reality” exists independently of any observer, but who is aware that we only have access to our own (mental) “conceptions”; for the constructivist, the relationship between reality and conception is principally subjective, and may be subject to negotiation between observers; any agreement may have to be adapted from time to time. [FHL⁺96]”*

Below are some examples of lexons:

- $\langle Cultural\ Domain\ Expert\ 1, Artist, with, of, Name \rangle$
- $\langle Cultural\ Domain\ Expert\ 1, Artist, with, of, First\ Name \rangle$
- $\langle VCard, VCard, with, of, Email\ Address \rangle$
- $\langle Cultural\ Domain\ Expert\ 2, Artist, with, of, Age \rangle$
- $\langle Cultural\ Domain\ Expert\ 1, Artist, born\ on, of\ birth\ of, Date \rangle$
- $\langle Offer\ #1\ of\ Organization\ A, Offer, with, of, Title \rangle$
- $\langle Offer\ #1\ of\ Organization\ A, Offer, valid, for, Date \rangle$
- $\langle RFP\ Documentation, RFP, with, matches, Offer \rangle$
- $\langle FOAF, Agent, with, of, Name \rangle$
- $\langle Cultural\ Domain\ Expert\ 3, Artist, contributing\ to, with\ contribution\ of, Sculpture \rangle$

- ...

These lexons are then used to construct commitments. The function ci maps terms in those lexons to concepts as they are understood and agreed upon in those contexts. It is possible that terms in different contexts refer to the same concept, e.g. $ci(Offer \#1 \text{ of } Organization A, Offer)$ and $ci(RFP Documentation, Offer)$. Stating that two context-term pairs refer to the same concept is also delegated to the commitment layer, which will be discussed later on.

The DOGMA Framework was previously defined as to contain two layers: the Lexon Base Λ and the commitment layer containing the commitments K . Later on in this thesis, however, a motivation for the introduction of an additional layer in order to do proper business between two or more autonomous information systems will be given. What is lacking is a layer in which the agreements and engagement of the community of stakeholders is captured to which everyone needs to comply with for proper semantic interoperation to be possible. In a way, these are similar to the commitments described in the definition above, but without the mapping of application symbols and only involving lexons belonging to a domain (and not to an application). From here onward, the distinction between *application commitments* and *community commitments* is made. The first will refer to the latter for to annotate existing applications, also providing additional enterprise-specific knowledge and the mappings. The community commitments will be motivated and described in later sections, but first the application commitments will be described in more detail.

3.2.1 Application commitments

This section describes the characteristics of the commitments K of a DOGMA Ontology Description. Such commitments describe how one individual application commits to a selection of lexons as well a description on how these applications uses these lexons (by means of constraints) and how its application symbols map onto the terms and roles inside that selection. Figure 3.1 depicts an example commitment. For more details on the syntax of commitments, the reader is referred to [VDBM04, TTM07]. This thesis will introduce a dialect of Ω -RIDL.

3.2.1.1 Constraints in an Application Commitment

The constraints in an application commitment are largely based on ORM [HM08] constraints. Object Role Modeling (ORM) [HM08] is the successor of NIAM³ [Win90] and is intended to model conceptual schemas for closed information systems. ORM has two basic constructs: objects types and relations called fact types. Object types correspond with the subjects or objects in sentences and the verbs these object types play are the roles (predicates) of the fact type. An object thus plays a role in relations with other objects. ORM also provides a graphical notation and a conceptual modeling method called

³Nijssen's Information Analysis Methodology, and later as Natural language Information Analysis Methodology and Binary Relationship Modeling

```

BEGIN SELECTION
<'Offer #1 of Organization A', Offer, with, of, Title>
<'Offer #1 of Organization A', Offer, contains, contained_in, Product>
<'Offer #1 of Organization A', Offer, made_by, makes, Vendor>
<'Offer #1 of Organization A', Vendor, located_on, location_of, Address>
...
END SELECTION
BEGIN CONSTRAINTS
EACH Offer contains at least 1 Product.
EACH Vendor located_on exactly 1 unique Address.
...
END CONSTRAINTS
BEGIN MAPPINGS
MAP 'APP_OFF'.'Title' ON Title of Offer.
MAP 'APP_VEN'.'ADDR' ON Address location_of Vendor.
...
END MAPPINGS

```

Figure 3.1: Example of a commitment for a particular application showing pieces of the three parts: selection σ , constraints c and annotations (or application symbol mappings) α .

the Conceptual Schema Design Procedure (CSDP). In DOGMA, the terms correspond to the object types and the role labels with the roles.

ORM is used to create conceptualizations using a graphical notation and a formalism that is grounded in first order logic and set theory. The graphical notation is used to create diagrams that capture elementary fact types.

ORM is attribute-free, which means it treats all elementary fact types as relationships and in this manner treats decisions for grouping facts into structures as implementation concerns irrelevant to meaning. By avoiding attributes in the base model, ORM improves semantic stability. Semantic stability is a measure of how well models or queries expressed in the language retain their original intent in the face of changes to the business domain. The more changes one needs to make to a model or query to cope with a business change, the less stable it is [HM08].

In [SMJ02, DJM02], the authors put forward the advantages of ORM as a semantically rich modeling language, also mentioning the advantages of being able to depict the models as natural language sentences (verbalization) and as an intuitive diagram.

For this thesis, some ORM constraints have been adopted and other constraints have been introduced. From ORM, the constraints necessary for creating referable terms are taken into account. A term is referable when that term is either lexical (thus its instances can be printed on a screen) or if that term has a set of attributes that *uniquely* and *totally identify* instances of the concept referred to by this term. The terms referred to in those attributes (i.e. played by the co-role of the term) have to be – in turn – referable as well.

1. “Uniquely” means a role played at most once by every instance;

2. “Totally” means that the role is mandatory, this role has to be played by every instance (a mandatory constraint);
3. “Identifying” means that every combination of instances of concepts referred to by each term playing the co-role refers to only one instance.

Assume that a floor is uniquely and totally identified by its floor number, and the floor number is lexical. Then the constraints will look as follows:

```
EACH Floor with AT MOST 1 Floor Number. # Unique
EACH Floor with AT LEAST 1 Floor Number. # Total
EACH Floor IS IDENTIFIED BY (Floor Number of Floor). # Identifying
EACH Floor Number IS LEXICAL.
```

“Floor” thus has a unique *simple* reference. A unique simple reference is one unique, total and identifying attribute. A unique *composite* reference has more than one attribute. Given the description of “Floor” from above, assume that each hotel room is uniquely, and totally identified by its room number (which is lexical) and the floor. This would give the following result:

```
EACH Hotel Room with AT MOST 1 Room Number. # Unique
EACH Hotel Room with AT LEAST 1 Room Number. # Total
EACH Hotel Room with AT MOST 1 Floor. # Unique
EACH Hotel Room with AT LEAST 1 Floor. # Total
EACH Hotel Room IS IDENTIFIED BY (Floor of Hotel Room)
AND (Room Number of Hotel Room). # Identifying
EACH Room Number IS LEXICAL.
```

In the first example, the identifying constraint was put on one role and therefore the same constraint as the uniqueness constraint. Uniqueness constraints are typically denoted with an arrow or line above the role in ORM. In the second example, the identifying constraint is modeled using an external uniqueness constraint. Here external means across multiple lexons. In ORM, they are typically denoted with a circle containing the letter ‘U’ (for uniqueness) or a line, connected with dotted lines the roles involved. Mandatory constraints are depicted with a bullet on the role and the distinction between lexical and non-lexical terms is made by using a full line around the term label for the first, and a dashed line around the term label for the latter. An ORM diagram of this example is shown in Figure 3.2.

Sometimes the instances of a lexical term are limited to a certain set, finite or not. These constraints are called *value constraints* and can be part of the domain and therefore shared. A value constraint is described in terms of a value range, which can be an explicit enumeration of elements, ranges or even regular expressions. For example, if one wants to limit the occurrences of category type to ‘A’, ‘B’ and ‘C’, he states:

```
EACH Category Type IN ('A', 'B', 'C').
```

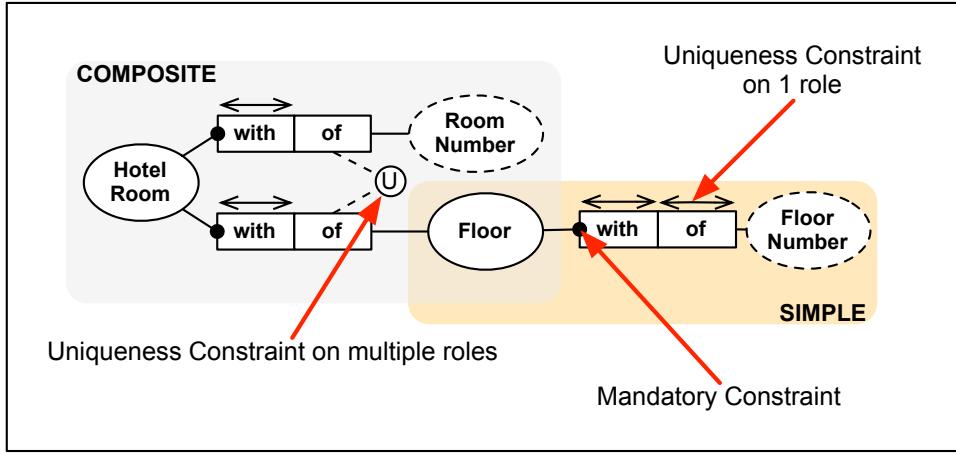


Figure 3.2: An example of a simple and a composite unique reference.

Another typical constraint is the subset constraint, which states that the population of a set of roles in fact types must be a subset of the population of another set of roles. Figure 3.3 depicts such a subset constraint using the ORM notation. This constraint states that the population of lexon F2 is a subset of the population of lexon F1. In other words, if an instance of A plays the role of Rn on an instance of B, that same instance of A must also play the role of R1 with the same instance of B.

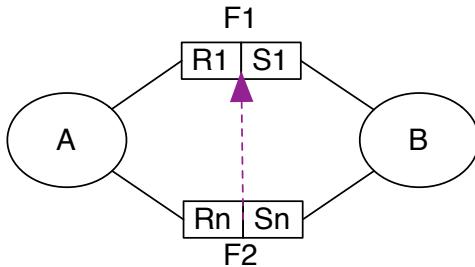


Figure 3.3: An example of a subset constraint.

3.2.1.2 Synonyms in Application Commitments

An application can commit to lexons coming from different contexts. Often, terms in those lexons from one context will be synonymous with terms in lexons from other contexts. It is even possible that different terms within one context can be synonymous. The commitment therefore needs to contain information on how different terms are linked with each other. Information on synonyms are part of the constraints as a predicate relating two term-context pairs. A more detailed description on the nature of synonymity will be given in Sections 3.3.4.

Note that the separation of concerns mentioned in the previous section is reflected here through the set of plausible lexons in the lexon base on one side, and the constraints, rules, etc. on a relevant selection of those lexons on the other. In fact there are no constraints or any other reasoning supports included in the lexon base, making for a so-called “light weight” ontology.

Also notice that a particular application is able to commit to a series of lexons originating from different contexts. When two or more autonomously developed information systems need to interoperate meaningfully, they will need to share at least the lexons. Nothing, however, prevents the people responsible for these application commitments to select other lexons. Moreover, it will sometimes be desirable or necessary to include additional information (from the organization owning the application, or other contexts) in order to enable interoperability.

Take for example the relational database depicted in Figure 3.4 capturing the relation of artists contributing to works of art (called “pieces”, in this particular application) in a table called `artistpiece`. This database, however, uses an artificial ID as a primary key for both instances of artists and works of art. The foreign keys used in the join table are evidently referring to these primary keys. These artificial IDs are not shared among other applications. In fact, those artificial IDs are not even part of the domain. Only rarely, one can find identifiers shared across applications (such as ISBN numbers for books), and even then applications often use an artificial primary key for various reasons, e.g. simplicity or increased performance. In order for this application to share information on who collaborated on what piece, the owners of this application will need to properly annotate the artificial IDs with their organization knowledge as well. This is reflected in the additional lexons originating from a different context, that of the organization owning the application, in Figure 3.5.

The example in Figure 3.5 shows how the constraints are used so that 1) the concepts of artists and works of art within the organization are synonymous with those of another context, and 2) the artificial IDs are used to uniquely, and totally identify instances of artists and works of art. The primary- and foreign keys are then annotated with this knowledge and an agent capable of interpreting Ω -RIDL can then construct queries necessary to join instances of artists with instances of works of art, and populating the lexon $\langle 'Cultural\ Domain\ Expert\ 1', Artist, contributed\ to, with\ contribution, Work\ Of\ Art \rangle$. Other applications are now able to retrieve information from this relation, without even the need to care about enterprise-specific lexons.

The application commitments allow organizations to disclose some of the organization’s knowledge, or at least those necessary for enabling interoperability. And as more organizations pick up lexons previously not shared by many, the more this lexon will become part of the shared understanding that constitutes the ontology. This is why one can include additional lexons and constraints, even when they are not (yet) shared.

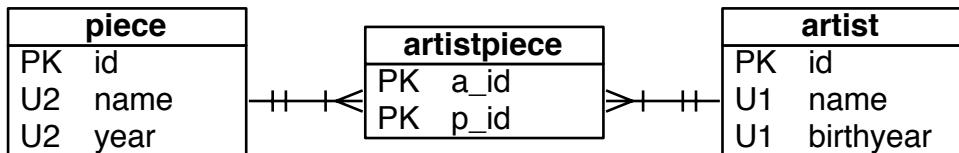


Figure 3.4: Small relational database modeled in (E)ER.

```

BEGIN SELECTION
# Selection of lexons shared by > 1 applications
<'Cultural Domain Expert 1',Artist,contributed to,with contributor,Work Of Art>
<'Cultural Domain Expert 1',Artist,having,of,Name>
<'Cultural Domain Expert 1',Artist,born in,of birth of,Year>
<'Cultural Domain Expert 2',Work Of Art,with,of>Title>
<'Cultural Domain Expert 1',Work Of Art,made in,of,Year>
# Enterprise-specific lexons
<'MyOrganization',Artist,with,of,AID>
<'MyOrganization',Work Of Art,with,of,WID>
END SELECTION
BEGIN CONSTRAINTS
LINK('Cultural Domain Expert 1',Artist,'MyOrganization',Artist).
LINK('Cultural Domain Expert 1',Work Of Art,'MyOrganization',Work Of Art).
LINK('Cultural Domain Expert 1',Work Of Art,
     'Cultural Domain Expert 2',Work Of Art).
# List enterprise-specific constraints
EACH Artist with AT MOST 1 AID.          #(1)
EACH Artist with AT LEAST 1 AID.          #(2)
EACH AID of AT MOST 1 Artist.            #(3)
EACH Work Of Art with AT MOST 1 WID.    #(4)
EACH Work Of Art with AT LEAST 1 WID.   #(5)
EACH WID of AT MOST 1 Work Of Art.      #(6)
END CONSTRAINTS
BEGIN MAPPINGS
MAP 'Artist'.'name' ON Name of Artist.
MAP 'Artist'.'birthyear' ON Year of birth of Artist.
MAP 'Artist'.'id' ON AID of Artist.
MAP 'piece'.'name' ON Title of Work Of Art.
MAP 'piece'.'year' ON Year of Work Of Art.
MAP 'piece'.'id' ON WID of Work Of Art.
MAP 'artistpiece'.'a_id' ON AID of Artist contributed to Work Of Art.
MAP 'artistpiece'.'p_id' ON WID of Work Of Art with contributor Artist.
END MAPPINGS

```

Figure 3.5: Example Ω -RIDL commitment for describing the database shown in Figure 3.4.

3.3 Towards a Hybrid Ontology Framework

This section extends the DOGMA Ontology Framework to support hybrid ontology engineering. In hybrid ontologies, all concepts, terms, etc. are represented not just on their own formal structures (e.g. by means of fact-orientation), but are *always* to be interpreted in a given context, which is the *community* that agrees on those formal structures. Agreements are made possible and are supported by *glosses* in natural language of which the shared understanding is implicit. In hybrid ontology engineering, communities consequently are promoted to “first-class citizens” by formalizing the social interactions that evolve the hybrid ontologies and by declaring the community as the context in which all processes take place.

In this thesis, the act of providing an informal, natural language definition is called *articulating* and the result thereof an *articulation*⁴. Articulation is defined as “*the result of articulating terms or lexons. Articulating means to explain; to put into words; to make something specific*”.

The previous section already explained that application commitments were not enough to guarantee proper interoperation between autonomous information systems belonging to a community of stakeholders. Indeed, the engagement of the community members to commit to certain lexons will be necessary. Those agreements will need to be captured in a *community commitment*, which will be discussed in depth in section 3.4.

Before introducing the community commitment, however, an artifact that will support and even drive these agreements, called a *glossary*, will need to be introduced. This artifact will capture the informal natural language descriptions of concepts. After describing the glossary and the community commitment, the social processes involved to make the community commitment evolve will be discussed. This will constitute the framework for hybrid ontology engineering, but not yet the method. The method will be described in the following chapter.

3.3.1 The Glossary and Concept Identifiers

Note that previous definitions imparts a well-defined hybrid aspect on ontologies as they are to be resources shared among humans working in a community as well as among networked systems such as exist in the World Wide Web. As the “unique concept” property mentioned above informally and intuitively results from a community agreement, it is useful to formalize a community precisely as such a context, and to name the resulting notion a hybrid ontology [MD10]. A special linguistic resource, called a *glossary*, is introduced. This glossary will record and support all the social processes.

Definition 3 (Hybrid Ontology Description)

A Hybrid Ontology Description is an ordered pair $H\Omega = \langle \Omega, G \rangle$ where Ω is a DOGMA ontology description where the contexts in Γ are labeled communities and G is a

⁴Note that articulation is not to be confused double-articulation introduced in the previous chapter.

glossary. A glossary G is an ordered triple $\langle g_1, g_2, EQ_G \rangle$, where

- g_1 is a function of the form $g_1 : \Gamma \times T \rightarrow \text{Gloss}$, the Term Glossary;
- g_2 is a function of the form $g_2 : \Lambda \rightarrow \text{Gloss}$, the Lexon Glossary;
- Gloss is a set of human-interpretable objects.
- EQ_G is a finite set of pairs $\text{Gloss} \times \text{Gloss}$ containing the agreement that two glosses refer to the same concept.

Context identifiers are thus pointers to a community. They can be a name, a URI to a website or even a URI to a document describing the community. To improve readability, names as context identifiers are used throughout this document. For example, given the DOGMA ontology description Ω from the previous example, the lexons will thus point to communities instead of their source of origin. Those lexons could thus look as follows:

- $\langle \text{Cultural Domain Community}, \text{Artist}, \text{with, of, Name} \rangle$
- $\langle \text{Cultural Domain Community}, \text{Artist}, \text{with, of, First Name} \rangle$
- $\langle \text{Address Community}, \text{VCard}, \text{with, of, Email Address} \rangle$
- $\langle \text{Cultural Domain Community}, \text{Artist}, \text{with, of, Age} \rangle$
- $\langle \text{Cultural Domain Community}, \text{Artist}, \text{born on, of birth of, Date} \rangle$
- $\langle \text{Vendor Community}, \text{Offer}, \text{with, of, Title} \rangle$
- $\langle \text{Vendor Community}, \text{Offer}, \text{valid, for, Date} \rangle$
- $\langle \text{RFP Community}, \text{RFP}, \text{with, matches, Offer} \rangle$
- $\langle \text{Address Community}, \text{Agent}, \text{with, of, Name} \rangle$
- $\langle \text{Cultural Domain Community}, \text{Artist}, \text{contributing to, with contribution of, Sculpture} \rangle$
- ...

Note that the lexons from VCard and FOAF are now in the same community. It could be that lexons from different sources were agreed upon by the same community. With those lexons, a hybrid ontology description can be constructed where G contains (among others):

- ($\langle \text{Address Community}, \text{Email Address} \rangle$, “The address of an email, a system of world-wide electronic communication in which a user can compose a message at one terminal that can be regenerated at the recipient’s terminal when the recipient logs in”)
- ($\langle \text{Vendor Community}, \text{Offer} \rangle$, “Represents the public announcement by a vendor to provide a certain business function for a certain product or service to a specified target audience.”)
- ($\langle \text{Vendor Community}, \text{Offer}, \text{contains, contained in, Products} \rangle$, “Represents the relation of a product for sale being included in an offer.”)

For the sake of understanding the text, it will be sufficient to think of Gloss as a set of natural language descriptions each providing an “explanation” for a term in T or a lexon in Λ adequate within a given community.

Those definitions should be of proper quality, in order to be useful. Jarrar gave several guidelines on the construction of such glosses [Jar06]: a gloss should (i) start with the term of the principal or supertype of the concept being defined; (ii) be written in the form of propositions; (iii) focus on the distinguishing characteristics of the concept being defined; (iv) be supportive (examples are encouraged); (v) be consistent with the formal axioms in the ontology and (vi) be sufficient, clear and easy to understand by the members of the community.

Before continuing to describe the role of glosses in hybrid ontology engineering and how concepts interplay with the concepts referred to by the *ci* function, a couple of functions on relations on concepts needs to be defined.

3.3.2 Community Calculus for Concepts

Given a lexon $\langle\gamma, t_1, r_1, r_2, t_2\rangle \in \Lambda$, the function *ci* returns for the community γ and term t_1 the concept that t_1 evokes within γ . That every community-term pair should refer to at most one concept is obvious, the community would be divided if two different subsets of that community would have a different concept in mind for a particular label. Yet, communities can agree that their terms could refer to the same concept and those agreements are captured in their respective concepts. Those agreements are captured and part of the concepts.

Definition 4 (Concept Equality Agreements inside a Concept)

A function *cea* for retrieving the concept equality agreements part of the concept $ci(\gamma, t)$ referred to by one community-term pair $\langle\gamma, t\rangle \in \Gamma \times T$ is defined as $cea : C \rightarrow 2^C$ returning for a given concept $ci(\gamma, t)$ the set of concepts $\{ci(\gamma', t') | ci(\gamma, t) \equiv_C ci(\gamma', t') \wedge \langle\gamma', t'\rangle \in \Gamma \times T\}$ where \equiv_C is the agreement of both communities that those terms refer to the same concept. Note that if $\gamma = \gamma'$, it refers to the agreement within one community that two labels refer to the same concept.

For example, given the situation depicted in Figure 3.6, the *cea* for each community-term pair are:

- $cea(ci(\gamma_1, t1)) = \{ci(\gamma_1, t1), ci(\gamma_2, t2), ci(\gamma_3, t3), ci(\gamma_4, t4)\}$
- $cea(ci(\gamma_2, t2)) = \{ci(\gamma_1, t1), ci(\gamma_2, t2), ci(\gamma_3, t3)\}$
- $cea(ci(\gamma_3, t3)) = \{ci(\gamma_1, t1), ci(\gamma_2, t2), ci(\gamma_3, t3)\}$
- $cea(ci(\gamma_4, t4)) = \{ci(\gamma_1, t1), ci(\gamma_4, t4), ci(\gamma_5, t5)\}$
- $cea(ci(\gamma_5, t5)) = \{ci(\gamma_4, t4), ci(\gamma_5, t5)\}$

Given two community-term pairs $\langle\gamma, t\rangle, \langle\gamma', t'\rangle \in \Gamma \times T$ with $ci(\gamma, t) = ci_1$, $ci(\gamma', t') = ci_2$, $g_1(\gamma, t) = gloss_1$ and $g_1(\gamma', t') = gloss_2$. The following functions are defined:

- **Concept-intersection** $\cap_C : C \times C \rightarrow C$ is the concept constructed with the intersection of the concept equality agreements of two concepts referred to by two

community-term pairs with the ci function, returning a concept with a cea with all the shared agreements of concepts referred to by the given community-term pairs. With ci_1 and ci_2 the resulting concept c_3 for $c_1 \cap_C c_2$ is constructed such that $\forall c' \in cea(c_3) : c' \in cea(c_1) \cap cea(c_2)$. For example, given the situation depicted in Figure 3.6, the intersection of $ci(\gamma_1, t1) \cap_C ci(\gamma_4, t4)$ is a concept containing the agreement of communities γ_1 and γ_4 that their respective terms are referring to the same concept and $ci(\gamma_2, t2) \cap_C ci(\gamma_5, t5)$ is a concept containing no agreement.

- $cea(ci(\gamma_1, t1) \cap_C ci(\gamma_4, t4)) = \{ci(\gamma_1, t1), ci(\gamma_4, t4)\}$
- $cea(ci(\gamma_2, t2) \cap_C ci(\gamma_5, t5)) = \emptyset$

- **Concept-union** $\cup_C : C \times C \rightarrow C$ is the concept constructed with the union of the concept equality agreements of two concepts referred to by two community-term pairs with the ci function, returning a concept with a cea with all the shared agreements of concepts referred to by the given community-term pairs. With ci_1 and ci_2 the resulting concept c_3 for $c_1 \cup_C c_2$ is constructed such that $\forall c' \in cea(c_3) : c' \in cea(c_1) \cup cea(c_2)$. In Figure 3.6, the cea of the union of $ci(\gamma_1, t1) \cup_C ci(\gamma_5, t5)$ would result in $cea(ci(\gamma_1, t1) \cup_C ci(\gamma_5, t5)) = \{ci(\gamma_1, t1), ci(\gamma_2, t2), ci(\gamma_3, t3), ci(\gamma_4, t4), ci(\gamma_5, t5)\}$
- **Concept-equality** $=_C : C \times C \rightarrow \{True, False\}$ returns true if and only if the agreements in the intersection of both concepts are the same as the agreements in the two concepts. In Figure 3.6, $ci(\gamma_2, t2) =_C ci(\gamma_3, t3)$ since both concepts contain the same set of agreements. In the same example, we have that $ci(\gamma_1, t1) \neq_C ci(\gamma_2, t2)$, since the first operand also contains the agreement that communities γ_1 and γ_4 agreed that their respective terms $t1$ and $t4$ refer to the same concept and this is not appearing in the set of agreements in the second operand.
- **Concept-inclusion** given c_1 and c_2 , $\subseteq_C : C \times C \rightarrow \{True, False\}$ which returns true if and only if $cea(c_1) \subseteq cea(c_2)$. With this definition, the following property holds: $(c_1 \cap_C c_2) \subseteq_C c_1$.

3.3.3 The Role of Glosses in Hybrid Ontologies

Glossaries turn out to require a fairly rich structure when to be deployed for (hybrid) ontology engineering, as they are used to build agreements in communities about concepts. This section describes how glosses interplay with concepts and how the combination of both will facilitate the agreement processes. However, first some functions and relations on glosses are introduced. As an example, the following gloss $gloss_E$ for the term “Email Address” in some community γ_E is used.

“An email address unambiguously names the location of an email box to which email messages are delivered. An example format of an email address is `lewis@example.net` which is read as lewis at example dot net.”

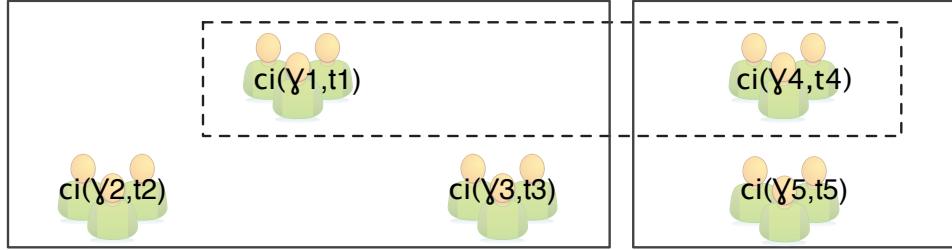


Figure 3.6: Example of interrelations between communities using the agreements on referring to the same concept. Communities γ_1 , γ_2 and γ_3 all agree that their term refers to the same concepts. The same holds for the terms of communities γ_4 and γ_5 . Communities γ_1 and γ_4 also agree that their terms t_1 and t_4 also refer to the same concept, something the other communities not necessarily agreed with.

- **Gloss-terms.** The function $terms : Gloss \rightarrow 2^S$ returns a set of terms used in the gloss using noun discovery techniques⁵. Using the Apache OpenNLP toolkit⁶, for example, the following nouns can be detected: $\{Location, Address, Email, Box, Example, .net\}$. Other solutions, such as the commercial API Alchemy⁷ would return $\{E-mail, E-mail address, Domain Name System, .net, .NET Framework\}$.

The occurrence of relevant terms inside a gloss that does not appear as a term in one of the communities' lexon provides a trigger for a new discussion. Indeed, if the community would deem the discovered term relevant, they will have to relate it with either the described term or another term from that community residing in a lexon.

- **Gloss-instances** $instances : Gloss \rightarrow 2^{Instance}$, where $Instance$ is the set of all instances in all glosses of a Hybrid Ontology Description. Instances are composed over an alphabet just like term- and role labels. Instances are to be mapped onto terms in the ontology existing in the ontology and maybe be used to reason over the constraints (e.g. constraints) as sample data. If the term on which an instance has to be mapped onto is missing from the ontology, the community can decide to enter a new lexon with that term. By doing so, the instance becomes part of the motivation of entering the new lexon. To detect instances in a gloss, one can use named entity extraction techniques or look for proper nouns by using a gazetteer. In the example above, "lewis" or "lewis@example.net" would have been identified as named entities, perhaps as instances of Person and Email Address respectively. One could assume that in a well-constructed gloss, would be properly indicated, e.g. by use of capitalization or quoted literals. However, such a restriction would be too restrictive to be practical.
- **Gloss-lexons.** The function $prelexons : Gloss \rightarrow 2^\Lambda$ returning unrefined "pre-lexons" in which one of the roles is not necessarily expressed. In fact, it can happen

⁵For which several off-the-shelf solutions exist. The quality of the output of the chosen solution, however, falls outside the scope of this dissertation.

⁶<http://incubator.apache.org/opennlp/>

⁷<http://www.alchemyapi.com/>

that both role and co-role are not appearing. Terms in those pre-lexons can actually refer to specific instances (being named entities). If both terms are referring to concepts, then a social process for entering this lexon - if not yet in the hybrid ontology - might be initiated. In the other case, the class of the instances has to be first determined before the lexon can be introduced to the ontology. This instance of a relation can then be used as sample data for reasoning later on. The pre-lexons that can be extracted from this gloss with the Alchemy API can be found in Table 3.1 and give the community an idea of what other lexons they can introduce to the ontology. From those pre-lexons, the first might be refined into the following lexon $\langle \text{Vendor Community}, \text{Email Address}, \text{names, name of, Email Box} \rangle$.

Table 3.1: Examples of pre-lexons retrieved from a gloss (using the Alchemy API), those pre-lexons give the community a hint of what other lexons they can introduce to the ontology.

Subject	Predicate	Object
An email address	Names	The location of an email box which is
An example format of an email address	Is	lewis@example.net which is read as lewis at example dot net
Which	Is read as	At example dot net

The identification of attributes of entities has always been a modeling issue in conceptual (database) modeling and because of the application independency of ontologies. For instance, assume the follow pre-lexons can be elicited from a gloss.

$$\langle \gamma', "chrdebru@vub.ac.be", \text{identifies, ., Email Inbox} \rangle$$

Here the γ' is the community in which the gloss was placed related to a term. The ‘.’ stands for the co-role that has not been expressed. The “chrdebru@vub.ac.be” is actually referring to an instance of an email address and the lexon will be refined as

$$\langle \gamma', \text{Email Address, identifies, identified by, Email Inbox} \rangle$$

Expressing both role and co-role proves has the advantage as the “attributeness” of a relation depends on the constraints on those roles. In this example, it is clear that the email address is an attribute of an email inbox. This is one of the benefits of using fact-oriented modeling, mentioned in the previous section.

- **Source of glosses.** The function $\text{source} : \text{Gloss} \rightarrow \mathcal{D}$ maps elements of Gloss to elements of \mathcal{D} , a set of pointers of Documents. Documents can include dictionaries, manuals, and even interviews with humans. Documents can reside both offline or on the Web. Those pointers are assumed to be identified unambiguously by some identifier.

This function allows us to group glosses according to their origin. The more glosses refer to the same source, the more it is assumed the source is trusted. The function

also acts as a means to detect gloss-equivalence. As mentioned earlier, communities will often describe the same thing with two glosses that are not completely identical (e.g. a different punctuation or use of cases). Glosses with a same origin and high lexical similarity are proposed to the communities as potentially equivalent glosses. Similarity is calculated using existing string metrics [CRF03].

It happens that different glosses for a term originating from the same source are reformulated or reinterpreted in such a way that heuristics (such as string metrics) return low similarity scores. The combination of a keeping sources and string metrics for detecting similarities cannot only be used to detect gloss-equivalences, but even to drive the communities in agreeing on one of several gloss-equivalent glosses that originate from the same source.

The proposed functions allow agreements on glosses and terms to drive the ontology engineering process, helping the community to start social processes to propose additional lexons in the ontology. As discussions are started, the community as a whole can also control which discussions fall out of the scope of the kind of information services the Hybrid Ontology Description has to support.

Onto the relation between concepts and glosses. It is natural to associate them with concepts (in a DOGMA ontology description through the terms of lexons). However, one needs to assure that the terms in a lexon are articulated before the lexon itself is articulated. This is called the *glossary coherence principle*.

Definition 5 (Glossary coherence principle)

Given a hybrid ontology description $H\Omega = \langle \Omega, G \rangle$, a glossary is said to be *coherent* when $\forall \lambda = \langle \gamma, t_1, r_1, r_2, t_2 \rangle \in \Lambda : \text{if } g_2(\lambda) \text{ is defined, then both } g_1(t_1) \text{ and } g_1(t_2) \text{ are defined.}$

Indeed, it would be not very useful to describe a relation between two terms if one or both terms playing the roles in that relation are not described themselves, implying that their intended meaning has not yet been made explicit.

3.3.4 Glossary-consistency and Gloss-equivalences

When two different terms are articulated with the exact same gloss, one would assume that the glosses and therefore also the described terms refer to the same concept. If this property holds, the hybrid ontology is said to be *glossary-consistent*.

Definition 6 (Glossary-consistency principle)

A hybrid ontology satisfies the *glossary-consistency principle* if for every two pairs $\langle \gamma_1, t_1 \rangle, \langle \gamma_2, t_2 \rangle \in \Gamma \times T$, if $g_1(\gamma_1, t_1) = g_1(\gamma_2, t_2)$ then $ci(\gamma_1, t_1) \equiv_C ci(\gamma_2, t_2)$. The converse does not necessarily hold.

In other words, if two terms in two communities are articulated with the exact same gloss, then those terms in those communities must refer to the same concept as well in order for the hybrid ontology to be glossary-consistent. For most purposes, this condition is too limiting since often glosses will express “the same thing” without being textually identical. It suffices that the communities agree on their equivalence; this leads to the following definition.

Definition 7 (Gloss-equivalence)

Given communities $\gamma_1, \gamma_2 \in \Gamma$ and terms $t_1, t_2 \in T$, the two term-glosses $g_1(\gamma_1, t_1)$ and $g_1(\gamma_2, t_2)$ are said to be *gloss-equivalent* EQ_G if the two communities agree that the described terms refer to the same concept.

There are two special cases of gloss-equivalence: one in which $\gamma_1 = \gamma_2$ and $t_1 \neq t_2$, and one in which $\gamma_1 \neq \gamma_2$ and $t_1 = t_2$. The first is called *community-equivalence* EQ_γ and the latter *term-equivalence* EQ_T .

Using gloss-equivalence, the glossary-consistency principle can be redefined as follows:

Definition 8 (Glossary-consistency principle, bis)

A hybrid ontology is said to satisfy the *glossary-consistency principle* if for every two pairs $\langle \gamma_1, t_1 \rangle, \langle \gamma_2, t_2 \rangle \in \Gamma \times T$, if $EQ_G(g_1(\gamma_1, t_1), g_1(\gamma_2, t_2))$ then $ci(\gamma_1, t_1) \equiv_C ci(\gamma_2, t_2)$. The converse does not necessarily hold.

Note that when two communities $\gamma_1, \gamma_2 \in \Gamma$ agree that the glosses used to describe their terms $t_1, t_2 \in T$ are gloss-equivalent EQ_G , that this does not automatically imply $ci(\gamma_1, t_1) \equiv_C ci(\gamma_2, t_2)$. Both terms *should* refer to the same concept; however, both agreements can be established separately. Gloss-equivalences are on the level of the glossary whereas \equiv_C agreements are on the level of the formal descriptions of the concepts (i.e. the lexons). It is necessary for an \equiv_C agreement that the terms must appear in a lexon. This implies that a term will only be in the community commitment if that term plays at least one role (otherwise the term has no purpose for this community). If the term would end up in a taxonomy, then it plays the role of being the sub- or supertype of another term (e.g. with the roles “is a/subsumes”), hence satisfying the condition. Communities can start gradually building their glossary before formally describing their concepts. However, nothing should prevent community members for having agreements on the “sameness” of descriptions across or within their own community. If the definition would impose \equiv_C on the formal descriptions, the community would first need to agree on at least one lexon concerning that term.

Another reason is validation of the equivalences. The glossary-consistency principle will pinpoint the descriptions used for terms that are EQ_G , but whose terms in those communities are not \equiv_C . The glossary-consistency principle does not become a property that needs to hold or else the ontology project fails, instead it becomes a means to drive the

community in establishing \equiv_C ; validating whether the gloss-equivalence was not misleading and both terms really do refer to the same concept.

This is particularly handy as the validity of the natural language descriptions and the equivalence of two such descriptions are relative to the communities participating in these discussions. If glosses have been ill defined and yet their gloss-equivalence agreed upon, a discussion over the articulated term's \equiv_C agreement might help the communities in discovering and rectifying their mistakes.

Gloss-equivalence is a symmetrical property as it captures the communities agreeing that the term-glosses refer to the same concept. Term-adoption, however, is asymmetric. The definition is given below.

Definition 9 (Term-adoption)

Given a hybrid ontology description $H\Omega = \langle \Omega, G \rangle$ and communities $\gamma_1, \gamma_2 \in \Gamma$ and $t_1 \in T$, a community γ_2 is said to adopt $\langle \gamma_1, t_1 \rangle$ when $gloss_1 = g_1(\gamma_1, t_1)$ and $gloss_2 = g_1(\gamma_2, t_1)$ are defined, and we have

- (i) $EQ_T(gloss_1, gloss_2)$, i.e. first “match” the two glosses; and
- (ii) $ci(\gamma_2, t_1) \Leftarrow_{cea} ci(\gamma_1, t_1)$, i.e. agree that both concepts are equal with γ_2 also incorporating the meaning agreements inside $ci(\gamma_1, t_1)$.

\Leftarrow_{cea} is an operator for adopting meaning agreements, allowing the first community to incorporate for their term all meaning agreements around the second community's term. This is achieved by asserting $c_i \equiv_C ci(\gamma_2, t_1)$ for every c_i in $cea(ci(\gamma_1, t_1))$.

In other words, by adopting the gloss of another community-term pair, the adopting community agrees with all existing \equiv_C agreements the adoptee has with other communities. Term-adoption allows γ_1 and γ_2 to agree their respective glosses refers to the same concept (a symmetric condition) and γ_2 agreeing to use t_1 as a term to refer to γ_1 's concept behind it (an asymmetric condition).

Important to note is that assertions of gloss-equivalences and synonymy are only symmetric, reflexive and transitive – i.e. an equivalence relation – *within one agreement process*. This constraint was put in place to avoid synonymy and gloss-equivalences to be propagated without each of the communities validating the new relations inferred from these assertions. If communities A, B and C get together and agree that their terms tA, tB and tC are synonymous, the following assertions are added: $ci(A, tA) \equiv_C ci(B, tB)$, $ci(B, tB) \equiv_C ci(C, tC)$ and $ci(A, tA) \equiv_C ci(C, tC)$. However, if community C and D afterwards agree that $ci(C, tC) \equiv_C ci(D, tD)$, then this does not imply that $ci(A, tA) \equiv_C ci(D, tD)$ or $ci(B, tB) \equiv_C ci(D, tD)$. The agreements on synonymy can be followed by the other communities, allowing them to start interactions to state the terms are indeed synonymous. The same holds for gloss-equivalences.

3.4 Community Commitments

What the definition above does not explain is how to achieve an agreement on how concepts should be referred to. For this, one option is to let the communities involve their application commitments and examine the reference structures for the concepts in those commitments.

While the description of concepts and glosses are formally adequate to be useful in practice, more details of the structure (i.e. organizations) and the processes by which such a community achieves agreement about lexons and about the commitment of a specific information system to the hybrid ontology are required. The first step is the requirement that a community viewed as a context must agree on unique concepts based on terms used in lexons. This is reflected by the definition of the concept identifier function c_i , which maps every community-term pair to exactly one concept.

The second step is the introduction of a *community commitment* to add structure to the agreement processes. A community commitment is similar to an application commitment that it contains a selection $\sigma \subset \Lambda$ and a predicate c over $T \cup R$ of that same selection. It differs from an application commitment in not containing annotations of application symbols and – as a natural consequence – not containing any enterprise-specific knowledge either. The community commitment is an *engagement* of the members within that community to commit to the lexons and constraints agreed upon by the community and captured in the community commitment. The community commitment will contain only lexons and constraints on these lexons describing the domain (thus not belonging to one individual application).

The introduction of a community commitment is motivated by the need for ensuring proper semantic interoperation between information systems. Depending on the goal of the ontology, instances shared across different autonomous information systems need to some degree to be compared for equivalence. One example is joining information from separate sources belonging to one instance of a concept. In order to achieve this, the members of the community have to agree upon a series of attributes that uniquely and totally identify the concepts *they share*. In other words, they have to agree on the reference structures described in Section 3.2.1. By sharing the same reference structures, the information systems are able to unambiguously interpret information describing instances and find the agreed corresponding instance in their data store (or of that of a third system).

By adding an additional layer and a glossary, the first to capture the agreements necessary for interoperability and the latter to facilitate agreement processes, a framework for hybrid ontology engineering has been set up. Figure 3.7 shows how the DOGMA and Hybrid ontology engineering frameworks relate to each other. Notice that application commitments in the hybrid ontology engineering framework are still able to commit to lexons not appearing in a community commitment for describing enterprise-specific knowledge. Application commitments can refer to one or more communities, but can also contain only enterprise-specific information. In that case, however, the annotation of that information system only makes sense for the organization owning that information system and semantic interoperation with that system therefore not guaranteed.

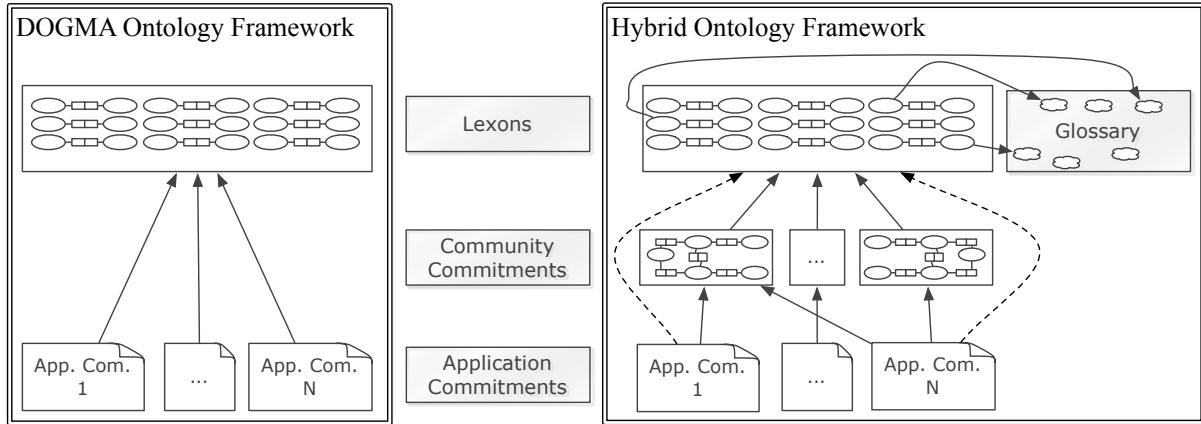


Figure 3.7: The different “layers” in the DOGMA ontology engineering framework and the Hybrid ontology engineering framework.

The hybrid aspect is reflected in a dual perspective on the ontology Ω , and in particular on the glossary underpinning its lexons within a community: the community members agree on unique concepts based on glosses while systems interoperate (reason) based on the relationships (lexons) that are deemed to exist between terms that refer to those same concepts.

It is important to state that constraints in the community commitment are only declared on the lexons currently agreed on by the community, as a community needs to come to an agreement for their specific semantic interoperability requirements. Declaring constraints over lexons from more than one community is possible in application commitments, as application commitments can refer to more than one community and even contain enterprise-specific lexons.

Figure 3.8 depicts a simplification of the iterative process involved. The interactions between the community result in ontology evolution operators applied to the ontology. These operators thus enact the externalization of the reflections of that community. The new ontology description, after a while, will be re-internalized as the community achieves (and discusses about) new insights. The Hybrid Ontology Description is used downstream (ref. Figure 3.9) to generate a knowledge base, e.g. as OWL-defined “storage structures” and constraints/rules implementing relevant commitments for the enterprise information systems to be served. The co-evolution of a community and its Hybrid Ontology Description is a natural consequence of this process. Externalization [DD08] - identifying the key conceptual patterns that are relevant from the discussions - results in a series of ontology evolution operators for the next version and (re-)internalization [DD08] - by committing instance bases to the new version of the ontology - changes the community’s composition: members depart when their goals differ too much from the common goal, or others join.

Before describing the social processes in the next sections, emphasis is again put on the fact that communities in a collaborative ontology engineering method are relevant only if there are two or more autonomously developed information systems that need to interoperate. When there is only one information system, the semantics resulting from

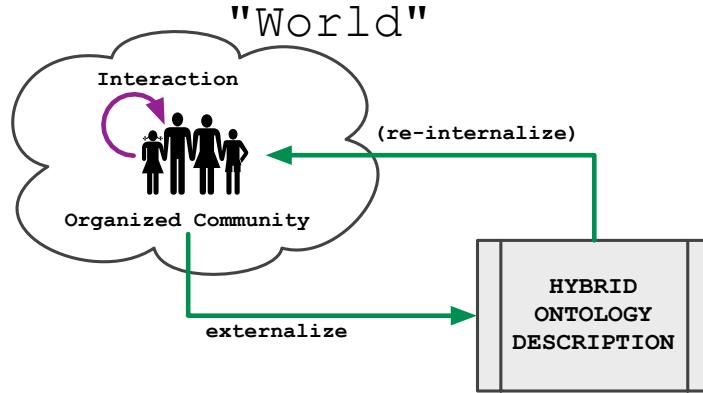


Figure 3.8: Feedback loop between an organized community and an ontology.

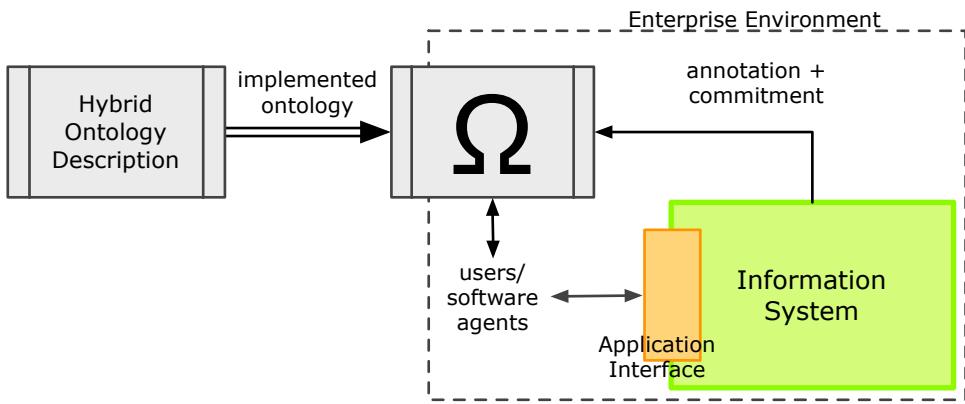


Figure 3.9: “Downstream” usage of the Hybrid Ontology Description to implement the ontology, used for annotating the application symbols of an information system as well as for assisting the validation of the ontology, consistency checks with reasoners, etc.. Users and software agents “recognize” the kind of annotated data provided by the information system and the ontology

that community (even if the number of people is greater than one) are those of that application. This would bring us no step further from going from a closed information system to open information systems.

3.5 Social Processes in Ontology Engineering

To support hybrid ontology engineering, one first needs to define a set of processes within a community that are intended to reflect its member interaction with the “real world” and with each other. Then those processes are “mapped” onto a sequence of ontology evolution operators, as defined by [DdMM07]. It is essential to observe that the ontology description evolves only as the result of agreements, viz. actions performed in principle by multiple community members.

Every ontology evolution operation is subject to discussion before approval during the

re-internalization phase mentioned earlier. Members request certain changes under the form of those operators with a motivation. Depending on the outcome of the discussion, the request is “approved” or “denied”. The latter is useful when the community agrees that the request falls out the scope of milestones closer by. Once the proposed changes have been accepted and the community decides to go to a next version of the hybrid ontology description, all changes are translated into ontology evolution operators.

The boundaries of one ontology engineering iteration and the whole ontology project in general need to be scoped. Scoping helps grounding discussions, preventing members of a community to go “off topic”. The first iteration consists of the initial community of members representing autonomously developed information systems that need to interoperate semantically. The discussions are based on a motivation and a problem scope. The motivation expresses why a Hybrid Ontology Description⁸ or an incremental extension of a Hybrid Ontology Description is needed. During this phase, members can also suggest the use of relevant sources from which inspiration can be drawn upon. Sources of inspiration can be legacy database schemas, standards, documentation, etc. Before going on to the next phase, the relevance of these sources needs to be agreed upon by the community.

The ontology evolution operators are subject to pre- and post-conditions, similar to the pre- and post-conditions for context-dependency management introduced by [DdMM07]. The conditions will be tested only after the proposed changes are approved by the community.

We will describe the social interactions that can take place in a community. Throughout this section, it is assumed that the interactions take place in some community γ . The outcome of some interactions will evolve the community commitment. The community commitment is part of the set of commitments in the hybrid ontology $H\Omega = \langle \langle \Lambda, ci, K \rangle, G \rangle$. This community commitment will be referred to with $\kappa = \langle \sigma, \alpha, c \rangle$.

Before getting started, the function λ is introduced: the function $\lambda : T \times K \rightarrow 2^\Lambda$ returns – given a term t and a commitment – the lexons in that commitment for which one of the terms is t .

Semantic interoperability is defined as the ability of two or more autonomously developed and maintained information systems or their computerized components to communicate data and to interpret the information in that data [DL09]. The scope of an ontology project are defined by semantic interoperability requirements, which are defined as:

Definition 10 (Semantic Interoperability Requirements)

A semantic interoperability requirements for a community $\gamma \in \Gamma$ $SIR(\gamma)$ consists of an ordered pair $\langle KT, GO \rangle$: a non-empty set of key terms $KT \subset \Gamma \times T$ for which descriptions of concepts referred to by these terms are needed and a non-empty set of goals GO , which contain the desired results the community envisions to obtain with the ontology they intend to develop expressed in that community’s language with their usual alphabet.

Since both key terms and goals are captured as sets, one can apply set operations to

⁸In the case of a first iteration, the hybrid ontology description is initially empty.

perform evolution of them.

- The function $kt : \Lambda \rightarrow 2^{KT}$ returns the set of key terms of a community;
- The function $go : \Lambda \rightarrow 2^{GO}$ returns the set of goals of a community.

Add (invite) member and remove member. Members can join (or invited) to take part in refining the motivation and scope and the subsequent ontology engineering processes. When the goals of a community differ to greatly from the interest of one of the stakeholder, a member can decide to leave the community.

Proposing resources that can be used to draw inspiration from. The community discusses which resources will be referred to as information sources from which the lexons and definition could elicited from. Sources can be referred to by a URI, reference, etc., as long as the community as a whole knows where to find and access this information. Examples of such re-sources can be the use of existing standards.

Request to add a key term k

Request to remove a key term k

Request to add a goal g

Request to remove a goal g

For both the addition of a key term or a goal for the SIR of a community, it is of course required that the term or goal is not yet appearing in respectively in K and GO . As for the removal of key terms and goals, they should appear in the SIR of that community in order for the social interaction to take place or the operation to be executed.

Other social interactions are:

Request to add a lexon l . In this social process, the community members discuss whether the lexon l will be included in the community commitment.

Pre-condition	$l \notin \sigma$
Post-condition	$l \in \sigma$

Request to add a constraint. A community discusses whether a constraint on the lexons in the community commitment should be part of that same community commitment. All constraints have the (obvious) pre-condition that the constraint should not appear in the community commitment and the inclusion in the community commitment as a post condition. For a constraint x , there is

Pre-condition	$x \notin c$
Post-condition	$x \in c$

For some specific constraints, different pre- and post-conditions apply. The constraints are handled as follows:

Internal and External Uniqueness Constraints An internal uniqueness constraint spanning one role can be put on either role of a lexon. The external uniqueness constraint will be put on two or more roles over two or more lexons, all referring to the same term. Depending on how one looks at an internal uniqueness constraints spanning one role, it can be either read as a functional role, or an identifying role. In other words, EACH Person has AT MOST ONE Name. is the name as EACH Person IS IDENTIFIED BY (Name of Person). With this knowledge, the internal uniqueness constraint spanning only one role can be seen as an identifying constraint, and can be treated in the same way as external uniqueness constraints identifying terms do.

An identifying constraint x identifies a term $t \in T$ via one or more paths π_1, \dots, π_n . Each path π_i is an *ordered* set of path-segments s_1, \dots, s_n of the form $\langle t_1, r, t_2 \rangle$ where $t_1, t_2 \in T$ and $r \in R$. Intuitively, a path-segment denotes a term playing a role on another term. Π is a function that returns the paths of a constraint.

Pre-condition	$\forall \pi \in \Pi(x) (\forall \langle t_1, r, t_2 \rangle \in \pi (\exists r' \in R (\langle \gamma, t_1, r, r', t_2 \rangle \in \sigma)))$ $\forall \pi \in \Pi(x) (\forall i \in [1, \pi - 1] (nth(i, \pi).t_2 = nth(i + 1, \pi).t_1))$ $\forall \pi \in \Pi(x) (nth(\pi , \pi).t_2 = t)$
Post-condition	

The first pre-condition states that there should exist a lexon in the community commitment for each segment in a path. The second states that every two segments in a path should be “chainable”, meaning that the second term of the first segment should be the first term in the second segment. Finally , the third pre-condition ensures that every last segment in the constraint ends with the term.

Mandatory Constraint A mandatory constraints states which roles must be played at least once by a term t . Similarly to uniqueness constraints, a mandatory constraint involves paths and segments. The difference, however, is that each path is composed of exactly one segment and that the role is the one played by the term t of the constraint x .

Pre-condition	$\forall \pi \in \Pi(x) (\forall \langle t_1, r, t_2 \rangle \in \pi (\exists r' \in R (\langle \gamma, t_1, r, r', t_2 \rangle \in \sigma)))$ $\forall \pi \in \Pi(x) (\pi = 1)$ $\forall \pi \in \Pi(x) (nth(1, \pi).t_1 = t)$
Post-condition	

Every segment should have a corresponding lexon in the community commitment for the mandatory constraint as well. The number of segments in a path should be exactly one, and every segment should start with the term.

Lexicality of Term

Synonymy Will be handled later on in this section.

Request to remove lexon $\langle \gamma, h, r, r', t \rangle$.

Pre-condition	$\langle \gamma, h, r, r', t \rangle \in \sigma$ $\forall x \in c(\Pi(x) \neq \emptyset \rightarrow$ $(\forall \pi \in \Pi(x)(\# \langle l, m, n \rangle \in \pi$ $((l = h \wedge m = r \wedge n = t) \vee (l = t \wedge m = r' \wedge n = h))))$ $ \lambda(h, \kappa) = 1 \rightarrow \# t' \in T(\# \gamma' \in \Gamma(Links(\gamma, h, \gamma', t') \in c)$ $ \lambda(t, \kappa) = 1 \rightarrow \# t' \in T(\# \gamma' \in \Gamma(Links(\gamma, t, \gamma', t') \in c)$ $ \lambda(h, \kappa) = 1 \rightarrow IsLexical(h) \notin c$ $ \lambda(t, \kappa) = 1 \rightarrow IsLexical(t) \notin c$
Post-condition	$\langle \gamma, h, r, r', t \rangle \notin \sigma$

The pre-conditions are fairly straightforward. If the lexon to be removed is used in a constraint, its removal cannot be executed. For mandatory and uniqueness constraints, the motivation is clear; “relaxing” the constraints by removing the paths in which this lexon is referred to, changes the whole meaning of the constraint. Discussing the constraints to be changed before the removal of this lexons thus seemed appropriate, as the community will then make a thoughtful decision. The same approach was adopted for synonymy links and the lexical nature of terms (only when the number of lexons in which this term is occurring equals to one). The constraints and synonyms have thus to be removed if this term will completely disappear from the community commitment.

Request to change the supertype of a term t . Allows a community to discuss the taxonomy of the community’s concepts. The concept hierarchy is constructed with a lexon whose roles bear a special meaning (the taxonomic relation, e.g. with role and co-role “is a” and “subsumes”). The taxonomic relation is assumed to be transitive. A function is defined isa which returns true when the first operand is a specialization of the second operand.

$$isa(x, y) = \begin{cases} 1 & \text{if } \langle \gamma, x, is\ a, subsumes, y \rangle \in \sigma \\ 1 & \text{if } \exists z(\langle \gamma, x, is\ a, subsumes, z \rangle \in \sigma \wedge isa(z, y)) \\ 0 & \text{Otherwise} \end{cases}$$

When no supertype was defined, a taxonomic relation is added between the two terms. For adding $\langle \gamma, t, is\ a, subsumes, s \rangle$

Pre-condition	$\langle \gamma, t, is\ a, subsumes, s \rangle \notin \sigma$ $\neg isa(s, t)$
Post-condition	$\langle \gamma, t, is\ a, subsumes, s \rangle \in \sigma$

When such a relation $t, is\ a, subsumes, s'$, where $s \neq s'$, already exists between the terms and another super terms, the existing taxonomic relation is removed before the creation of the new one. The *additional* pre- and post-conditions for this case are added:

Pre-condition	$\langle \gamma, t, is\ a, subsumes, s' \rangle \in \sigma$
Post-condition	$\langle \gamma, t, is\ a, subsumes, s' \rangle \notin \sigma$

Request to change “super lexon” of a lexon. Which indicates that the population of a lexon is a subset of the more general lexon. A special operation, as it corresponds with a subset constraint on both roles of the two lexons [HM08, TTM07].

The subset constraint, however, will be modeled to create role hierarchies, hence the reference to “super lexon”. Pre-conditions for this constraint is that both terms of the more specialized lexon are in a taxonomic relation with the corresponding terms of the more general lexon in direct line. The subset constraint is – as its name implies – a constraint. But since it will be used to create role hierarchies, a special interaction was assigned to this constraint in a similar way that subtype relations are modeled via a special interaction.

The following conditions for a subset constraint x between lexons $\langle \gamma, l, m, n, o \rangle$ and $\langle \gamma, p, q, r, s \rangle$ apply.

Pre-condition	$\langle \gamma, l, m, n, o \rangle \in \sigma$ $\langle \gamma, p, q, r, s \rangle \in \sigma$ $(isa(p, l) \vee isa(l, p)) \wedge (isa(s, o) \vee isa(o, s))$ $x \notin c$
Post-condition	$x \in c$

Request to remove a constraint. The removal of a constraint has no implication on the community commitment or on the glossary. The pre- and post-condition of this social interaction, if accepted, are trivial.

Pre-condition	$x \in c$
Post-condition	$x \notin c$

Request to add gloss ϕ , for a particular term $t \in T$ or lexon $\lambda \in \Lambda$ in a community $\gamma \in \Gamma$, or request to $g_1(\gamma, t) \leftarrow \phi$ or $g_2(\lambda) \leftarrow \phi$. In order to maintain glossary coherence, terms in a lexon need to be articulated before that same lexon can be articulated.

For a:	term t	lexon $\lambda = \langle \gamma, a, b, c, d \rangle$
Pre-condition	$\nexists x \in Gloss(\langle \gamma, t, x \rangle \in g_1)$	$\exists x \in Gloss(\langle \gamma, a, x \rangle \in g_1)$ $\exists x \in Gloss(\langle \gamma, d, x \rangle \in g_1)$ $\nexists x \in Gloss(\langle \lambda, x \rangle \in g_2)$ $\lambda \in \sigma$
Post-condition	$\langle \gamma, t, \phi \rangle \in g_1$	$\langle \lambda, \phi \rangle \in g_2$

A gloss is always gloss-equivalent with itself. So when there was already a community using a particular gloss and this gloss is used to describe the concept a term or lexon is referring to, the community has immediate knowledge of all terms and lexons annotated with that same gloss by the previous community.

In the situation that the community uses a gloss that has already been used by another community and inside a gloss-equivalent agreement with different gloss, the gloss-equivalences remain. However, if that community disagrees with the gloss-equivalence, either the community starts processes to remove this equivalence with all involved communities, or they change the gloss to reflect the difference.

Request to remove a gloss ϕ . To remain glossary coherent, one can choose to remove all articulated lexons around a described term when the gloss of that term is removed, or one can choose not to allow this operation to happen yet. The second approach was adopted in this thesis, as changing the gloss would be more appropriate and also to preserve the agreements on term glosses.

For a:	term t	lexon $\lambda = \langle \gamma, a, b, c, d \rangle$
Pre-condition	$\langle \gamma, t, \phi \rangle \in g_1$ $\# \lambda = \langle \gamma, l, m, n, o \rangle \in \sigma($ $(t = l \vee t = o)$ $\wedge \exists x \in \text{Gloss}(\langle \lambda, x \rangle \in g_2))$	$\langle \lambda, \phi \rangle \in g_2$ $\lambda \in \sigma$
Post-condition	$\langle \gamma, t, \phi \rangle \notin g_1$	$\langle \lambda, \phi \rangle \notin g_2$

Removing a gloss from a term or lexon has no impact on any existing gloss-equivalence agreements. Indeed, it is not because a community deems that a particular gloss should be removed from a term, that this gloss should cease to exist and the agreement with another community that this gloss refers to the same concept as another gloss removed. If no term or lexon of the first community is linked with the gloss, the other community still agrees that the glosses were gloss-equivalent, this gloss-equivalence and thus remain in EQ_G . This explanation is depicted graphically in Figure 3.10 using terms, but the same applies for lexons.

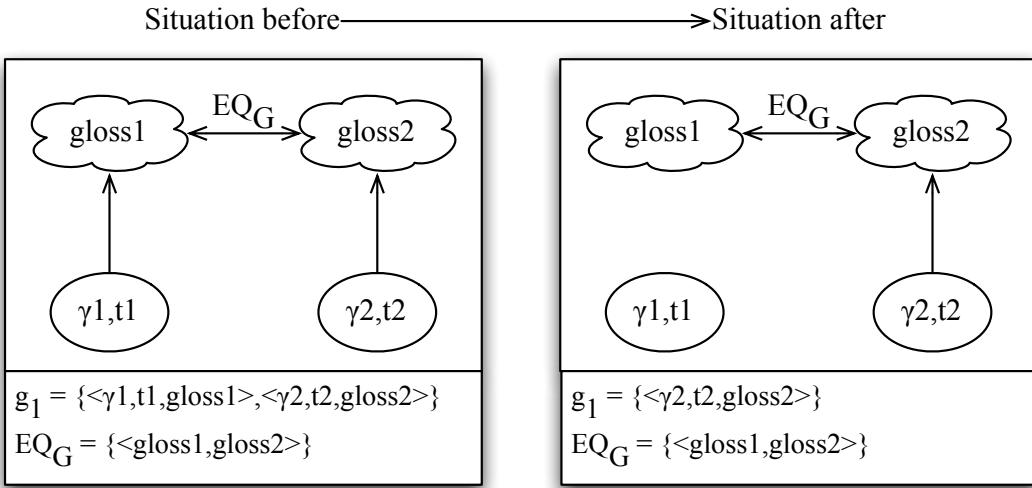


Figure 3.10: Visualization of the situation of the state after the removal of a gloss from a term.

Request to change a gloss with ϕ .

For a:	term t	lexon $\lambda = \langle \gamma, a, b, c, d \rangle$
Pre-condition	$\langle \gamma, t, \phi \rangle \notin g_1$ $\exists x \in \text{Gloss}(x \neq \phi \wedge \langle \gamma, t, x \rangle \in g_1)$	$\langle \lambda, \phi \rangle \notin g_2$ $\exists x \in \text{Gloss}(\langle \lambda, x \rangle \in g_2)$ $\lambda \in \sigma$
Post-condition	$\langle \gamma, t, \phi \rangle \in g_1$ $\# x \in \text{Gloss}(x \neq \phi \wedge \langle \gamma, t, x \rangle \in g_1)$	$\langle \lambda, \phi \rangle \in g_2$ $\# x \in \text{Gloss}(\langle \lambda, x \rangle \in g_2)$

The situation where a community decides to change the gloss is trickier. A community can have multiple terms and lexons pointing to the same gloss. In that way, the community either decides to change the gloss for all these terms or lexons, or just a few. The problem here is what should happen with any previous gloss-equivalences? Similarly to a gloss-removal, any gloss-equivalences remain as they have been previously agreed upon. For all terms and lexons whose gloss has been changed, however, social interactions are started to investigate gloss-equivalences with the update. In other words, if a community γ_1 wishes to change the gloss ϕ_1 of term t into ϕ_2 . Social interactions for gloss-equivalence are started between all communities in

$$\begin{aligned} \{\gamma_2 | \exists \phi \in \{\langle \phi_1, \phi \rangle \in EQ_G\} \wedge \\ (\exists t \in T(\langle \gamma_2, t \rangle \in g_1)) \vee \\ (\exists t, t' \in T(\exists r, r' \in R(((\langle \gamma_2, t, r, r', t' \rangle, \phi) \in g_2)))) \\ \}\end{aligned}$$

This explanation is depicted graphically in Figure 3.11 using terms, but the same applies for lexons.

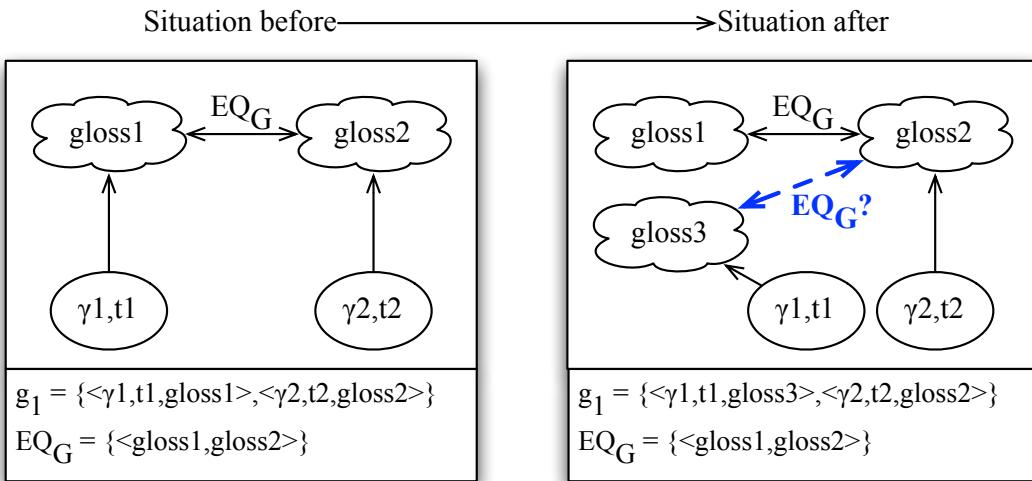


Figure 3.11: Visualization of the situation of the state after the change of a gloss from a term.

Request to add synonym $s = link(\gamma_1, t_1, \gamma_2, t_2)$. A request to link terms across different communities, so that $ci(\gamma_1, t_1) \equiv_C ci(\gamma_2, t_2)$ where $t_1, t_2 \in T$ and $\gamma_1, \gamma_2 \in \Gamma$.

Pre-condition	$s \notin c_1 \wedge s \notin c_2$ $\exists a(\exists b(\exists c(\langle \gamma, t_1, a, b, c \rangle \in \sigma_1)))$ $\exists a(\exists b(\exists c(\langle \gamma, t_2, a, b, c \rangle \in \sigma_2)))$
Post-condition	$s \in c_1 \wedge s \in c_2$

Request to remove synonym. This social interaction can be initiated when two communities deem that the definitions of both concepts referred to by these terms diverge in such a way that they do not handle about the same anymore, or it becomes clear that the terms referred to different concepts.

Pre-condition	$s \in c_1 \wedge s \in c_2$
Post-condition	$s \notin c_1 \wedge s \notin c_2$

Request to add gloss-equivalence. This social interaction can be initiated between two communities γ_1 and γ_2 if the two communities deem that their respective glosses ϕ_1 and ϕ_2 used to articulate one of their terms or lexons are referring to the same concept.

Pre-condition	$EQ_G(\phi_1, \phi_2) \notin EQ_G$
Post-condition	$EQ_G(\phi_1, \phi_2) \in EQ_G$

Request to remove gloss-equivalence. Two communities γ_1 and γ_2 can initiate this social process to remove from EQ_G the assertion that two glosses ϕ_1 and ϕ_2 which the communities respectively used to annotate their terms or glosses do not refer to the same concept anymore.

Pre-condition	$EQ_G(\phi_1, \phi_2) \in EQ_G$
Post-condition	$EQ_G(\phi_1, \phi_2) \notin EQ_G$

At any given point, in order to achieve unification, discussions between users can take place. One can compare such discussions with posts and Web forums. By linking posts with their replies, one can create threads. An item in such discussion can be a trigger for an ontology operation or a task assigned to a person. A link is therefore kept between a task and an ontology operation if the post in question was the source of this action. For example, users who do not feel comfortable with formal ontology operators or do not know how to solve a problem might request an edit.

Request for edit. A general request for edit (or solving a problem). For instance used when a member feels he has not enough responsibility over the concept to propose the actual changes.

Request for information. Not to be confused with a request for edit for glosses, but rather a request for clarification. Such a request might result in a request for edit or as a request for an ontology operation.

Request for peer review. An invitation to review some aspects of the ontology, e.g. inviting members of the community to give comments to certain proposed changes, even though they are not immediately affected by the concepts in question.

Request for help in contributing to certain aspects of the Hybrid Ontology Description.

Comment. A comment to a post or a concept, a general class of posts that are not related to the other types of posts.

Reply. All posts not belonging to any category in a thread. For DILIGENT, Pinto *et al.* provided the following types of arguments in a discussion [PST04]: elaboration, justification, example, counterexample, evaluation, and alternative. NeOn adopted this classification, but did not take into account alternative proposal [DEMB⁺08]. NeOn does, however, provide more structure. They propose: supporting and objecting examples, supporting and objecting evaluations, and supporting and objecting justifications. For this thesis, the classification used by NeOn is used, but the alternative is added. Thus, for hybrid ontology engineering, the types of replies are:

- Supporting example;
- Supporting justification;
- Supporting evaluation;
- Objecting example;
- Objecting justification;
- Objecting evaluation;
- Alternative.

Concluding a discussion. All social interactions have to be concluded. When concluding, one member of the community summarizes the outcome of the discussion. On acceptance, the operation on the hybrid ontology that has been discussed is carried out. The operation carried out might be different from the initial proposition. For instance, the community can discuss a new gloss for articulating a term, yet agree on a different version of that gloss that will be entered.

The removal of a term-gloss results in the removal of the lexon-gloss if the lexon(s) involved were articulated as well. All gloss-equivalence assertions around this term-gloss are removed as well. Here again, if a particular term is again articulated with the same gloss at a later term, those assertions need again to be agreed upon. The same happens when glosses are changed. The gloss-equivalences have to be reconsidered in order to be kept.

This thesis follows [Gua98, Hep08] in that an ontology conceptualization should be separated from its instances. Instances should therefore not be part of the community commitment. Examples, however, often clarify the concepts that are being modeled by the community. We therefore introduce social processes for managing examples that will serve as so-called *test populations* [HM08].

Request to add an example.
Request to update an example.
Request to remove an example.

Note that these examples are not the same as the supporting and objecting examples in the *reply* social process. Supporting examples however, can be introduced via these

requests. Note also that these instances will *not* be included in the community commitment. Rather, they will be stored separately as to follow the strict separation between conceptualization and instances. This artifact can be seen as a “sandbox” application commitment.

Chapter 6 will later on describe how these examples in that application commitment as well as those of real applications will be used to test statements made by the community. For example, by testing whether a constraint violates the population of lexons made with the examples. The outcome of these tests will steer the social processes within the community.

3.6 Conclusions

This chapter started from the DOGMA framework for hybrid ontology engineering and limited the context identifiers of lexons to pointers to a community, as all agreements take place within one or across communities. The ownership of knowledge is now given back to the community. To facilitate meaning agreements within a community, a glossary was introduced which facilitates the alignment of a community to formally describe concepts. The glossary adds also an additional layer of meaning agreements; in this framework communities can agree that not only terms in labels refer to the same concept, but also individual glosses. To facilitate interactions, the possible social interactions in this framework have been described.

A framework is only one thing. One needs a method to guide the community in reaching a consensus. As Vrandecic *et al.* reported in an experiment, not prescribing processes will render the ontology engineering method inefficient [VPST05]. This motivates the need for a method. By limiting the context identifiers to pointers to a community, part of the ontology’s ownership is given back to the community. The next chapter will present a method that will “orchestrate” all social interactions described in this chapter in such a way that communities are guided in reaching consensus and communities are in charge of making decisions. The method will therefore support the community in owning and maintaining their own ontology.

Chapter 4

Hybrid Ontology Engineering Method

4.1 Introduction

Chapter 3 introduced a framework for hybrid ontology engineering that is built on top of DOGMA, a fact-oriented ontology engineering framework. We also defined several social processes enabling a community to alter the hybrid ontology towards a closer approximation of that community's domain.

This chapter introduces a method for hybrid ontology engineering. A method prescribes certain guidelines and steps to achieve a certain goal; here the construction of a hybrid ontology. The method adopts and orchestrates the social processes defined in the previous chapter. To exemplify the claims and definitions, a fictional case in the cultural domain will be presented and used as a running example.

The method in this chapter is called GOSPL, which stands for **G**rounding **O**ntologies with **S**ocial **P**rocesses and natural **L**anguage. Figure 4.1 summarizes the different processes in GOSPL. Starting from co-evolving communities and requirements, the articulation of key terms have to be gathered before formally describing those concepts. These formal descriptions can be constrained and then committed to by applications using Ω -RIDL application commitments. During the processes from creating the glossary to committing to the hybrid ontology description, the communities can make agreements on gloss-equivalences and synonyms. The hybrid ontology, and the data described with those commitments can then be re-internalized by the community for another iteration, gradually approximating the domain that needs to be captured by the ontology.

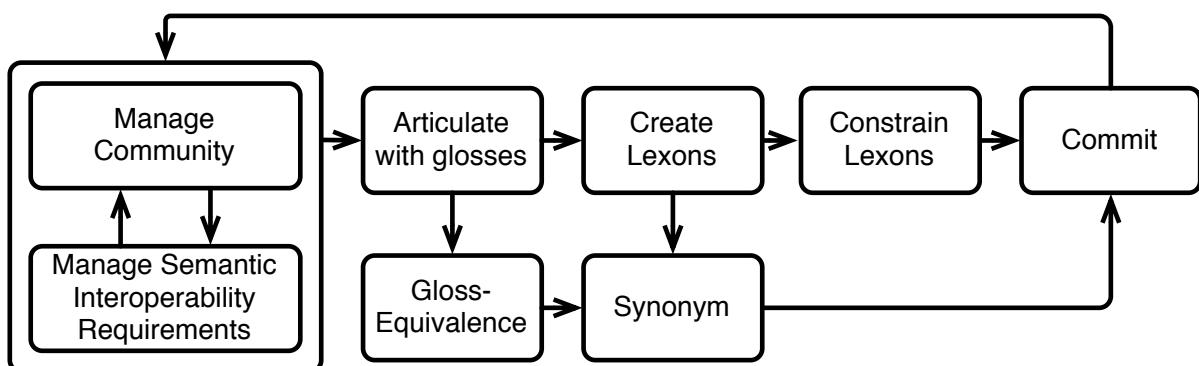


Figure 4.1: The GOSPL method.

4.1.1 Running Example

To exemplify the claims and definitions, this section presents a running example inspired from a case study in the cultural domain (see Chapter 8).

Brussels is becoming popular amongst tourists for weekend getaways (also called “city breaks”) and holidays, as observed in an increase of overnight stays¹. Brussels has almost 90 museums, dozens of theaters and as many galleries. All are offering cultural events that could possibly interest tourists. However, getting appropriate information on when, where and for what price events are available, might be cumbersome for tourists to find out, certainly while walking around the city. Luckily, some organizations, such as BOZAR², provide detailed information on the different events taking place in their buildings. The events are, however, often limited to the ones organized by that organization. Other organizations – such as VISITBRUSSELS asbl³ – aim at providing a portal with information about cultural events organized around the city. The information on such portals, however, are shallow and most information are unstructured. One also needs to know the existence of those organizations and websites in order to find the information.

As tourists often walk in front of buildings hosting cultural events, we wonder if information could be delivered to them based on their proximity to these buildings. The Open Semantic Cloud for Brussels⁴ project aims at providing the cultural sector a service for uploading pictures of a building returning – if possible – information about the events taking place in that building. The different organizations form a community with the developer of that service. Portals aggregating the different events will be able to directly request the data from the organizers rather than duplicating the same information in separate databases, which may become out of sync as organizers update information about events. Furthermore, such portals will be able to provide more detailed and structured information to their users.

In this running example, the different stakeholders (BOZAR, Agenda.be, etc.) will constitute a community that will be called the “Cultural Domain Community”.

4.2 Semantic Interoperability Requirements

GOSPL is tailored towards communities of stakeholders representing autonomously developed and maintained information systems with the need to have their information systems exchange information in a meaningful manner. That need is then translated into a goal, and the goal is represented by the semantic interoperability requirements (SIRs, see Definition 10 in Section 3.5). From this follows that a community is partly identified by its semantic interoperability requirements. More specifically, communities are identified by their requirements and its set of members.

The semantic interoperability requirement for the cultural domain community in this

¹http://visitbrussels.be/bitc/static/front/img/db/img_7291.pdf

²<http://www.bozar.be/>

³<http://www.visitbrussels.be/>

⁴<http://www.oscb.be/>

example can look as follows:

```
A description of
- Event
- Schedule
- Location
- Ticket price
- Target audience
- Artist
- Work of Art
for
- the exchange of information of events in the cultural domain.
```

To discuss (changes to) the semantic interoperability requirements, the community starts by the following social processes:

- Request to add a key term;
- Request to remove a key term;
- Request to add a goal;
- Request to remove a goal.

The post-conditions that both sets of key terms and goals are non-empty, need to be met.

4.3 Building the Glossary

Semantic interoperability is achieved by annotating the application symbols of an information system with terms and relations in the hybrid ontology. In the previous chapter we already described how – in a hybrid ontology – terms are on one hand articulated with natural language descriptions called glosses for humans, and described formally for annotating information systems and their computerized systems on the other.

The term- and role labels in lexons evoke concepts within the members of a community. Different terms may evoke similar concepts to member of the community, while the use of the same term does not always mean that the same concept is evoked to different members of that community. The relation between a symbol (in this case the labels), the concept it evokes and the referent the symbol stands for and referred to by the concept has been captured in the famous semiotic triangle [OR23], shown in Figure 4.2. The semiotic triangle implies that the referent of an expression (which can be a word, a sign or a symbol) is relative to different interpreters. The expression invokes for the *interpreter* a certain idea (thought) that refers to something (the referent) that symbol is standing for. Note that [Pei35] extended this semiotic triangle with the notion of interpreters as symbols stand to somebody for something in some respect or capacity.

To ensure all members of a community are referring to the same referent for a particular label, the community needs to align their ideas of the concept symbolized by the term. This process is called *alignment*. Alignment is achieved by (1) describing the concepts

referred to by these labels and (2) having the community members agree on such one description per label.

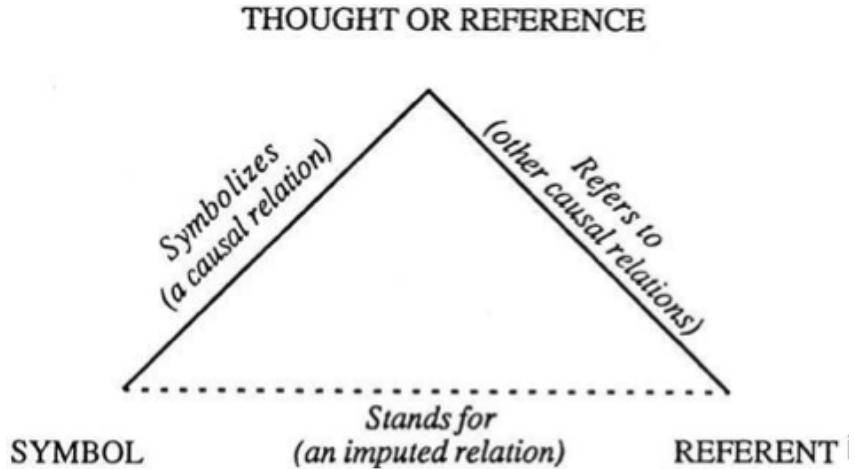


Figure 4.2: The semiotic triangle by Ogden and Richards [OR23]

There are two ways to describe a concept. On the one hand, concepts can be described by providing an (unambiguous) definition in natural language that everyone can understand. The meaning of this term is then determined by this definition if this definition is agreed upon by the community. On the other hand, one can describe the term formally by decomposing and agreeing upon every little detail, which in turn will need to be decomposed and agreed upon as well. The descriptions of these details will be based on other concepts and therefore hampered by the fact that they also need to be described formally.

To facilitate alignment, GOSPL imposes terms to be articulated before formal descriptions are added, starting with the list of key terms in the semantic interoperability requirement. To this end, the community can use the following social processes:

- Request to add a gloss to a term
- Request to change a gloss of a term
- Request to remove a gloss of a term

During the first iteration, there are no lexons. A community needs to wait for lexons to emerge before they can start articulating those. Lexons can be articulated with a gloss only if both its terms are articulated in order to maintain glossary coherence.

A community is able to articulate all the lexons. However, GOSPL strongly encourages articulating at least those lexons whose internal uniqueness constraints span more than one role. In other words, GOSPL encourages the articulation of non-attributive relations. In the absence of an internal uniqueness constraint, the uniqueness constraint is assumed to be spanning the two roles (as with the CSDP procedure for the development of closed information systems [HM08]). Such relations must correspond with a concept in the domain that needs to be approximated by the ontology. This is in contrast with so called

“attributive” relations, which can be too “trivial” to fully articulate as they are often meronymic (i.e., part-whole relations).

Take for instance the lexon $\langle \text{Cultural Domain Community}, \text{Artist}, \text{contributed to, with contributor, Work Of Art} \rangle$. This lexon can be losslessly decomposed into a series of lexons with functional roles as seen in Figure 4.3. As this figure shows the link between a complex relation and a concept, it becomes clear why the community is encouraged to articulate such lexons: the lexon symbolizes a reference for an referent in the real world; in this case a relation. Note that this also enables communities to obtain gloss-equivalences between lexon- and term-glosses. In the example of $\langle \text{C2}, \text{Artist}, \text{born on, of birth of, Date} \rangle$ with an artist born on at most one Date, date (of birth of) becomes an attribute of Artist. Describing the relation as being the occurrence of people having a birth date is therefore not needed. Non-attributive relations thus denote concepts and therefore need to be articulated by the community.

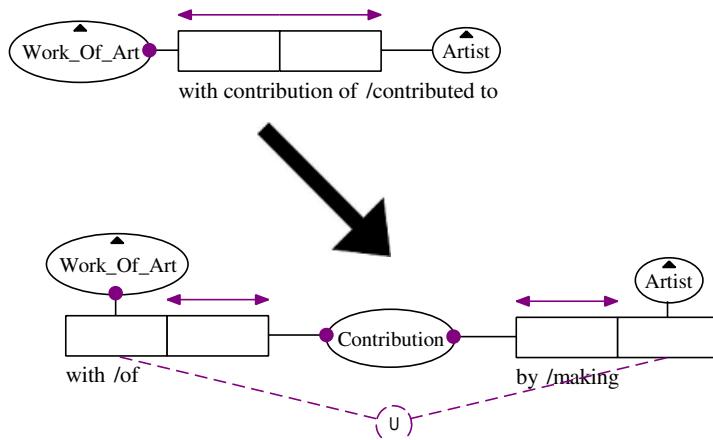


Figure 4.3: Decomposing a lexon with a uniqueness constraint spanning two roles into a two lexons; a lossless transformation.

The social processes for articulating lexons are similar to those of articulating terms:

- Request to add a gloss to a lexon
- Request to change a gloss of a lexon
- Request to remove a gloss of a lexon

If no internal uniqueness constraint is specified on a lexon, it is implied that a uniqueness constraint spanning both roles is holding. In other words, the absence of a uniqueness constraint spanning only one role on a lexon implies that the roles are non-attributive.

During the first iteration, the cultural domain community needs to start articulating the key terms and their semantic interoperability requirements as per the aforementioned social process. The members in that community can look up definitions in dictionaries, in house documentation or even come up with new definitions that will hold in their community. After such an iteration, the glosses could look as follows⁵:

⁵The glosses come from the experiment conducted for this thesis (see Chapter 8). The “quality” of

- (*Cultural Domain Community, Location*), “A location describes the position of an object or a place on Earth. It can be identified by either specifying an address or its geographical coordinates.”)
- (*Cultural Domain Community, Artist*), “An artist is a person who engages in one or more activities that are considered a work of art.”)
- (*Cultural Domain Community, Work of Art*), “A cultural object/act that was created with the intent of being aesthetic, rather than serving some practical purpose.”)
- ...

4.4 The Creation of Lexons

Lexons can only be entered in the lexon base if and only if at least one of the terms in those lexons have already been articulated. Indeed, it would be undesirable to describe a relation between two terms if both terms playing the roles in that relation are not described themselves, meaning that their intended meaning has not yet been made explicit. If at least one of the terms is described, one can assume that the lexon proposed around that term is in function of the articulation of that term and/or the semantic interoperability requirements⁶.

Figure 4.4 depicts this process. In this figure, where blank circles denote terms with no articulation, non-blank circles are terms that are articulated and an edge between two circles denotes a relation between two terms. Step 1 represents the concepts taken from the semantic interoperability requirement. Those terms are – in that stage – not yet articulated. Communities need to articulate those terms (step 2) before adding relations between those terms (step 3). Lexons added in step 3 may contain terms that are not yet defined, which will first need to be articulated (step 4) before using them in other lexons (step 5). Ultimately, all the terms should be articulated in the hybrid ontology for that community (step 6).

At this stage, the community can perform the following social processes provided at least one of the terms is articulated:

- Request to add lexon
- Request to remove lexon

Using these two requests, a community is able to agree upon relations that are plausible in their universe of discourse. The community can for instance agree that the following lexons hold in their domain (where ‘C’ stands for “Cultural Domain Community”):

```
<C, Affiliation, of, with, Artist>
<C, Artist, born on, of birth of, Year>
```

these glosses may be disputed. The communities, however, at the time deemed those glosses adequate for alignment.

⁶The relation between glosses and their impact on the community commitment will be described in Chapter 5.

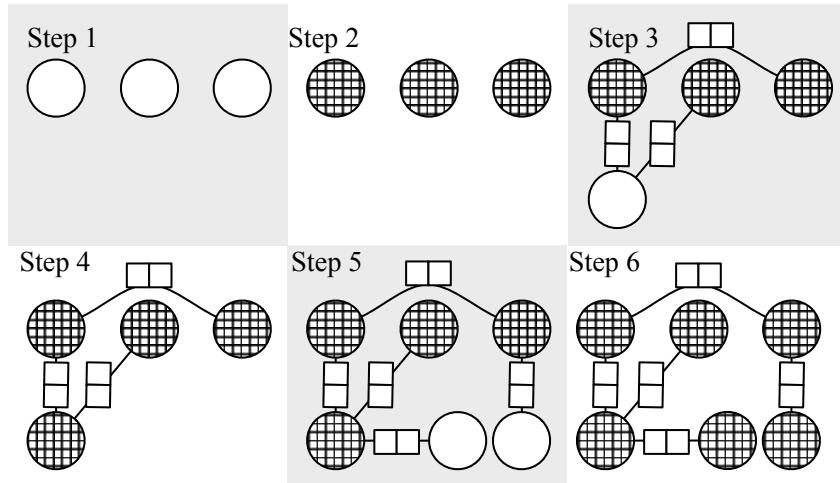


Figure 4.4: Lexons can only be added around a term if and only if the community articulates the term.

```

<C, Artist, contributed to, with contribution of, Work Of Art>
<C, Artist, with, of, Gender>
<C, Artist, with, of, Name>
<C, Location, has, of, Address>
<C, Location, is positioned at, positions, Coordinate>
...

```

These lexons will help the community in exchanging information across heterogeneous information systems. Furthermore, they enable a community to define the relations holding between concepts in their domain. In the previous chapter, a motivation for a special type of community – the community commitment – was given to capture the shared constraints. This will ensure proper interoperability. The creations of constraints on the shared lexons will be explained in the next section.

4.5 Constraining Lexons

Application commitments contain next to a selection of lexons from the hybrid ontology and mappings from application symbols to terms and roles in that selection also constraints on that selection that indicates how that particular application uses those concepts. Some of these constraints have to be shared and agreed upon by the community in order to meet the interoperability requirements. Those constraints should not stem from the individual applications, but be part of the domain that is being modeled.

An important combination of such constraints occurs when they constitute a unique, total identification⁷ for a given (usually stored) concept. Such identifiers are called a *unique simple reference* when only one lexon is involved, or a *unique composite reference* in the

⁷The terms “Unique”, “total” and “identifying” are all used as in ORM, see Section 3.2.1.1.

case of more than one lexon. Definitions for both simple and unique compound references are given below and Figure 4.5 shows an example of a simple and a unique composite reference using the graphical notation by [HM08].

Definition 11 (Unique Simple Reference)

Let A be a non-lexical object type, B an object type and $\langle A, r_1, r_2, B \rangle$ a fact type between them. Then $\langle B r_2 \rangle$ is called a unique simple reference for A if and only if r_2 is an identifying role and r_1 is a identifying and mandatory role.

Definition 12 (Unique Composite Reference)

Let A be a non-lexical object type, B_1, \dots, B_n object types and $\langle A, r_{11}, r_{12}, B_1 \rangle, \dots, \langle A, r_{n1}, r_{n2}, B_n \rangle$ a fact type between them. Then $\langle B r_{12}, \dots, B r_{n2} \rangle$ is called a unique composite reference for A if and only if there is a uniqueness constraint identifying A and involving exactly the roles r_{12}, \dots, r_{n2} and each of the r_{11}, \dots, r_{n1} is a identifying and mandatory role.

A classic example of such a combination is *book* being uniquely, and totally identified by its *ISBN number*. Unique references are needed to ensure proper semantic interoperation between the different systems, and hence proper business between the different organizations owning those systems. As described in the previous chapter, those constraints will – once agreed upon – be stored in the community commitment.

The community thus might need to agree on constraints to meet the goals captured by their semantic interoperability requirements. A distinction is made between two constraints: on terms or on (roles of) lexons. In both cases, the GOSPL method imposes the terms to be articulated with a gloss. Indeed, it would be unreasonable to constrain the use of a term, a role, or a lexon whose intended meaning has not yet been made explicit and agreed upon by the community.

Agreements can be made on all constraints described in the previous chapter (mandatory constraint, uniqueness constraint, etc.). In our method, the emphasis will be put on the necessary constraints to create reference structures. To achieve agreement on identification or disambiguation any term referring to a type of concept, indeed must also necessarily possess a lexical reference construct that allows to uniquely refer, at the type level, to every eventual instance of that concept. In other words, any such (“non-lexical”) type term must have a fixed set of one or more attributes (a first normal form candidate key, in database speak) that are agreed/declared to totally and uniquely identify any given instance of that type. Such a concept/non-lexical object type is then called referable. Note that the attributes involved can be lexical, or non-lexical if the latter recursively are referable, and that in general such reference schemes will be “information bearing”, i.e. need to be subject of a formal commitment to the lexon base.

Reference schemes for non-lexical object types are essential for system and enterprise interoperability as they are part of the domain rules that help software agents distinguish

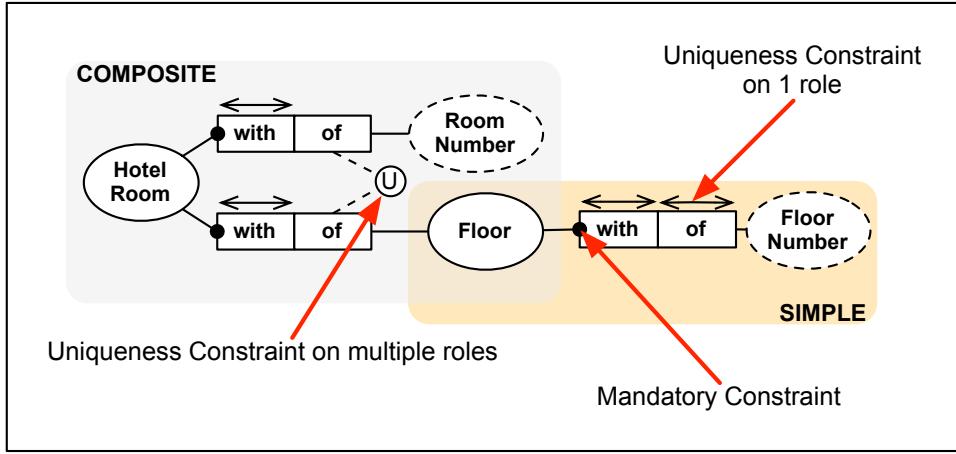


Figure 4.5: An example of a simple and a composite unique reference.

or merge instances. The same holds for the Semantic Web and Linked Data, where one only has URIs to identify resources and (debatable) OWL constructs to establish equivalences. An application that wishes to exploit the availability of billions of triples will keep its semantics inside its program (e.g. rejecting a bunch of triples belonging to a resource not fulfilling the requirements), thus offering little improvement when these agents want to exchange data. For a theoretical treatment of the complexity issues of identification and unique referencing in databases when considered in a semantic web context, see [CDGL01], who represents functional dependencies in Description Logic.

Given the definition of RM-referability, every “leaf” in the RM-referability structure needs to be a lexical object type. To this end, users are allowed to agree upon the fact that some terms in the hybrid ontology are lexical. Unlike the mandatory or uniqueness constraint, this is a constraint on the term (in lexons thus a term) and not on a role.

The reason for focusing on these constraints is that there exists an algorithm that transforms (maps) an ORM schema into a (normalized) relational database schema with additional constraints. The algorithm is described in detail in Appendix A.

For managing constraints in a hybrid ontology, the social processes are:

- Request to add a constraint
- Request to remove a constraint

For the social process “Request to change superlexon of lexon”, it is required that the four terms of both lexons involved be articulated. Indeed, how can one imply that an instance playing a particular role “r1” implies that same instance playing another role “r2” if the terms or the relation itself are not specified. Lexons can be articulated as well if and only if both its terms are articulated.

- Request to change supertype of a term (as the subsumption relations ship is a lexon whose role labels have a special interpretation, e.g. “is a / subsumes”).

In the running example, it is assumed that the above-mentioned social interactions lead to the following constraints:

```

EACH Year IS LEXICAL.
EACH Artist with AT MOST 1 Name.
EACH Artist with AT LEAST 1 Name.
EACH Artist IS IDENTIFIED BY (Name of Artist) AND (Year of birth of Artist).
EACH Artist born on AT LEAST 1 Year.
EACH Artist born on AT MOST 1 Year.

```

Those constraints have been agreed upon by the community because these constraints are either general (thus holding in every domain related to these concepts and relations) or because they need to be complied with by the different stakeholders to ensure proper interoperability. When committing the different applications to the ontology, stakeholders will often add extra lexons and enterprise-specific constraints. Either to express their view (i.e. their intended use of the concepts and relations) or to ensure proper annotation of their information. The next section describes how enterprise-specific knowledge aids stakeholders in annotating foreign keys to connect the pieces of information in their databases.

4.6 Committing to the Hybrid Ontology

Once there is a first version of the hybrid ontology, stakeholders will already be able to start annotating their information systems with the hybrid ontology by means of an application commitment.

Each application commitment is an ordered triple $\langle \sigma, \alpha, c \rangle$ where $\sigma \subset \Lambda$ is a selection of lexons from the lexon base, $\alpha : \Sigma \rightarrow T \cup R$ is a mapping called an annotation from the set Σ of application (information, system, database) symbols to terms and roles occurring in that selection, and c is a predicate over $T \cup R$ of that same selection expressed in a suitable first-order language.

Application commitments can even contain enterprise-specific lexons and applications specific constraints over all the lexons, capturing how the application uses the lexons represented by these lexons. A typical use for enterprise-specific lexons and constraints is to cope with identifiers used within one organization. The lexons and constraint inside this commitment originate from the community commitment to which this particular application is committing to.

Assuming that the community commitment is depicted graphically in Figure 4.6 (the context identifier is implied to refer to the Cultural Domain Community). Take for instance the lexon, $\langle \text{Cultural Domain Community}, \text{Artist}, \text{contributed to}, \text{with contributor}, \text{Work Of Art} \rangle$. From Figure 4.6, it shows that an artist is identified by its name and date of birth. Many artists can contribute to many works of art, and a work of art can be the result of the contribution of many artists. Assume there is an application with a relational database using artificial keys to identify artists and work of arts.

Figure 4.7 depicts a (E)ER diagram of a relational database. An application commitment for this application could look as follows: see Figure 4.8.

In the application commitment of Figure 4.8, the lexons and constraints agreed upon

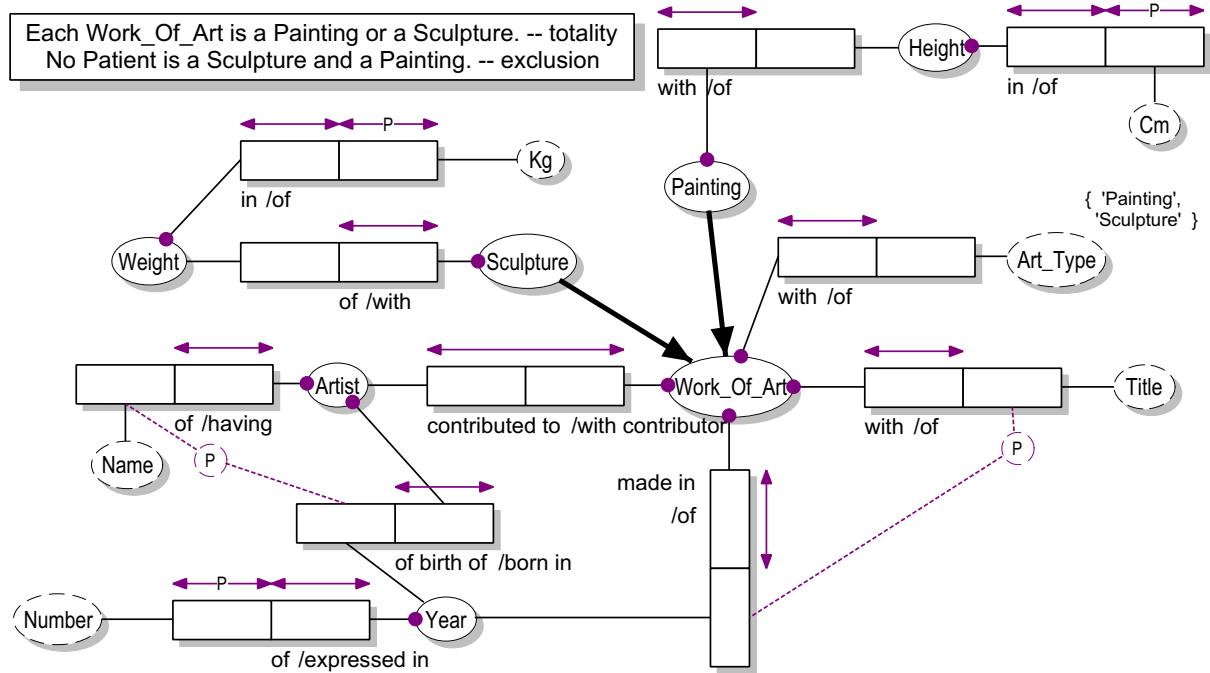


Figure 4.6: Example of a RM-reference complete ORM schema

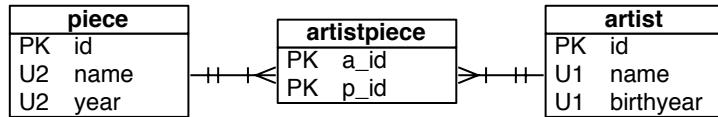


Figure 4.7: (E)ER diagram of a relational database.

by the stakeholder owning this information system are incorporated by referring to the community with [’Cultural Domain Community’]. For instance, the community agreed that all instances of Year have exactly one Number and that each Number identifies a Year.

The enterprise-specific lexons are not (necessarily) shared across applications, and the information annotated with that knowledge is not necessarily understood by other applications. Every instance is uniquely identified by their artist ID AID (cfr. lines with numbers 1, 2 and 3) and every instance of a work of art by their Work of Art ID WID (cfr. lines with numbers 4, 5 and 6). The foreign keys in the join table are annotated with those identifiers. With those annotations, it is for external applications now possible to ask questions such as LIST Name of Artist contributed to Work Of Art with Title ‘Bal au Moulin Rouge’ as the Ω -RIDL parser will be able to perform a join based on those IDs, populating the lexon \langle Cultural Domain Community, Artist, contributed to, with contributor, Work Of Art \rangle in the correct way.

The application commitments allow for the different information systems to be annotated and enable the exchange of information residing in those systems. With every (closer) approximation of the domain, the commitments will provide access to instances of concepts that can be used for defining and/or refining the definitions, lexons and constraints

```

BEGIN SELECTION
# Selection of the community.
['Cultural Domain Community']
# Enterprise-specific lexons
<'MyOrganization', Artist, with, of, AID>      # AID stands for Artist ID
<'MyOrganization', Work Of Art, with, of, WID> # WID stands for Work of Art ID
END SELECTION
BEGIN CONSTRAINTS
LINK('Cultural Domain Community', Artist, 'MyOrganization', Artist).
LINK('Cultural Domain Community', Work Of Art, 'MyOrganization', Work Of Art).
# List enterprise-specific constraints
EACH Artist with AT MOST 1 AID.          #(1)
EACH Artist with AT LEAST 1 AID.          #(2)
EACH AID of AT MOST 1 Artist.            #(3)
EACH Work Of Art with AT MOST 1 WID.    #(4)
EACH Work Of Art with AT LEAST 1 WID.   #(5)
EACH WID of AT MOST 1 Work Of Art.     #(6)
END CONSTRAINTS
BEGIN MAPPINGS
MAP 'Artist'.'name' ON Name of Artist.
MAP 'Artist'.'birthyear' ON Year of birth of Artist.
MAP 'Artist'.'id' ON AID of Artist.
MAP 'piece'.'name' ON Title of Work Of Art.
MAP 'piece'.'year' ON Year of Work Of Art.
MAP 'piece'.'id' ON WID of Work Of Art.
MAP 'artistpiece'.'a_id' ON AID of Artist contributed to Work Of Art.
MAP 'artistpiece'.'p_id' ON WID of Work Of Art with contributor Artist.
END MAPPINGS

```

Figure 4.8: Example Ω -RIDL commitment for describing the database shown in Figure 4.7.

in the hybrid ontology description. How these instances are used for the creation and refinement of definitions, lexons and constraints is described more fully in 6.

The instances accessible via the application commitments furthermore influence the internalization of the ontologies by providing extensional interpretations of the hybrid ontologies, which can be used by the community to check and test the validity of the hybrid ontologies.

4.7 Gloss-equivalences and Synonyms

At any point in time, two communities can agree that the glosses describing their respective terms and lexons actually refer to the same concept, even when glosses are not yet used to articulate terms or lexons. This can be achieved by asserting a gloss-equivalence between two glosses. Note that there are two special cases of gloss-equivalence: term equivalence occurs when two (or more) communities share the same terms, while commu-

nity equivalence occurs when different terms represent the same concept within a single community.

Gloss-equivalences can be created and removed with the following social processes:

- Request to add gloss-equivalence
- Request to remove gloss-equivalence

Note also that for every two community-term pairs whose glosses are identical or considered gloss-equivalent, there should be an agreement that the terms referring are referring to the same concept. This is a necessary property for a hybrid ontology to be glossary-consistent (cfr. Section 3.3.4). The inverse should not necessarily hold, two concepts can be deemed synonyms by the communities, but their glosses not equivalent.

Synonyms can be managed with the following two requests:

- Request to add synonym
- Request to remove synonym

The distinction between these two requests allows agreements to be made not only at the level of the glossary, but also at the level of the formal lexons. Although two (or more) communities may agree that terms are synonyms, they may disagree that their glosses are equivalent. The following example gives two glosses for the term color originating from two different communities. Their glosses (seem to) refer to the same concept, and therefore the community agrees that both community-term pairs are synonyms. However, their glosses are not equivalent as they describe different aspects of the same concept from different perspectives.

- $g_1(\gamma_{lcd}, \text{color}) = \text{"The quality of an object or substance with respect to light reflected by the object, usually determined visually by measurement of hue, saturation, and brightness of the reflected light; saturation or chroma; hue."}$ ⁸
- $g_1(\gamma_{physics}, \text{color}) = \text{"The electromagnetic radiation characterized by its wavelength (or frequency) and its intensity. When the wavelength is within the visible spectrum (the range of wavelengths humans can perceive, approximately from 390 nm to 750 nm), it is known as visible light."}$ ⁹

Unlike the previous requests, which happen at the level of one community. The requests involving gloss-equivalences and synonyms can and will occur across communities, making these inter-community social processes.

Again, when two communities $\gamma_1, \gamma_2 \in \Gamma$ agree that the glosses used to describe their terms $t_1, t_2 \in T$ are gloss-equivalent EQ_G , then it is not automatically implied that $ci(\gamma_1, t_1) \equiv_C ci(\gamma_2, t_2)$. Both terms *should* refer to the same concept; however, both agreements can

⁸Definition taken from <http://dictionary.reference.com/browse/color>, June 2011

⁹Definition taken from <http://en.wikipedia.org/wiki/Color>, June 2011

be established separately. Gloss-equivalences are on the level of the glossary whereas \equiv_C resides on the formal descriptions of the concepts (i.e. the lexons). It is necessary that for \equiv_C , the term must appear in a lexon. This implies that a term will only be in the community commitment if and only if that term plays at least one role (otherwise the term has no purpose for this community). If the term would end up in a taxonomy, then it plays the role of being the sub- or supertype of another term (e.g. with the roles “is a/subsumes”), hence satisfying the condition. Communities can start gradually building their glossary before formally describing their concepts. However, nothing should prevent the community for having agreements on the “sameness” of descriptions across or within their own community. If the definition would impose \equiv_C on the formal descriptions, the community first needs to agree on at least one lexon concerning that term.

The glossary-consistency principle then becomes a driver for agreement processes, as terms in two communities are supposed to be synonymous when the two communities agreed that two descriptions – used to articulate those terms – were referring to the same concept. Glossary-consistency will thus aid in validating previously established agreements.

4.8 Community and SIR Co-evolution

Currently, this chapter explained how a community starts the development of hybrid ontologies by first defining their semantic interoperability requirements, articulating the key terms and gradually constructing agreements on lexons, glosses, constraints, gloss-equivalences and synonyms. Communities and their semantic interoperability requirements are, however, not static. They are evolving and even co-evolving. In this section, we do not intend to provide a taxonomy of reasons why semantic interoperability requirements or a community changes, but briefly elaborate on this co-evolution.

With the addition of a new stakeholder, the community changed not only with the presence of a new member, but also with the addition of new ideas, new perspectives and possibly new requirements for the community. External forces, such as legislation changes, may also lead to the articulation of new requirements. The community’s constitution does not necessarily need to change for the semantic interoperability requirements to evolve, a community can come to the conclusion that the current approximation of the domain by the hybrid ontology description does not suffice to meet their needs even though it complied with the requirements. In some cases, a better understanding of the domain may lead a community to changing/adapting the original requirements through negotiation.

Changes to the constitution of a community or changes to the original semantic interoperability requirements inherently means the community has evolved. As described in the beginning of this chapter, a community is identified by its members and their semantic interoperability requirements, but an evolving community is not the same thing as a new community. This means that the insights of the evolved community could lead to a reconsideration of previously established agreements.

4.9 GOSPL w.r.t. other DOGMA Based Methods

Now that the method has been defined, the GOSPL method will be compared with other methods built around DOGMA. The first section describes the GOSPL framework and method with respect to DOGMA-MESS, while the second section takes a closer look at the differences with Business Semantics Management (BSM). Both DOGMA-MESS and BSM have been described in Chapter 2.

4.9.1 Compared with DOGMA(-MESS)

There are important differences between GOSPL and DOGMA-MESS [dMDM06] even though both methods start from the DOGMA ontology engineering framework. Below, a comparison between GOSPL and DOGMA-MESS is given.

- In GOSPL, glosses and community agreements are first-class citizen. In DOGMA-MESS the context identifiers refer to information sources from which the lexons are elicited from, whereas the context identifiers are pointers to communities in GOSPL.
- In [DdMM07], which described the context-dependency management in DOGMA-MESS, the articulation of a concept was defined as follows:

Definition 13 (Articulation in DOGMA-MESS)

$\text{artConcept}(\langle \gamma, t \rangle, c)$ maps a term $t \in T$ in a particular context $\gamma \in \Gamma$ onto a concept $c \in C$, provided $ct(\gamma, t)$ is not defined.

In this definition, the function ct corresponds with this thesis' function ci to map context-term¹⁰ pairs to a concept. [Jar05] also stated that every context-term pair is mapped onto one unique *ConceptID*. However, in GOSPL, the ci corresponds with the concept invoked by a certain term in a community and is in no way an explicit operation performed by the community (as done in [DdMM07]).

- Another important difference between DOGMA-MESS and hybrid ontology engineering is how concepts and glosses relate to each other. The Concept Definition Server (CDS) [DBSM04, DdM05, Jar06] in DOGMA-MESS is based on the popular lexicon WordNet [Fel98]. The CDS serves as a database in which one can query with a term and get a set of different meanings or concept definitions (called senses in WordNet) for that given term. That concept definition is identified unambiguously by a natural language description. In DOGMA-MESS, an *injective* mapping between concept identifiers and concept definitions is defined [DdMM07].

In hybrid ontology engineering, different communities can agree that their described terms refer to the same abstract concept, while employing different glosses. This

¹⁰Remember that the context identifiers are limited to communities in GOSPL.

implies that one concept can be related to many glosses. Also, given that in an ideal situation, every term described with the same gloss should refer to the same concept, this kind of constraint would be too restricting to be practical. From this follows that one gloss can be related to many concepts. The redefinition (on the use) of glosses and concepts allows for scalability (by letting communities diverge and converge over time) and accentuates the importance of capturing the community agreements in the ontology engineering process.

- In [DdMM07], three ontology evolution operators for changing the taxonomy were defined: $defineGenus(\langle\gamma, t_1\rangle, \langle\gamma, t_2\rangle)$ to insert a lexon stating that $\langle\gamma, t_2\rangle$ is a child of $\langle\gamma, t_1\rangle$, $pullUp(\langle\gamma, t_1\rangle, \langle\gamma, t_2\rangle)$ to pull up an already defined concept (including its children) higher in the taxonomy as a child of $\langle\gamma, t_2\rangle$ and $pullDown(\langle\gamma, t_1\rangle, \langle\gamma, t_2\rangle)$ to pull down an already defined concept (including its children) lower in the taxonomy as a child of $\langle\gamma, t_2\rangle$. All these operations can be achieved with the ontology operator resulting from a request to change type.
- The operations *introduceTerm* and *dropTerm* were introduced in [DL09]. The first was to enter a term t as a subtype of *Thing*, thus the lexon $\langle\gamma, t, \text{is a, subsumes, } Thing\rangle$ was added to the ontology. The latter was to remove a term from the ontology. Different strategies for handling the lexons around that term were proposed; the choice fell on: for every lexon around that term, add a lexon in which that term is replaced with the term of all its immediate children.

In Hybrid Ontology Engineering, the action “drop Term” is not proposed. Instead if a particular term needs to be removed from the ontology, all lexons involving that term should be removed. It is thus the responsibility of the community to build the ontology step by step.

- The *defineDiff* operation is used to add a series of non-taxonomical lexons to the ontology. This corresponds to the ontology evolution operator resulting from a request to add lexon. The difference however is: (i) that the *defineDiff* operator is used to add a series of lexons around a particular $\langle\gamma, t\rangle$ and (ii) that the $\langle\gamma, t\rangle$ has to be already be somewhere in the ontology. In [DdMM07], terms are first introduced via the *defineGenus* operator before they can be used in other lexons. Although not explicitly defined in [DdMM07], every ontology contains implicitly the term *Thing*, denoting the most general concept in the community and is the parent of all other concepts.
- Finally, [DdMM07] defined two operators *specialiseDiff* and *generaliseDiff* to replace (one of the) terms in a lexon with more specific or more general terms respectively. It is by definition a “destructive” operation in which the replaced lexon is removed from the ontology. GOSPL is restricted to operations for adding and removing lexons based on and after community agreement.

It is important to note one more difference. DOGMA-MESS and the above-mentioned operators were defined to support context-dependencies. This section will not go in too much depth on their nature, and the reader is referred to [DdMM07] for more details. But

the idea of context-dependencies was to capture ontology evolution by letting the stakeholder extend an ontology with their plausible lexons that represents their view (called a perspective). These resulted in several versions of the ontology and those perspectives needed to be unified. Users were able to immediately perform those operations on the ontology, and stored in a different branch for that stakeholder. In Hybrid Ontology Engineering, every operation is the result of a social process and is not performed until the community agrees with the operation. The ontology evolution operators are thus the result of a social process, whereas the social processes (meaning negotiation) happens after everyone has performed the ontology evolution operators to generate their perspective in DOGMA-MESS.

There are some differences between the notion of gloss and its role in formal ontology engineering proposed by in [Jar06]. Jarrar gave several guidelines for a proper gloss [Jar06]. A gloss should:

1. start with the principal or supertype of the concept being defined. In other words, a gloss should start with a term.
2. be written in the form of propositions.
3. focus on the distinguishing characteristics of the concept being defined.
4. be supportive (examples are encouraged).
5. be consistent with the formal axioms in the ontology.
6. be sufficient, clear and easy to understand by the members of the community.

The author did not give a formal definition for glosses, but one can deduce that glosses are given a context-term pair and reside in a namespace. Each gloss is identified by a URI, the question whether two glosses with the same lexical values are identical has not been addressed. In [Jar06], glosses are only used for high level reasoning among human stakeholders, the guidelines provided by the author are a means to facilitate the process and ensure quality of the glosses. The purpose of a gloss is not only to provide or catalog general information and comments about a concept, but also to render factual knowledge that is critical to understanding a concept, but that is unreasonable, implausible, or very difficult to formalize (e.g., the context in which one uses a particular concept). The glosses and concepts are given a URI and those URIs are used to implement an ontology in, for instance, OWL. The URIs also contain the source of the gloss. The problem, however, is that if the community agrees on adopting another gloss from another source, that URI should change. This brings down the scalability of ontology versioning. In GOSPL, the lexons (the formal part) are transformed into other formalisms and the glosses are used to provide documentation and links between the formal parts.

4.9.2 Compared with Business Semantics Management

Business Semantics Management (BSM) [DCM10] draws from best practices in ontology management [DLM08] and ontology evolution [HDdS08]. The representation of business semantics was originally based on the DOGMA approach. BSM adopts the Semantics of Business Vocabulary and Business Rules (SBVR) [OMG09] to capture concepts and their relationships in fact types. Like DOGMA, SBVR is a fact-oriented modeling approach.

As DOGMA's lexons and constraints are fully compatible with SBVR (supported by OMG), BSM recently adopted SBVR for representing the business domain and rules. SBVR does provide constructs that were not available in the DOGMA frame-work, such as support for unary fact types to represent characteristics of a business entity (e.g. Project is terminated).

BSM consists of two complementary cycles: *semantic reconciliation* and *semantic application* (see Figure 4.9) that each groups several activities:

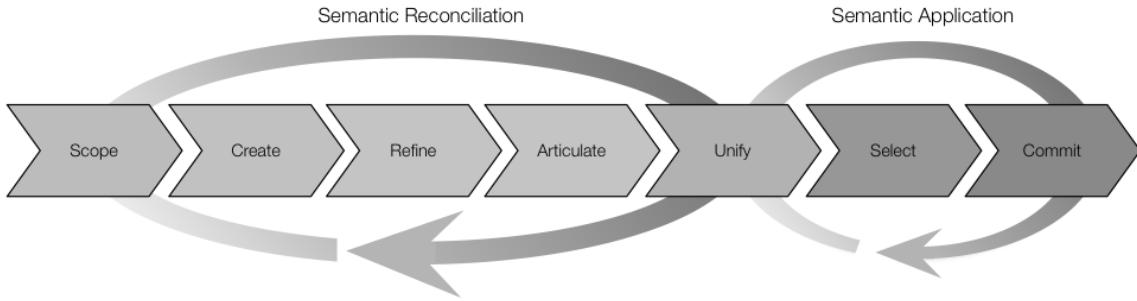


Figure 4.9: Processes in the Business Semantics Management method

- *Semantic Reconciliation* is the first cycle of the method. In this phase, business semantics are modeled by extracting, refining, articulating and consolidating lexons from existing sources such as natural language descriptions, existing metadata, etc. Ultimately, this results in several consolidated language-neutral semantic patterns that are articulated with glosses (e.g. WordNet [Fel98] word senses). These patterns are reusable for constructing various semantic applications. This process is supported by the Business Semantics Glossary.
- *Semantic Application* is the second cycle. During this cycle, existing information sources and services are committed to a selection of lexons, as explained earlier. In other words, a commitment creates a bidirectional link between the existing data sources and services and the business semantics that describe the information assets of an organization. The existing data itself is not moved nor touched.

The first cycle, semantic reconciliation, is supported by the Business Semantics Glossary (BSG) shown in Figure 4.10 (taken from [DDS⁺11]). This figure shows a screenshot of the term "Project" (within the "Project" vocabulary of "CERIF" speech community that is part of the "FRIS" semantic community). The software is currently deployed at EWI for managing business semantics of CERIF terms. A term (here "Project") can be defined using one or more attributes such as definitions, examples, fact types, rules sets, categorization schemas (partly shown in taxonomy), and finally milestones for the life cycle. "Project" in this case is a subtype of "Thing" and has two subtypes: "large academic project" and "small industrial project". Re governance: in the top-right corner is indicated which member in the community (here "Pieter De Leenheer") carries the role of "steward", who is ultimately accountable for this term. The status "candidate"

indicates that the term is not yet fully articulated: in this case “Project” only 37.5%. This percentage is automatically calculated based on the articulation tasks that have to be performed according to the BSM method. Tasks are related to defining attributes and are distributed among stakeholders and orchestrated using workflows.

The screenshot shows the Collibra BSG interface for the 'Project' entity. At the top, there is a navigation bar with tabs for 'Roles', 'Tasks', 'Tools', and 'Workflows'. On the right side of the header, there are links for 'Administrator', 'Dashboard', 'Glossary', and 'Settings'. A search bar labeled 'Search...' is also present.

The main content area displays the 'Project' entity details. It includes:

- Information:** Status: Candidate, Concept Type: Object Type, Steward: Pieter De Leenheer, Articulated: 37.5%.
- Attributes:** A section to 'Select an attribute...' with an 'Add Attribute' button.
- Relations:** A taxonomy tree under 'Thing' with 'Project' as the root node, which further branches into 'Large Academic Project', 'Small Industrial Project', and 'Specialized Term'.
- Stakeholders:** A list of users: Felix Van De Maele, Administrator, and Geert Van Grootel.

Below these sections, there are several input fields and tables:

- Definitions:** A text input field containing the definition: "Planned set of interrelated tasks to be executed over a fixed period and within certain cost and other limitations." with edit and delete icons.
- Examples:** A text input field containing "Large Hadron Collider" with an edit icon.
- Fact Types:** A table showing fact types for 'Project':

	Fact Type	(Candidate)	Edit	Search	Delete
<input type="checkbox"/>	Project has Title	(Candidate)	edit	search	X
<input type="checkbox"/>	Project funded by Funding Programme	(Candidate)	edit	search	X
<input type="checkbox"/>	Project executed by Organisation	(Candidate)	edit	search	X
<input type="checkbox"/>	Project has Budget	(Candidate)	edit	search	X
<input type="checkbox"/>	Project has Duration	(Candidate)	edit	search	X
<input type="checkbox"/>	Project ends on Date	(Candidate)	edit	search	X
<input type="checkbox"/>	Project starts on Date	(Candidate)	edit	search	X
<input type="checkbox"/>	Project described by Discipline Code	(Candidate)	edit	search	X
- General Rule Set:** A table showing rule sets for 'Project':

	Rule Set Description	Edit	Search
<input type="checkbox"/>	Project described by at most 3 Discipline Code.	edit	X
<input type="checkbox"/>	Project executed by at least 1 Organisation.	edit	X
<input type="checkbox"/>	Project ends on exactly 1 Date.	edit	X
<input type="checkbox"/>	Project has at least 1 Title.	edit	X
<input type="checkbox"/>	Project starts on exactly 1 Date.	edit	X

Figure 4.10: Screenshot of Collibra’s BSG supporting the semantic reconciliation process of the BSM method by providing domain experts means to enter simple key fact types in natural language, natural language definitions of fact types and terms in those fact types as well as constraints (in BSG called “rule sets”).

The *scope* activity is used to set out – during an iteration of the method – the scoped terms that are needed to establish semantic interoperability. During the *create* activity, every scoped term is syntactically defined as well as rules for these terms and the roles they play in their fact types are created. During this phase, inspiration can be drawn from existing sources (manuals, users, standards, etc.). While *refining* fact types and constraints that were created during the creation activity are refined so that they are understandable to both business and technology. The refined fact types and constraints should be (i) correct,

(ii) useful, (iii) reusable, and (iv) elegant. In the articulation process, informal meaning descriptions are created as extra documentation. These descriptions include definitions and examples and can serve as anchoring points when stakeholders have used different terms for the same concepts (i.e. detecting synonyms). Where available, already existing descriptions can be used to speed up the process and facilitate reuse.

The main difference with GOSPL is that BSM is not driven by the glosses. Instead, a community first needs to create and refine the formal descriptions. Thus agreements on the formal descriptions are not driven by alignment processes, even though key terms have to be defined.

Users with the appropriate rights, can change the formal description. Additions have a flag of their status (e.g. candidate, accepted, etc.). Deletions, however, are immediately carried out. The versioning behind BSG allows for easy rollbacks. They can possibly, as mentioned in the state-of-the-art, render discussion or agreement processes less efficient as ontology evolution is not the immediate result of discussion. One also needs to examine the status of each part of the ontology to see which parts have been accepted or deleted. This is not so much a problem as the documentation actually imply that with every iteration of the method, a new version of the ontology is actually “compiled and published” into something that is needed for the ontology project (e.g. and XSD on the Web).

4.10 Conclusions

This chapter answered the second research question: “How can hybrid ontology construction be supported in a necessarily complex collaborative setting?” and presented a method for hybrid ontology engineering called GOSPL, which stands for Grounding Ontologies with Social Processes and natural Language. The method is built on top of the hybrid ontology engineering framework introduced earlier in this thesis and was designed so that the ontology engineering process is driven by the informal concept descriptions by appropriately orchestrating the social processes defined in Chapter 3. The ontology ownership belongs to the community, which is a consequence of ontology evolution being defined as the result of incremental agreements processes within that community. Each of the different processes in GOSPL was exemplified with a running example stemming from a use case. Also a comparison with other methods built on top of DOGMA was presented.

Now that a method has been presented, one can ask how different parts or processes of this method can be (semi-)automated. Two important artifacts within this framework are the glosses and the application commitments. In subsequent Chapters, the use of both glosses and commitments to steer or even start social processes within a community will be examined.

Chapter 5

Evolving Glosses

A gloss is a (brief) description of a concept and thus helps in providing the meaning of a term or lexon. The purpose of a gloss is not to provide or catalog general information and concepts about a concept, as conventional dictionaries and encyclopedias do, but is supposed to render factual knowledge that is critical to understanding a concept in ontology engineering [Jar05].

A gloss is composed of one or more sentences constructed with the community's usual alphabet. Those sentences have to be themselves human-interpretable in order for the gloss to become understandable. \mathcal{S} is used to denote the set of all possible sentences that can be constructed with those alphabets. Whether this set contains sentences (or parts thereof) that are valid syntax- and grammar wise is "ignored" in this thesis. As the community will choose and discuss the elements of this set used for constructing a gloss, they will make sure that what is chosen makes sense (at least for this community).

Use of the word "truth" will be avoided in this chapter, since we are not in a formal logical context. "Truth" will be almost exclusively used in the context of the mapping between an ontology and application symbols (the formal semantics). Occasionally and carefully, the word will be used for addressing the agreement on validity assumed to exist in the community. In other words, truth is relative to the community; if there is an agreement on something, that something is assumed to be valid.

Every part of a gloss should contribute to a better understanding of the concept described. As a consequence, some of these parts should correspond with parts of the formal description of that concept. In other words, as the glosses evolve, so should the lexons and constraints.

This chapter will thus present how the evolution of glosses used and agreed upon by the community has an impact on the hybrid ontology description presented in Chapter 3. Key for this process will be to describe how different parts of a gloss relate to each other.

5.1 Discrete Gloss Evolution

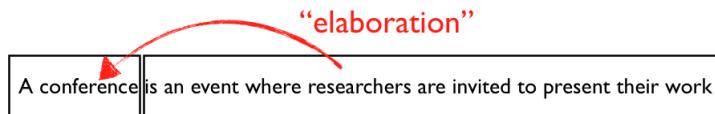
The articulation of a term or a lexon with a gloss can and will evolve over time. Those changes happen for a reason, which is then discussed by the community. This reason is captured by the motivation of the change and the communities' discussion. How the gloss changes, can be formalized. There are two types of gloss updates. The first is a complete change of a gloss. In this thesis, this kind of update is deemed to happen only accidentally; as such a change would imply that the community – as a whole – misinterpreted the term

being described in the context of that community (and their goals).

The second type of change is a (more gradual) refinement of the gloss. Glosses are composed of one or more sentences. Sentences or parts of sentences can be added or removed. The sentences that have been added or removed serve (or served in the case of the latter) a particular *modality* for that gloss. This modality captures and describes how the gloss and those sentences were related. One can define many such modalities. In the context of discourse relation, these modalities correspond with *Discourse Segment Purposes* [GS86]. In this thesis, the set of modalities is referred to by Θ .

Example 1

In the following example of a modality, the second span has the modality of “elaborating” on what has been stated in the first span.

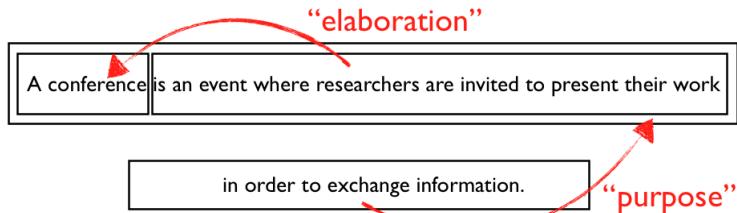


Gloss evolution is a mapping from a set of glosses to another set of glosses that is the result of a community applying a modality in Θ to add or remove parts of a gloss. The *linguistic amalgamation operators* to add or remove (a part of) a sentence from a gloss are defined as:

- $\oplus : \text{Gloss} \times \Theta \times \mathcal{S} \rightarrow \text{Gloss}$ for adding an element of \mathcal{S} to a gloss
- $\ominus : \text{Gloss} \times \Theta \times \mathcal{S} \rightarrow \text{Gloss}$ for removing an element of \mathcal{S} from a gloss

Example 2

Starting from the previous example, a gloss-amalgamation operator can be applied on that gloss to add a third span providing the modality of “purpose” on the whole.



For the sake of presentation, the exact locations where sentences would be added (before, after or somewhere in another sentence) or which part of a sentence is removed are not displayed. Those are assumed to be additional parameters of above-mentioned operators.

5.2 Gloss Evolution Modalities

This section will provide a proposal for the elements in Θ . Many approaches are possible, we choose to draw inspiration from discourse theory, and more concretely Rhetorical Structure Theory [MT88]. Elements from RST are furthermore refined and complemented with elements introduced specifically for GOSPL. One such element specific to GOSPL is adopting a gloss via *term-adoption* (See Definition 9 in Chapter 3), where one community explicitly states to adopt the gloss of another community.

The goal is not to provide an exhaustive list, as it can vary depending on the type of community or even the type of language used. The framework is therefore defined in such a way that new elements can be introduced to Θ .

A distinction is made between elements that affect the community commitment and elements that merely serve to provide additional text to (better) understand the concept described.

The application of some gloss-amalgamation operators might imply the introduction of lexons, constraints and even instances in the community commitment. Lexons and constraints are useful for building up the lexon base and community commitments. The instances are useful to validate the constraints explicitly agreed upon by the community.

5.2.1 Drawing Inspiration from Rhetorical Structure Theory

For the elements of Θ , inspiration is – as already stated above – first drawn from RST. RST was originally developed as part of research on computer-based text generation. RST was intended to describe texts by means of two types of “relations”, each at a different level. The first is the “nucleus-satellite relation” and is the most frequent structural pattern. It involves two (usually adjacent) spans¹ which are related in such a way that one of them has a specific role relative to the other. The other type is “multinuclear relations”, grouping a set of nuclei².

In RST, somewhat unfortunately the roles that spans play on other spans are confusingly labeled “relations”, a term obviously not suitable for computer science texts, such as this thesis. The authors of RST are linguists and the “relations” they propose actually refer to “functionalities”, their purpose. Therefore these “functionalities” will be called modalities, as the elements in Θ . This will not pose a problem as we explicitly state which of the RST modalities will be adopted and included in Θ .

RST thus allows one to describe how two segments of discourse are connected to one another. With elements of the first type, the nucleus (N) is part of the text on which the satellite (S) will play a particular role.

¹A span is a part of a sentence or text.

²<http://www.sfu.ca/rst/>, on March 2012

Example 3

Consider the sentence: “*Employees are urged to complete new beneficiary designation forms for retirement or life insurance benefits whenever there is a change in marital or family status.*” The part “*Employees are urged to complete new beneficiary designation forms for retirement or life insurance benefits*” is the *nucleus*. The part “*whenever there is a change in marital or family status*” is the *satellite* and expresses a *condition*.

Glosses need to concisely describe terms or lexons employed by the community. Their (implicit) “meaning” also has to be agreed upon by that same community. Opinions or statements in favor of a particular gloss are part of the discussion leading to an agreement, and not part of a gloss itself. As RST provides modalities with sometimes a subjective nature (e.g. the *antithesis* that describes ideas favored by the author), only a subset of these modalities that is deemed relevant for gloss evolution will be presented. This is in line with the guidelines on the construction of glosses given in [Jar06]: a gloss should (i) start with the term of the principal or supertype of the concept being defined; (ii) be written in the form of propositions; (iii) focus on the distinguishing characteristics of the concept being defined; (iv) be supportive (examples are encouraged); (v) be consistent with the formal axioms in the ontology and (vi) be sufficient, clear and easy to understand by members of the community.

The modalities provided by RST that were not taken into consideration are:

1. Modalities for expressing opinions: antithesis, concession, and justify.
2. Modalities aimed at enabling the reader in undertaking actions: enablement.
3. Modalities at interpreting and evaluating text: restatement, interpretation and evaluation.
4. Modalities concerned at relating information with causes and effects: non-volitional cause, non-volitional result, volitional cause and volitional result.

In the following examples, the community employing the term is implied and denoted with γ . Throughout the descriptions, N will be used to refer to a nucleus and S to refer to a satellite.

- With **background**, S is used to facilitate the understanding of N. S thus adds additional background information for understanding the information inside the gloss N by the community γ as well as other communities who wish to negotiate the sameness of concepts across communities.

Example 4

Given the gloss g for the term “color”: “*The visual perceptual property corresponding in humans to the categories called RGB and others. Color derives from the spectrum of light interacting in the eye with the spectral sensitivities*

of the light receptors.” and the sentence $s = “(distribution\ of\ light\ energy\ versus\ wavelength)”$, the application of $g \oplus Background(s)$ results in *“The visual perceptual property corresponding in humans to the categories called RGB and others. Color derives from the spectrum of light (distribution of light energy versus wavelength) interacting in the eye with the spectral sensitivities of the light receptors.”*

- **Circumstance** is the modality that denotes that a part of a gloss S sets the framework for interpreting N, another part of that same gloss. It is different from *background* in that both N and S are about a single situation.

Example 5

Given gloss g for the term “opportunity cost” as “The cost of passing up the next best choice when making a decision”. The circumstance of “The cost of passing up the next set choice” is “when making a decision.”

- A **condition** is a modality used for stating that the truth-value accorded to the proposition in a part N of the gloss depends on the truth-value accorded to the proposition in another part S of that same gloss.

Example 6

S = “that are in order” expresses the condition for a set to be a sequence in the following gloss for “sequence”: “*A sequence is a set of things (usually numbers) that are in order. If the sequence goes on forever it is called an infinite sequence, otherwise it is a finite sequence.*”

The **Unconditional** is a modality that relates S and N in such a way that the truth-value of the proposition in N does not depend on the truth-value of the proposition in S. Such sentences typically start with “even if”, “even when”, etc.

Otherwise is a modality used to state that if the truth-value of the proposition in N is not true, the truth-value of the proposition in S will be. This modality is typically used to create so-called *if-then-else* statements. Such statements are for instance useful to describe the conditions to classify an instance of a concept as an instance of one of that concept’s subtypes in the type hierarchy. In the example above, this modality is used to state that a sequence is a finite sequence when the sequence does not go on forever.

Unless expresses the modality that the truth-value of the proposition in N is true only when the truth-value of the proposition in S is false. As the name of this modality implies, such satellites typically start with – as the name implies – “unless”.

- **Elaboration** denotes modality of adding information in a part of the gloss S to already available information in another part of the gloss N. It is different from the background modality in that the additional information provides further detail on the concept described in the gloss rather than providing a framework for better understanding the remainder of the gloss.

Example 7

Given the following gloss g for the term “color”: “*The visual perceptual property corresponding in humans to the categories called RGB and others*”. One would want to provide additional information. For instance, the addition of sentence $s = \text{“Color derives from the spectrum of light interacting in the eye with the spectral sensitivities of the light receptors.”} \in \mathcal{S}$. The following operation $g \oplus \text{Elaboration}(s)$ would result in the following gloss $g' = \text{“The visual perceptual property corresponding in humans to the categories called RGB and others. Color derives from the spectrum of light interacting in the eye with the spectral sensitivities of the light receptors.”}$

RST described some specific types of elaboration modalities:

1. **Instantiation.** N describes the abstract concept, and S presents an instance.
2. **Meronymy.** N presents the whole, and S provides a part.
3. **Step.** N describes a process in which S is a step.
4. **Attribute.** N describes an object, and S presents an attribute of that object.
5. **Specification.** N describes a “supertype”, and S provides a subtype.
6. **Generalization.** N describes a “subtype”, and S presents supertype.
7. **Membership.** N describes a set, and S presents an element of that set.

Note that those specific types of elaboration modalities are not exhaustive. In fact, a special case concerning the identification of the concept described will be proposed in the next section.

All but *Membership* were taken into account for this thesis. *Membership* relates a member in a satellite to the set described in the nucleus. In the context of hybrid ontology engineering, terms and lexons are sets. The elements of these sets are examples of the concepts represented by these terms and lexons. As the goal of a gloss is to describe a concept, the **instantiation** modality already covers relating examples as members of a set.

- **Evidence** Evidence is a sentence S that supports a claim N . In this case, the claim is expressed by the gloss. *Examples* are typically used as evidence; they support the definition contained in the gloss. Examples, however, are already present as a special case of the elaboration modality (instantiation). The type of evidence is therefore restricted to information supporting a claim.

Another type of evidence are *analogies*. By drawing comparisons between two different things, one can help illustrate or clarify one of the two. In this case,

comparisons are made to show the reader similarities. When comparisons are made to show the differences, a **contrast** (see later) is made. Below an example of the use of evidence³.

Example 8

The gloss $g = \text{"The cost of passing up the next best choice when making a decision. For example, if an asset such as capital is used for one purpose, the opportunity cost is the value of the next best purpose the asset could have been used for. Opportunity cost analysis is an important part of a company's decision-making processes, but is not treated as an actual cost in any financial statement."}$ illustrates the use of an example to support the definition of “opportunity cost”.

- **Means** is a modality used by a community to denote the sentence S presenting an instrument used for achieving the concept described in sentence N.

Example 9

Given the gloss $g = \text{"An algorithm is a process or set of rules to be followed in calculations or other problem-solving operations"}$ for the term “algorithm”, adding the means $s = \text{“; esp. by a computer.”}$ to this gloss with $g \oplus \text{Means}(s)$ results in $\text{"An algorithm is a process or set of rules to be followed in calculations or other problem-solving operations, esp. by a computer."}$

- With the **Preparation** modality, the sentence S is used to prepare the reader to expect and interpret the sentence in N. According to the guidelines of [Jar06], a gloss should start with the term or its supertype. This corresponds with the preparation, as the term will give an idea to the reader what the gloss’ subject will be. However, all sentences that aid the reader orienting his thoughts are considered preparations. This modality is different from the background modality in that it prepares the reader for what to expect, and the background modality is used to provide information needed to understand the rest of the gloss.

Example 10

For example, given the following gloss g for the term “Color” $\text{"The visual perceptual property corresponding in humans to the categories called RGB and others."}$ with $s = \text{“Color is”}$, the community can thus apply $g \oplus \text{Preparation}(s)$ returning $\text{"Color is the visual perceptual property corresponding in humans to the categories called RGB and others."}$

³Gloss taken from http://www.investorwords.com/3470/opportunity_cost.html, retrieved on February 2012.

- The **Purpose** modality is used to describe that an activity in N needs to be initiated in order to achieve what is described in S. In other words, S describes the purpose of doing the activity described in N.

Example 11

For instance, in the following gloss for “cardiopulmonary resuscitation” given below, the sentence S “*for restoring normal heartbeat and breathing to victims of heart failure, drowning, etc.*” expresses the purpose in the type of emergency medical procedure N: “*Cardiopulmonary resuscitation is an emergency medical procedure.*”

“Cardiopulmonary resuscitation is an emergency medical procedure for restoring normal heartbeat and breathing to victims of heart failure, drowning, etc.”

- In RST, *Solutionhood* describes the modality of N presenting a solution to the problem described in S. In other words, the concept described in N provides a solution for the problem described in S. Solutionhood is defined as: “*A solutionhood relation is an interpropositional relation in which a proposition(s) is presented as an answer or remedy for a problem, such as one of the following, communicated in another proposition(s).*⁴”

In RST most modalities describe the role of sentence S on sentence N. Solutionhood, however, was described from the perspective of N. In other words, the role N plays on S. To have the direction of modalities consistent, solutionhood was renamed to **problem-for**. This way, the form of each modality in Θ from RST is defined in a consistent way.

Example 12

In the following gloss, the problem is a dispute or conflict and the settlement is a solution: “*A settlement is an official agreement intended to resolve a dispute or conflict.*”

All the elements of Θ described so far involve a satellite playing a role on a nucleus. However, RST also describes modalities that relate multiple nuclei.

All but two of these types are taken into account. The multinuclear restatement was not considered, as for the same reason as the restatement modality; it aids at interpreting another sentence without providing additional information. Also the joint modality was not considered, as it is used to “glue” two pieces of text that are not related. As a gloss needs to present a brief description of the described term, all parts in that gloss need to be relevant.

The modalities involving multiple nuclei used in this thesis are:

⁴<http://sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsASolutionhoodRelation.htm>

- **Conjunction.** The items are conjoined to form a unit in which each item plays a comparable role. Items can be combined with words such as “and” and “nor”.
- **Disjunction.** An item presents an alternative for the other(s). The disjunction is not necessarily an exclusive disjunction.
- **Contrast** is used for at most two nuclei. The two are understood to be similar (or the same) in many respects and to differ in a few respects, and both are compared with respect to those differences.
- **List** for linking items are comparable to each other and **sequence** for linking items with a logical succession, e.g. steps to perform a task.

The multi-nuclei modalities are included in Θ so that one sentence plays a particular role on another sentence. This means that for the multi-nuclei modalities that can have more than two nuclei, the sentences have to be nested with their modality. In other words, if a conjunction A and B and C were to hold, this would have to be rewritten as $(\text{and } A (\text{and } B C))$. Later on will be described how the community members will have access to syntactic sugar taking care of this.

5.2.2 Other Gloss Evolution Modalities

RST provides a foundation for choosing gloss-evolution modalities. However, the hybrid ontology engineering framework also needs to treat certain processes that result in gloss evolution not covered by these modalities.

One example of a social interaction evolving glosses not covered by the RST modalities are term-adoptions (See Definition 9 in Chapter 3). Given two communities $\gamma_1, \gamma_2 \in \Gamma$ and their respective terms $t_1, t_2 \in T$ and γ_2 has articulated t_2 with a gloss g , community γ_1 is able to adopt g for describing t_1 . It is obvious that this operation evolves the gloss for $\langle \gamma_1, t_1 \rangle$. The implications of the adopted gloss on the hybrid ontology remain within the original community, but are known to the adopting community as this operator links both terms.

Chapter 3 explained the importance of the identification of the set of attributes that uniquely and totally identify instances of a concept in database management and knowledge representation in IT. As a natural consequence, semantic interoperability between autonomously developed information systems should take into account these constraints to properly identify instances in each of those systems. RST does not provide modalities to capture this attribute. A special kind of modality is thus introduced: **identifies**. With this modality, a community relates a sentence S to a sentence N such that S provides information on how to identify instances of the concept described in N . One will then later be able to identify these attributes – by means of simple NLP techniques, see Section 5.2.3 – and start the necessary social processes to agree on those lexons and constraints (unique, total and identifying).

5.2.3 A Grammar to Structure and Process Glosses

In order to illustrate the mechanism for annotating glosses, we defined a simple grammar to create a gloss modality parser. The grammar – developed with the ANTLR Parser Generator⁵ – is shown below. The parsers takes appropriate measures depending on the modality encountered. The different measures are presented in a Table at the end of this section. Note that white spaces are ignored. In modalities involving a nucleus and a satellite, the satellite provides additional information with a particular “purpose”. The satellite, however, may appear on either side of the nucleus. The grammar expresses this with `modalityDirection`.

```
// PARSER RULES
// Top rule, starting point of span
rule: span ;
span: sentence | modality | multinuclei ;
// Simple sentence: a unstructured string
sentence: STRING ;
// Application of a modality
modality: '(' span modalityDirection span ')';
modalityDirection: ('<- ' modalityName) | (modalityName '>-') ;
modalityName: LITERAL;
// Application of a multinuclei
multinuclei: '{' LITERAL (span)+ '}';
// LEXER RULES
LITERAL: ('A'...'Z')+;                                // Words in capital letters
STRING: '"' ACTUALSTRING '"';                         // Quoted Strings
fragment ACTUALSTRING: ~('\'|'"')*;
```

A possible annotation of the gloss “*A planet, in astronomy, is one of a class of celestial bodies that orbit stars. Examples are Mercury, Mars and Earth.*” to parse with this grammar is shown in Example 13.

Example 13

```
((("A planet," <- BACKGROUND "in astronomy,")
  <- GENERALIZATION ("is one of a class of celestial bodies"
    <- ELABORATE "that orbit stars.")
    <- INSTANTIATION
      {CONJUNCTION "Examples are Mercury," "Mars" "and Earth."})
```

One is now able to parse structured glosses and to some extent reason on this structure with this grammar. In the next section, this structure will be examined to support glossary and community commitment co-evolution. The grammar is defined in such a

⁵ANTLR Parser Generator: <http://www.antlr.org/>

way that the possible gloss-evolution modalities are not pre-determined. Software agents implementing this grammar can thus implement different strategies depending on the label of a modality encountered.

5.3 Glossary and Commitment Co-evolution

We are now ready to introduce one of the key methodological principles of our approach, which links the management of glosses to the engineering of ontologies within a given community. This principle is called *co-evolution*.

The elements in Θ and the amalgamation operators enable us to implement the support for the so-called co-evolution of communities, ontologies and glossaries. Changes in the requirements of the community are reflected in the formal part of the ontology and possibly require the refinement of the glosses based on the newly defined gloss evolution operators. In turn, these changes might start a series of social processes for the formal part of the hybrid ontology to reflect those changes accordingly. At any time, changes in both the lexon base and the glossary will influence the communities' next decisions. Some elements of Θ provide only additional information to the community for understanding the gloss. Other elements however, can and *should* have an impact on the hybrid ontology. These elements influence the hybrid ontology at three levels:

1. The introduction of one or more *pre-lexons* in the Lexon Base.
2. The introduction of *pre-constraints*.
3. The introduction elements in the population of a term or a lexon.

Pre-lexons are “raw” lexons that have not yet been refined by the community (e.g. proper stemming of verbs in roles, the introduction of the co-role, etc.). Some gloss-evolution modalities result in lexons of which roles, concepts or generalization of concepts are known. For instance, when describing a specialization of a concept, the roles *is a* / *subsumes* – interpreted as the taxonomic relation – will be proposed. *Pre-constraints* are constraints expressed in terms of the pre-lexons.

Example 14

Given some community $\gamma \in \Gamma$ wishing to articulate the term “Car” with a gloss, the application of the following generalization (a type of elaboration): “A car” \oplus *Generalization*(“is a road vehicle”) results in the following pre-lexon:

- $\langle \gamma, \text{Car}, \underline{\text{is a}}, \underline{\text{subsumes}}, \text{road vehicle} \rangle$

The roles are underlined as they are pre-filled and have a special interpretation. Nothing prevents the community to refine this pre-lexon and change its role labels. But as the gloss has evolved with a generalization, one would expect that this would reflect with the addition of a taxonomic relation in the hybrid ontology.

Once refined, the execution of this gloss evolution triggers social processes for adding this lexon.

Example 15

Taking the following gloss g for “Car” in some community $\gamma \in \Gamma$: “*A car is a road vehicle*”. One can elaborate on this term by adding the following sentence s “*powered by an internal combustion engine and able to carry a small number of people.*”. The sentence s is actually the result of a conjunction $s' \oplus \text{Conjunction}(s'')$ where $s' =$ “*powered by an internal combustion engine*” and $s'' =$ “*and able to carry a small number of people.*”

Elaborating g with the conjunction contained in s with $g \oplus \text{Elaboration}(s)$ results in the following pre-lexons:

- $\langle \gamma, \text{Car}, \text{powered by}, ., \text{an internal combustion engine} \rangle$
- $\langle \gamma, \text{Car}, \text{able to carry}, ., \text{a small number of people} \rangle$

These pre-lexons have to be refined by the community by starting social processes adding new lexons in the hybrid ontology (cf. the social processes defined in 3). Either a community member proposes an initial refinement of the lexon or the community will immediately discuss how to refine the pre-lexons.

- $\langle \gamma, \text{Car}, \text{powered by}, \text{powering}, \text{Internal Combustion Engine} \rangle$
- $\langle \gamma, \text{Car}, \text{carrying}, \text{carried by}, \text{Group} \rangle$

Instances can be elicited from glosses and used as a test population in hybrid ontology engineering via requests to add examples (see Chapter 3).

Example 16

Given a gloss $g =$ “*A planet, in astronomy, is one of a class of celestial bodies that orbit stars.*” for the term “planet”, one can elaborate this gloss by giving examples. $s =$ “*Examples are Mercury, Mars and Earth.*”

$g \oplus \text{Instantiation}(s) =$ “*A planet, in astronomy, is one of a class of celestial bodies that orbit stars. Examples are Mercury, Mars and Earth.*”

The instances of planets are proposed to be taken into account, and will – once accepted – serve as a test population for ontology engineering. $\text{Population}(\gamma, \text{Planet}) = \{\text{Mercury, Mars, Earth}\}$

What is particularly interesting about instances is that they refer to elements in the community’s domain and therefore the information about these instances should fit in one of that term’s reference structures. In other words, the labels used to refer to actual instances of a concept should be attributes that uniquely and totally identify those instances that have agreed upon by the community. A member of the community will thus be asked – as will be shown later on – to select either existing total, unique identifying lexons in which the instances can be placed, or propose a set if none (adequate) are found.

There are several ways to discover constraints in glosses. Gloss-evolution modalities such as the **conditional** can express subset constraints between roles of lexons. Others, such as the elaboration, contain hints on frequency or totality constraints within a pre-lexon.

Example 17

Given a gloss $g = "A \text{ proposal results in a project}"$ for the term “Proposal” in a certain community $\gamma \in \Gamma$ and assuming that the lexon $\langle \gamma, \text{Proposal}, \text{results in, result of, Project} \rangle$ is already present, by adding a condition by applying $g \oplus \text{Condition}$ (“when the proposal is accepted by the review board.”), the following pre-lexon and subset constraint are elicited:

- $\langle \gamma, \text{Proposal, is accepted by, ., review board} \rangle$
- Subset constraint from “is accepted by” to “results in”.

The community can refine the pre-lexon as well as the subset constraint and trigger social processes to accept these in the formal part of the hybrid ontology.

Eliciting and refining pre-lexons, -constraints and instances are community processes. To aid the community, these processes can be partly derived by a software agent that applies natural language processing techniques and reason over the modalities the community has annotated their glosses with.

While extracting the pre-lexons for a modality f_1 , the detection of terms in the nucleus will often follow the same steps. In essence, if the nucleus is a sentence, then noun-extraction will be applied on this sentence. However, if the satellite is a modality f_2 , then the nucleus of that modality should be examined. If the modality f_1 would be applicable to the nucleus of this modality f_2 , then the structure of this gloss would look different.

This becomes clear if the application of elements of Θ for glosses are visualized as a tree (see Figure 5.1). On the left, the wheels are part of the car, which is the nucleus of the modality. On the right, however, the **meronymy** modality is pointing to the road vehicle. It is up to the community to ensure that the structure of the gloss corresponds makes sense.

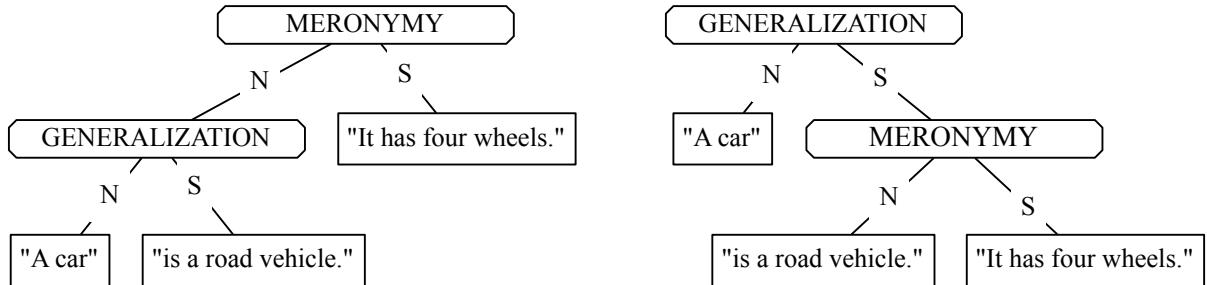


Figure 5.1: Tree representation of differently structured glosses that appear the same.

When the gloss is defined by means of prepositions, those prepositions can easily be

structured according to these rules. In case of a multi-nucleic modality, the terms will be detected in each element of that relationship. The multi-nucleic modalities are mainly used to illicit a series of pre-lexons that serve a particular modality. For instance, in the case of an **identification** modality in Example 18 in some community γ , the disjunction modality will denote distinct reference structures, whereas the conjunction will indicate which attributes belong together.

Example 18

Given the following gloss for the term “employee” in some community γ ,

```
("An employee" <- IDENTIFIES
  {DISJUNCTION {CONJUNCTION "is identified by their first-"
    "and last name"}
  "or an employee number"})
```

the pre-lexons in the example are:

- $\langle \gamma, \text{employee}, \text{with, of, first-} \rangle$
- $\langle \gamma, \text{employee}, \text{with, of, last name} \rangle$
- $\langle \gamma, \text{employee}, \text{with, of, employee number} \rangle$

The pre-constraints (in terms of these pre-lexons) are:

```
EACH employee with AT LEAST 1 first-.
EACH employee with AT MOST 1 first-.
EACH employee with AT LEAST 1 last name.
EACH employee with AT MOST 1 last name.
EACH employee IS IDENTIFIED BY (first- of employee)
  AND (last name of employee).
EACH employee with AT LEAST 1 employee number.
EACH employee with AT MOST 1 employee number.
EACH employee IS IDENTIFIED BY (employee number of employee).
```

When the community refines those pre-lexons, the constraints are updated accordingly.

The table below presents what information we elicit for each of the described gloss-evolution modalities. It is assumed that both S and N are analyzed independently as they can be themselves structured with elements of Θ . For each modality, a check mark indicates whether pre-lexons, -constraints and lexons can be elicited. The description briefly explains – at a high level – how. Cells that are grayed out mean that the modality has no implication on the hybrid ontology either because the modality provides information not immediately related to the concept being described (e.g. the **background**

modality) or because the modality is only useful in combination with other modalities (e.g. the *conjunction*).

	Pre-lexons	Pre-constraints	Instances
Gloss-evolution modality			
Background. Meanly intended for providing more information to comprehend the nucleus. This modality does not have any implication on the formal part of the hybrid ontology. The structure of the satellite, however, will be analyzed separately.			
Circumstance. Similar to background .			
Conditional. The conditional modality expresses a constraint on the nucleus, namely that the description (lexons and constraints) in the satellite must be true in order for the description in the nucleus to be true. Two possibilities are identified. If the condition is on a generalization , a subtype definition is identified. Otherwise, a subset constraint is identified. Pre-lexons can be elicited from the condition if not mentioned elsewhere.	✓	✓	
Elaboration. The elaboration modality has several special cases (each denoted with an arrow) that are treated somewhat differently. When this element is used, however, one will look for the verb phrases within the satellite to define the role the term in the nucleus is playing on (a) term(s) in the satellite.	✓	✓	✓
→ Attribute. The satellite provides one or more attributes of the concept being described. As an attribute, the concept plays the attributive role on each and one of these attributes at most once. Thus next to some pre-lexons with the roles “with / of”, a series of pre-constraints are also proposed.	✓	✓	
→ Generalization. The satellite provides information on a more general concept denoted in the nucleus. One thus needs to detect the term in the nucleus and construct “is a/subsumes” pre-lexons with co-terms found in the satellite.	✓		
→ Instantiation. The goal of the instantiation modality is to provide examples of the terms or lexons described within one community. Here, the goal is thus to identify these examples and propose the community to use these example as test population in the hybrid ontology engineering process. Looking for noun phrases and proper nouns in the satellite identifies the examples. Some noun phrases, however, are for this particular modality filtered. For instance, in the fragment “Examples of planets are Mercury, Mars and Earth”, the label “Example” is also a noun phrase.			✓
→ Meronymy. The satellite provides information of the concepts that are part of the concept described in the nucleus. One thus needs to detect the term in the nucleus and construct “with part/part of” pre-lexons with co-terms found in the satellite.	✓		
→ Specification. The satellite provides information on a more specific concept denoted in the nucleus. The term in the nucleus thus needs to be detected to construct “subsumes/is a” pre-lexons with co-terms found in the satellite.	✓		

→ Step. With this modality, the satellite provides information of a step belonging to a process described in the nucleus. To capture this information, the process and sub-processes are elicited from the nucleus and satellite respectively and introduce these processes as subtypes of the concept “Process”. The lexon “Process containing / part of Process”, if not yet present in the hybrid ontology, is furthermore introduced.	✓		
Evidence. The type of evidence is restricted to information, such as analogies as explained in Section 5.2.1. Hence, it is treated in the same way as the background modality.			
Means. With this modality, the satellite presents a method or tool for the concept described in the nucleus. Thus, next to providing the pre-lexons with roles “uses / used for”, the pre-lexons “X is s / subsumes Tool” and “X is a / subsumes Method” where X is the method or tool presented in the satellite are also presented to the community. It is then up to the community to refine and choose the appropriate pre-lexons.	✓		
Otherwise presents an alternative to a conditional or unless modality. Again here, subtype definitions or set-constraints are proposed depending on the type of modality the conditional or unless modality was put on.	✓	✓	
Preparation. Similar to background			
Problem-for. The satellite provides a problem solved with the concept described in the nucleus. “solving / solved by” pre-lexons are elicited from this gloss as proposals for new lexons in the hybrid ontology.	✓		
Purpose. With this modality, the satellite provides the purpose of the activity described in the nucleus. This allows the distill subclasses of the concept “process” in the nucleus and subclasses of the concept “goal” in the satellite.	✓		
Unconditional. This modality relates S and N such that the truth-value of the proposition in N does not depend on the truth-value of the proposition in S. It thus does not provide information about possible lexons (otherwise it would be an elaboration), information about a constraint or instances. However, it remains interesting to keep this modality to make a distinction with an elaboration modality.			
Unless. This modality is similar to the conditional modality, but this time the description (lexons and constraints) in the satellite must be false in order for the information in the nucleus to be true. Again, here one can distill a subtype definition or an exclusion constraint, depending whether this modality is on a generalization modality or another. Pre-lexons can be elicited from this modality.	✓	✓	
Conjunction and Disjunction . For the definition of glosses, the use of disjunction and conjunction only make sense when applied in a modality. Indeed, the sentences have to be written to describe a term or lexon. The combination of the logical connectives “and” and “or” are used to provide more information on the mandatoriness of lexons when combined.			
Contrast. The nuclei in a contrast are not used to distill any information. It is purely informational. A contrast is used to highlight the difference between two concepts. However, this should actually be contained in the description of the other concept by means of a generalization and conditions.			
Both List and Sequence will be used to iterate over the spans.			

Term adoption. Term adoption does not have any implications on the hybrid ontology except for all meaning agreements in the adopted gloss automatically being used by the adopting community. As these can be considered constraints on the interpretation of the concept described, we checked the second column.	✓		
Identifies. The satellite provides a set of attributes that uniquely and totally identify the concept described in the nucleus. This allows an agent to propose to the community the constraints necessary to build the reference structure.	✓	✓	
Other. Not applicable. One can indeed parse the sentence and look for any pre-lexons. We, however, decided to store the information as is.			

5.3.1 Processing Structured Glosses

Even though any details on the implementation of tool support for the method proposed in this chapter of the thesis are not for until Chapter 7, a demonstration of community commitment and gloss co-evolution becomes more tangible by showing a prototype implementing these ideas.

Figure 5.2 depicts a screenshot of a client which – after prompting for the community member’s credentials – retrieves the gloss for a particular term or lexon and parses it. After parsing (1) the “pretty gloss” is shown to the member as well as information about the gloss’ tree-structure. In this picture, the **Instantiation** modality was selected for processing and retrieving the instances mentioned in the gloss (2). These instances should correspond with a reference structure for the term being articulated. If that is the case, the community member should be able to choose a set of lexons to populate. The community member can choose to use from the community commitment an existing set of unique, total and identifying attributes (see Section 3.2.1.1), or propose a new set which can contain new lexons (3), of which details are shown in Figure 5.3. After this step, social processes for new lexons, new constraints and the acceptance of lexon- and term populations are launched (4).

The social processes that are triggered are motivated by the gloss-evolution purpose being processes, and thus also the gloss. How these social processes are implemented is not yet of importance in this chapter, but details thereof can be found in Chapter 7.

Selecting and processing other modalities yield in different proposed pre-lexons and pre-constraints, depending on the actions that have been taken, described in the table of previous section. For example, using the same gloss of Figure 5.2, processing the instantiation modality would result in the following pre-lexon: $\langle \gamma, \text{Planet}, \text{is a, subsumes, class of celestial bodies} \rangle$. Processing the elaboration modality would have resulted in discovering $\langle \gamma, \text{class of celestial bodies, orbits, ., stars} \rangle$.

5.4 Relation with Related Work

Concerning the modalities adopted in this thesis, we would like to note the work of Hovy [Hov93]. Hovy discussed and provided a taxonomy for several discourse-relation

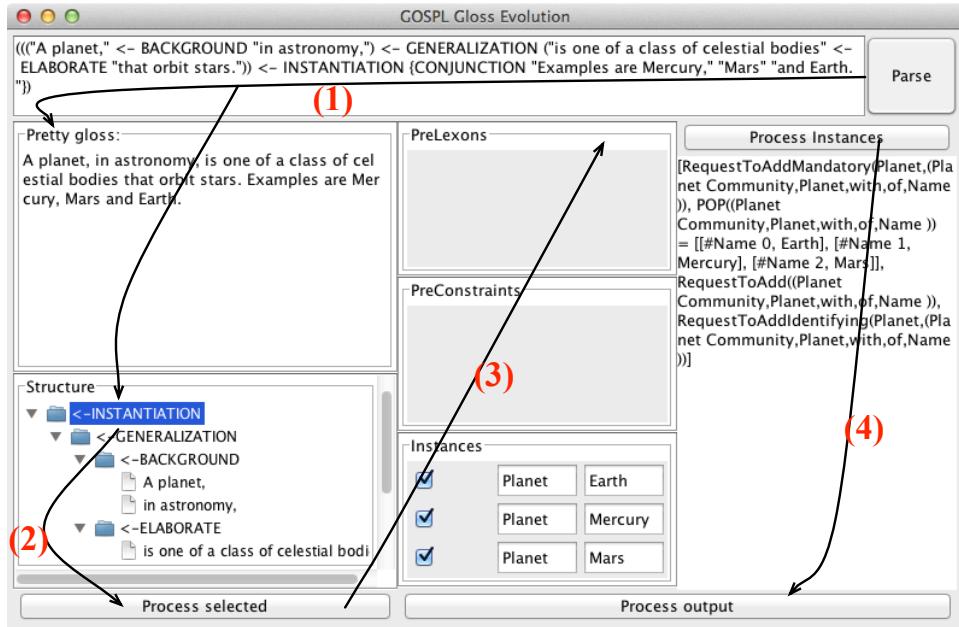


Figure 5.2: Processing a structured gloss for the term ‘Planet’ in the ‘Planet Community’.

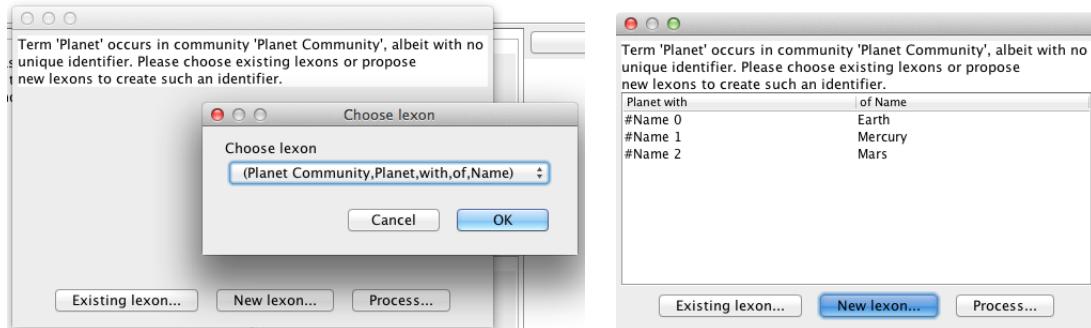


Figure 5.3: Populating the attributes to process the mentioned instances.

approaches and makes a distinction between formalist and functionalist analyses. Formalist analyses focus on the structure of the text, but are not that suitable for describing the actual content of the resources. Functional analyses assume that the internal structure of a discourse are defined by a communicative purpose and define the “functions” that segments of text have on other segments. Functional analyses, however, tend to be less suitable for describing the actual structure of the text. Hybrid approaches are also possible. As the content of the gloss is vital for our approach, we have adopted a method of the second category: Rhetorical Structure Theory. Hovy created a taxonomy of modalities in which he noted the number of researchers taking into account each modality. Most modalities of RST that are not subjective in nature appeared in that taxonomy, which gives an indication of the confidence on the relevance of those modalities in discourse relations [Hov93]. We thus analyzed the use of RST as a method for structuring and managing glosses and the impact on the formal descriptions of the described concepts thereof, thereby using glosses for more than merely meaning alignment

or meaning negotiation. However, not all modalities adopted in this chapter came from RST. We introduced some modalities that are specific to GOSPL.

Our approach to extract semantics from natural language definitions is very different from related work on ontology learning from text: we do not apply machine learning techniques and require the community to annotate the (type of) contribution they make while evolving the glosses. Buitelaar *et al.* provided a “layer cake” in which related work on ontology learning from text can be classified: terms > synonyms > concepts > concept hierarchies > relations > rules [BCM05]. What is interesting for us is to look at the techniques that learn relations and rules from text. As stated by Buitelaar *et al.* in that same paper: “most of the work on text mining combines statistical analysis with more or less complex levels of linguistic analysis, e.g. by exploiting syntactic structure and dependencies for relation extraction.” In [BOS04], for instance, the authors developed a framework for parsing texts via NLP techniques and declare how some structures should be transformed into statements in OWL. [CGR⁺05] start from texts in which the named entities and corresponding ontology concepts have been identified and parses this input to obtain the structure of the text to create a graph between the instances and concepts. This graph is then used to “learn” relations via statistical analysis. [GGA⁺02] proposed a method in which an agent first learns and classifies the positions of named entities in a corpus and then applies so-called “interpretation rules” to declare how specific classes need to be interpreted to construct the ontology.

5.5 Conclusions

Chapter 3 provided a framework for hybrid ontology engineering in which social interactions lead to formal descriptions of concepts and those social interactions are supported by a glossary to facilitate meaning agreements. This chapter provided a description on how the evolution of glosses has an impact on the hybrid ontologies by triggering social interactions that depend on the kind of gloss evolution, thus providing additional support for creating formal descriptions of these concepts. This was achieved by modeling (i) discrete gloss evolution, (ii) define a non exhaustive list of modalities relating sentences and (iii) propose how each modality could (or should) impact the community commitment by generating the necessary social interactions that will take place. This chapter thus contributed in helping the externalization processes of the communities. The next step is to aid those communities in re-internalizing the hybrid ontologies after agreements has been reached on glosses of- and formal descriptions of concepts. The next chapter will thus focus on the use of the formal descriptions of concepts and the annotated information systems to steer discussions within a community.

Chapter 6

Using Commitments for Steering Social Interactions

Commitments provide valuable information on the terms and lexons the different members of the community commit to. These were the results of the social interactions leading to the formal descriptions of concepts (described in Chapter 3) and the impact of gloss evolution on the community commitments (Chapter 5, more precisely in Section 5.2.1). Now we will describe how the formal representations of concepts can be used to steer the discussions in the community.

This chapter presents how the formal descriptions of concepts in both community- and application commitments will be examined to drive social processes within the community. First, the lexons and constraints in commitments will be translated in a suitable Description Logic dialect that preserves a bijective mapping between all permitted populations of the commitment and its translation. In other words, a *lossless* transformation. The Chapter then proceeds to explain how the formal descriptions and annotated datasets are used to drive the discussions by looking for support for certain claims. Finally, this chapter also shows how this translation can be used to retrieve instances from annotated datasets via the lexons, thus using the lexons as a query language.

Note to the reader: this chapter will occasionally mention the word “ontology”. In this chapter, an ontology will refer to a DL ontology and not to a hybrid ontology, i.e. it refers to the DL implementation of the hybrid ontology.

6.1 Translating Commitments into DL

This section presents a *lossless schema transformation* [DTMP83] for community commitments in GOSPL in $\text{DL-Lite}_{A,id}$. A lossless schema transformation is a transformation of a schema that allows one to preserve each permitted population [DTMP83]. In other words, the populations can be reconstructed unambiguously and thus a bijective mapping between both sets of permitted populations must exist. The “losslessness” of a transformation needs to be shown.

This section first introduces $\text{DL-Lite}_{A,id}$ and the translation of statements in $\text{DL-Lite}_{A,id}$ into first-order logic (FOL). This is followed by a presentation of the translation of fact types and constraints of the DOGMA ontology engineering framework and demonstrate the population equivalence.

6.1.1 The Description Logic DL-Lite_{A,id}

The alphabet of DL-Lite_{A,id} [CDL⁺08] consists of symbols for atomic concepts, value-domains, atomic roles, atomic attributes, and constants. The value-domains considered for this description logic are those adopted from RDF – i.e. XSD data types – representing sets of values T_1, \dots, T_N that are pairwise disjoint.

\mathcal{C} denotes the alphabet for constants assumed to be partitioned in two sets: \mathcal{C}_V as the set of constants for values (in turn partitioned into $\mathcal{C}_{V_1}, \dots, \mathcal{C}_{V_n}$ for each of the value-domains) and \mathcal{C}_O for the set of constant symbols for objects.

Before providing the specification of the language, first the notation used by [PLC⁺08] is introduced. A denotes an atomic concept, which is denoted by a name. B stands for a basic concept and C for a general concept. The syntax for both basic and general concepts will be given in the table below. Finally, T_C stands for the universal concept. A basic value-domain is denoted with E and F stands for a value-domain expression. T_D finally stands for the universal value-domain. P, Q and R will be used to refer to an atomic role, a basic role and a general role respectively. An atomic role is a role denoted by a name. Basic and general roles are role expressions, which will be described later on. U denotes an atomic attribute, and V_C a general attribute. An atomic attribute is an attribute denoted by a name, and a general attribute is a concept expression. The syntax of these concept expressions will follow.

Given an attribute U , the domain of an attribute refers to the set of objects that U relates to values. The domain of an attribute is denoted with $d(U)$. The domain of an attribute refers thus to a concept. Similarly, the range of an attribute U refers to the set of values that the attribute relates to objects. The range of an attribute is denoted with $r(U)$. The range of an attribute refers to a value-domain.

The expressions in DL-Lite_{A,id} is defined as follows:

Concept expressions	Value-domain expressions
$B ::= A \exists Q d(U)$	$E ::= r(U)$
$C ::= T_C B \neg B$	$F ::= T_D T_1 \dots T_n$
Role expressions	Attribute expressions
$Q ::= P P^-$	$V_C ::= U \neg U$
$R ::= Q \neg Q$	

Note that only basic concepts can be negated and explicit disjunction is not allowed [CDL⁺05].

The semantics of this description logic is given in terms of FOL interpretations. An interpretation I is a pair $(\Delta^I, .^I)$, where Δ^I is the interpretation domain and $.^I$ is the interpretation function. The interpretation domain is the union of two disjoint sets: Δ_O^I (the domain of objects) and Δ_V^I (the domain of values). The domain of values is the union of $v(T_1), \dots, v(T_n)$ where $v(T_i)$ refers to the set of instances of a particular data type T_i . Each $a \in \mathcal{C}_V$ is interpreted as one specific value and is denoted as $v(a)$. The interpretation function $.^I$ assigns an element of Δ^I to each constant in \mathcal{C} , a subset of Δ^I .

to each concept and value-domain, and a subset of $\Delta^I \times \Delta^I$ to each role and attribute in such a way that [PLC⁺08]:

- For each $a \in \mathcal{C}_V$, $a^I = v(a)$
- For each $a \in \mathcal{C}_O$, $a^I \in \Delta_O^I$
- For each $a, b \in \mathcal{C}$, $a \neq b \rightarrow a^I \neq b^I$
- For each T_i , $T_i^I = v(T_i)$
- And the following conditions are satisfied:

$T_C^I = \Delta_O^I$	$A^I \subseteq \Delta_O^I$	$r(U)^I = \{v \exists o : (o, v) \in U_C^I\}$
$T_D^I = \Delta_V^I$	$(\neg U)^I = (\Delta_O^I \times \Delta_V^I) - U_C^I$	$d(U)^I = \{o \exists v : (o, v) \in U_C^I\}$
$P^I \subseteq \Delta_O^I \times \Delta_O^I$	$(\neg Q)^I = (\Delta_O^I \times \Delta_O^I) - Q^I$	$(P^-)^I = \{(o, o') (o', o) \in P^I\}$
$U_C^I \subseteq \Delta_O^I \times \Delta_V^I$	$(\neg B)^I = \Delta_O^I - B^I$	$(\exists Q)^I = \{o \exists o' : (o, o') \in Q^I\}$

The authors in [PLC⁺08] defined the interpretation in such a way that the unique name assumption is adopted. In other words, each constant is interpreted differently in the domain. An ontology in DL-Lite_{A,id} is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ where \mathcal{T} represents the terminology box (TBox) containing the intensional knowledge and \mathcal{A} is the assertion box (ABox) containing the extensional knowledge. The assertions in the TBox in this description logic are of the following forms:

- $B \sqsubseteq C$ - concept inclusion
- $Q \sqsubseteq R$ - role inclusion
- $E \sqsubseteq F$ - value-domain inclusion
- $U \sqsubseteq V_C$ - attribute inclusion
- $(funct\ Q)$ - role functionality
- $(funct\ U)$ - attribute functionality
- $(id\ B\ \pi_1, \dots, \pi_n)$ - identification assertion

A role functionality assertion expresses the functionality of a role. In the case where $Q = P$, the functionality constraint is imposed on an atomic role, while in the case where $Q = P^-$, it is imposed on the inverse of an atomic role. An attribute functionality assertion expresses the functionality of an atomic attribute.

Identification assertions were first introduced in [CDL⁺08]. In identification assertions, every π_i is a path. A path is either:

- An atomic role or the inverse of an atomic role;
- An atomic attribute or the inverse of an atomic attribute;
- A composition of two paths π_a, π_b denoted as $\pi_a \circ \pi_b$, where \circ denotes the composition operator on two paths;
- A test relation $D?$ representing the identity relation on instances of D (either a basic concept or a value-domain). Test relations are used to impose involving instances of a certain concept or value-domain in the paths.

At least one of the paths in an identification assertion has to have a length of one, i.e. be an atomic role or attribute (or the inverse thereof). To define the semantics of identification constraints, the semantics of paths first need to be specified. The extension $\pi^{\mathcal{I}}$ of a path π in an interpretation \mathcal{I} is defined as follows:

- If $\pi = S$, then $\pi^{\mathcal{I}} = S^{\mathcal{I}}$ (where S is an atomic role or attribute);
- If $\pi = D?$, then $\pi^{\mathcal{I}} = \{(o, o) | o \in D^{\mathcal{I}}\}$
- If $\pi = \pi_1 \circ \pi_2$, then $\pi^{\mathcal{I}} = \pi_1^{\mathcal{I}} \circ \pi_2^{\mathcal{I}}$

Negative inclusion assertions are assertions of the form $B_1 \sqsubseteq B_2$, and are also possible for role and value-domain assertions.

Some notation:

- As a notation, [CDL⁺08] proposed to write $\pi^{\mathcal{I}}(o)$ to denote the set of π -fillers for o in \mathcal{I} . In other words, $\pi^{\mathcal{I}}(o) = \{o' | (o, o') \in \pi^{\mathcal{I}}\}$. This notation will be used to define the interpretation to satisfy an identification constraint.
- An atomic attribute U is called an identifying property in a TBox \mathcal{T} , if \mathcal{T} contains a functionality assertion (*funct U*). An atomic role Q is called an identifying property in a TBox \mathcal{T} , if \mathcal{T} contains a functionality assertion (*funct Q*).
- Let X be an atomic attribute or a basic role. X is said to be appearing positively in the RHS of an inclusion assertion a if a has the form $Y \sqsubseteq X$ and X is said to be appearing negatively in the RHS of an inclusion a if a has the form $Y \sqsubseteq \neg X$.
- An atomic attribute or a basic role is called primitive in a TBox if it does not appear positively in the RHS of an inclusion assertion and does not appear in an expression of the form $\exists Q.C$ in that same TBox.
- A DL-Lite_{A,id} TBOX is said to be a finite set of DL-Lite_{A,id} intensional assertions satisfying the condition that every identifying property in this TBox is primitive. In other words, identifying properties cannot be specialized by appearing in the RHS of an inclusion assertion.

Given a TBox \mathcal{T} in DL-Lite_{A,id}, an interpretation I satisfies:

- A concept inclusion assertion $B \sqsubseteq C$ if $B^I \subseteq C^I$
- A value-domain inclusion assertion $E \sqsubseteq F$ if $E^I \subseteq F^I$
- A role inclusion assertion $Q \sqsubseteq R$ if $Q^I \subseteq R^I$
- An attribute inclusion assertion $U \sqsubseteq V_C$ if $U^I \subseteq V_C^I$
- A role functionality assertion (*funct Q*) if

$$\forall o, p, q \in \Delta_O^I : (o, p) \in Q^I \wedge (o, q) \in Q^I \rightarrow p = q$$

- An attribute functionality assertion (*funct U*) if

$$\forall o \in \Delta_O^I, \forall v, w \in \Delta_V^I : (o, v) \in U^I \wedge (o, w) \in U^I \rightarrow v = w$$

- An identification constraint assertion (*id C π₁, ..., π_n*) if

$$\forall o, o' \in C^I : (\pi_1^{\mathcal{I}}(o) \cap \pi_1^{\mathcal{I}}(o') \neq \emptyset \wedge \dots \wedge \pi_n^{\mathcal{I}}(o) \cap \pi_n^{\mathcal{I}}(o') \neq \emptyset) \rightarrow o = o' \quad (6.1)$$

I is a model of a DL-Lite_{A,id} TBox \mathcal{T} , i.e. I satisfies \mathcal{T} if and only if I satisfies all intensional assertions in \mathcal{T} . This is written as $I \models \mathcal{T}$.

Note that even though conjunction and disjunction are not explicitly allowed in DL-Lite_{A,id}, one can simulate these constructs in a few cases [CDGL⁺07]. A disjunction on the lefthand side of a concept inclusion $A \sqcup B \sqsubseteq C$ is equivalent with the pair of assertions $A \sqsubseteq C$ and $B \sqsubseteq C$. A conjunction on the righthand side of a concept inclusion $A \sqsubseteq B \sqcap C$ is equivalent with the pair of assertions $A \sqsubseteq B$ and $A \sqsubseteq C$.

6.1.2 Lossless Schema Transformation

A Lossless Schema Transformation is defined as follows [DTMP83]: given \mathcal{S} a graph of lexons and \mathcal{C} a set of declared constraints, \mathcal{P} is a permitted population of $(\mathcal{S}, \mathcal{C})$ if and only if \mathcal{P} satisfies all constraints in \mathcal{C} . $\Pi(\mathcal{S}, \mathcal{C})$ defines the set of all permitted populations of $(\mathcal{S}, \mathcal{C})$. A transformation $t = (t_S, t_P)$ is a lossless schema transformation if and only if $t_S : (\mathcal{S}, \mathcal{C}) \rightarrow (\mathcal{S}', \mathcal{C}')$ such that $t_P : \Pi(\mathcal{S}, \mathcal{C}) \rightarrow \Pi(\mathcal{S}', \mathcal{C}')$ is bijective.

One thus needs to prove that the proposed translation into $DL_{A,id}$ is lossless. Both the model in the community commitment and the translation in $DL_{A,id}$ are translated in a set of FOL formulas and tested for their equivalence to shot this. The first translation is done by adopting the formalization provided by [Hal89] and the latter by [BHS08] (see Chapter 2). Testing their equivalence means showing that one formula is the logical consequence of the other and vice versa.

Description logics are decidable fragments of FOL. Concept names are unary predicates and role names are binary predicates. Concept descriptions correspond to FOL formulas with one free variable, which will be bound when used in a concept inclusion statement [BHS08]. The translation of assertions in a DL into FOL formulas are provided by [BHS08]. The translation of concept description C into a FOL formula with one free variable $\tau_x(C)$ is defined as follows:

1. $\tau_x(A) := A(x)$ for all concept names A
2. $\tau_x(C \sqcap D) := \tau_x(C) \wedge \tau_x(D)$
3. $\tau_x(C \sqcup D) := \tau_x(C) \vee \tau_x(D)$
4. $\tau_x(\neg C) := \neg \tau_x(C)$
5. $\tau_x(\forall r.C) := \forall y(r(x, y) \rightarrow \tau_y(C))$, where variable y is different from x
6. $\tau_x(\exists r.C) := \exists y(r(x, y) \wedge \tau_y(C))$, where variable y is different from x

Given a TBox \mathcal{T} with concept-inclusions, the translation $\tau(\mathcal{T})$ of \mathcal{T} is given by:

$$\tau(\mathcal{T}) := \bigwedge_{C \sqsubseteq D \in \mathcal{T}} \forall x(\tau_x(C) \rightarrow \tau_x(D))$$

6.1.2.1 Binary Fact Types

The translation of a binary fact type into DL-Lite_{A,id} is shown in Table 6.1. To prove that this translation is lossless, their equivalence will be demonstrated by means of a

semantic tableau¹ to demonstrate that $\models \Sigma \leftrightarrow \Phi$ holds.

All roles with the same label are translated in such a way that the labels become unique. For instance, in $\langle \gamma, A, r, s, B \rangle$ and $\langle \gamma, C, r, s, D \rangle$ the labels of the first roles are the same, but the roles are different: the first has domain A and range B , the second domain C and range D . During translation, they are thus transformed into r_1 and r_2 . The same is done for both roles with label s .

Table 6.1: Translating a binary fact type into DL-Lite_{A,id}.

DOGMA	DL
	$\exists R.\top \sqsubseteq A$ (6.2) $\exists R^-.top \sqsubseteq B$ (6.3)
DOGMA to FOL Σ	DL to FOL Φ
$\forall x(\forall y(R(x, y) \rightarrow (A(x) \wedge B(y))))$ (6.4)	$\forall x(\exists y R(x, y) \rightarrow A(x))$ (6.5) $\forall x(\exists y R(y, x) \rightarrow B(x))$ (6.6)

The semantic tableaux in Figures 6.1 and 6.2 both close, meaning there are no counterexamples for both $\Sigma \models \Phi$ and $\Phi \models \Sigma$. Since each is a consequence of the other, both sets of FOL formulas are equivalent. As they are equivalent, so are their possible extensions. Because of this equivalence, it follows naturally that both sets of formulas are population equivalent.

6.1.2.2 Mandatory Constraints

The translation of a mandatory constraint into DL-Lite_{A,id} is shown in Table 6.2. First: a mandatory constraint over one role. As the reader can see below, both translations in FOL yield the exact same formula and are thus equivalent.

¹Note the term “semantic” in semantic tableau is different from how this term is used in this thesis.

Semantic tableaux are an efficient and convenient means to test whether a formula ϕ is a logical consequence of a set of formulas Σ in FOL. This is done by trying to make ϕ false with respect to Σ by looking for counterexamples for the sequent $\Sigma \circ \phi$. Sequents are two sets of formulas ϕ_1, \dots, ϕ_n and ψ_1, \dots, ψ_m separated by the symbol \circ . An evaluation V is called a counterexample of a sequent $\phi_1, \dots, \phi_n \circ \psi_1, \dots, \psi_m$ if $V(\phi_1) = \dots = V(\phi_n) = 1$ and $V(\psi_1) = \dots = V(\psi_m) = 1$. When a formula ϕ occurs on both sides of the sequent, the evaluation of ϕ returns both 1 and 0. In that case, the sequent contains a contradiction and thus has no counterexample.

Formulas on the LHS of a sequent have to be made true and formulas on the RHS of a sequent false. For instance, in $\Sigma, \alpha \wedge \beta \circ \Pi$, $\alpha \wedge \beta$ is true if and only if both α and β are true, which then yields the subproblem $\Sigma, \alpha, \beta \circ \Pi$ (by using the $\wedge L$ rule, where the ‘L’ stands for left). A branch is considered closed if it contains the same formula both on the LHS and RHS in one of the sequents of a branch. Otherwise the branch is open and a counterexample is found. A counterexample is a model that makes the LHS of the top sequent true, but the RHS false. For more details on semantic tableaux and the notation adopted in this thesis, see [vBvDHK⁺03].

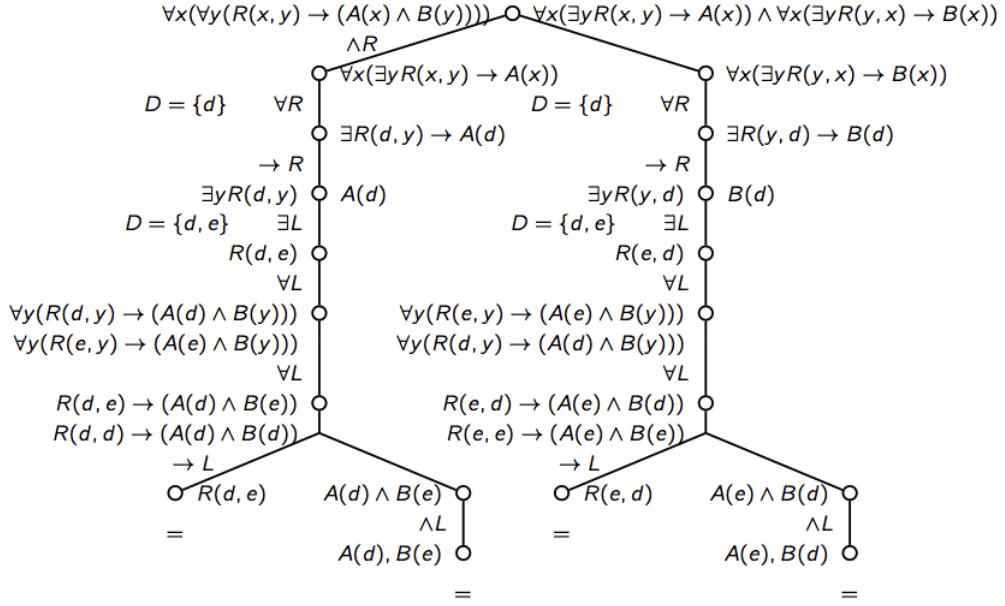


Figure 6.1: Semantic tableau to show that $\Sigma \models \Phi$ (cf. Table 6.1)

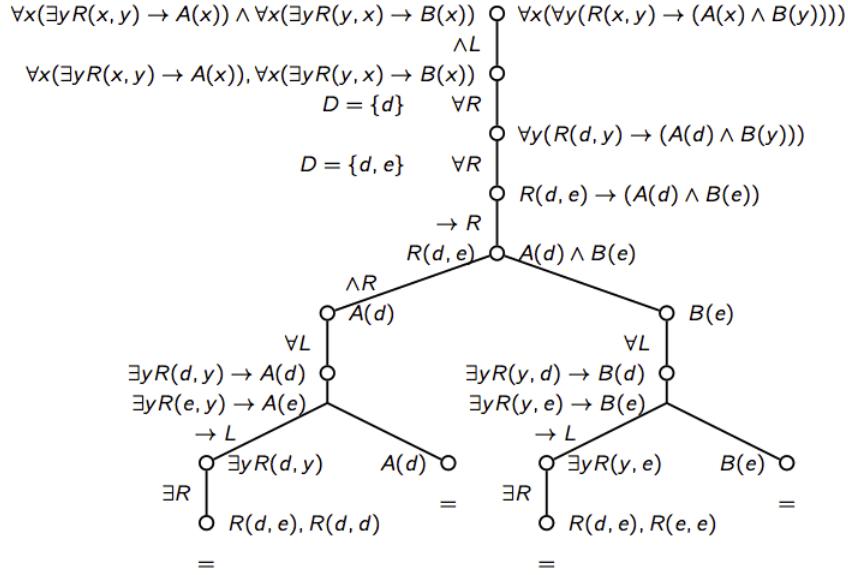


Figure 6.2: Semantic tableau to show that $\Phi \models \Sigma$ (cf. Table 6.1)

Inclusive mandatory constraints are easily translated into DL, but one needs to have concept disjunction in order to achieve this. This construct is not included in the DL dialect that is adopted for this thesis and can thus not be translated. However, such a translation would be lossless if it were possible to include this DL construct (see Section 6.1.3).

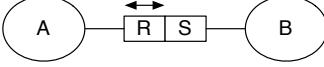
Table 6.2: Translating mandatory constraints into DL-Lite_{A,id}

DOGMA	DL
	$A \sqsubseteq \exists R. \top$ (6.7)
DOGMA to FOL Σ	DL to FOL Φ
$\forall x(A(x) \rightarrow \exists y(R(x, y)))$ (6.8)	$\forall x(A(x) \rightarrow \exists yR(x, y))$ (6.9)

6.1.2.3 Internal Uniqueness Constraints

Since only binary fact types are considered for this thesis, only five cases need to be examined: 1) no external uniqueness constraint, 2) one spanning only the first role, 3) one spanning only the second role, 4) one spanning the first role and one spanning the second role (a so-called one-to-one relation) and finally 5) one spanning both roles. Cases 1 and 5 are the same. This section will only elaborate on an internal uniqueness constraint on the first role. The third case is the same as the second, but using the inverse role. The fourth is a combination of cases 2 and 3. The translation is shown in Table 6.3 and the translations of both the DOGMA and DL-Lite_{A,id} into FOL are equivalent.

Table 6.3: Translating internal uniqueness constraints into DL

DOGMA	DL
	(funct R) (6.10)
DOGMA to FOL Σ	DL to FOL Φ
$\forall x(\forall y(\forall z((R(x, y) \wedge R(x, z)) \rightarrow y = z)))$ (6.11)	$\forall x(\forall y(\forall z((R(x, y) \wedge R(x, z)) \rightarrow y = z)))$ (6.12)

6.1.2.4 External Uniqueness Constraints

The translation of an external uniqueness constraint into DL-Lite_{A,id} is shown in Table 6.4. Equation (6.1) for the interpretation of an identification constraint actually states that: for any two instances of $a_1, a_2 \in A$, if the intersections of the interpretation of each

path π in the identification constraint for these two instances are not empty, then these two instances are actually the same instance.

Note that in DOGMA, one does not have the notion of “test populations” for identification constraints, which limit the population taken into account for a particular filler (or combination of fillers). First the atomic roles are treated, as they are important for RM-referability.

The interpretation of a particular path π is limited to the interpretation of an atomic attribute or role. In other words, if two instances share for every atomic role or attribute at least one instance in their range of these roles and attributes, they are deemed to be the same instance. Sharing an instance for a particular role implies that the intersection of the interpretations for that role for two instances of A is not empty. This corresponds with testing whether a_1 and a_2 play the same role with some concept y . So even though the interpretation in equation (6.1) was in terms of set theory, its translation into FOL is the same as that of the DOGMA model.

Table 6.4: Translating external uniqueness constraints into DL

DOGMA	DL
	$(id \ A \ R_1 \dots \ R_n) \quad (6.13)$
DOGMA to FOL Σ	DL to FOL Φ
$\forall x_1(\forall x_2(\forall y_1(\dots\forall y_n((R_1(x_1, y_1) \wedge \dots \wedge R_n(x_1, y_n) \wedge R_1(x_2, y_1) \wedge \dots \wedge R_n(x_2, y_n)) \rightarrow x_1 = x_2)\dots))) \quad (6.14)$	$\forall x_1(\forall x_2(\forall y_1(\dots\forall y_n((R_1(x_1, y_1) \wedge \dots \wedge R_n(x_1, y_n) \wedge R_1(x_2, y_1) \wedge \dots \wedge R_n(x_2, y_n)) \rightarrow x_1 = x_2)\dots))) \quad (6.15)$

It is interesting to note that using the graphical notation of ORM, the external uniqueness constraint is tested by means of a *conceptual join*. In an ORM schema, to navigate from one predicate to another, one must pass through an object type, performing a conceptual join on that object type [HM08]. The external uniqueness constraint is then tested by performing the natural join on the populations of each predicate in that constraint and checking whether the join does not contain any duplicates for that set of attributes. But as already noted in [Hal89], neither NIAM nor ORM diagrams can handle the following ambiguity arising from the graphical notation. Take for example the ORM diagram in Figure 6.3. In this diagram, it is not clear which of the following two constraints written in RIDL mentioned in the Figure are true. Hence, such a diagram in NIAM or ORM is illegal, as explained in [HM08]. In other words, RIDL is more expressive for modeling external uniqueness constraints.

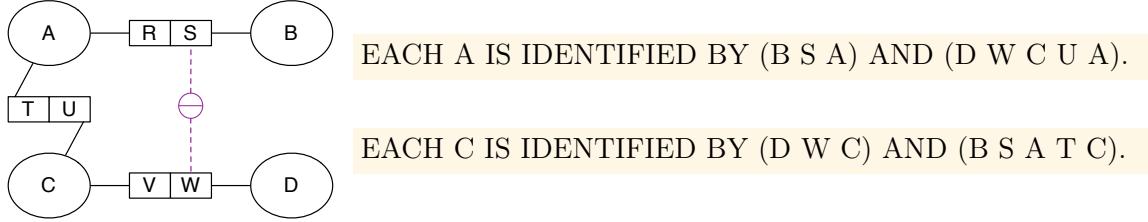


Figure 6.3: Example of an ambiguous external uniqueness constraint with the ORM diagram notation.

6.1.2.5 Subtyping

Translating concept hierarchies in DOGMA and DL-Lite_{A,id} into FOL is straightforward. Graphically, subtype relations are denoted by an arrow from the more specialized object type to the more general object type in ORM. Notice that the translation of a subtype declaration – shown in Table 6.5 – into FOL provided by [Hal89] is the same as the translation of the corresponding concept-inclusion in DL-Lite_{A,id}, and thus equivalent.

Table 6.5: Translating subtypes into DL-Lite_{A,id}

DOGMA	DL
	$B \sqsubseteq A$ (6.16)
DOGMA to FOL Σ	DL to FOL Φ
$\forall x(B(x) \rightarrow A(x))$ (6.17)	$\forall x(B(x) \rightarrow A(x))$ (6.18)

The problem with subtyping, however, is that the instances of all non-subtype object types are considered to be disjoint. Halpin calls these object types *primitive* [Hal89]. For any conceptual schema, there will be a finite number of such primitive object types. The disjointness of the instances of these object types are given with the following rule: given A_1, \dots, A_n primitive object types

$$\forall x(\neg(A_1(x) \wedge A_2(x)) \wedge \neg(A_1(x) \wedge A_3(x)) \wedge \dots \wedge \neg(A_{n-1}(x) \wedge A_n(x))) \quad (6.19)$$

In other words, it is prohibited for an instance to be a member of two object types from the set of primitive object types.

In order to be population equivalent, this same restriction needs to be modeled in the DL language adopted in this chapter. The problem, however, is that DL-Lite_{A,id} has no means for describing disjointness in an explicit way. In order to solve this, the disjoint

concepts need to be modeled via binary Horn inclusions: the concept-inclusion $A \sqsubseteq \neg B$ is asserted for every two concepts A and B that are disjoint. $A \sqsubseteq \neg B$ states that an instance of A is not an instance of B . It is not necessary to assert $B \sqsubseteq \neg A$ as well, as the translation of both concept inclusions into FOL show that both formulas are equivalent as one is a quantification of the contraposition of the other formula: $\forall x(A(x) \rightarrow \neg B(x)) \leftrightarrow \forall x(B(x) \rightarrow \neg A(x))$.

For every two object types in the primitive object types of the DOGMA model, such a concept-inclusion is added in the translation into DL-Lite_{*A,id*}. Now we need to show that the translation of these concept-inclusions into FOL is equivalent with the FOL formula in equation (6.19). Again, this is shown by means of the two semantic tableaux in Figures 6.4 and 6.5.

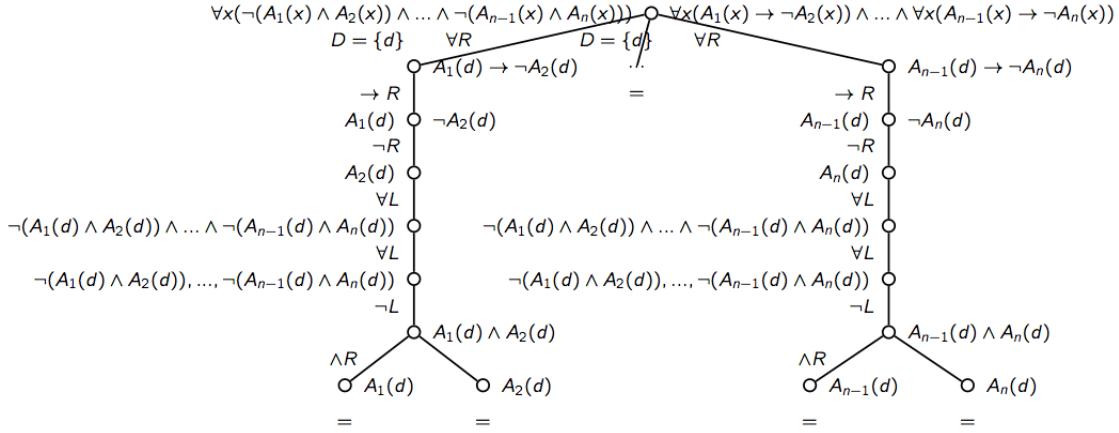


Figure 6.4: Semantic tableau

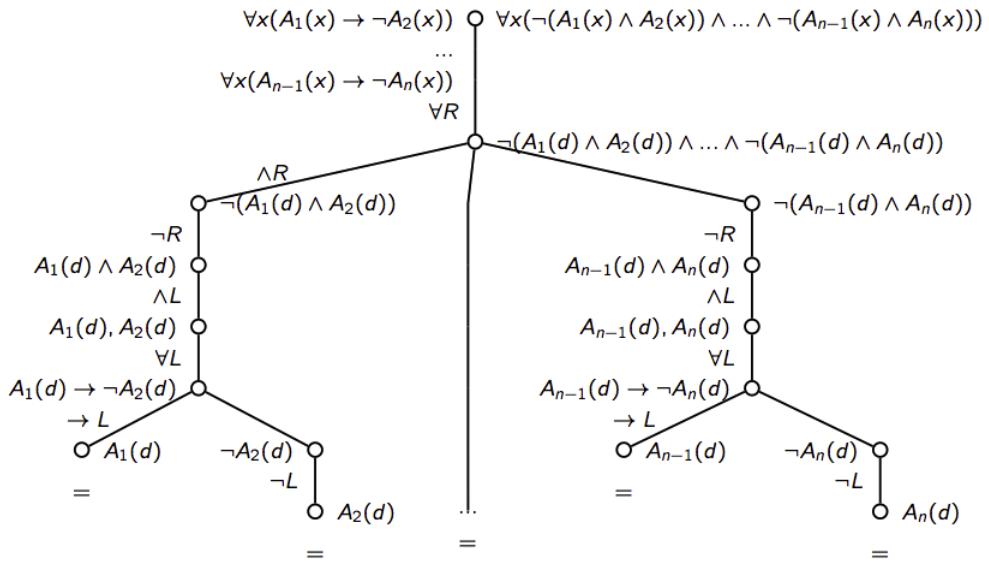


Figure 6.5: Semantic tableau

6.1.3 “Untranslatable” Constraints

$\text{DL-Lite}_{A,id}$ has been developed to remain tractable, as all members of the DL-Lite family are. This, however, has as a consequence that some ORM constraints cannot be translated into that specific DL. Luckily, this poses not too much of a problem as all constraints necessarily for conceptual reference structures can be translated into equivalent statements in $\text{DL-Lite}_{A,id}$. This section provides translations for these constraints that are not mapped. It is important to note, however, that one then must adapt a DL that is more expressive.

6.1.3.1 Inclusive Mandatory Constraint

The translation of an inclusive mandatory constraint in ORM to DL is straightforward. The translation is given in the Table 6.6. The table is followed by the two semantic tableaux in Figures 6.6 and 6.7 showing the equivalence of the two FOL formulas.

Table 6.6: Translating inclusive mandatory constraints into $\text{DL-Lite}_{A,id}$.

DOGMA	DL
	$A \sqsubseteq \exists R_1.\top \sqcup \dots \sqcup \exists R_n.\top \quad (6.20)$
DOGMA to FOL Σ	DL to FOL Φ
$\forall x(A(x) \rightarrow \exists y(R_1(x, y) \vee \dots \vee R_n(x, y))) \quad (6.21)$	$\forall x(A(x) \rightarrow (\exists yR_1(x, y) \vee \dots \vee \exists yR_n(x, y))) \quad (6.22)$

In ORM, total participation is depicted graphically with a full circle to which all participating is-a relations are connected with a dashed line. This constraint actually imposes that all instances of A must be an instance of B or an instance of C . In other words, $A = B \cup C$. Note that instances can be an instance of B and C at the same time. The translation into FOL is given below. It is clear that the translation of the corresponding statement in DL into FOL is the same; and thus equivalent. This translation is shown in Table 6.7.

6.1.4 Relation with Related Work

In the last few years, several authors addressed the problem of providing an encoding for ORM diagrams in DL knowledge bases [Kee07, Jar07b, Jar07a, HJ10, FMS12]. Only

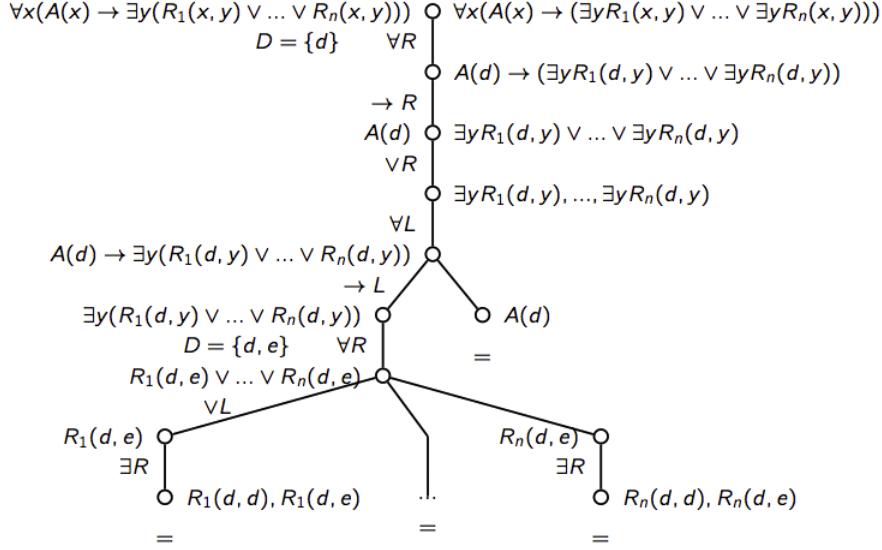


Figure 6.6: Semantic tableau to show that $\Sigma \models \Phi$ (cf. Table 6.6)

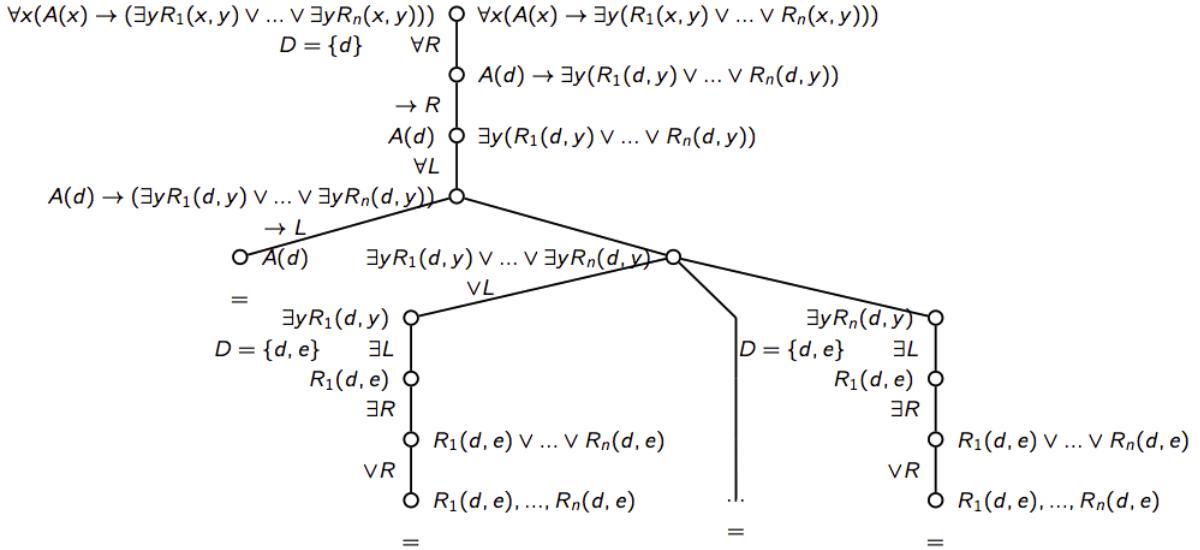


Figure 6.7: Semantic tableau to show that $\Phi \models \Sigma$ (cf. Table 6.6)

the work of Keet [Kee07], and Franconi and Mosca [FMS12] can be considered to have tackled the problem from a formal perspective. At the time of writing this thesis, the work of Franconi and Mosca was only available as an extended abstract. The authors do, however, have a technical paper providing more details on the linear transformation from ORM to a DL [FM12].

The inadequacy of the mapping proposed in [Jar07a, HJ10] can easily be shown by means of semantic tableaux. Note that the translation in [HJ10] contains quite a few syntactical errors, but builds further upon the work presented in [Jar07a]. Consider the binary fact type:

Table 6.7: Translation of totality on subtypes into DL-Lite_{A,id}.

DOGMA	DL
	$A \sqsubseteq B \sqcup C$ (6.23)
DOGMA to FOL Σ	DL to FOL Φ
$\forall x(A(x) \rightarrow (B(x) \vee C(x)))$ (6.24)	$\forall x(A(x) \rightarrow (B(x) \vee C(x)))$ (6.25)



The translation proposed in [Jar07a, HJ10] is as follows:

$$A \sqsubseteq \forall R.B \quad (6.26)$$

$$B \sqsubseteq \forall S.A \quad (6.27)$$

$$R \sqsubseteq S^- \quad (6.28)$$

Not only is the last statement incorrect and should be replaced with $R \equiv S^-$, one can still find – after correction – a counterexample for the translation of the binary fact type into FOL according to Halpin and the translation of these DL statements into FOL. The formulas below provide the latter translation.

$$\forall x(A(x) \rightarrow \forall y(R(x, y) \rightarrow B(y))) \quad (6.29)$$

$$\forall x(B(x) \rightarrow \forall y(S(x, y) \rightarrow A(y))) \quad (6.30)$$

$$\forall x(\forall y(R(x, y) \leftrightarrow S(y, x))) \quad (6.31)$$

Indeed, an interpretation \mathcal{I} with $\mathcal{I}(R) = \{\langle d, e \rangle\}$, $\mathcal{I}(S) = \{\langle e, d \rangle\}$, $\mathcal{I}(A) = \{\}$ and $\mathcal{I}(B) = \{\}$ is a model for above FOL formulas, but not for the translation provided by Halpin: $\forall x(\forall y(R(x, y) \rightarrow (A(x) \wedge B(y))))$. In other words, there are counterexamples and therefore there is not a bijective mapping between the two.

In [Jar07b, Kee07], both Jarrar and Keet provided a translation of ORM into a DL that supports n -ary relations where $n \geq 2$, namely the dialect \mathcal{DLR}_{ifd} [CDGL98]. The problems with translation proposed by Jarrar were examined by Keet in the second version of this paper². Keet criticized the inaccuracy of Jarrar's work with respect to

²The first version of her paper was published in 2007 in the Computer Research Repository. Later on, she provided a second version of her paper with corrections, more extensive related work, etc. in 2009. Both versions of the paper can be found here: <http://arxiv.org/abs/cs.LO/0702089>

the syntax and semantics. Franconi, in turn, provides critique on Keet's work on several inaccuracies [FM12, FMS12]. Both proposals are thus inadequate for the translation proposed in this thesis, even though the idea was appealing. However, as DOGMA limits itself to the use of binary lexons and DL-Lite_{A,id} provides constructs for a lossless translation, there is no need for constructs to support arbitrary n-ary relations.

Franconi and Mosca provided a translation of ORM into \mathcal{ALCQI} , thus using the DL \mathcal{ALC} extended with qualified cardinality restrictions and inverse roles. In essence, they "reify" fact types with uniqueness constraints spanning two or more roles by first introducing a new concept and then transform each of the involved roles into a DL role where the domain is the newly introduced concept and the range is the object type to which the ORM role was connected to. Those new roles are then declared to be functional. Indeed, each instance of that relation only plays each role once. As they claim, their translation is indeed sound and complete. Every model of the ORM translation into FOL is also a model for their DL translation into FOL. But, as will be seen, the inverse is not true.

Their translation is actually a lossy schema transformation (as shown in Figure 6.8). The figure presents a binary fact type and its lossless schema transformation using only attributive fact types. Lets call this translation (A). The figure also contains the "corresponding" ORM diagram for the DL translation proposed by Franconi and Mosca. Lets call this translation (B). Assuming that the FOL translation of (A) is contained in Σ and that of (B) in Φ . It is easy to see that due to the additional constraints in (A), every model of Σ is also a model for Φ , but the inverse is not true. Φ is thus a logical consequence of Σ , but not the other way round. Since the sets of formulas are not equivalent, there cannot exist a bijective mapping between the two. Hence, the translation proposed by Franco and Mosca is not lossless.

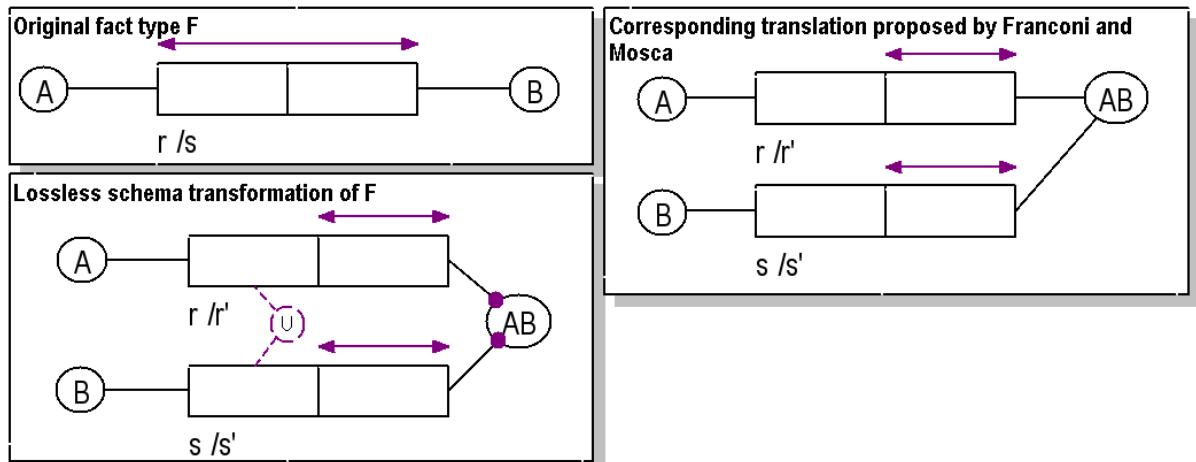


Figure 6.8: A binary fact type, its lossless schema transformation using only attributive fact types and the "corresponding" lossy translation by Franconi and Mosca [FMS12].

6.1.5 Conclusions

The first section in this chapter proposed a translation of community- and application commitments in $\text{DL-Lite}_{A,id}$. This DL dialect contains all constraints necessary for modeling reference structures and was therefore adopted in this thesis. This particular dialect is furthermore proven to be tractable. The translation that was presented here was lossless, meaning that there is a bijective mapping between the two sets of permitted populations. The losslessness of this translation was proved by means of semantic tableaux. Some constraints in GOSPL, such as the inclusive mandatory constraint, are not supported by the DL adopted.

More expressive DLs are able to support these translations, albeit loosing the tractability of reasoning and querying tasks. Whether this is an issue is left for future investigation.

Starting from this mapping, the next section will present how this translation is used to annotate legacy databases from closed information systems to look for support and counterexamples for claims made by the community.

6.2 Application Commitments in the Feedback Loop

Commitments provide valuable information on the terms and lexons the different members of the community representing their organization commit to. This selection will now be used to inform those members when changes are requested (and occur) in the ontology as to stimulate discussion.

The mapping α in those commitments is furthermore used to delve into the annotated data in search for support or counterexamples for certain statements made by the community, e.g. to notify the community whether proposed constraint is true for all annotated information systems currently known in the community. This process will guide the community in its dialogue to achieve agreement. This is done by generating the necessary queries using the commitments of each of the applications, populating the lexons in the conceptual schema and then reason over the data in terms of lexon populations. This tool is called Ω -DIPPER. Figure 6.9 extends and depicts the place of Ω -DIPPER in the feedback loop.

6.2.1 Semantics of Constrained Lexons

The previous section presented an encoding of lexons and constraints in Description Logic (DL) so that DL-reasoners can be utilized for reasoning tasks over the resulting ontologies.

The Open World Assumption (OWA) in DLs allows the existence of unknown information. However, in many cases, information needs to be as complete as possible to support the goal for which semantic interoperability between the information systems is needed - as defined by the semantic interoperability requirements of a community of stakeholders. In other words, the instances in the annotated information systems must follow certain business rules to ensure proper business.

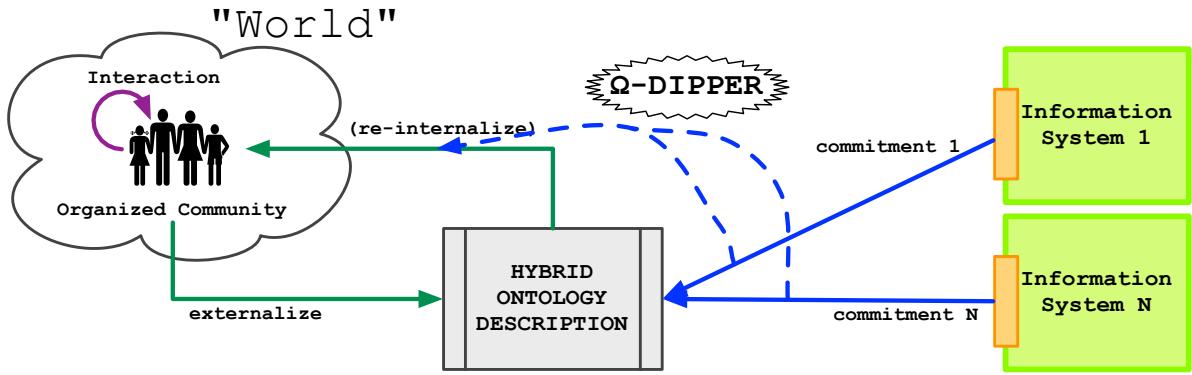


Figure 6.9: Feedback loop from the ontologies to the community by not merely taking into account the lexons committed to by the application, but the data in the annotated organization information systems as well.

For example, in some cultural domain community, it could be that every “Event must have at least one explicit associated Location”. The OWA, however, cannot capture this intuition of constraints that need to be imposed, a different semantics for the translation of constrained lexons in DL is needed to treat those constraints as OWL integrity constraints [MHS09, TSBM10]. The purpose of this section is to present how the translated constraints can be used as integrity constraints for the annotated data.

To achieve this, a combination of the Open World Assumption and Closed World Assumption (CWA) will be adopted. OWA is used in reasoning to derive new knowledge, and CWA is adopted when validating the integrity of the application data. A survey of existing approaches for OWL integrity constraints is presented³ that will lead us to an interpretation suitable for this section. This thesis follows [Gua98, Hep08] in that an ontology conceptualization should be separated from its instances. For presentation’s sake, however, both the axioms and assertions are considered when discussing the integrity constraints in this section. Also for improving the readability of the example, we use a set of assertions as an abbreviation of a model.

Definition 14 (Integrity constraints by consistency [Kow78])

An ontology \mathcal{O} satisfies an integrity constraint IC if and only if $\mathcal{O} \cup IC$ is satisfiable.

³Note that some authors will refer to an ontology as a knowledge base.

Example 19

Suppose that \mathcal{O}_1 consists of the following axioms

$$\exists has.\top \sqsubseteq Event \quad (6.32)$$

$$\exists has^-.top \sqsubseteq Location \quad (6.33)$$

$$MusicEvent \sqsubseteq Event \quad (6.34)$$

$$MusicEvent(boomtown) \quad (6.35)$$

and IC_1 contains only

$$Event \sqsubseteq \exists has.\top \quad (6.36)$$

It is easy to see that, under OWA, $\mathcal{O}_1 \cup IC_1$ is satisfiable. So \mathcal{O}_1 satisfies IC_1 by Definition 14. However, it does not fit the understanding of the constraint as done in this thesis: wanting that every event has explicit location, *boomtown* is an event but its location is not explicitly presented.

Definition 15 (Integrity constraints by entailment [Rei88])

An ontology \mathcal{O} satisfies an integrity constraint IC if and only if $\mathcal{O} \models IC$.

Example 20

Considering another example in which IC_2 contains only the constraint (6.36), \mathcal{O}_2 consists of all axioms of \mathcal{O}_1 and $\{has(boomtown, ghent), Location(ghent)\}$. Intuitively, one might think that \mathcal{O}_2 satisfies IC_2 . However, there is also a model $\mathcal{I}_1 = \{MusicEvent(boomtown), has(boomtown, ghent), Event(boomtown), Location(ghent), Event(polepole)\}$ of \mathcal{O}_2 for which $\mathcal{I}_1 \not\models IC_2$. By Definition 15, \mathcal{O}_2 does not satisfy IC_2 . That contradicts the intuition.

Definition 15 states that the integrity constraints must be entailed by all models. The above example, however, suggests that entailment in Definition 15 should be restricted to minimal models of \mathcal{O} . \mathcal{I} is a *minimal model* of \mathcal{O} if and only if \mathcal{I} is a model of \mathcal{O} and there is no model \mathcal{J} of \mathcal{O} such that $\mathcal{J} \subset \mathcal{I}$. Therefore, Definition 15 should be formalized as follows: \mathcal{O} satisfies IC if and only if all minimal models of \mathcal{O} entail IC . This idea has been nicely captured in [MHS09, MHS07], where ontology axioms are expressed as FOL formulas [Bor96] and *skolemization* [NW01a] is applied to deal with existential quantifiers.

Definition 16 (ICs by minimal models & skolemization [MHS09, MHS07])

Let $\pi(\mathcal{O})$ and $\pi(IC)$ be FOL formulas that express axioms in \mathcal{O} and IC respectively, $sk(\mathcal{O})$ be the set of formulas obtained by outer skolemization of $\pi(\mathcal{O})$. Outer skolemization generates new Skolem functions that get all the universally quantified variables as arguments the formula under consideration depends on [NW01b]. \mathcal{O} satisfies integrity constraint IC if and only if $\mathcal{I} \models \pi(IC)$ for every minimal Herbrand model \mathcal{I} of $sk(\mathcal{O})$. We sometimes write $\mathcal{I} \models IC$ instead of $\mathcal{I} \models \pi(IC)$.

Reconsidering Examples 19 and 20 with respect to Definition 16. In Example 19, the only minimal Herbrand model of \mathcal{O}_1 is $\mathcal{I}'_1 = \{MusicEvent(boomtown), Event(boomtown)\}$. By Definition 16, \mathcal{O}_1 does not satisfy IC_1 because $\mathcal{I}'_1 \not\models IC_1$. It follows the intuitive interpretation of avoiding an unknown *Location* for *Event*. In example 20, the only minimal Herbrand model of \mathcal{O}_2 is $\mathcal{I}_2 = \mathcal{I}_1 \setminus \{Event(polepole)\}$ and $\mathcal{I}_2 \models IC_2$, then \mathcal{O}_2 satisfies IC_2 . This also fits the intuitive interpretation, however in some cases the skolemization could lead to unexpected consequences.

Example 21

Let \mathcal{O}_3 consists of following axioms:

$$\exists has.\top \sqsubseteq Event \quad (6.37)$$

$$\exists has^-.top \sqsubseteq Location \quad (6.38)$$

$$MusicEvent \sqsubseteq Event \quad (6.39)$$

$$Event \sqsubseteq \exists has.\top \quad (6.40)$$

$$MusicEvent(boomtown) \quad (6.41)$$

IC_3 contains only the axiom: $MusicEvent \sqsubseteq \exists has.\top$

The minimal Herbrand model of \mathcal{O}_3 is of the form $\mathcal{I}_3 = \{MusicEvent(boomtown), hasLocation(boomtown, u), Event(boomtown), Location(u)\}$ where u is generated by skolemization of axiom (6.38) and (6.40). We see that $\mathcal{I}_3 \models IC_3$, so \mathcal{O}_3 satisfies the IC_3 although the exact location of *boomtown* is unknown.

Definition 16 almost captures the intuition of integrity constraints, but it has some drawbacks as discussions in [TSBM10]. To avoid the problem of unknown individual and avoid unintuitive meaning of integrity constraints, this thesis proposes an alternative semantics for OWL integrity constraints. Before that, however, the *Herbrand-based model* of a DL ontology is first defined, which is similar to the Herbrand model in FOL.

Definition 17 (Herbrand-based model)

Given an ontology $\mathcal{O} = \{\mathcal{T}, \mathcal{A}\}$. We call \mathcal{I} a *Herbrand-based model* of \mathcal{O} if \mathcal{I} is a

model of \mathcal{O} , $\Delta^{\mathcal{I}}$ consists of ABox's individuals, and every individual is mapped to itself.

A Herbrand-based model \mathcal{I} of an ontology \mathcal{O} is *minimal* if there is no Herbrand-based model \mathcal{J} of \mathcal{O} such that: $\Delta^{\mathcal{I}} \subset \Delta^{\mathcal{J}}$ and interpretation function $\cdot^{\mathcal{J}}$ contains every mapping in the interpretation function $\cdot^{\mathcal{I}}$.

Now the *integrity constraint interpretation* is defined for interpreting the translated GOSPL constraints. For each ontology, there is at most one such interpretation. In case of unsatisfiable ontologies, there is no such interpretation.

Definition 18 (Integrity constraint interpretation)

Given an ontology $\mathcal{O} = \{\mathcal{T}, \mathcal{A}\}$ and $\mathcal{M} = \{\mathcal{I}_1, \dots, \mathcal{I}_n\}$ where each \mathcal{I}_i is a minimal Herbrand-based model of \mathcal{O} . Concept name A , role R , and individual d in integrity constraints are interpreted by the following *integrity constraint interpretation* $\mathcal{I}_{IC} = \{\Delta^{\mathcal{I}_{IC}}, \cdot^{\mathcal{I}_{IC}}\}$, where $\Delta^{\mathcal{I}_{IC}}$ is a set of all individuals in \mathcal{A} , as follows:

$$\begin{aligned} A^{\mathcal{I}_{IC}} &= \{d^{\mathcal{I}_{IC}} \mid d^{\mathcal{J}} \in A^{\mathcal{J}}, \text{for all } \mathcal{J} \in \mathcal{M}\} \\ R^{\mathcal{I}_{IC}} &= \{(c^{\mathcal{I}_{IC}}, d^{\mathcal{I}_{IC}}) \mid (c^{\mathcal{J}}, d^{\mathcal{J}}) \in R^{\mathcal{J}}, \text{for all } \mathcal{J} \in \mathcal{M}\} \\ d^{\mathcal{I}_{IC}} &= d \end{aligned}$$

The extension of \mathcal{I}_{IC} to inverse roles and complex concepts is done as normal.

Now the definition of how an ontology satisfies an integrity constraint. Note that this thesis only considers satisfiable ontologies. In other words, ontologies for which its *integrity constraint interpretation* exists.

Definition 19 (Integrity constraint satisfaction)

Given an ontology \mathcal{O} and its integrity constraint interpretation \mathcal{I}_{IC} , \mathcal{O} satisfies an integrity constraint IC if and only if $\mathcal{I}_{IC} \models IC$.

Reconsidering the previous examples and check whether Definition 19 captures the intuition of integrity constraints.

Example 22

Reconsidering Example 19 with respect to Definition 19. Every minimal Herbrand-based model of \mathcal{O}_1 is of the form $\mathcal{I}_i = \{MusicEvent(boomtown), Event(boomtown), has(boomtown, loc_i), Location(loc_i)\}$ where loc_i is different in each model. We have $Event^{\mathcal{I}_{IC}} = \{boomtown\}$ but $(\exists has. \top)^{\mathcal{I}_{IC}} = \emptyset$. Thus, \mathcal{O}_1 does not satisfy IC_1 . This matches the intuition of integrity constraints in this section.

It is easy to check that Definition 19 also matches the intuition in Example 20. Reconsider Example 21 in the light of Definition 19, we have:

Example 23

Every minimal Herbrand-based model of \mathcal{O}_3 is of the form $\mathcal{I}_i = \{\text{Location}(u_i), \text{MusicEvent}(\text{boomtown}), \text{has}(\text{boomtown}, u_i), \text{Event}(\text{boomtown})\}$ where u_i is different in each \mathcal{I}_i . We have $\text{MusicEvent}^{\mathcal{I}_{IC}} = \{\text{boomtown}\}$, but $(\exists \text{has}. \top)^{\mathcal{I}_{IC}} = \emptyset$. Thus \mathcal{O}_3 does not satisfy IC_3 as expected.

Our proposal resembles the *IC-interpretation* in [TSBM10]. This approach differs from [TSBM10] in using minimal Herbrand-based models instead of classical models. To validate integrity constraints, the approach described [TSBM10] is adopted with one difference: adopting the Unique Name Assumption instead of the Weak Unique Name Assumption [TSBM10].

The next part presents the process of checking integrity constraints.

6.2.2 Checking Constraints over Annotated Data

Note that in hybrid ontology engineering, the proposed constraints are not yet in the community commitment if the community has not yet agreed on this constraint. To ensure proper semantic interoperability between the information systems belonging to that community, it is possible that some constraints have to be complied with by all information systems. To this end, the previous section was devoted to defining an integrity constraint satisfaction function. The integrity constraint satisfaction function in Definition 18 whether an ontology is valid with respect to an integrity constraint IC .

Inspired by [TSBM10], we show how IC validation is done by querying the annotated datasets. In [TSBM10], SPARQL ASK queries were constructed for IC constraints such that the evaluation of the query (a Boolean value) determines whether this IC is violated. For our purpose, however, it is more interesting to find the counterexamples, as the community will examine those counterexamples during their discussion. Queries are thus written in such a way that a non-empty set of results indicate counterexamples for a given IC.

Some SPARQL queries (without prefix) to check the validity of those integrity constraints and get the counterexamples are shown below. To get complete answers, the HermiT⁴ reasoner was used to classify the ontology and then perform *materialization* before running those SPARQL queries. Note that the method presented here works correctly with ontology languages in which classification tasks can be done without taking assertions into account.

Mandatory constraint $C \sqsubseteq \exists R. \top$

⁴<http://www.hermit-reasoner.com>

```

PREFIX ont: <http://path.to.my.ontology/#>
SELECT ?x WHERE {
    ?x a ont:C.
    OPTIONAL {?x ont:R ?y.}
    FILTER (!BOUND(?y))
}

```

Internal uniqueness constraint (*funct R*)

```

PREFIX ont: <http://path.to.my.ontology/#>
SELECT ?x WHERE {
    ?x ont:R ?y1.
    ?x ont:R ?y2.
    FILTER (?y1 != ?y2)
}

```

External uniqueness constraint (*id C R₁ ... R_n*)

```

PREFIX ont: <http://path.to.my.ontology/#>
SELECT ?x1 ?x2 WHERE {
    ?x1 a ont:C.
    ?x1 ont:R1.
    ...
    ?x1 ont:Rn ?yn.
    ?x2 a ont:C.
    ?x2 ont:R1 ?y.
    ...
    ?x2 ont:Rn ?yn.
    FILTER (?x1 != ?x2).
}

```

Is Lexical Constraint

For testing whether a term is truly of lexical nature in the annotated datasets, the following steps needs to be taken: determine all fact types with a role played by this term (as those concepts disappear in the OWL implementation of the hybrid ontology). For every corresponding co-role r_1, \dots, r_n , look up the OWL predicates R_1, \dots, R_n and use these predicates to construct the following query:

```

PREFIX ont: <http://path.to.my.ontology/#>
SELECT DISTINCT ?s ?p ?o WHERE {
    { ?s ?p ?o. FILTER(!isLiteral(?o) && ?p = ont:R1). }
    UNION
    ...
    UNION
    { ?s ?p ?o. FILTER(!isLiteral(?o) && ?p = ont:Rn). }
}

```

This section investigated the adoption of a proper interpretation for constraints for examining annotated data. To this end, the tension field between both Open and Closed World Assumptions were analyzed. Since data will be annotated for semantic interoperability (e.g. to do business), it was opted to go for a closer world assumption and treat constraints as integrity constraints. Constraints can therefore be tested by: 1) applying a reasoned to the annotated data and 2) translating those constraints into SPARQL queries to look for counterexamples.

6.3 Controlled Natural Language Querying

An application commitment contains information on how a particular information relates to the ontology via mappings and – possibly – additional enterprise-specific knowledge and constraints. One can thus obtain information from different annotated applications via those commitments. We explain how the translation of the community commitment into DL-Lite_{A,id} is used to query those databases via the lexons in the community commitment. This is done in two steps. First, we explain how queries with lexons are translated into queries using the DL translation. For this part, it is assumed that there exist such annotated datasets with this particular DL translation. Secondly, we explain how such DL annotations can be generated from an application commitment.

In this section, the namespace declaration and details of the OWL ontologies in the SPARQL queries have been omitted. The predicates are intentionally kept as human readable as possible to render the examples clear.

Information retrieval can be made easier for the user by hiding some of the complexity (e.g. learning standards such as XML, RDF and OWL) [Sch05]. Some of these standards are at odd with how users perceive the world, which is often a natural language perspective. Here, an approach to bridge this gap between questions in natural language and the popular Semantic Web query language SPARQL [PS08] – a W3C Recommendation for querying RDF – will be presented.

Human communication constitutes the exchange of facts, which can be generalized into fact types. In those fact types, concepts are playing roles on other concepts, and the name (or label) of that role changes depending on the direction. For instance, people *have* names, but a name is *of* a person. This bi-directionality is present in questions such as: “What are the names of artists with a gender with code F”. OWL supports inverse object properties (relations between non-lexical), but not for data properties (between a non-lexical and a lexical). The following SPARQL query returns all the names of artists with gender code ‘F’:

```
PREFIX myOnto0: <http://localhost:8080/gospl/ontology/10#>
SELECT DISTINCT ?name WHERE {
  ?a myOnto0:Artist_having_Name ?name .
  ?a myOnto0:Artist_with_Gender ?gender .
  ?gender myOnto0:Gender_with_Code 'F' .
}
```

This example shows clearly that constructing this query starts from artists having a name, rather than the names of artists. So, not only do users have to learn new formalisms in order to model and query information, users also have to consider that some of the predicates in those formalisms have to be addressed in the other direction in order to obtain the information. This section thus explores how lexons can be adopted for querying annotated datasets while the community discusses changes in the ontology.

To query data via lexons, a query language using those binary fact types needs to be either created or adopted. For this thesis, the latter was chosen and the fact-oriented query language RIDL [Mee82] was adopted.

RIDL, which stands for Reference and IDEa Language, was created to provide formal syntactic support for information and process analysis, semantic specification, and constraint definition as well as a query/update language at a conceptual level in the early eighties. The RIDL language manipulated, defined and restricted information structures and flows described using the NIAM [Win90] method, albeit restricted to binary fact types. RIDL was one of the first query languages to access the data via the conceptualization, which resulted from a natural language discourse between the users (of an information system). Because of its groundings in natural language, it was easier for users to retrieve information out of the system. A guide and description of the RIDL grammar are described in [DTMP83].

RIDL is a Controlled Natural Languages (CLN), which are subsets of natural language whose grammars and lexicons have been restricted, making it less complex and ambiguous [Sch05]. CLNs makes information retrieval and ontology engineering tasks easier on the user by hiding some of the complexity (e.g. learning standards such as XML, RDF and OWL) [Sch05]. RIDL also inspired Ω -RIDL [TTM07], a language to describe mappings between application symbols (e.g. fields in a database) to concepts in an ontology. Using a concatenation of lexons, sentences can be constructed to describe those application symbols.

Statements entered by the user are parsed following a grammar based on the original RIDL language (see [DTMP83]). The part adopted from RIDL has been refined to cope with Hybrid Ontology Descriptions. The RIDL language for fact-oriented querying of RDF – dubbed R-RIDL – has been defined using ANTLR [Par07], which can be found in Appendix C.

The goal of R-RIDL is not to replace SPARQL, which is excellent for building application on top of annotated data. The goal of R-RIDL is to allow exploring annotated data via the lexon, which represents knowledge grounded in natural language. As a natural consequence, those queries will be closer to the language used by the community.

R-RIDL statements are transformed into intermediate SPARQL queries used for populating the object types and lexons, applying set-operations on those populations to construct the result of the R-RIDL query. The specific set-operations depend on keywords such as AND, OR, NOT and so forth. At all times, a link to a community commitment is needed to bridge between the RDF and lexons in the hybrid ontology description. How an application commitment can be used to atomize a relational database as RDF triples will be explained later on.

One major difference between RIDL and R-RIDL is that the former did not allow listing of non-lexical object types. In other words, it was impossible to query all artists. Instead, lexical attributes of non-lexical object types had to be queried. In the context of the Semantic Web, everything should have a URI (or is a blank node). Moreover, in the context of Linked Data, everything must have a URI. Since URIs identify resources, users are able to formulate such queries and display the URI in the listing. When a resource happens to have more than one URI (e.g. via an `owl:sameAs` predicate), one is just chosen. This poses no problem as the URI can serve as a starting point to explore that resource and thus observe the other URIs.

Some statements in R-RIDL and their equivalent SPARQL queries will be shown below. The namespaces have been omitted, the namespace `myOnto0` is assumed to point to the OWL implementation of the Hybrid Ontology Description.

Example 1

R-RIDL: LIST Artist SPARQL: SELECT DISTINCT ?a WHERE { ?a a myOnto0:Artist }

Notice that in this example, the type of the resource needs to be specified in the SPARQL query, whereas a list of artist can be immediately obtained via the term label in R-RIDL.

Example 2

R-RIDL: LIST Name of Artist SPARQL: SELECT DISTINCT ?n WHERE { ?a a myOnto0:Artist . ?a myOnto0:Artist_having_Name ?n. }
--

With the name being a non-lexical object type, this example shows clearly the unidirectional nature of SPARQL.

Example 3

R-RIDL: LIST Name of Artist with Gender with Code = 'F' SPARQL: SELECT DISTINCT ?n WHERE { ?a myOnto0:Artist_having_Name ?n. ?a myOnto0:Artist_with_Gender ?g. ?g myOnto0:Gender_with_Code 'F'. }
--

Example 4

R-RIDL: LIST Artist NOT with Gender with Code = 'M' SPARQL: SELECT DISTINCT ?a WHERE { ?a a myOnto0:Artist . OPTIONAL { ?g myOnto0:Gender_of_Artist ?a. ?g myOnto0:Gender_with_Code ?c. } FILTER(?c != "M" !bound(?c)) }

In this example, all artists not having a gender with code ‘M’ are listed. This includes the artists whose gender was not explicitly stated. For the equivalent SPARQL query, one thus needs to specify that gender is optional. The tension field between open and closed world assumption is here apparent. This is done with the OPTIONAL clause, which will leave the variables unbound if no such information is available.

But merely testing the whether variable ?c does not equal ‘M’ does not suffice. As apart from bound, all functions and operators that operate on RDF will produce a type error if any arguments are unbound. Thus the result of a boolean test can be true, false or error. Testing whether ?c != ‘M’ will thus result in an error and the result will thus not taken into account for this query. One therefore needs to test whether the variable does not equal ‘M’ or the variable is unbound. The logical-and and logical-or truth table for true (T), false (F), and error (E) of SPARQL is given in Table 6.8.

Table 6.8: The logical-and and logical-or truth table for true (T), false (F), and error (E) in SPARQL [PS08]

A	B	$A \vee B$	$A \wedge B$	A	B	$A \vee B$	$A \wedge B$
T	T	T	T	T	E	T	E
T	F	T	F	E	T	T	E
F	T	T	F	F	E	E	F
F	F	F	F	E	F	E	F
				E	E	E	E

Example 5

```
R-RIDL: LIST Name of Artist born_in Year >
          (THE Year of Work_Of_Art with Title = 'A Bigger Splash')
SPARQL: SELECT DISTINCT ?n WHERE { ?a a myOnto0:Artist .
                                     ?a myOnto0:Artist_having_Name ?n .
                                     ?a myOnto0:Artist_born_in_Year ?y .
                                     ?w a myOnto0:Work_Of_Art .
                                     ?w myOnto0:Work_Of_Art_with_Title "A Bigger Splash".
                                     ?w myOnto0:Work_Of_Art_made_in_Year ?y2 .
                                     FILTER (?y > ?y2) }
```

In R-RIDL set-expressions, as the name implies, return sets. When values need to be compared to an occurrence in a set, the THE or ANY clauses can be used. The first returns the only occurrence of a set (reporting an error when the cardinality of a set does not equal to 1) and the latter returning any of the occurrences (reporting an error if the cardinality is not at least 1). The THE clause in this example is thus used to retrieve the year of a particular work of art to compare this value with the values of another attribute.

Example 6

```
R-RIDL: LIST Name
SPARQL: SELECT DISTINCT ?n WHERE {{ ?a myOnto0:Artist_having_Name ?n. }
                                UNION { ?a myOnto0:Art_Movement_with_Name ?n. } }
```

Assume that in the Hybrid Ontology Description, both artists and art movements have names, which are lexical. In R-RIDL, one merely needs the term label to obtain the set of all names. This is not possible in SPARQL as lexical attributes result in object properties with their ranges being instances of `rdfs:Literal`. To achieve the same effect, (i) one needs to look up all the lexons in which that term plays a role, (ii) find the corresponding data properties and (iii) construct the SPARQL query using the `UNION` operator for each of those data properties.

6.4 Atomizing Relational Databases with Ω -RIDL

A mapping in an Ω -RIDL application commitment ω captures how an application symbol relates to concepts and relations in the selection component of that commitment. For a relational database (RDB), such a mapping will describe how a field in a table corresponds to one or more lexons. A mapping μ is thus a triple $\langle T, F, p \rangle$ where T and F are respectively labels for a table and a field in a relational database and p is a non-empty set of lexons in the selection component of ω .

This section will explain how one can use an application commitment to “atomize” the information inside a relational database into RDF triples. By “atomization”⁵ we mean the process of generating triples out of RDBs, XML files, etc.

This process consists of two steps. First, one needs to be able to expose the contents of relational databases as RDF triples. For this, an existing solution, D2R Server⁶, shall be adopted. Such solutions, however, often can only base the RDF or OWL Schema merely on the “flat” relationships between attributes and the entities in the database schema without reference to the underlying “semantics” present when the database was developed [SHH⁺09] – presumably for one organization. The resulting triples are thus not properly semantic in nature⁷. The meaning of the generated classes and properties only make sense for that organization. One will thus need to “augment” or refine the generated classes and properties with the agreed upon semantics. One will even need to be able to generate additional classes and properties based on the selection inside a commitment and the lexons needed to achieve interoperability between the different information systems. The process of adding semantics to the triples constitutes the second part of this process.

⁵A term coined by Robert Meersman in 2010, but which he never published or claimed.

⁶<http://www.d2rq.org/>

⁷Remember, that semantics stem from the agreement of a community of stakeholders.

6.4.1 Creating RDF out of Relational Databases

A survey of RDB-to-RDF initiatives was given in [SHH⁺09]. Most of the methods follow the steps proposed by [BL98]: a record is an RDF node; the field (column) name is RDF propertyType; and the record field (table cell) is a value. The authors observed that automatically generated mappings often do not capture complex domain semantics that are required by many applications, but can serve as a starting point to create more customized, domain-specific mappings. The survey concluded that one of the important aspects of mapping RDB to RDF is the potential to explicitly model information that was either implicitly modeled or not represented at all in the database. This motivates the importance of using a domain ontology - developed and agreed upon by a community of stakeholder - in addition to information from the database schema [SHH⁺09].

This section will address the problem of enriching the generated RDF with more meaning coming from the hybrid ontologies via application commitments.

Even though there is currently an initiative under development for mapping relational databases to RDF by means of the R2RML mapping language by the W3C RDB2RDF Working Group⁸, D2R server was adopted for its maturity. The R2RML, or RDB to RDF Mapping Language is a W3C Candidate Recommendations as of February 23, 2012 and implementations of that standard are on their way. However, since their goals are similar, the approach we are presenting will be easily applicable to other mapping languages as well.

The D2RQ Platform provides means for accessing relational databases as virtual RDF graphs by annotating the relational database with special predicates provided by the D2RQ Mapping Language [CBG⁺12]. The platform not only allows databases to be queried using SPARQL, but the content of that database is also published as Linked Data: returning a human readable page when browsing and RDF for software agents.

Even though there is currently an initiative under development for mapping relational databases to RDF by means of the R2RML mapping language by the W3C RDB2RDF Working Group⁹, D2R server is adopted for its maturity. The R2RML, or RDB to RDF Mapping Language is a W3C Recommendations as of September 27, 2012 and implementations of that standard are on their way. However, since their goals are similar, the approach presented here will be applicable to other mapping languages as well.

D2RQ provides a tool for generating a mapping by analyzing the schema of an existing database, thereby using any primary keys, indexes and foreign keys specified. The foreign keys, for instance, will be used to detect join tables and create an object property between the two concepts referred to by the foreign keys. As RDF only supports binary relations, this is only done for join tables joining exactly two tables. Join tables joining three or more tables are converted into classes.

Take for instance the following DDL statement for a work of art:

⁸<http://www.w3.org/2001/sw/rdb2rdf/>

⁹<http://www.w3.org/2001/sw/rdb2rdfs/>

```

CREATE TABLE piece (
    id int(10) unsigned NOT NULL AUTO_INCREMENT,
    name varchar(250) NOT NULL,
    year int(4) NOT NULL,
    price_value bigint(20) DEFAULT NULL,
    currency varchar(3) DEFAULT NULL,
    PRIMARY KEY (id), UNIQUE KEY name (name,year))

```

The annotation that D2RQ generated for this table is shown in Appendix B. This example will be used to explain some of the predicates provided by the D2RQ mapping language. More information of the D2RQ mapping language can be found in [CBG⁺12].

The following definition of a class map is taken from [CBG⁺12]: “A `d2rq:ClassMap` represents a class or a group of similar classes of an OWL ontology or RDF(S) schema. A class map defines how instances of the class are identified.” In other words, it relates all records of a table - identified by a set of attributes - as instances of a given class. The mapping generated creates such a class on the fly based on the name of that table, here `vocab:piece`. Another important predicate in this annotation is the `d2rq:uriPattern`, which specifies a URI pattern that will be used to identify instances of this class map. The generated class map uses the primary key to construct this pattern.

```

map:piece a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "piece/@@piece.id@@" ;
  d2rq:class vocab:piece;
  d2rq:classDefinitionLabel "piece".

```

A `d2rq:ClassMap` has a set of `d2rq:PropertyBridge` descriptions which attach RDF properties to the instances. In the snippet below, the field “year” of table “piece” is assigned to the class map via the `d2rq:column` and `d2rq:belongsTo ClassMap` properties. A property is created on the fly and attached to this property bridge with a `d2rq:property` predicate.

The example below creates a relation between the value of a field and instances of a class. To create RDF properties between instances of two classes, one needs to specify the class map the property refers to with the `d2rq:refersToClassMap` property.

```

map:piece_year a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_year;
  d2rq:propertyDefinitionLabel "piece year";
  d2rq:column "piece.year";
  d2rq:datatype xsd:int.

```

D2RQ allows us to atomize databases and dump the content of that database as RDF triples on the Web. The mappings generated with D2RQ need to be refined with ontologies if one wants to achieve interoperability. The fields in a table, for instance, are

considered by the D2RQ mapping generator as attributes of the concept represented by that table. However, this is often not the case. Imagine a community agreeing on the fact types that prices are represented by a combination of a value and a currency code. In the table given in the example, two fields represent the price of a work of art (price_value and currency). The two fields became properties of a work of art via the property bridges.

```
map:piece_price_value a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_price_value;
  d2rq:propertyDefinitionLabel "piece price_value";
  d2rq:column "piece.price_value";
  d2rq:datatype xsd:long.

map:piece_currency a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_currency;
  d2rq:propertyDefinitionLabel "piece currency";
  d2rq:column "piece.currency".
```

If one wishes to represent price as a concept and instances of price having the attributes value and currency code, additional class maps and property bridges have to be defined. Also the classes and properties agreed upon by the community need to be added at the appropriate places.

Ω-RIDL application commitment files provide us with a means to leverage this task of refining a D2RQ mapping file (or any other similar mapping language, for that matter). The creation of the appropriate classes and properties can be guided by analyzing the annotations of applications symbols (here, tables and fields) with the terms and roles occurring in the selection. This process can even be automated, which will be shown below.

6.4.2 Augmenting the D2RQ Mapping File

The mapping of fields of a table onto terms and roles in the selection component of an Ω-RIDL application commitment always looks as follows:

```
MAP 'table'.'field' ON <LOT> (<ROLE> <NOLOT>)+.
```

This is clear as a field always represents a lexical attribute of a non-lexical object type, possibly playing roles with other non-lexical object types. Take for example the following annotation:

```
MAP 'artist'.'gender' ON Code of Gender of Artist.
```

Notice how every term-role-term combination actually correspond with one of the following lexons:

- $\langle \text{Cultural Domain Community}, \text{Artist, with, of, Gender} \rangle$
- $\langle \text{Cultural Domain Community}, \text{Gender, with, of, Code} \rangle$

Fields and tables are thus mapped onto a concatenation of lexons in one reading direction. It is clear that a term-role-term combination can be used to identify a lexon in the selection component of a commitment, and use this lexon to identify 1) which missing class maps and property bridges have to be constructed, 2) which OWL classes have to be connected to the class maps, and 3) which OWL data- or object properties have to be connected to the property bridges. The OWL classes and properties can be found in the OWL implementation of the Hybrid Ontology Description.

For every term-role-term combination in a mapping, the first will correspond with a data property (a link between a non-lexical and a lexical entity), the rest of the combinations correspond with an object-property. Most of the annotations can be processed in this way, only the annotations of join tables needs to be treated differently.

The DOGMA framework's Lexon Base is a (possibly vast) set of plausible binary fact types. The advantages of binary fact types are manifold. First, all fact-oriented diagrams can be losslessly transformed into diagrams with only binary fact types as we have explained in Section 6.1.2. Another advantage of only using binary fact types is avoiding nested fact types. Fact types containing n roles with an internal uniqueness constraint spanning $n - 1$ cannot be nested. Given an attributive lexon, it should not be possible for the community to even discuss nesting this lexon as this would lead to an invalid diagram. By not allowing the use of nesting, one avoids such problems. Even though nesting allows for seemingly less verbose diagrams and are generally convenient, the pitfall of nesting is to "skip" thinking about the concept denoted by that many-to-many relation between two object types. The relation denoted by the lexon is a concept playing roles with other concepts, and we therefore ask the community to label that concept (with a term). One can finally argue that by restricting the number of types of fact types the community needs to agree upon, a part of the agreement process is leveraged in the sense that teaching the community fact-oriented modeling becomes easier by presenting them with only one construct for modeling knowledge.

Many-to-many relations in the domain are either represented as (1) lexons with no functional roles or (2) as a non-lexical object type totally and uniquely identified by the involved object types. In solution (1), the 7-step algorithm for creating a relational model of a fact-oriented model (see Appendix A) would result in a so-called join table with foreign keys pointing to field in the tables representing the other object types. For (2), the result would be the same (1) provided there are no lexons connected to that concept other than those for the reference structure. Figure 6.10 contain an example of both (1) and (2) on the left and right respectively. The figure shows clearly that the resulting table for the many-to-many relation is the same except for the additional attribute on the right.

The mapping generated with D2R uses the foreign key constraints to determine whether a table is a join table. This is done by checking whether a table is a reference table (all fields are part of the primary key) and all fields are part of the foreign key constraints. Note that D2R server only checks this for binary relations, as RDF only supports binary relations. N-ary relations, where N is different from 2, have to be represented by means of

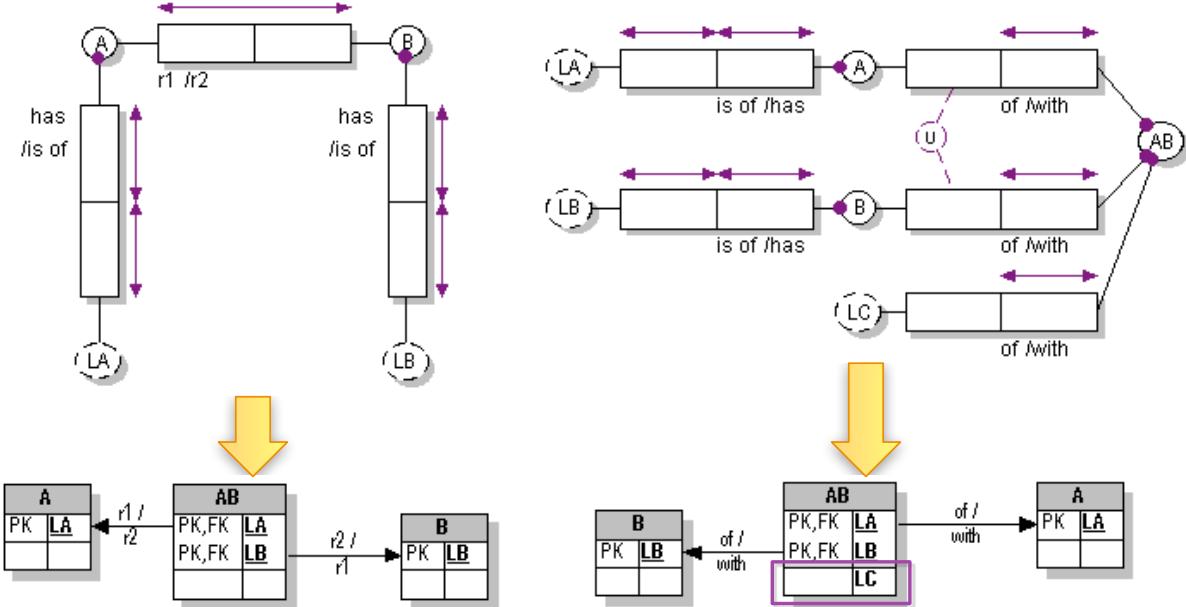


Figure 6.10: Transforming many-to-many relations in ORM to the relational model.

classes. If the foreign keys are not specified, a `d2rq:ClassMap` for that table is generated. This gives us four possibilities:

D2R identified m:n? n:m as lexon?	Yes	No
Yes	A	B
No	C	D

- (A) The relation is represented as a lexon in ω and as a m:n relation in δ . If the lexon is part of the Hybrid Ontology Description, find the corresponding object property in the OWL file generated from that Hybrid Ontology Description. If that lexon is part of selection component of the commitment that is enterprise-specific, then generate a property that D2R server can use. Check the direction of the object property, if the domain and range of that property bridge are switched with respect of the object property, take the inverse of that object property.
- (B) The relation is represented as a lexon in ω and as a class map in δ . Create the `d2rq:PropertyBridge` between the two `d2rq:ClassMap` descriptions. Use the annotations in the two class maps to construct the appropriate joins in that property bridge. In other words, look what the primary keys are in the two class maps and use those to describe a join between the foreign key and primary keys. The class map describing the join table can remain in δ .
- (C) The relation is represented as a term in ω and as a m:n relation in δ . Generate a new resource of the type `d2rq:ClassMap` as done in the previous sections. Generate the appropriate URI patterns for giving instances of this class a unique URI. Generate two `d2rq:PropertyBridge` instances to connect this class map with the two class maps composing this relation. The old m:n representation in δ can remain.

- (D) The relation is represented as a term in ω and as a class map in δ . No particularities in this case, this falls in the same process as the previous section.

If a join table corresponds with a lexon, a pattern can be easily recognized and processed accordingly. As an example we use the diagram and ER model of the left part of Figure 6.10: for instance, take the following lexons: $\langle \gamma, A, \text{with}, \text{of}, \text{LA} \rangle$, $\langle \gamma, B, \text{with}, \text{of}, \text{LB} \rangle$ and $\langle \gamma, A, r1, r2, B \rangle$. With each A being totally and uniquely identified by LA , each B totally and uniquely identified by LB , and each A can play the role of $r1$ on many B s and each B can play the role of $r2$ on many A s. Assuming one would have a join table AB with fields LA and LB , the mapping of this join table to the lexon $\langle \gamma, A, r1, r2, B \rangle$ would look as follows:

```
MAP 'AB'.'LA' ON LA of A r1 B.
MAP 'AB'.'LB' ON LB of B r2 A.
```

So, if there is a table with two annotations whose last term-role-term combinations can reconstitute a lexon from the selection, there is an annotation of a join table for that lexon. All the annotations are describing the attributes identifying the terms in that lexon. Those attributes can in turn be either lexical or non-lexical.

This section explained how one can use every term-role-term combination in a mapping to identify the corresponding OWL classes and properties and the identification or creation of the class maps and property bridges. Special care needs to be given in for lexons with no functional roles and join tables, which has been extensively discussed in this section. The algorithm that brings all proposed steps together is shown in Algorithm 1.

Algorithm 1 Semantically augmenting a D2RQ mapping file δ with an Ω -RIDDL application commitment ω

```
1:  $\delta \leftarrow \text{copyOf}(\delta)$ 
2:  $\text{processNonJoinMappings}(\omega, \delta)$  // Described in Algorithm 2
3:  $\text{processJoinMappings}(\omega, \delta)$  // Described in Algorithm 3
4: return  $\delta$ 
```

As an example, the community commitment in the running example of Chapter 4 is extended as follows (where ‘C’ stands for “Cultural Domain Community”):

```
<C, Affiliation, of, with, Artist>
<C, Affiliation, of, with, Art Movement>
<C, Art Movement, with, of, Name>
<C, Artist, born on, of birth of, Year>
<C, Artist, contributed to, with contribution of, Work Of Art>
<C, Artist, with, of, Gender>
<C, Artist, with, of, Name>
<C, Work Of Art, made in, of Year>
<C, Word Of Art, with, of, Title>
```

The constraints of this community commitment are:

Algorithm 2 Semantically augmenting a D2RQ mapping file δ with an Ω -RIDL application commitment ω

```

1: for all  $\mu \in mappings(\omega)$  do
2:    $list \leftarrow termRoleTermsIn(\mu)$ 
3:   if  $\mu \in joinMappings(\omega)$  then
4:     // Remove last Term-role-term combination, will be processed later
5:      $list \leftarrow list - \{last(list)\}$ 
6:   end if
7:    $ontC_{prev} \leftarrow undefined$ 
8:    $cMap_{prev} \leftarrow undefined$ 
9:   for all  $trt \in list$  do
10:     $ontC_2 \leftarrow ontClassOf(secondTermIn(trt))$ 
11:     $cMap_2 \leftarrow getConceptMappingFor(tableOf(\mu), secondTermIn(trt))$ 
12:     $Assert(commitsTo(cMap_2, ontC_2))$  in  $\delta$ 
13:     $ontP \leftarrow ontPropertyOf(lexonCorrespondingWith(trt))$ 
14:     $pMap \leftarrow getPropertyMappingFor(tableOf(\mu), fieldOf(\mu))$ 
15:     $assert(commitsTo(pMap, ontP))$ 
16:    if  $equals(trt, first(list))$  then
17:       $assert(domainCorrespondsWith(pMap, cMap_2))$  in  $\delta$ 
18:    else
19:       $assert(domainCorrespondsWith(pMap, cMap_{prev}))$  in  $\delta$ 
20:       $assert(rangeCorrespondsWith(pMap, cMap_2))$  in  $\delta$ 
21:    end if
22:     $ontC_{prev} \leftarrow ontC_2$ 
23:     $cMap_{prev} \leftarrow cMap_2$ 
24:   end for
25: end for

```

Algorithm 3 Semantically augmenting a D2RQ mapping file δ with an Ω -RIDL application commitment ω

```

1:  $visited \leftarrow \emptyset$ 
2: for all  $\mu_1 \in joinMappings(\omega)$  do
3:   if  $\mu_1 \notin visited$  then
4:      $\mu_2 \leftarrow correspondingJoinMapping(\mu)$ 
5:      $visited \leftarrow visited \cup \{\mu_1, \mu_2\}$ 
6:      $trt \leftarrow last(termRoleTermsIn(\mu_1))$ 
7:      $cMap_1 \leftarrow getConceptMappingFor(tableOf(\mu_1), firstTermIn(trt))$ 
8:      $cMap_2 \leftarrow getConceptMappingFor(tableOf(\mu_1), secondTermIn(trt))$ 
9:      $ontP \leftarrow ontPropertyOf(lexonCorrespondingWith(trt))$ 
10:     $pMap \leftarrow getPropertyMappingFor(tableOf(\mu_1), fieldOf(\mu_1))$ 
11:     $assert(commitsTo(pMap, ontP))$  in  $\delta$ 
12:     $assert(domainCorrespondsWith(pMap, cMap_1))$  in  $\delta$ 
13:     $assert(rangeCorrespondsWith(pMap, cMap_2))$  in  $\delta$ 
14:   end if
15: end for

```

EACH Artist IS IDENTIFIED BY (Name of Artist) AND (Year of birth of Artist).
 EACH Gender with AT LEAST 1 Code.
 EACH Gender with AT MOST 1 Code.
 EACH Gender IS IDENTIFIED BY (Code of Gender).
 EACH Affiliation of AT LEAST 1 Artist.
 EACH Affiliation of AT LEAST 1 Art_Movement.
 EACH Affiliation of AT MOST 1 Artist.
 EACH Affiliation of AT MOST 1 Art_Movement.
 EACH Affiliation IS IDENTIFIED BY (Artist with Affiliation)
 AND (Art_Movement with Affiliation).
 EACH Price with AT LEAST 1 Value.
 EACH Price with AT LEAST 1 Currency.
 EACH Price with AT MOST 1 Value.
 EACH Price with AT MOST 1 Currency.
 EACH Price IS IDENTIFIED BY (Value of Price) AND (Currency of Price).
 EACH Currency with AT LEAST 1 Currency Code.
 EACH Currency with AT MOST 1 Currency Code.
 EACH Currency IS IDENTIFIED BY (Currency Code of Currency).

And finally, for a particular application there is the following application commitment:

```

BEGIN SELECTION
[ 'Cultural Domain Community' ]
< 'MyOrganization', Artist, with, of, AID >
< 'MyOrganization', Art Movement, with, of, AMID >
< 'MyOrganization', Work Of Art, with, of, WID >
END SELECTION
BEGIN CONSTRAINTS
LINK( 'Cultural Domain Community', Artist, 'MyOrganization', Artist).
LINK( 'Cultural Domain Community', Work Of Art, 'MyOrganization', Work Of Art).
LINK( 'Cultural Domain Community', Art Movement, 'MyOrganization', Art Movement).
EACH Art Movement with AT LEAST 1 AMID.
EACH Art Movement with AT MOST 1 AMID.
EACH Art Movement IS IDENTIFIED BY (AMID of Art Movement).
EACH Artist with AT LEAST 1 AID.
EACH Artist with AT MOST 1 AID.
EACH Artist IS IDENTIFIED BY (AID of Artist).
EACH Work Of Art with AT LEAST 1 WID.
EACH Work Of Art with AT MOST 1 WID.
EACH Work Of Art IS IDENTIFIED BY (WID of Work Of Art).
END CONSTRAINTS
BEGIN MAPPINGS
MAP 'artist'.'id' ON AID of Artist.
MAP 'artist'.'name' ON Name of Artist.
MAP 'artist'.'birthyear' ON Year of birth of Artist.
MAP 'piece'.'id' ON WID of Work Of Art.
MAP 'piece'.'name' ON Title of Work Of Art.
MAP 'piece'.'year' ON Year of Work Of Art.

```

```

MAP 'piece'.'price value' ON Value of Price of Work Of Art.
MAP 'piece'.'currency' ON Currency Code of Currency of Price of Work Of Art.
MAP 'artist'.'gender' ON Code of Gender of Artist.
MAP 'artistpiece'.'artist id' ON AID of Artist contributed to Work Of Art.
MAP 'artistpiece'.'piece id' ON WID of Work Of Art with contribution of Artist.
MAP 'movement'.'id' ON AMID of Art Movement.
MAP 'movement'.'name' ON Name of Art Movement.
MAP 'artistmovement'.'artist id' ON AID of Artist with Affiliation.
MAP 'artistmovement'.'movement id' ON AMID of Art Movement with Affiliation.
END MAPPINGS

```

Remember that [SHH⁺09] observed that automatically generated mappings often do not capture complex domain semantics that are required by many applications, but can serve as a starting point to create more customized, domain-specific mappings. With this approach, one can easily provide richer annotations. For instance, some fields in a table may not be lexical attributes of the entity stored in a record, but rather attributes of an entity playing a role with the entity stored in the record.

D2RQ generates a lexical attribute for `artist.gender` in the database. However, the community has decided that a gender is identified by a code and artists play a role with gender. If one wants to obtain a list of all instances of genders, a class map for a projection of that table needs to be generated. Gender is identified by its code and thus that attribute is used to create its URL pattern. The gender code is also used to create a property bridge. Once that is done, a property bridge between the two class maps (artist and gender) is generated (note that namespaces have been omitted, the OWL implementation of the hybrid ontology is assumed to reside in `myOnto`):

```

map:Gender_artist a d2rq:ClassMap;
d2rq:class myOnto0:Gender;
d2rq:dataStorage map:database;
d2rq:uriPattern "Gender/@@artist.gender?urlify00".
map:Gender_with_Code_artist a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:Gender_artist;
d2rq:column "artist.gender";
d2rq:property myOnto0:Gender_with_Code.
map:Gender_of_Artist_artist a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:Gender_artist;
d2rq:property myOnto0:Gender_of_Artist;
d2rq:refersToClassMap map:Artist_artist.

```

The join table `artistpiece` was annotated with the lexon $\langle \gamma, \text{Artist}, \text{contributed_to}, \text{with_contribution_of}, \text{Work_Of_Art} \rangle$. The identifying attributes of both artists and works of art were used to create two class maps used by the property bridge.

```

map:Work_Of_Art_artistpiece a d2rq:ClassMap;
d2rq:class myOnto0:Work_Of_Art;
d2rq:dataStorage map:database;

```

```

d2rq:uriPattern
"Work_Of_Art/@@artistpiece.p_id|urlify@@"
map:Artist_artistpiece a d2rq:ClassMap;
d2rq:class myOnto0:Artist;
d2rq:dataStorage map:database;
d2rq:uriPattern
"Artist/@@artistpiece.a_id|urlify@@".
map:Artist_contributed_to_Work_Of_Art_artistpiece
a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:Artist_artistpiece;
d2rq:property myOnto0:Artist_contributed_to_Work_Of_Art;
d2rq:refersToClassMap map:Work_Of_Art_artistpiece.

```

Next, the relation between an artist and an art movement were stored in a join table, but the community decided to represent it as a concept: **Affiliation**. The table is used in the construction of a class map for affiliation and the class maps for artist and art movement were connected with appropriate property bridges.

```

map:Affiliation_artistmovement a d2rq:ClassMap;
d2rq:class myOnto0:Affiliation;
d2rq:dataStorage map:database;
d2rq:uriPattern "Affiliation/
@@artistmovement.a_id|urlify@@/
@@artistmovement.movement_id|urlify@@".
map:Artist_with_Affiliation_artistmovement
a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:Artist_artistmovement;
d2rq:property myOnto0:Artist_with_Affiliation;
d2rq:refersToClassMap map:Affiliation_artistmovement.
map:Art_Movement_with_Affiliation_artistmovement
a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:Art_Movement_artistmovement;
d2rq:property myOnto0:Art_Movement_with_Affiliation;
d2rq:refersToClassMap map:Affiliation_artistmovement.
map:Art_Movement_artistmovement a d2rq:ClassMap;
d2rq:class myOnto0:Art_Movement;
d2rq:dataStorage map:database;
d2rq:uriPattern
"Art_Movement/@@artistmovement.movement_id|urlify@@".
map:Artist_artistmovement a d2rq:ClassMap;
d2rq:class myOnto0:Artist;
d2rq:dataStorage map:database;
d2rq:uriPattern
"Artist/@@artistmovement.a_id|urlify@@".

```

6.5 Conclusions

Chapter 3 described a framework for hybrid ontology engineering, in which the social interactions of a community lead to formal descriptions of concepts. Those agreements are supported by a special linguistic resource called the glossary. Chapter 5 then described how gloss evolution has an impact on the hybrid ontologies, thereby supporting the externalization processes of the community (or communities) involved in those interactions. In this chapter, we describe how the result of that externalization process can be used to the fullest to support the community in re-internalizing that result, thereby closing the circle of social interactions, glossaries and formal descriptions.

Community- and application commitments in GOSPL are used to drive the discussions between community members. The use of application commitments beyond semantic interoperability is done in two ways: (i) by looking for counterexamples for statements made by the community and (ii) by exploring the annotated datasets via the lexons in the community commitment.

The first allows to validate the formal descriptions in the community commitment or even ongoing discussions in the community. This required a suitable translation into a Description Logic (DL) and an interpretation function for that DL to cope with the closer world assumption, both presented in this chapter.

For the second, RIDL was adopted to provide a fact-oriented way for browsing and querying datasets annotated with those DL translations of community commitments.

Chapter 7

Implementation

An experiment took place in March, April and May 2011 that mainly validated the social processes and the hybrid ontology engineering framework presented in Chapter 3. This experiment – reported in [DM11] – adopted an existing method and tool namely Business Semantics Management (BSM) and the Business Semantics Glossary (BSG). For a description of both BSM and BSG, see Chapter 2. BSM, however, did not prescribe a methodological sequence of processes on how articulation and the formal descriptions should evolve. Instead, BSM also followed a wiki paradigm allowing for changes to be immediately visible without prior discussion within that system. The method and tool furthermore did not support changes to follow automatically from the outcome of a discussion. Based on these experiences, the GOSPL method was proposed. For GOSPL, specific tool support was needed that followed principles defined in Chapters 3 and 4. BSG was deemed unfit mostly because of the wiki technology on which it was based; on one hand it was not discussion oriented and the wiki paradigm was difficult to adapt to support a specific sequence of processes. The decision therefore was to develop a tool specifically designed for GOSPL rather than spending more effort in tweaking an existing framework in following the GOSPL method.

This chapter presents a demonstration of a tool set that supports the GOSPL method for hybrid ontology engineering. The tool bears the same name as the method. GOSPL is supported by a web front-end, which is communicating with a hybrid ontology server. The hybrid ontology server exposes services supporting the method allowing additional clients to be developed, e.g. for advanced knowledge-engineering tasks. The examples used in this chapter stem from various case studies in which GOSPL was used to model ontologies for establishing semantic interoperability. These case studies will be described in the next chapter.

7.1 Architecture

The choice for a web-based system is motivated by the fact that a web-based application saves on-site installation time. For more advanced knowledge engineering tasks or analysis, however, the web-based application might not suffice. The design of the application thus took into account a server onto which other clients could connect to. The web-based client thus serves as an interface for the server.

The hybrid ontology server is developed as a J2EE project running in an application server (e.g. JBoss Application Server¹) using beans as an interface for the clients. The

¹<http://www.jboss.org/>

client is developed by means of a combination of JavaServer Pages² and JQuery³.

The reasons a new tool was prototyped from scratch was manifold. First, the choice of the Java programming language as most of the libraries available for Semantic Web languages and technologies are developed in that language and would thus facilitate integration. Second, most of the collaborative web-platforms aimed at ontology engineering were not made for fact-oriented formalisms. Indeed, there is the Business Semantics Glossary⁴, but its platform was not adopted as it was difficult to incorporate the social processes defined in GOSPL. This leads us to the third reason; existing platform demanded often too much time to tailor an instance to support hybrid ontology engineering. After briefly experimenting with some existing platforms, the author of this thesis felt that all (mostly technical) obstacles or challenges in adopting an existing platform did not outweigh the effort of creating a new application to demonstrate some principles.

The JBoss Application Server is widely used in industry and the J2EE aspect of the application makes it relatively easy for clients to connect to a server. It helps also to ensure the scalability in collaborative environments. On the server side, the following modules were developed:

1. Community-management: management of the key terms and goals of a community and the members inside a community.
2. Commitment-management: managing the agreed-upon lexons, constraints, and synonyms within and across communities. This module also takes into account managing application commitments known to the platform (e.g. to reason over the annotated data). This module is also responsible for the analysis of a hybrid ontologies' coherence and glossary-consistency and the NLP analysis of a gloss.
3. Glossary-management: the management of glosses and agreements on the equivalence of glosses within and across communities.
4. Other modules are available for, for instance, transforming a community commitment into OWL ontology.

For this research, two additional prototypes that act as clients for above mentioned server have been developed; one to support discrete gloss evolution and one for querying RDF via R-RIDL.

7.2 The Collaborative Ontology Engineering Platform

After following the address of an instance of GOSPL, the user is able to either login or register (see Figure 7.1). After logging in, the user has access to a list of communities currently working together on the development of hybrid ontologies. This is shown in Figure 7.2. Note that a user is also able to start a new community. The discussions involving the key terms or goals of a community are social processes, which will be discussed later on.

²http://en.wikipedia.org/wiki/JavaServer_Pages

³<http://jquery.com/>

⁴<http://www.colibra.com/>

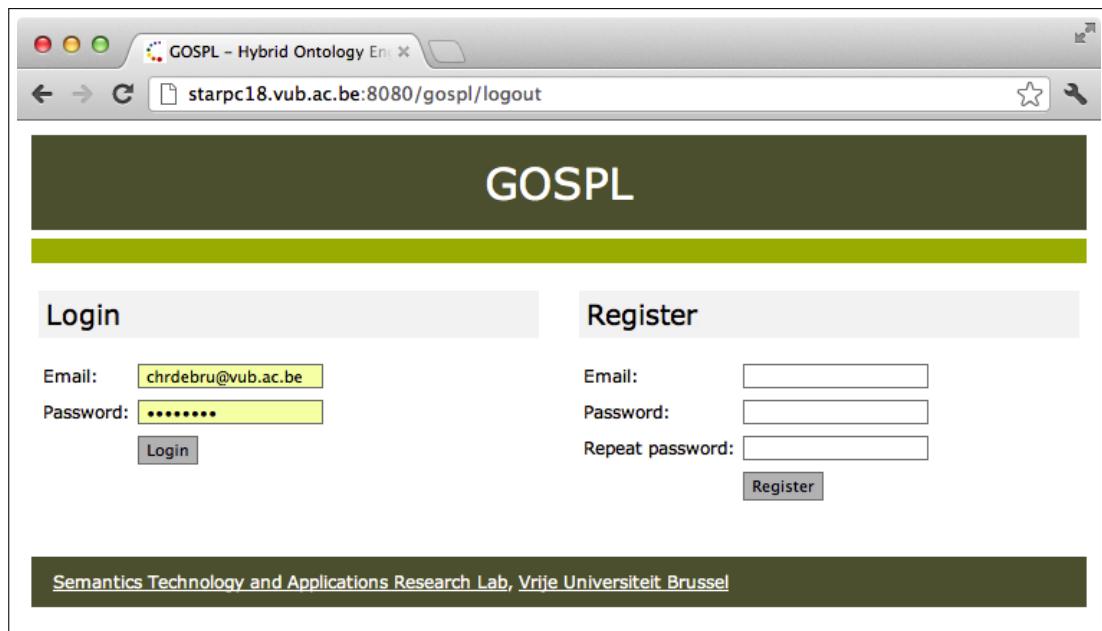


Figure 7.1: GOSPL login and registration screen.

The screenshot shows a web application titled 'Communities'. At the top, there is a search bar and a table showing a list of communities. The table has columns for 'ID', 'Label', 'Creation Date', 'Creator', and 'Member?'. The data is as follows:

ID	Label	Creation Date	Creator	Member?
2	Cultural Domain	2011-10-06		YES
3	Sandbox community	2011-10-24		YES
4	TermOnto Community	2011-11-30		YES
5	Date and Time Community	2012-04-24		YES
6	Address Community	2012-04-25		YES
7	Ticket Community	2012-04-25		YES
8	Act Community	2012-04-25		NO
9	Album Community	2012-04-25		YES
10	Venue community	2012-04-25		NO
11	Concert Community	2012-04-25		YES

Below the table, a message says 'Showing 1 to 10 of 54 entries' and a navigation bar shows 'First' through 'Last'. Underneath the table, there is a form for creating a new community, labeled 'Create a new community'. It includes fields for 'Label' (with a redacted input) and 'Goal' (with a redacted text area), and a 'Add Community' button.

Figure 7.2: A list of all available communities of one particular GOSPL instance.

7.2.1 The Community Commitment and Glossary

Clicking on a community brings you to a new screen with several tabs. Each of the tabs will be described in order. The first tab is called “ontology” and lists the lexons and constraints currently agreed-upon by the community. This tab thus displays the current community commitment. The lexons and constraints in Figure 7.3 have been respectively filtered by keywords “Time” and “Minute” to make the screenshot fit the page.

The screenshot shows the 'Community: Date and Time Community' interface. At the top, there is a navigation bar with tabs: Ontology, Glossary, Discussions, Members, Commitment, OWL/RDFS, and Activity. The 'Commitment' tab is active. Below the navigation bar, there are two main sections: 'Lexons' and 'Constraints'.

Lexons: A table showing lexons filtered by 'Time'. The columns are Head, Role, Corole, and Tail. The rows are:

Head	Role	Corole	Tail
DateTime	has	is of	Date
DateTime	has	is of	Time
DateTime	has	is of	Timezone
Time	has	is of	Hour
Time	has	is of	Minute
Time	has	is of	Second

Showing 1 to 6 of 6 entries (filtered from 9 total entries) | First | Previous | 1 | Next | Last

Constraints: A table showing constraints filtered by 'Minute'. The columns are Type and Description. The rows are:

Type	Description
LEX	Minute is lexical.
MAND	Time has at least 1 (Minute is of Time)
UNIQ	Minute is identified by (Time has Minute) [Time has at most one Minute]
UNIQ	Time is identified by (Second is of Time) and (Minute is of Time) and (Hour is of Time)

Showing 1 to 4 of 4 entries (filtered from 26 total entries) | First | Previous | 1 | Next | Last

Figure 7.3: The community commitment of the Date and Time Community.

Clicking on a term directs the user to a page that displays more information about the gloss of that term (if any), including information about term adoption. It also provides the user information about other terms (in other communities) that are currently considered synonymous with the displayed term. Figure 7.4 displays a screenshot displaying the community-term page of `DateTime` in the Date and Time Community.

Figure 7.5 provides a screenshot of the second tab called “glossary”. This tab lists the glosses the community currently uses to articulate the terms and lexons that are part of the community commitment. The tab also provides information about any gloss-equivalences on these glosses if any.

Community: [Date and Time Community](#)

Term: [DateTime](#)

Find glosses or facts for this term [here](#).

Gloss

Gloss	DateTime represents instants of time, optionally marked with a particular time zone offset.
Creator	DATUM PROJECT
Creation Date	2012-05-02 09:53:48.0

Term-communities adopting this gloss:

- None found for this gloss.

Glosses of other communities adopted by this community for this term:

- None found for this gloss.

[Analyze gloss](#) [Gloss equivalence](#) [Adopt this gloss!](#)

Synonyms

Show [10] entries	Search: <input type="text"/>	
Community	Term	Direct
Comment Community	DateTime	true
Date and Time Intervals Community	DateTime	true
News Community	DateTime	true

Showing 1 to 3 of 3 entries [First](#) [Previous](#) [1](#) [Next](#) [Last](#)

Figure 7.4: Screenshot of a community-term page.

Community: [Date and Time Community](#)

Ontology	Glossary	Discussions	Members	Commitment	OWL/RDFS	Activity
----------	----------	-------------	---------	------------	----------	----------

Term-glosses

- [Date](#) Time stated in terms of the day, month, and year.
- [DateTime](#) DateTime represents instants of time, optionally marked with a particular time zone offset.
- [Day](#) Also called civil day the period of time, the calendar day, of 24 hours' duration reckoned from one midnight to the next
- [Hour](#) Will contain the Hour value stated in Time
- [Minute](#) A way to identify the Minute value stated in Time
- [Month](#) A unit of time corresponding approximately to one cycle of the moon's phases, or about 30 days or 4 weeks.
- [Second](#) A unit of time equal to one sixtieth of a minute.
- [Time](#) Time stated in terms of the hours, minutes and seconds
- [Timezone](#) Any of the 24 divisions of the Earth's surface used to determine the local time for any given locality. Each zone is roughly 15° of longitude in width, with local variations for economic and political convenience. Local time is one hour ahead for each time zone as one travels east and one hour behind for each time zone as one travels west. The International Meridian Conference in 1884 established the prime meridian as the starting point for the 24 zones
- [Year](#) Way to identify the concept of year value stated in GregorianDate

Lexon-glosses

Figure 7.5: The glossary of the Date and Time Community.

7.2.2 Social Processes in GOSPL

GOSPL is discussion-oriented and both the ontology and glossary evolve only if the community reaches an agreement. This results in traceability both on decision level and on change level. Clicking on the tab “Discussions” brings the user to a list of discussions of that community, as shown in Figure 7.6. Different discussion can be started. Depending whether a person is a member of the community, some discussions might not be available. However, all users can leave comments and all users can start “informal” discussions (even when they are not part of the community). In other words, not only who changes what is recorded, but also the reasons certain changes have been made by linking changes to discussion on the platform. This was possible by formalizing the social processes and its corresponding operators.

Discussions					
Show 10 entries		Search: DateTime at most			
ID	Creator	Date	Title	Status	
3456	ISO/IEC JTC1 SC22/WG1	2012-05-24 12:07:36.0	Date is identified by (DateTime has Date) [DateTime has at most one Date]	Accepted	
3171	ISO/IEC JTC1 SC22/WG1	2012-05-21 11:33:14.0	Date is identified by (DateTime has Date) [DateTime has at most one Date]	Accepted	
3381	ISO/IEC JTC1 SC22/WG1	2012-05-23 15:35:56.0	Request to remove constraint: Date is identified by (DateTime has Date) [DateTime has at most one Date]	Accepted	
3382	ISO/IEC JTC1 SC22/WG1	2012-05-23 15:36:31.0	Request to remove constraint: Time is identified by (DateTime has Time) [DateTime has at most one Time]	Accepted	
3455	ISO/IEC JTC1 SC22/WG1	2012-05-24 12:07:13.0	Time is identified by (DateTime has Time) [DateTime has at most one Time]	Accepted	
3172	ISO/IEC JTC1 SC22/WG1	2012-05-21 11:33:51.0	Time is identified by (DateTime has Time) [DateTime has at most one Time]	Accepted	
3173	ISO/IEC JTC1 SC22/WG1	2012-05-21 11:34:17.0	Timezone is identified by (DateTime has Timezone) [DateTime has at most one Timezone]	Accepted	

Showing 1 to 7 of 7 entries (filtered from 105 total entries)

First Previous 1 Next Last

Figure 7.6: The discussions taking place in the Date and Time Community.

On the same page, users have the possibility to start new discussions. Depending whether the user is a member of that community, some requests are not accessible. For instance, one should be a member to contribute to the lexons, constraints or glosses of that community. Non-members have the possibility to request to become a member. Users that are part of another community are able to initiate a social process to discuss a gloss-equivalence of glosses or the synonymy of terms. Figure 7.7 depicts the form for a request to remove a particular constraint, and Figure 7.8 depicts the screen containing the thread after submitting the form.

A quasi-anonymous voting system is used to gather the opinion of people without the need of participating in the discussion (see Figure 7.7). It is quasi-anonymous in the

Add discussion: Request to remove constraint

Constraint Hour is lexical.

Motivation An hour shouldn't be lexical, the number of an hour is lexical.

Figure 7.7: Requesting the removal of a constraint.

sense that anyone can see who has voted, but not what they voted on a scale from 1 (strong agreement) to 5 (strong disagreement). This gives community members an idea inside the community are interested in what parts of the hybrid ontology.

Community: Date and Time Community

Request to remove constraint: Hour is lexical.

ID	Community	Creator	Date
3774	Date and Time Community	2012-08-03 14:55:50.0	
Post	An hour shouldn't be lexical, the number of an hour is lexical.		
Constraint	Request to remove constraint: Hour is lexical.		
Reply			

Dotmocracy Sheet

Strong agreement (0)	Agreement (0)	Neutral (0)	Disagreement (0)	Strong disagreement (0)
Cast your vote!				
<input type="button" value="Strong Agreement"/>	<input type="button" value="Agreement"/>	<input type="button" value="Neutral"/>	<input type="button" value="Disagreement"/>	<input type="button" value="Strong Disagreement"/>
Voters:				

Analysis

Analysis not supported for model.community.social.RequestToRemoveConstraint

Service Counterexamples

0

Resolve

Conclusion:

Action

Accept
 Deny

Figure 7.8: The resulting thread after submitted the form depicted in Figure 7.7

Some requests can be analyzed with the annotated application the platform is aware of. The application commitments belonging to community members describe how the application symbols of their system commit to the ontology, allowing the information in those database systems to be retrieved through the ontology. Of course, the discovery of counterexamples does not necessarily mean that the statement is false, however, this information might direct the discussion into another direction. Figure 7.9 shows a dataset has over 13000 counterexamples for the mandatory constraint on “has” between “Person” and “Other Name”.

Person Name has at least 1 Other Name			
ID	Community	Creator	Date
348	CERIF Person Ontology		2011-09-13 15:27:17.0
Post	Person Name has at least 1 (Person Name consists of Other Name)		
Constraint			
Reply			

Figure 7.9: Finding counterexamples for statements in the hybrid ontology.

The management of application commitments that enable this feature will be described in the next section.

7.2.3 Managing Application Commitments

Application owners are able to keep track of their application commitments within the platform. An application commitment can actually commit to several community commitments at once. Just as with the addition of (enterprise-specific) lexons and constraints, it is up to the owner of that application to make sure that the whole is consistent. Different versions of the application commitment are stored, which can be used to observe the changes between versions. In fact, it is possible to even detect the difference between two arbitrary application commitments. Figure 7.10 depicts what is shown to the user when two versions of a commitment are compared.

In the current implementation, users are able to associate an Ω -RIDL application commitment with a SPARQL endpoint. The SPARQL endpoints make use of the OWL implementation of the hybrid ontology (see Figure 7.11), which can be viewed by clicking on the appropriate tab. That same figure also shows that users are given a link on which the OWL file can be accessed.

7.2.4 Community Activity

As the last tab implies, all interactions between users and between a user and the system are logged (see Figure 7.12).

Application Commitment Differences

- 2 equal application commitment items:
 - Equal community:
 - Art community
 - Equal mandatory constraint:
 - Artist is identified by (Year_of_birth_of Artist) and (Name of Artist)
- 3 differences for application commitment id 2 (*new commitment items*):
 - Specific community difference:
 - Date and Time Community
 - Specific identity constraint difference:
 - DateTime has at least 1 (Time is_of DateTime)
 - Specific mandatory constraint difference:
 - Time is identified by (Hour is_of Time) and (Minute is_of Time) and (Second is_of Time)
- 2 differences for application commitment id 1 (*deleted commitment items*):
 - Specific community difference:
 - Address Community
 - Specific mandatory difference:
 - Address is identified by (House_Number is_of Address) and (City has Address) and (Street has Address) and (Country has Address) and (Post-Office_Box is_of Address) and (Postal_Code is_of Address)

Figure 7.10: Comparing two versions of an application commitment.

OWL/RDFS

An OWL/RDFS implementation of the current ontology can be found [here](#).

```
<?xml version="1.0"?
<rdf:RDF xmlns="/gospl/ontology/5#"
  xmlns:base="/gospl/ontology/5"
  xmlns:ontology8="http://starpcl8.vub.ac.be:8080/gospl/ontology/27#"
  xmlns:ontology9="http://starpcl8.vub.ac.be:8080/gospl/ontology/47#"
  xmlns:ontology4="http://starpcl8.vub.ac.be:8080/gospl/ontology/50#"
  xmlns:ontology3="http://starpcl8.vub.ac.be:8080/gospl/ontology/37#"
  xmlns:ontology6="http://starpcl8.vub.ac.be:8080/gospl/ontology/41#"
  xmlns:ontology5="http://starpcl8.vub.ac.be:8080/gospl/ontology/24#"
  xmlns:ontology="gospl/ontology/5#"
  xmlns:ontology2="http://starpcl8.vub.ac.be:8080/gospl/ontology/14#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#>
<owl:Ontology rdf:about="/gospl/ontology/5"/>
<!--
// Annotation properties
// -->

```

<http://starpcl8.vub.ac.be:8080/gospl/ontology/5>

Figure 7.11: OWL implementation of the hybrid ontology.

7.3 Discrete Gloss Evolution

As stated in Chapter 5, the support on discrete gloss evolution and its impact on the community commitment is supported by the development of:

1. The creation of a grammar for discrete gloss evolution, called the gloss-modality parser. The grammar – developed with the ANTLR Parser Generator⁵ – is shown below. The agent using this parser then uses a strategy pattern to take appropriate measures depending on the gloss-evolution modality, allowing for extensibility. Note that white spaces are ignored.

⁵ANTLR Parser Generator: <http://www.antlr.org/>

Activity

- 2012-08-03 14:55:50.0: started Request to remove constraint: Hour is lexical.
- 2012-05-25 21:26:34.0: closed and accepted Request to add synonym: SYN((Person Community,Date of birth),(Date and Time Community,Date))
- 2012-05-25 21:26:29.0: started Request to add synonym: SYN((Person Community,Date of birth),(Date and Time Community,Date))
- 2012-05-25 11:49:35.0: closed and accepted Request to add synonym: SYN((Movie community,Time),(Date and Time Community,Time))
- 2012-05-25 11:49:31.0: started Request to add synonym: SYN((Movie community,Time),(Date and Time Community,Time))
- 2012-05-24 16:21:56.0: closed and accepted Request to add synonym: SYN((Date and Time Intervals Community,DateTime),(Date and Time Community,DateTime))
- 2012-05-24 16:21:38.0: started Request to add synonym: SYN((Date and Time Intervals Community,DateTime),(Date and Time Community,DateTime))
- 2012-05-24 15:45:01.0: closed and accepted Request to add synonym: SYN((Exposition Items Community,Date),(Date and Time Community,Date))
- 2012-05-24 15:44:57.0: started Request to add synonym: SYN((Exposition Items Community,Date),(Date and Time Community,Date))
- 2012-05-24 13:40:59.0: closed and accepted Request to add synonym: SYN((Date and Time Intervals Community,Date),(Date and Time Community,Date))
- 2012-05-24 13:40:53.0: closed and accepted Request to add synonym: SYN((Date and Time Intervals Community,Time),(Date and Time Community,Time))
- 2012-05-24 13:40:49.0: started Request to add synonym: SYN((Date and Time Intervals Community,Time),(Date and Time Community,Time))
- 2012-05-24 13:40:39.0: started Request to add synonym: SYN((Date and Time Intervals Community,Date),(Date and Time Community,Date))
- 2012-05-24 12:07:48.0: closed and accepted Date is identified by (DateTime has Date) [DateTime has at most one Date]

Figure 7.12: Activity in a community.

```
// PARSER RULES
// Top rule, starting point of span
rule: span ;
span: sentence | modality | multinuclei ;
// Simple sentence: a unstructured string
sentence: STRING ;
// Application of a modality
modality: '(' span modalityDirection span ')';
modalityDirection: ('<->' modalityName) | (modalityName '>-') ;
modalityName: LITERAL;
// Application of a multinuclei
multinuclei: '{' LITERAL (span)+ '}';
// LEXER RULES
LITERAL: ('A'...'Z')+;                                // Words in capital letters
STRING: '"' ACTUALSTRING '"';                         // Quoted Strings
fragment ACTUALSTRING: ~('\'|'')*;
```

2. A Strategy Pattern [ERRJ95] was adopted to handle the specific operations for each type of modality. Depending on the modality; pre-lexons, pre-constraints or instances are detected that are then refined by the user before triggering the social processes.
3. The Apache OpenNLP Toolkit⁶ is a machine learning based toolkit for the processing of natural language text that has been adopted to tokenize, segment and apply part-of-speech tagging, etc. on the structured glosses to distill nouns, stem the verbs, etc. for the pre-lexons.

⁶<http://opennlp.apache.org/>

Figure 7.13 depicts a screenshot of a small client which – after asking for the user's credentials – retrieves the gloss for a particular term or lexon and parses it. After parsing (1) the “pretty gloss” is shown to the user as well as information about the gloss' tree-structure. In this picture, the **Instantiation** modality was selected for processing and retrieved the instances mentioned in the gloss (2). These instances should correspond with a reference structure for the term being articulated. If that is the case, the user should be able to choose a set of lexons to populate. The user can choose to use an existing set of unique, total and identifying attributes, or propose a new set which can contain new lexons (3, of which details are shown in Figure 7.14). After this step, social processes for new lexons, new constraints and the acceptance of lexon- and term populations are launched (4).

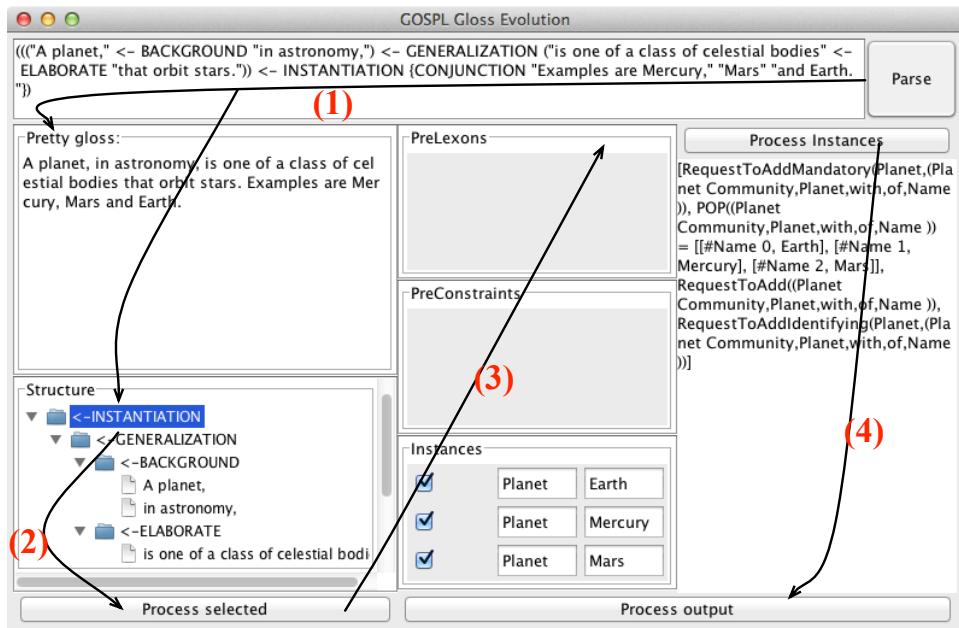


Figure 7.13: Processing a structured gloss for the term “Planet” in the ‘Planet Community’.

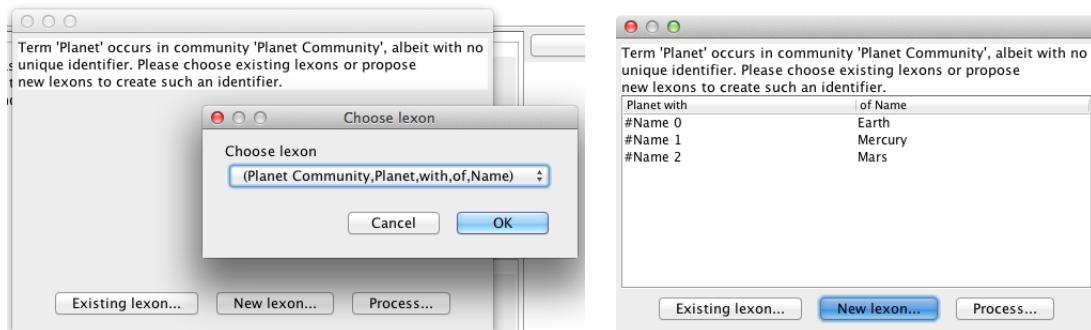


Figure 7.14: Populating the attributes to process the mentioned instances.

Selecting and processing other modalities yield in different proposed pre-lexons and pre-constraints.

7.4 Querying RDF with R-RIDL

The tool for querying RDF with R-RIDL consists of two modules: one to generate RDF out of relation databases via a Ω -RIDL application commitment and a module for transforming queries in R-RIDL into a series of SPARQL queries and return the information.

The first module uses Apache Jena⁷ to query the OWL implementation of the ontology and the RDF triples in the D2RQ mapping file. It also uses Jena to add additional statements to the model provided by this mapping file. The output can be saved to a file that can be used to start the D2RQ server⁸ (see Figure 7.15). Note, however, that in order for D2RQ mapping file a to be executable by the D2RQ server, minimal information about the database (username, password, host, etc.) needs to be provided in the D2RQ mapping file or added/modified later on.

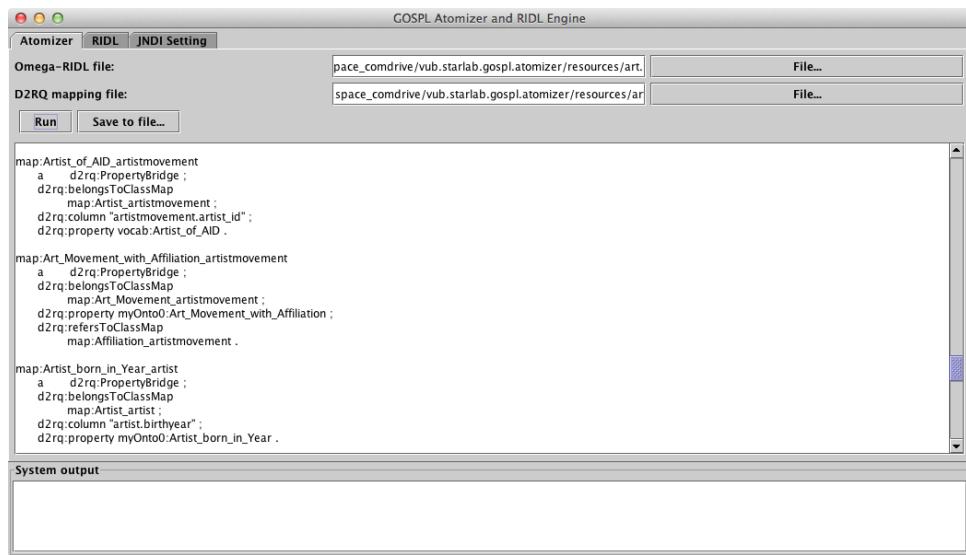


Figure 7.15: Screenshot of the Atomizer.

Figures 7.16 and 7.17 contain screenshots of a user interface built around a library developed for R-RIDL. In Figure 7.17, the Ω -RIDL commitment contains a reference to the community, and thus also to the hybrid ontology and its agreed upon lexons. Also a SPARQL endpoint - using the predicates defined in the OWL implementation of the hybrid ontology - is given. The results are shown in columns. Values of the result set are either resources, or values. In the case of the former, a description of the resource is retrieved. In the case of the latter, the value is shown. In this example of the LIST statement, the resource returns a human-readable description of that resource as a Web page following Linked Data principles [BL98]. Figure 7.17, shows a FOR-LIST statement in R-RIDL. The LIST statement returns a set of instances, which can be regarded as a set of unary tuples. The FOR-LIST statement allows the user to create queries returning a set of tuples of arity $n > 1$.

⁷<http://jena.apache.org/>

⁸<http://www.d2rq.org/>

The screenshot shows the GOSPL Atomizer and RIDL Engine interface. The top menu bar has tabs for 'Atomizer', 'RIDL' (which is selected), and 'JNDI Setting'. Below the tabs, there are two input fields: 'Omega-RIDL commitment file' containing '.comdrive/vub.starlab.gospl.atomizer/resources/a' and 'SPARQL endpoint' containing 'http://localhost:2020/sparql'. A query input field contains 'LIST Work_Of_Art with_contribution_of Artist NOT with Gender with Code = 'M''. On the right, a 'Query' button is visible. The main window displays the results of the query. On the left, a sidebar shows a list of 'Work_Of_Art' resources, one of which is expanded. The expanded view shows the 'Description of http://localhost:2020/resource/Work_Of_Art/3' with its resource URI. Below this, a table lists properties and their values for this specific resource. The table has columns 'Property' and 'Value'. Properties include: is myOnto0:Artist_contributed_to_Work_Of_Art of <http://localhost:2020/resource/Artist/3>, is myOnto0:Artist_contributed_to_Work_Of_Art of <http://localhost:2020/resource/Artist/4>, is vocab:Price_of_Work_Art of <http://localhost:2020/resource/Price/350000/EUR>, myOnto0:Work_Of_Art_made_in_Year 1915, vocab:Work_Of_Art_of_WID 3, myOnto0:Work_Of_Art_with_Title Flamenco Singer, and rdf:type myOnto0:Work_OF_Art. A note at the bottom says 'Generated by D2R Server'. At the bottom of the interface, there is a 'System output' section.

Figure 7.16: Example of the LIST statement in R-RIDL

The screenshot shows the GOSPL Atomizer and RIDL Engine interface. The top menu bar has tabs for 'Atomizer', 'RIDL' (selected), and 'JNDI Setting'. Below the tabs, there are two input fields: 'Omega-RIDL commitment file' containing '.comdrive/vub.starlab.gospl.atomizer/resources/a' and 'SPARQL endpoint' containing 'http://localhost:2020/sparql'. A query input field contains 'FOR Artist with Gender with Code = 'F' LIST Name of, Year_of_birth_of'. On the right, a 'Query' button is visible. The main window displays the results of the query. On the left, a sidebar shows a list of 'Artist' resources, one of which is expanded. The expanded view shows the 'Description of http://localhost:2020/resource/Artist/4' with its resource URI. Below this, a table lists properties and their values for this specific resource. The table has columns 'Property' and 'Value'. Properties include: myOnto0:Artist_born_in_Year 1885, myOnto0:Artist_contributed_to_Work_Of_Art <http://localhost:2020/resource/Work_Of_Art/3>, myOnto0:Artist_having_Name Sonia Terk, vocab:Artist_of_AID 4, is myOnto0:Gender_of_Artist of <http://localhost:2020/resource/Gender/F>, and rdf:type myOnto0:Artist. A note at the bottom says 'Generated by D2R Server'. At the bottom of the interface, there is a 'System output' section.

Figure 7.17: Example of the FOR-LIST statement in R-RIDL

7.5 Conclusions

This chapter demonstrated the tool that supports the hybrid ontology engineering framework, method as well as the use of gloss evolution and commitments presented in the previous chapters. The tool offers a zero-install approach by providing a web-based collaborative environment. The hybrid ontologies developed with this tool allow ontologies

to be “implemented” in DL-Lite_{A,id} for so-called downstream usage of the ontologies.

For this research, two prototypes that act as clients for the GOSPL server have been developed; one to support discrete gloss evolution and one for querying RDF via R-RIDL. These two prototypes have yet to be integrated with the collaborative platform. A reason for the first is efficiency; the machine learning algorithm for the natural language processing takes a considerable amount of time to train, which would have been a nuisance in a web environment. Accepted (structured) glosses need to be retrieved with the client in order to distill lexons, constraints and instances from the gloss.

A second reason is a limitation of the technology adopted. The server needs to be deployed as a jar. All libraries and resources (e.g. training data) required to support the NLP, as well as querying and reasoning over RDF with SPARQL need to be made available to that jar. That would mean putting third party libraries in the JBoss library folder and deploy the resources next to the jar. Integration is thus postponed until a more elegant solution to this problem has been found. Until now, this has not yet been a problem to validate the ideas.

Chapter 8

Case Studies

In this chapter, we present two large ontology engineering experiments in which the GOSPL method and tool were adopted. Both experiments are part of larger case studies in which GOSPL is deployed. To evaluate the tool, we assessed the user satisfaction concerning system usefulness, information quality and interface quality. As for the method, an analysis of the interactions and the resulting ontologies provide insights on the method's effectiveness.

Before we report on the results of the two experiments, we first present the user satisfaction survey we have adopted for this thesis. The results of the first and second experiment were partly reported in [CD12] and [DC13] respectively.

8.1 Assessing the User Satisfaction with PSSUQ

Usability is defined by the ISO-9241 standard [ISO98] as *the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in particular environments*. Usability is a key factor in making the (computer) systems easy to learn and to use. Usability testing has been extensively studied and applied by Lewis [Lew12] at IBM Software Group. The results of the usability testing improve the design of a system by evaluating the organization, presentation and interactivity of the system interface.

Satisfaction was measured using the standardized Post-Study System Usability Questionnaire (PSSUQ) [Lew93, Lew02] developed by IBM. PSSUQ originally consisted of 19 questions, each question being a statement about the usability of the system. We used the short version, which consists of only 16 questions. Participants need to answer each statement using a Likert scale of 7 points, where 1 indicates that the user “strongly agrees” with the statement, whilst 7 indicates that the user “strongly disagrees” with it. PSSUQ is based on a comprehensive psychometric analysis, providing scales for three sub-factors, namely: (1) system usefulness; (2) information quality; and (3) interface quality. The short (and most recent) version of PSSUQ, illustrated in Table 8.1, was used, in order to save time. In Table 8.1, the questions correspond with the sub-factors as follows:

- System usefulness: the average of items 1 through 6;
- Information quality: the average of items 7 through 12;
- Interface quality: the average of items 13 through 15;
- Overall: the average of items 1 through 16.

The reason for choosing PSSUQ for this study is mainly because of the rich information it provides, with little effort from the user, and the extensive IBM documentation and

Table 8.1: PSSUQ - short version [Lew12]

Item	Item Text
Q1	Overall, I am satisfied with how easy it is to use this system.
Q2	It was simple to use this system.
Q3	I was able to complete the tasks and scenarios quickly using this system.
Q4	I felt comfortable using this system.
Q5	It was easy to learn to use this system.
Q6	I believe I could become productive quickly using this system.
Q7	The system gave error messages that clearly told me how to fix problems.
Q8	Whenever I made a mistake using the system, I could recover easily and quickly.
Q9	The information (such as online help, on-screen messages and other documentation) provided with this system was clear.
Q10	It was easy to find the information I needed.
Q11	The information was effective in helping me complete the tasks and scenarios.
Q12	The organization of information on the system screens was clear.
Q13	The interface of this system was pleasant.
Q14	I liked using the interface of this system.
Q15	This system has all the functions and capabilities I expect it to have.
Q16	Overall, I am satisfied with this system.

experience for the statistics it can provide. Besides the 16 items in the test, the test participants can make comments and elaborate on their answers. Based on these comments, conclusions are drawn and recommendations for improving the human-system interaction were provided.

The goal of the test is to evaluate the usability of GOSPL in two dimensions: formative and summative, from the user satisfaction point of view. The formative usability testing aims at identifying the usability problems of the tool. The summative usability test consists of a series of measurements (e.g. effectiveness, efficiency, satisfaction) that are performed in order to compare the usability results against a set of predefined objectives.

8.2 Case 1: Annotating Cultural Events in Brussels

The Open Semantic Cloud for Brussels (OSCB)¹ project started on February 1st, 2011 and aims to cover Brussels, metaphorically speaking of course, with a cloud of structured and interlinked information elements produced by “atomizing” a collection of relevant databases and other resources into RDF triples. The first use case in the OSCB project aims at semantic linking cultural events in Brussels.

In this project, access was given to two datasets: one by Agenda.be in XML and a relational database dump by BOZAR. Agenda.be² is an initiative of the Foundation for the Arts and Visit Brussels to provide a federated, up-to-date overview of cultural events

¹<http://www.oscb.be/>

²<http://www.agenda.be/>

in Brussels. BOZAR³ is the Centre of Fine Arts in Brussels and stores information about their cultural events and those of affiliated organizations in their own database. Both data sets contain information about venues and cultural events mostly in Brussels. The level of detail, and perspective on this domain is very different when analyzing both data schemas. One of the goals is thus the creation of an ontology suitable for annotating both data sets. Next to an ontology mainly developed by the consortium in order to support this use case, an experiment in the same domain was conducted. The resulting ontologies of that experiment were subsequently taken into account to refine the ontology built by the consortium.

The experiment involved 43 people (1 supervisor and 42 students) and lasted about three months (from March 2012 to May that same year). All students took part in this experiment as part of an obligatory course on ontologies and ontology engineering, which is part of the MSc. in Applied Sciences and Engineering: Computer Science curriculum of the Vrije Universiteit Brussel. Most participants had a background (BSc) in computer science, software- or hardware engineering. All participants were asked to form groups of 3 to 5 people and were free to choose with whom they wished to collaborate. 11 groups were formed with one group consisting of only 2 people, which were working students and had limited possibilities working during office hours. Each group also had to choose a particular type of cultural event. An overview of each group (group-size and topic) is given below.

#	Topic	Size	#	Topic	Size
1	Theater plays	4	7	Exhibitions	5
2	Movies	2	8	Movies	4
3	Concerts	4	9	Concerts	4
4	Festivals	4	10	Concerts	4
5	Exhibitions	4	11	Lan Parties	4
6	Festivals	3			

First the groups were asked to develop their own portal for displaying information about cultural events in their domain. All groups were then asked to develop hybrid ontologies to enable 1) semantic interoperation between the different portals, 2) enable federation of the data contained in the different portals. Groups were also asked to use the developed ontologies to annotate the BOZAR relational database and demonstrate the use of the annotated data.

The experiment resulted in 46 different community commitments. This amount seems excessive, but the participants had a natural tendency to separate concerns and an important part of the communities became redundant as time passed by. The first could be very well explained by their background; most had a background in computer science or engineering. As for the latter, we will later on explain why communities will not be removed as one can never know which applications already or ever will commit to those community commitments. More details about this decision will be given while some of the usability issues identified in the usability study will be addressed.

³<http://www.bozar.be/>

In total, all participants were responsible for almost 7200 social interactions on this platform including setting up communities, starting discussions, closing discussion, giving replies, and voting.

8.2.1 Summative and Formative User Satisfaction.

The satisfaction test was undertaken by a group of 15 out of the 42 participants. The overall usability testing was carried out both implicitly by analyzing the data logs and the user-system interactions and explicitly, by collecting the user feedback via several questionnaires. We would like to note that forms filled in for user satisfaction assessment had no influence on the grade the students obtained for their assignment, which has been clearly communicated to the student.

Summative User Satisfaction. The results delivered by the PSSUQ questionnaire are as follows: the overall (average) user satisfaction shows a value of 3.4, the system usefulness 3.1, the information quality 3.9 and the interface quality 3.4 (see Table 8.2). There were four groups of volunteers involved in the study, denoted as: A, B, C and D. The group most satisfied overall is group C, the group most satisfied with the system is the same group C, the group most satisfied with the information quality is group B and the group most satisfied with the interface quality is group A. The overall satisfaction ranking over the four groups is therefore: group C (most satisfied), group B, group D and group A (least satisfied).

Table 8.2: Summative user satisfaction

Metric	A	B	C	D	Average
System usefulness	3.4	3.0	2.8	3.1	3.1
Information Quality	4.1	3.8	3.9	3.9	3.9
Interface Quality	3.1	3.6	3.3	3.6	3.4
Overall	3.6	3.4	3.3	3.5	3.4

These results are compared with the system logs indicating the number of user-system interactions per group, in order to justify the differences on the satisfaction levels per group. The logs show that the average number of interactions per group corresponds to the overall satisfaction per group: 1) group A - 94 interactions⁴; 2) group B - 270 interactions; 3) group C - 350 interactions; and 4) group D - 176 interactions. These results are also correlated with the quality of work of each group, reflected also by the final project grade. It is curious, however, to observe that the group which has used the interface least often (group A) is most satisfied with the interface quality, but are overall least satisfied with the system.

Formative User Satisfaction. The problems identified by the users in the comments section of each item are illustrated in Table 8.3. The problem they faced most often

⁴Averages of interactions are based on the whole group, but only three of the four people in that group participated in the survey. The average number of interactions for the three users that participated is 85.

Table 8.3: Summative user satisfaction. INF = Information Quality, INT = Interface Quality and SYS = System Usefulness. Number of reports is indicated on the right.

	Usability problem	Nature	#
1	The (error) messages displayed by the system were often not clear to the user. There was in general no online help or documentation available.	INF	6
2	There is no “undo” or “edit” option available	INF, INT	5
3	No (top menu) link to the current community in the discussion page	INT, INF	5
4	It took a while to understand how the system works	SYS	1
5	Sometimes, listing items in the dynamic tables did not go well when after returning to a page it displayed the first item again.	INT	3
6	There was no “delete” option for the communities that “died” during the process.	INF, INT	2
7	The user name is not clear (just email addresses appear)	INT	1
8	Sometimes, more clicking necessary than that one would expect (e.g. when browsing through several discussions).	SYS	1

was linked to the absence of intelligible (error) messages and online help/documentation. Other often cited problems are related to the lack of “edit” and “undo” options; also, the users mention the need for a better organization and links regarding the (current and visited) communities. Other problems include clarification of the user names, the “back” button or the missing request to add gloss in the list box.

Taking the satisfaction results obtained from PSSUQ and the user comments, the following conclusions are derived: out of the three sub-factors identified by PSSUQ, the system usefulness measure performed best (3.1); the information quality of the system is the sub-factor that needs most improvement (3.9). This also corresponds to the (information quality-related) problems most cited by the users: usability problems 1, 2 and 3 in Table 8.3. Therefore, summative and formative usability testing deliver similar results when it comes to user satisfaction in this study. The next section will describe how the aforementioned problems were tackled for the second experiment.

8.2.2 Impact of Glosses on Meaning Negotiation

Concerning the method we observed that terms that were articulated before lexons around this term were entered into a community commitment were less likely to have changes in their formal description than those that were not [DV13]. We analyzed the interactions involving terms in a community with the following criteria: (1) the term had to be non-lexical, meaning that instances of this concept cannot be printed on a screen, only its lexical attributes can, (2) the term was the subject of at least 4 interactions (not including gloss-equivalences and synonyms, thus focusing on the formal and informal descriptions around this term), and (3) the term took part in at least one lexon. We took into account terms with a fair amount of activity. This is due to the fact that the communities employed terms that are only relevant to their application, and therefore only inspired discussions within that group. These discussions are not interesting as the community tended to agree on what has been decided for their application.

We then analyzed how much of these terms changed in terms of their formal description when the gloss is provided. With these criteria, we identified 49 terms. Of these 49 terms, 38 started with the natural language description as described by the GOSPL method. Of these 38 terms, 11 of them had changes in their formal description (29%). Of the remaining 11 terms that did not start with the informal description, 5 of terms had changes in their formal description (45%).

The reason we left out lexicals is that they often play an attributive role. Lexons are supposed to be entered when at least one of the terms is articulated. At the start, the key terms are often described first. And when the second term concerns a lexical in an attributive role, the community tends to agree on the meaning of this attribute based on the label of that term. If we were to take lexicals into account, we again observe that terms that did not start with an informal description are more likely to change its formal description: 18 terms out of 46 started with a gloss and 6 terms out of 12 did not start with a gloss.

From this observation we can conclude that aligning the meaning of labels within a community by means of natural language definitions does contribute to more stable evolution of formal descriptions of those concepts. Stable here means that previous agreements are less likely to be reverted.

8.2.3 Using the Ontology

Participants then each annotated a part of the relational database provided by BOZAR and demonstrated how the ontologies (actually, the OWL implementation thereof) are used to retrieve and use the annotated information. One example⁵ is shown in Figure 8.1, in which information about musical pieces and composers of musical events in BOZAR are retrieved and later on were enriched with information in DBpedia⁶.

8.3 Changes in the Prototype

After the first experiment, the following functionalities have been added to the GOSPL prototype:

- Explicit social processes for defining the key terms and goals that constitute the semantic interoperability requirements of a community.
- Social processes for communities to agree that terms and roles in formal descriptions refer to the same concepts as classes and properties in other ontologies stored somewhere on the Web (e.g. OWL ontologies).
- Tool support for collaboratively managing application commitments.
- An RSS feed such that users did not need to check the platform for any new discussions and observe the activity by means of an RSS reader. The RSS feed has

⁵Developed by S. Van Laere, P. Stroobants, K. Tanaka and W. Van Rossem.

⁶<http://dbpedia.org/>

The screenshot shows a web page with the following elements:

- Vrije Universiteit Brussel** logo and name.
- DBpedia** logo with a building illustration.
- SPARQL (W3.ORG)**, **BOZAR (BOZAR.BE)**, and **DBPEDIA (DBPEDIA.ORG)** links.
- TERUG** link.
- INFORMATION** section with links to:
 - SPARQL Queries
 - Bozar Database
 - DBpedia Project
 - Demonstration
- SPARQL queries on the Bozar DB and DBpedia** heading.
- A Demonstration** heading.
- Demonstration A** section listing numerous SPARQL query results, including titles like "00:00", "3 Lieder (Cäcilie, op. 27/2 - Befreit, op. 39/4 - Zueignung, op. 10/1)", and "Der Rosenkavalier, Wälzersuite Nr. 1, op. 59".
- Demonstration B** section.
- Terug naar boven** link at the bottom right.

Figure 8.1: Using the annotated BOZAR database.

for each community commitment a separate channel, allowing one to filter to the community of interest.

- A reputation framework, reported in [DN13]. The reputation framework provides users how well he or she performs with respect to following the method. The reputation framework also took into account an evaluation of other users on a user's action. The users were presented "scores" as to encourage them to do better.

The problems reported in the first experiment shown in Table 8.3 were addressed as follows:

Problem 1. While teaching the method to the participants, they were offered a document and slide set (available online) in which the method and tool were explained. A running example for the creation of an application commitment was also provided.

Problem 2. Regarding the "delete" and "edit" options, the students were explained that we were reluctant for allowing editing of posts and comments (e.g. to prevent abuse). Most of them would be content with a feature that allows a post to be edited within several seconds. The problem reported here was thus not so much to undo an action, but rather to edit mistakes such as typos. Also, the outcome of a discussion sometimes differs from the initial proposition. For some social interactions, the users are now able to conclude with the final outcome.

Problem 3. Users mentioned that there is no link to the current community in the discussion page. Such a link did exist, but the users did not notice it. Even though it was styled as a link with the website's style sheet, the users seemed to have overlooked it. The style of the prototype has been adapted and the link to the community is made more obvious.

Problem 4. The availability of online documentation should solve that problem.

Problem 5. This remark basically boiled down to search parameters being stored in a session such that users did not have to enter the same filter every time they leave the page. This was easily solved by storing the filters in a cookie.

Problem 6. There was a wish from the users to delete communities, in particular the communities that became obsolete (or – as the users put it – "dead") as the different communities evolved. Even though they understood that even those communities might once again become active, they would be happy to be able to "filter" the dead communities from the list and toggle that filter. We did not wish to provide such a feature, as one can never know when a particular community can have an uptake. We therefore did not provide such functionality for the next experiment.

Problem 7. We do not request users to provide an additional username, instead we strongly encouraged the users to use their institution's address or an address containing their names.

Problem 8. In communities with many discussions, browsing through the different discussions could have been cumbersome. This has been partly tackled by storing the filters in a cookie.

8.4 Case 2: Research Information Systems

Research information systems – also called Current Research Information Systems [JA10] – are information systems used by the various actors in the research process and their use ranges from “*the documentation of research projects and their results over easing the management of research to simplifying research assessment*” [BM11]. Research information systems require a clear focus on a bounded set of indicators or items of interest that are agreed upon by the stakeholder group to ensure the proper support of aforementioned research process [BM11]. Information concerning research activities is often stored at different places on the Web and across several organizations (e.g. databases owned by each knowledge institution or federated databases by each public administration).

The goal of the second case is to test the use of a knowledge management platform for the creation of ontologies for publishing the data of a research information system as RDF on the Web. To this end, we were given a relational database dump by the Department of Economy, Science and Innovation of the Flemish Government (from here on called EWI) of their FRIS portal⁷. FRIS stands for Flanders Research Information Space and aims to provide a portal for all research activities within Flanders. A problem, however, of this database is that it is not linked with other datasets found on the Web, such as DBLP⁸.

The experiment involved 36 students and lasted about three months (from March 2013 to May that same year). All students took part in this experiment as part of an obligatory course on ontologies and ontology engineering, which is part of the MSc. in Applied Sciences and Engineering: Computer Science curriculum of the Vrije Universiteit Brussel. Again, most participants had a background (BSc) in computer science, software- or hardware engineering. All participants were asked to form groups of 3 to 4 people and were free to choose with whom they wished to collaborate. 10 groups were formed, with one group consisting of only 2 people, which were participants who arrived late. Each group also had to choose (a part of) an existing research information system. Details on each group are given below.

#	Application	Size
1	DBLP	4
2	FRIS (focus on organizations)	4
3	Microsoft Research	4
4	ACM DL	4
5	CORDIS (focus on projects)	4
6	IBM Research	4
7	Journal TOCs	3
8	CORDIS (focus on funding programs)	4
9	CORDIS (focus on organizations)	3
10	CORDIS (focus on funding programs)	2

First, the groups were asked to develop their own system based on the existing application they have chosen by “reverse engineering” the conceptual schema. All groups were then

⁷<http://www.researchportal.be/index.html>

⁸<http://www.informatik.uni-trier.de/~ley/db/>

Table 8.4: Summative user satisfaction: results looking at the groups, average of all complete groups, average all respondents and results of the previous study.

Metric	A	B	C	D	E	Average groups	Average all users
System usefulness	2.9	2.9	4.0	3.4	2.3	3.1	3.2
Information quality	3.2	3.5	4.3	3.3	3.8	3.6	3.5
Interface quality	3.1	2.8	5.0	3.1	2.3	3.3	3.3
Overall	3.1	3.1	4.4	3.3	2.8	3.3	3.3

asked to develop hybrid ontologies to enable 1) semantic interoperation between the different systems, 2) enable federation of the data contained in the different system. Groups were also asked to use the developed ontologies to annotate the FRIS relational database and demonstrate the use of the annotated data.

The experiment resulted in 24 different community commitments. Again, the participants had a natural tendency to separate concerns and a part of the communities became redundant as time passed by. In total, all participants were responsible for around 6890 social interactions on this platform including setting up communities, starting discussions, closing discussion, giving replies, and voting. We would like to note again that forms filled in for user satisfaction assessment had no influence on the grade the students obtained for their assignment, which has been clearly communicated to the student.

8.4.1 Summative and Formative User Satisfaction.

The satisfaction test was undertaken by a group of 23 out of the 36 participants. The overall usability testing was carried out both implicitly by analyzing the data logs and the user-system interactions and explicitly, by collecting the user feedback via several questionnaires.

Summative User Satisfaction. The results delivered by the PSSUQ questionnaire are as follows (cfr. Table 8.4):

- System usefulness: the average of all groups remained the same as in the previous study. The overall average, however, had a small decline in satisfaction with 0.1 points.
- Information quality: compared with the previous study, the system performed better in terms of information quality with 0.3 points for the groups and 0.4 points for the overall average.
- Interface quality: for both the group average and the overall average, the interface quality was deemed more satisfying with 0.1 points compared to the previous study.
- Overall satisfaction: the system performed slightly better in terms of user satisfaction.

The average number of interactions for each group are: 263 for group A, 423 for group

B, 175 for group C, 89 for group D and 224 for group E. In this experiment, we did not observe that overall user satisfaction did not increase with use. Group C was overall the least satisfied of all aspects. Although members of group C had “only” an average of 175 interactions per person, they were involved in most communities. The negative scores within that group could thus reflect the difficulty of having cross-community interactions.

Formative User Satisfaction. Of the 23 respondents, 17 have left comments. These comments were analyzed to pinpoint the problems of the tool. When indicating an occurrence, this corresponds with a respondent making a remark about the issue at least once. Some respondents commented about a certain issue multiple times in different comment sections of the survey. The problems identified by the users in the comments section of each item are as follows:

- Keeping an overview of the discussions (10 occurrences). Some proposals have been made to tackle this problem: 2 respondents proposed a central notification system, one respondent proposed the ability to follow the actions of a particular user, another proposed an RSS feed per community complementing the overall feed⁹. Other proposals were: identifying the “hottest” discussions, offering the changes after last login and even a search function over the whole system.
- The new version of the prototype offered possibilities for creating and managing versions of application commitments, which are built according to a particular grammar (5 occurrences). Users noted that the errors while parsing application commitments were often too obscure to be practical and had to rely too much on help.
- Correcting mistakes (5 occurrences). The changes made to the tool – to cope with mistakes or changes in a discussion – proved inadequate to improve the user’s satisfaction. They wished the ability to “undo” or “cancel” an interaction. One respondent also suggested the possibility to alter a comment.
- Problems creating constraints (4 occurrences). Surprisingly, the functionality of building constraints has not changed and yet 4 people reported that the construction of constraints was confusing. 2 of these 4 respondents also mentioned that the verbalization of these constraints were not clear.
- The voting mechanism (4 occurrences). 3 respondents wished the system would require a justification when one is against a proposal. With an additional respondent stating the voting system to be inadequate for stimulating the discussion.
- Even though documentation was available as well as a running example in the slide set, participants wished for documentation within the tool next to the material offered (5 occurrences) and more examples (4 occurrences). Three participants merely noted there was not enough documentation without providing further details.
- Availability of concrete (worked out) examples and tutorials next to the documentation (3 occurrences).
- Availability of help functionality within the tool rather than online in separate documents (2 occurrences).
- Lastly, there were 4 complaints of the back button resulting in a warning on a discussion page. This needed to be added as the reputation framework kept track

⁹Even though one could filter on channel

of the discussions visited by a user and one of the popular browsers not capturing the event of clicking the back-button properly. To this end, we asked the users whether they “wished to leave the page”, whereupon a click on the button “Yes” called the method for logging it.

8.4.2 Recommendations for Improvement

Taking the satisfaction results obtained from PSSUQ and the user comments, we drive the following conclusions: out of the three sub-factors identified by PSSUQ the system usefulness performed best. The users of this study seemed to be less satisfied than in the previous study. Information quality had a fairly important improvement in terms of satisfaction with respect to the previous study. Taking into account the complaints on error handling of the commitment manager (which was added to the prototype), we can conclude this is a very positive evolution. Both the interface quality and overall satisfaction evolved positively. The following steps will be taken to improve the system:

1. Investigating how the overview of all the discussions can be improved.
2. Improving the interface for managing the application commitments.
3. Allowing actions to be undone (i.e. “cancel” or “undo”) in case of error.
4. Improving the verbalization and forms for constructing constraints.
5. Participants complained that the voting system did not require users not agreeing to a proposal to justify their opinion. The goal of the voting system was to allow users to participate to discussions in a “lightweight” manner. After the experiment, however, we feel that the voting mechanisms did not contribute to the discussion and sometimes lead to confusion. We therefore will most likely remove the voting mechanism.

We observed a need for more worked out examples and the availability of help functionality within the tool rather than online in a separate document.

8.4.3 Using the Ontology

Participants then each annotated a part of the relational database provided by EWI and demonstrated how the ontologies (actually, the OWL implementation thereof) are used to retrieve and use the annotated information. One such application is the detection of conflicts of interest. This is particularly important for funding agencies that need to construct review boards for the evaluation of project proposals. The application¹⁰ depicted in Figure 8.2 depicts which person one most probably would not include based on past collaboration in projects and organizations. Publications were not taken into account since the dataset provided by EWI – at the time of writing – only contained some publications of one knowledge institution. The query to detect paper collaborations, however, can easily be extended.

¹⁰Developed by W. Bellen, J.P. Hubrecht, B. Moerman, and L. Van den Heuvel.

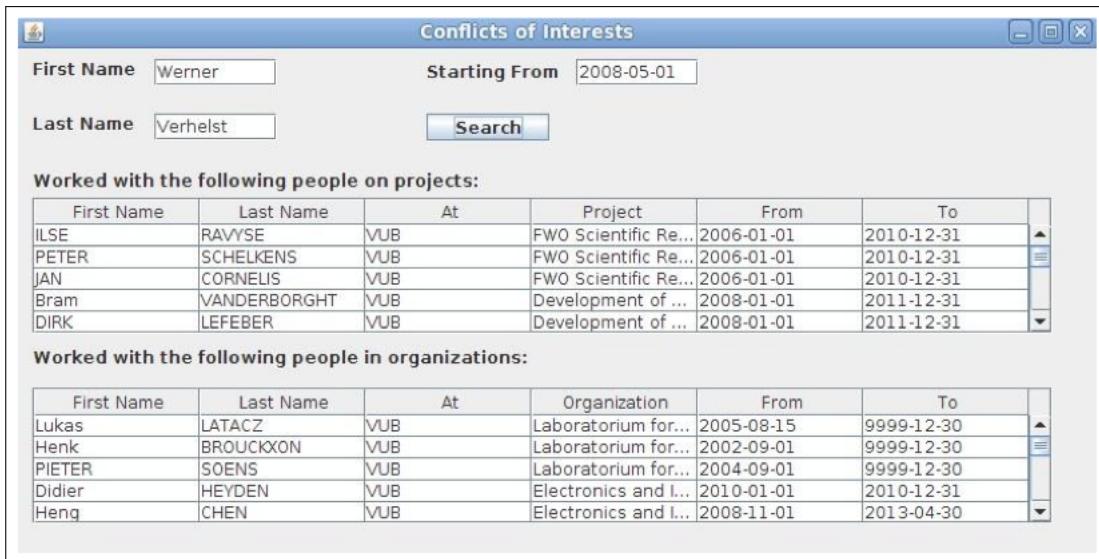


Figure 8.2: Using the annotated EWI database.

8.5 Discussion

In this chapter, we described two experiments in which the GOSPL method and tool were adopted. On overall, the users were successful in achieving semantic interoperability between their systems and the existing system we provided them with. Most problems reported concerned usability and a lack of documentation/guidance within the system, which we will thus need to improve. Between the two experiments, the tool was refined and additional functionality to the users was provided. Notwithstanding the problems experienced by the users with the new functionality provided in the second version of the tool, the tool did manage to improve its user satisfaction in terms of system usefulness, information- and interface quality. We also showed that the articulation of concepts with glosses does have a positive impact on agreements on the formal descriptions of these concepts.

Every method needs to be teachable, repeatable and traceable. The GOSPL method for hybrid ontology engineering complies with all three criteria.

Teachable. The DOGMA framework for ontology engineering, on which GOSPL is based upon, drew inspiration from database design methods and techniques such as NIAM and ORM. NIAM/ORM and therefore also DOGMA are fact-oriented approaches in which stakeholders communicate fact types expressed in natural language. Fact-oriented approaches differ from frame-oriented approaches (e.g. UML) by eliminating the distinction between attributes and relations; everything is a fact between concepts. This reduces the learning curve. The use of natural language to express these fact types also facilitates the knowledge elicitation processes.

Repeatable. Ontology engineering processes and possible interactions have been formalized and therefore repeatable by a community who have been trained or have

access to the documentation. Because the method is repeatable, the third aspect – traceability – is a logical consequence.

Traceable. In order to support ontology evolution, one needs to record the changes over time. As in software engineering, it is a good practice to also document why certain changes have been made. The different evolution operators on the formal parts are therefore traceable (who, why, when, etc.), what is not often captured is the whole process of reaching a decision, with GOSPL, the social processes leading to a change in the ontology will have been formalized and stored for future reasoning.

Chapter 9

Conclusions

This chapter provides a reflection on the results presented in this thesis as well as future work that yet has to be done. First, the research questions are restated, and results are summarized and discussed according to each chapter. Finally, the limitations of the work conducted in this thesis are touched upon as well as the theoretical and practical research challenges to be addressed in future research will be described.

9.1 Contributions

This thesis addressed the following research questions:

- Q1 What are hybrid ontologies?
- Q2 How are hybrid ontologies constructed?
- Q3 How can hybrid ontology construction be supported in a necessarily complex collaborative setting?
- Q4 How does the evolution of a natural language definition of a concept influence the formal part of a hybrid ontology?
- Q5 How does the annotation of application symbols drive the hybrid ontology engineering process?

These were translated in the following research objectives.

- O1 **Provide an analysis of the state-of-the-art on ontology engineering with respect to the use of social processes and the use of natural language definitions to support the evolution of meaning agreements within a community.**

Starting from a definition of ontology used in computer science that has been refined in this thesis, Chapter 2 provides a survey on ontology languages and ontology engineering methods and tools. In this survey, it became apparent that most method took little or no attention to the use of natural language definitions of concepts (called *glosses* in this thesis) for evolving ontologies. As natural language definitions are a day-to-day tool for humans to align their thoughts and avoid misunderstanding, a motivation for investigating an ontology engineering method in which those definitions leading to agreements within a community emerged.

O2 Develop the notion of hybrid ontologies to support social processes with natural language definitions for concepts.

Chapter 3 provided a more precise definition for ontologies in computer science, in which the community aspects were emphasized. Starting from an existing fact-oriented ontology engineering method in which knowledge is grounded in natural language, communities were given a more prominent role in ontology artifacts by providing communities with a special linguistic resource which drives their meaning agreements. This resulted in the notion of hybrid ontologies: ontologies in which concepts are both formally described by means of a mathematical formalism and natural language definitions. Chapter 3 also described and provided a motivation for the nature of “sameness” agreements of natural language definitions and labels in the formal descriptions. The social interactions that take place in those communities were described, which provides one a framework for hybrid ontology engineering.

Chapter 3 furthermore motivated the need for two types of ontological commitments: community- and application commitments. Community commitments contain a selection of *shared* lexons and constraints on top of these lexons necessary for a successful semantic interoperation between the different information systems. Application commitments commit to one or more community commitments next to an additional selection of (enterprise-specific) lexons and constraints with mappings from application symbols to terms and roles in that selection. The *double articulation principle* [SMJ02] in GOSPL is not violated, as there is still an explicit separation between the lexon base containing all plausible fact types and the restatement of these lexons as selection in both community- and application commitments. In fact, when a community agrees to remove a lexon, some stakeholders can decide to retain this lexon in their application commitment since it has not been removed from lexon base.

O3 Develop a method for hybrid ontology engineering.

Chapter 4 adopts the hybrid ontology engineering framework described in Chapter 3 for proposing a method for hybrid ontology engineering called GOSPL, which stands for Grounding Ontologies with Social Processes and natural Language.

The method prescribed constraints that need to be fulfilled in order for some community interactions to take place and the outcome of these community interactions to be executed. Key in this chapter is that the community aligns their thoughts on their shared concepts by means of glosses before formally describing these concepts. The agreement of natural language definitions thus precedes the formal description of concepts. The conditions for community interactions concerning the creation of lexons and constraints are in terms of the existence of these natural language definitions.

O4 Determine which parts of the method can benefit from the evolution of those natural language definitions.

Chapter 5 provides an answer for the 4th research question by describing the notion of discrete gloss evolution, thereby reaching the 4th objective. Discrete gloss evolution assumes that glosses to describe concepts evolve over time for a reason and that

this evolution should have an impact on the formal description of these concepts. Rhetorical Structure Theory (RST) [MT88] provides a list of possible modalities sentences have on other sentences. In this thesis, a selection of these modalities was adopted for discrete gloss evolution as well as the definition of additional modalities specific for GOSPL.

The adopted modalities give a hint on the type of lexons, constraints and instances than can be elicited from natural language definitions. Discrete gloss evolution thus allows the start of several interactions within the community to agree on these lexons, constraints and instances that should naturally follow from this particular gloss. In the end, the verbalization of the lexons and constraints of a concept should as closely resemble the gloss of that concept or, in other words, the community should deep the verbalization of lexons and constraint of this concept to be gloss-equivalent with their gloss of that concept.

O5 Determine which parts of the method can benefit from the annotation of existing (legacy) systems.

Chapter 6 determined how commitment – and in particular application commitments – are used to steer social interactions within a community by pinpointing the problems in the formal descriptions of concepts. This was done in several steps. First, an appropriate logic was adopted in which the lexons and constraints in commitments were translated in a lossless manner. Lossless means there exists a bijective mapping between all possible permitted populations of the lexons and constraints, and the translation thereof. This translation is then used to annotate the legacy datasets.

Secondly, an appropriate interpretation function for constraints was defined to give an answer to the tension field between the open and closed world assumptions. This is necessary as information systems that need to properly interoperate in a semantic way require some of the shared and agreed upon constraints to be followed by all systems. Queries are then formulated in such a way that for each constraint, counterexamples are sought. These counterexamples provide the communities working on their community commitment to analyze some of the consequences of the formalization discussed and allow them to take appropriate measures.

Finally, with the translation of lexons and constraints into an ontology implementation language and the annotation of the legacy databases with that translation, the community can use the lexons to explore the annotated data. For this third part, an existing conceptual query language for binary fact types was adopted. Queries in this query language are translated into intermediate SPARQL queries to obtain the results. The goal of this work was not to provide yet another query language for building services, but rather to have a truly conceptual layer on top of annotated datasets.

O6 Develop a tool that is based on this aforementioned method.

Chapter 7 presented a description of a web base collaborative tool for hybrid ontology engineering based on the methods described in Chapter 4 and the use of discrete

gloss evolution and application commitment, respectively presented in Chapters 5 and 6.

O7 Evaluate the contributions and – by consequence – their conceptual design.

Finally, we presented the results of a usability study of the method and prototype in a use case in the cultural events domain involving 40+ users in Chapter 8. Users were overall successful in achieving semantic interoperability. They were thus able to achieve all agreements necessary for annotating legacy datasets. There were little remarks on the method; mostly interface issues and unexpected behaviors were reported. Analysis of the data did show that concepts that were articulated before a formal description was given tended to be more stable in their description than concepts that were first formally described.

The contribution of this thesis was a method and tool for collaborative and distributed hybrid ontology engineering, in which communities own the ontology and agreements lead to ontology evolution. In other words, it is the outcome of these interactions (e.g. dialogues) that gradually evolves the ontology to better approximate the community’s reality to support semantic interoperability between their information systems.

9.2 Limitations and Future Work

- **Lossless schema transformation of commitments in more expressive DLs.** A lossless transformation of community- and application commitments into $\text{DL-Lite}_{A,id}$ was described in Chapter 6. The choice for this particular DL was motivated by its fair expressiveness (certainly with respect to relational databases) as well as the MASTRO [CDGL⁺11] framework which allows for efficient querying and reasoning of relational databases annotated with DL ontologies in that dialect. This DL dialect was expressive enough for (and actually intended for) supporting unique, total identifiers for concepts. More expressive DLs, e.g. allowing for a translation of inclusive mandatory constraints, are not tractable. As future work, an investigation of translating these “problematic” constraints into these DLs and providing a proof for lossless schema transformation would be in place.
- **Inclusion of non-binary fact types in DOGMA and GOSPL.** During experiment conducted in 2011 and 2012, of which results are reported respectively in [DM11] and [DM12], the participants where taught that unary fact types could be easily transformed a lossless manner into a binary fact type in which the role is played exactly once by the object type and the number of instances of the newly introduced object type is limited to two.

During the experiment, however, the participants did state that the introduction of unary fact types would be welcome, as unary predicates are useful to describe

what SBVR [OMG09] calls *characteristics*¹: abstraction of a property of an object or set of objects that serve as qualifiers, e.g. “being green”, “is terminated”, etc.

One can imagine that an interface for hybrid ontology engineering could support automatic translation of unary fact types into lexon (and the necessary constraints). However, this would render the creation of application commitments more difficult as this translation need to be transparent. Another solution would be the extending the DOGMA framework to support unary predicates. This extension would be straightforward: redefining the lexon base, extending the g_1 function to cope with all sorts of fact types, defining new social processes, and so on. The translation into a DL and a Ω -RIDL language to support unary fact types will, however, require more effort.

- **Classifying users according to their social interactions and evolution of application commitments.** Future work includes the investigation of automatically and dynamically assigning user-roles (and therefore responsibilities) to members in a community on the collaborative platform by mining their social interactions and evolution of their application commitments. In previous work reported in [DLDP09], it was already shown that simple data mining techniques could be applied to classify users according to their social interactions. This work, however, was preliminary and needs to be further developed. This research direction is important as quite a few methods assume an exhaustive set of user-roles (to start with).

However, we assume that the user-roles may vary over communities; not rejecting that commonalities across communities cannot be found. It would therefore be beneficial to identify emerging types of users in a community, label them and then assign them processes, responsibilities accordingly. Examples would be to identify community leaders who would have the right to close and record the outcome of a discussion. By classifying users, one could also automatically configure a social interaction that needs to take place to work out a concept in the community commitment by looking for members with different competences.

- **Capturing agreements outside the GOSPL prototype.** One important limitation of the GOSPL tool is that it is not able to capture agreements made outside the system. Indeed, when agreements need to be made, nothing is better to have the stakeholders around a table for a face-to-face meeting when that is feasible. Nowadays, even teleconferences with video such as Skype allow people to sit virtually together to discuss matters. While discussing, people use all sorts of means to structure and get their thoughts across: mind mapping, brainstorming, searching for documents, and so on. All these interactions between community members as well as the man-machine interactions are useful to store, reason upon and even use for mining user categories as mentioned in the item above. To this end, one could draw inspiration from frameworks for computer supported cooperative work such as presented in [LCD08]. In [LCD08], a framework was presented in which all sorts of existing tools could be plugged in and all interactions between users and

¹A feature that the Business Semantics Glossary (<http://www.colibra.com/>) supports.

man-machine is logged. This framework could be adopted to provide the community to adopt teleconferences, mind mapping tools and the like to reach agreements to develop the hybrid ontology to eventually store the outcome on the GOSPL platform.

- **Capturing the dependencies in meaning agreements across communities.** An aspect not addressed in this thesis is the reason two or more communities agree on labels or glosses referring to the same concept. Reasons could be the wish to reuse a formal description of a concept already defined elsewhere. Another reason might be the specialization of a concept (i.e. creating a subtype), or even extending an existing concept. This all depends on the semantic interoperability requirements of the community.

Related is the work of [DdMM07], which introduced the notion of *context dependencies* in the DOGMA framework. Context dependencies constrain the possible relations between the entity described and its context. For instance, given the term *task* and its relations in some context γ (which is called a template by the authors), a *specialization dependency* states that each other context with a specialization dependency on *task* must provide a specialization of that template. In this case, that could mean the relations in that template are refined with subtypes of each of the concepts in that relation. [DdMM07] provided four simple dependencies of which one is more concerned on the versioning of ontologies. Context dependencies were used to let multiple organizations define their perspective, e.g. by means of specializing a template, and then negotiate a common agreement from those perspectives via a meaning evolution support system [dMDM06].

In GOSPL, the context identifiers are limited to communities and there is an explicit distinction between community- and application commitments. However, the notion of *community dependencies* could start from the work done by [DdMM07]. The community dependencies between commitments (both community and application) emerge from and need to be stored in the outcome of social interactions.

Rather than seeing all commitments as one big graph via the synonyms and gloss-equivalences introduced by all communities, one can reason over these dependencies to consult parts of the graph. For instance, given community $\gamma_1 \in \Gamma$, γ_1 's community commitment is considered reference complete if and only if all terms in the community commitment are referable, or one of the synonyms in another community commitment is referable, regardless whether that the other community commitment is referable as well.

Appendix A

7-step Algorithm

This appendix presents an algorithm that transforms (maps) an ORM schema losslessly into a normalized relational database schema with constraints added.

A condition for this algorithm to work is to have a RM-reference complete ORM schema. An ORM Schema is said to be RM-reference complete if every object type (OT) in that schema RM-referable. An object type A in an ORM schema is RM-referable if and only if either:

- A is a lexical object type (LOT);
- A is a non-lexical object type (NOLOT) and
 - A has a unique simple reference (see Definition 11);
 - A has a unique composite reference (see Definition 12);
 - A is the subtype of an RM-referable object type.

Note that any object type may be referable in more than one way and that non-lexical object types have a reference declared at type level, which means that the reference is expressed in terms of object types and declared constraints only. In other words, its reference is independent of the schema's population.

When all object types are RM-referable, the Relational Model “7-step” Mapping Algorithm can be applied to transform the ORM model into a relational model. The references will thus be used to create primary keys and foreign keys in the relational database schema. The steps are shown in Algorithm 4¹. The seven steps of this algorithm are as follows:

Algorithm 4 The Relational Model “7-step” Mapping Algorithm for ORM

- 1: Verify RM-reference completeness
 - 2: Group around non-subtype entities
 - 3: Group around subtypes (and add supertype reference)
 - 4: Map “ $n \geq 2$ uniqueness” constraints
 - 5: Make lexical (= choose references)
 - 6: Map remaining constraints
 - 7: Eliminate “reference” relations
-

¹In this thesis, a simplified version of the RM-algorithm was adopted, used in the Information Systems course developed by prof. dr. Robert Meersman. Another version of this mapping algorithm is presented in [HM08].

Verify RM-reference completeness. We already covered the RM-reference completeness of an ORM schema in the previous section. Figure A.1 contains an example of a RM-reference complete ORM schema which will be used to demonstrate the remaining six steps. The ORM was intentionally kept very simple for the sake of the example.

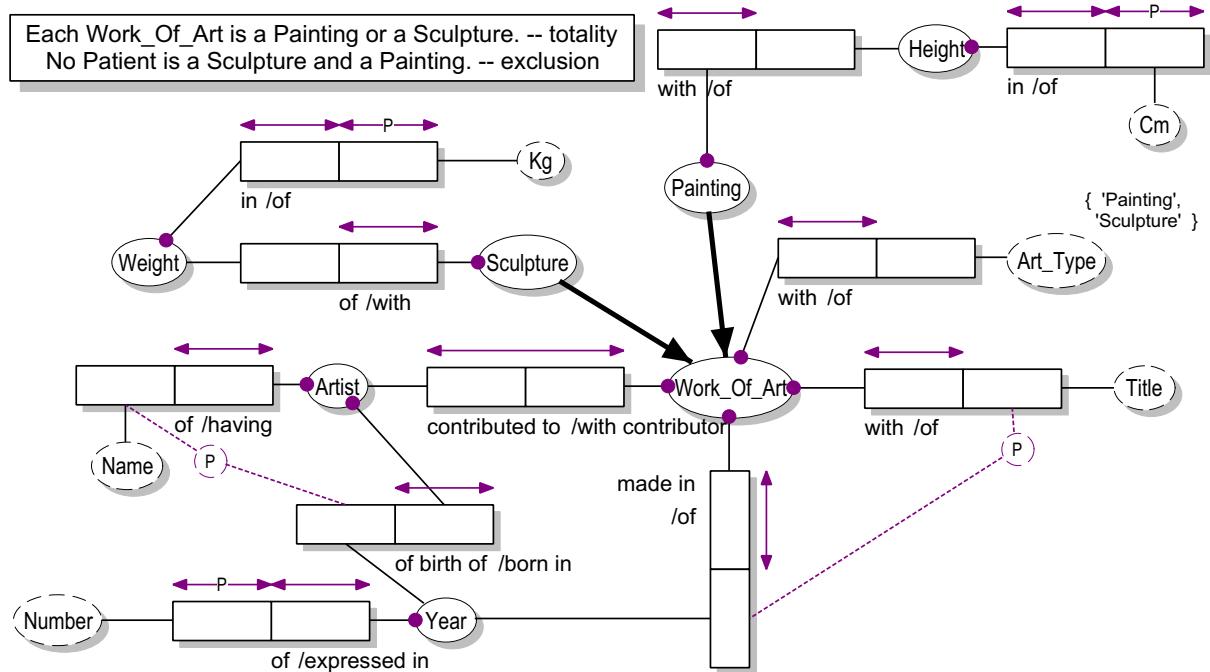
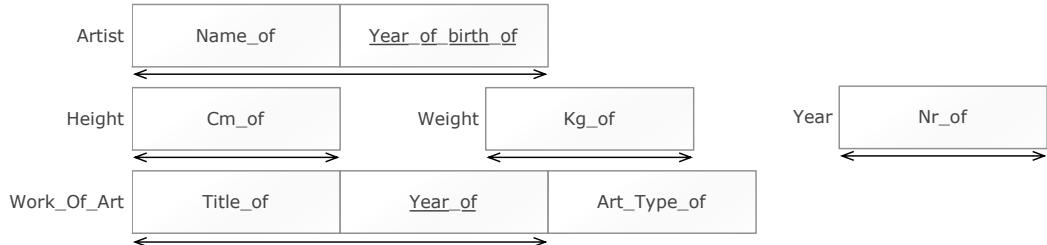


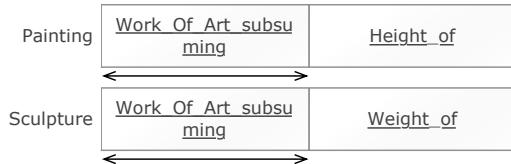
Figure A.1: Example of a RM-reference complete ORM schema

Group around non-subtype entities. Choose a non-subtype non-lexical object type, lets call this the focus-NOLOT. For each OT (LOT or NOLOT) that is connected to the focus-NOLOT by a fact type not marked as “grouped”, and such that the role played by the focus-NOLOT is identifying: concatenate the OT name with its co-role label in that fact type, and add this OT-co-role as field into a record type structure; give this grouped structure the label of the focus-NOLOT. Mark each such fact type in the ORM schema as “grouped”. Mark with a double arrow all OT-co-roles combinations that identify focus-NOLOT as unique. There must be at least one since the schema is RM-reference complete. Mark with underline all NOLOT-co-role (non-lexical) fields. Mark with parentheses those OT-co-role fields that connect with non-total (non-mandatory) roles on the focus.

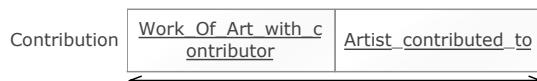
For the example, this step would result in:



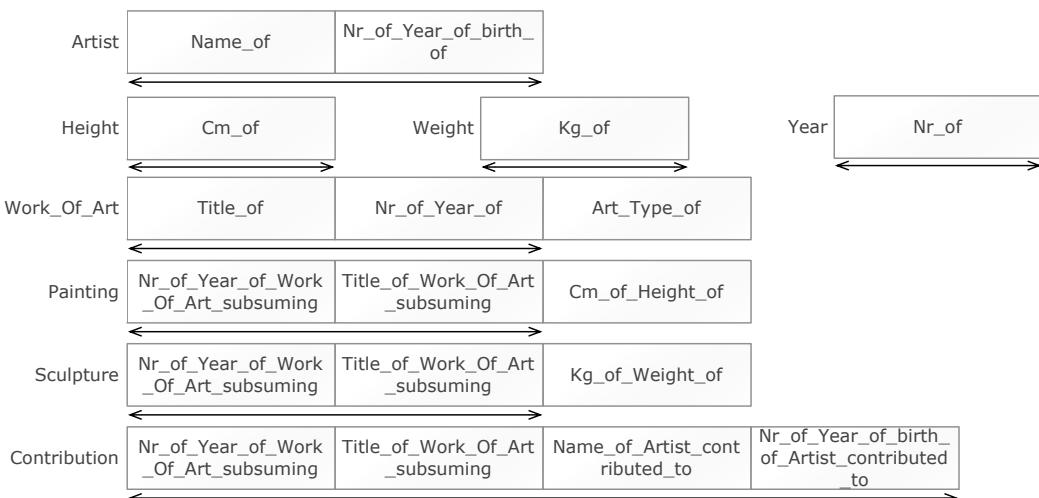
Group around subtypes. Groups as before, but add an identifying non-lexical field for each NOLOT that is a direct supertype. In this example, this would result in:



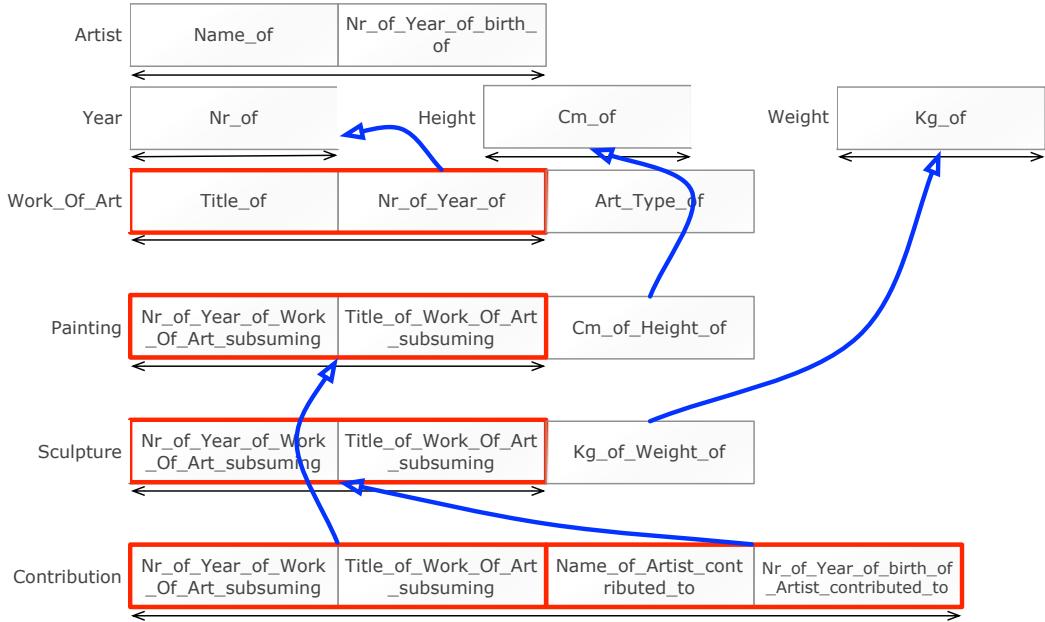
Map “ $n \geq 2$ uniqueness” constraints. The DOGMA ontology engineering framework is limited to the use of binary fact types. For every fact type with an internal uniqueness constraint spanning more than 1 role. Create a record type structure and give this structure a label. Add references to the two OTs in that fact type and mark them as identifying one tuple in that relation. This results in:



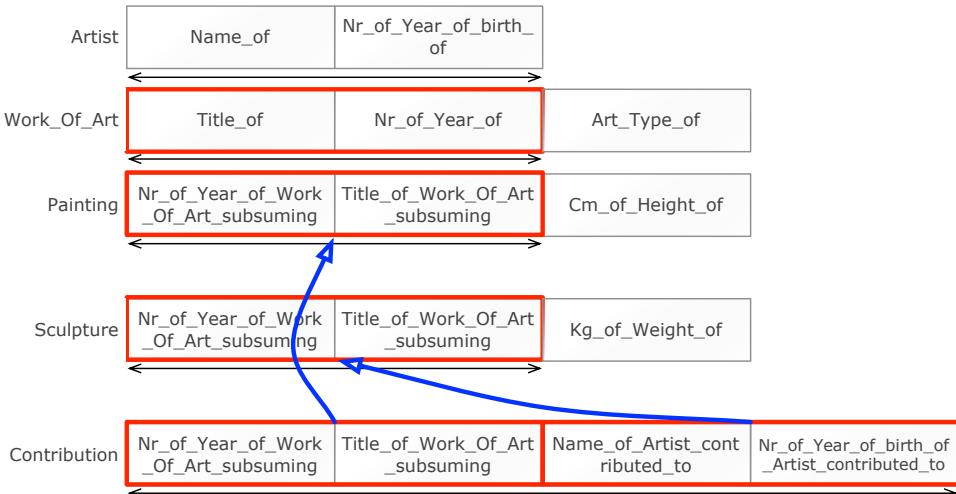
Make lexical. For each of the fields with a dashed underline, replace that field with all lexical attributes identifying that OT:



Map remaining constraints. Constructing the foreign key constraints as well as other ORM constraints (e.g. subset constraints). Note that - depending on the particular DBMS - some of these constraints cannot be mapped and have to reside at the application layer. In the image below, the foreign constraints are represented with arrows.



Eliminate “reference” relations. Reference relations are relations in which all fields constitute the primary key. Except when one wants to keep such a relation to keep an exhaustive list of instances of the corresponding OT or even store tuples that do not necessarily play a role with other OTs, those relations can be removed. In the example, for instance, one might not wish to keep such a list for years, heights and weights and therefore chooses to remove those relations.



Appendix B

D2RQ Generated Mapping

```
CREATE TABLE piece (
    id int(10) unsigned NOT NULL AUTO_INCREMENT,
    name varchar(250) NOT NULL,
    year int(4) NOT NULL,
    price_value bigint(20) DEFAULT NULL,
    currency varchar(3) DEFAULT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY name (name,year))
```

The annotation that D2RQ generated for this table is given below.

```
@prefix map: <http://d2r-mappings/mappingOnlyPiece.ttl#> .
@prefix db: <> .
@prefix vocab: <http://localhost:2020/vocab/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d2rq: <http://www.wiwiiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix jdbc: <http://d2rq.org/terms/jdbc/> .

map:database a d2rq:Database;
  d2rq:jdbcDriver "com.mysql.jdbc.Driver";
  d2rq:jdbcDSN "jdbc:mysql://localhost/art2";
  d2rq:username "myUserName";
  d2rq:password "mySecret :)";
  jdbc:autoReconnect "true";
  jdbc:zeroDateTimeBehavior "convertToNull".
map:piece a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "piece/@@piece.id@@";
  d2rq:class vocab:piece;
  d2rq:classDefinitionLabel "piece".
map:piece__label a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property rdfs:label;
  d2rq:pattern "piece #@@piece.id@@".
map:piece_id a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_id;
```

```
d2rq:propertyDefinitionLabel "piece id";
d2rq:column "piece.id";
d2rq:datatype xsd:unsignedInt.
map:piece_name a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_name;
  d2rq:propertyDefinitionLabel "piece name";
  d2rq:column "piece.name".
map:piece_year a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_year;
  d2rq:propertyDefinitionLabel "piece year";
  d2rq:column "piece.year";
  d2rq:datatype xsd:int.
map:piece_price_value a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_price_value;
  d2rq:propertyDefinitionLabel "piece price_value";
  d2rq:column "piece.price_value";
  d2rq:datatype xsd:long.
map:piece_currency a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:piece;
  d2rq:property vocab:piece_currency;
  d2rq:propertyDefinitionLabel "piece currency";
  d2rq:column "piece.currency".
```

Appendix C

R-RIDL Grammar

```
grammar QueryRIDL_FLAT;
options {language = Java; k=2;}
@header {package vub.starlab.gospl.ridl.parser;}
@lexer::header {package vub.starlab.gospl.ridl.parser;}
/*-----
 * PARSER RULES
 *-----*/
rule:
    (LIST setExpression)
    | (FOR setExpression
        LIST forElementarSetExpression (AS string)?
        (COMMA forElementarSetExpression (AS string)?)*);
/*-----
 * FOR LIST SPECIFIC PARSER RULES
 *-----*/
forElementarSetExpression:
    (ANY forElementarSetExpression)
    | (roleLabel (forSetReference)?);
forSetReference:
    forReference
    | (NOT forSetReference);
forReference: roleLabel forElementarSetExpression?;
/*-----
 * LIST PARSER RULES
 *-----*/
anyExpression: ANY elementarSetExpression;
booleanOperator: AND | OR;
elementarSetExpression:
    (termLabel (setReference)?)
    | setDefinition
    | occurrence;
expression: term (termOperator term)*;
factor: number | string | (LBRACE expression RBRACE);
factorOperator: DIVISION | TIMES ;
number: NUMERICCONSTANT;
occurrence:
    expression
    | ( LBRACE theExpression RBRACE )
    | ( LBRACE anyExpression RBRACE )
    | theExpression
```

```

| anyExpression;
operator: EQ | NEQ | LT | GT | LTE | GTE;
reference: roleLabel elementarSetExpression ;
roleLabel: LITERAL;
setDefinition: LBRACKET (occurrence (COMMA occurrence)*)? RBRACKET;
setExpression:
    elementarSetExpression
    (setOperation elementarSetExpression)*;
setOperation: UNION | INTERSECTION | SETMINUS;
setReference:
    reference
    | (NOT setReference)
    | (LBRACE setReference (booleanOperator setReference)* RBRACE)
    | (operator occurrence);
string: STRING;
term: factor (factorOperator factor)*;
termLabel: LITERAL;
termOperator: MINUS | PLUS;
theExpression: THE r=elementarSetExpression;
/*-----
 * LEXER RULES
-----*/
WHITESPACE: (' ' | '\t' | '\r' | '\n' | '\u000C')+;
UNION: 'UNION';
INTERSECTION: 'INTERSECTION';
SETMINUS: 'MINUS';
ANY: 'ANY';      THE: 'THE';
OR: 'OR';        AND: 'AND';      NOT: 'NOT';
AS: 'AS';        COMMA: ',';     DIVISION: '/';
EQ: '=';          FOR: 'FOR';    GT: '>';
GTE: '>=';        LBRACE: '(';   LBRACKET: '[';
LIST: 'LIST';    LITERAL: (HC|LC)(HC|LC|DIGIT|'_')*;
LT: '<';          LTE: '<=';    MINUS: '-';
NEQ: '<>';       NUMERICCONSTANT: ('-')? DIGIT+ ('.' DIGIT+)?;
PLUS: '+';        RBRACE: ')';   RBRACKET: ']';
STRING: '\'' ACTUALSTRING '\''; TIMES: '*';
fragment ACTUALSTRING: ~('\'|'\''')*;
fragment DIGIT: '0'..'9';
fragment HC: 'A'..'Z';
fragment LC: 'a'..'z';

```

Bibliography

- [ADF09] M. M. Al-Debei and G. Fitzgerald. OntoEng: A design method for ontology engineering in information systems. *Proceedings of the ACM OOPSLA'09, ODiSE*, 2009:1–25, 2009.
- [ANT⁺11] P. R. Alexander, C. Nyulas, T. Tudorache, P. L. Whetzel, N. F. Noy, and M. A. Musen. Semantic infrastructure to enable collaboration in ontology development. In W. W. Smari and G. Fox, editors, *CTS*, pages 423–430. IEEE, 2011.
- [AVCFLGP03] J. C. Arpírez Vega, Ó. Corcho, M. Fernández-López, and A. Gómez-Pérez. WebODE in a nutshell. *AI Magazine*, 24(3):37–48, 2003.
- [BBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the \mathcal{EL} envelope. *International Joint Conference on Artificial Intelligence*, 19:364, 2005.
- [BCM⁺03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.
- [BCM05] P. Buitelaar, P. Cimiano, and B. Magnini. Ontology learning from text: An overview. *Ontology learning from text: Methods, evaluation and applications*, 123:3–12, 2005.
- [BDFS04] C. Bussler, J. Davies, D. Fensel, and R. Studer, editors. *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, volume 3053 of *LNCS*. Springer, 2004.
- [Ber10] M. Bergman. A brief survey of ontology development methodologies. Via <http://www.mkbergman.com/906/a-brief-survey-of-ontology-development-methodologies/> (retrieved March 2012), 2010.
- [BG04] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, W3C, February 2004.
- [BHS08] F. Baader, I. Horrocks, and U. Sattler. Description logics. *Foundations of Artificial Intelligence*, 3:135–179, 2008.
- [BL98] T. Berners-Lee. Relational databases on the semantic web. <http://www.w3.org/DesignIssues/RDB-RDF.html> (last retrieved December 2012), September 1998.
- [BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.
- [BM11] Sven Bittner and André Müller. Social networking tools and research information systems: Do they compete? *Webology*, 8(1), 2011.
- [Bor96] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.
- [BOS04] P. Buitelaar, D. Olejnik, and M. Sintek. A protégé plug-in for ontology extraction from text based on linguistic analysis. In Bussler et al. [BDFS04], pages 31–44.
- [BTF05] C. Bussler, V. Tannen, and I. Fundulaki, editors. *Semantic Web and Databases, Second International Workshop, SWDB 2004, Toronto, Canada, August 29-30, 2004, Revised Selected Papers*, volume 3372, 2005.
- [BvHH⁺04] S. Bechhofer, F. van Harmelen, J. A. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. W3C Recommendation, W3C, February 2004.

- [CBG⁺12] R. Cyganiak, C. Bizer, J. Garbers, O. Maresch, and C. Becker. The D2RQ mapping language. <http://d2rq.org/d2rq-language>, March 2012.
- [CD12] I. Ciuciu and C. Debruyne. Assessing the user satisfaction with an ontology engineering tool based on social processes. In P. Herrero, Hervé P., R. Meersman, and T. S. Dillon, editors, *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, volume 7567 of *LNCS*, pages 242–251. Springer, 2012.
- [CDGL98] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In A. O. Mendelzon and J. Paredaens, editors, *PODS*, pages 149–158. ACM Press, 1998.
- [CDGL01] D. Calvanese, G. De Giacomo, and M. Lenzerini. Identification constraints and functional dependencies in description logics. In B. Nebel, editor, *IJCAI*, pages 155–160. Morgan Kaufmann, 2001.
- [CDGL⁺07] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated reasoning*, 39(3):385–429, 2007.
- [CDGL⁺11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D.F. Savo. The Mastro system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.
- [CDL⁺05] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 602–607. AAAI Press / The MIT Press, 2005.
- [CDL⁺08] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Path-based identification constraints in description logics. In G. Brewka and J. Lang, editors, *KR*, pages 231–241. AAAI Press, 2008.
- [CFLGP03] Ó. Corcho, M. Fernández-López, and A. Gómez-Pérez. Methodologies, tools and languages for building ontologies. where is their meeting point? *Data and Knowledge Engineering*, 46(1):41 – 64, 2003.
- [CFLGPLC03] Ó. Corcho, M. Fernández-López, A. Gomez-Perez, and A. López-Cima. Building legal ontologies with METHONTOLOGY and WebODE. In V. R. Benjamins, P. Casanovas, J. Breuker, and A. Gangemi, editors, *Law and the Semantic Web*, volume 3369, pages 142–157, 2003.
- [CFLGPV02] Ó. Corcho, M. Fernández-López, A. Gómez-Pérez, and Ó. Vicente. WebODE: An integrated workbench for ontology representation, reasoning, and exchange. In A. Gómez-Pérez and R. Benjamins, editors, *EKAW*, volume 2473 of *LNCS*, pages 138–153. Springer, 2002.
- [CGR⁺05] M. Ciaramita, A. Gangemi, E. Ratsch, J. Saric, and I. Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 659–664. Professional Book Center, 2005.
- [CRF03] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *Data Cleaning Workshop in Conjunction with KDD*, 2003.
- [DBSM04] J. De Bo, P. Spyns, and R. Meersman. Assisting ontology integration with existing thesauri. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, volume 3290 of *LNCS*, pages 801–818. Springer Berlin / Heidelberg, 2004.
- [DC13] C. Debruyne and I. Ciuciu. User satisfaction of a hybrid ontology-engineering tool. In Hervé P. and Y. Tang Demey, editors, *On the Move to Meaningful Internet Systems: OTM 2013 Workshops*, volume 8186 of *LNCS*, pages 414–423. Springer, 2013.
- [DCM10] P. De Leenheer, S. Christiaens, and R. Meersman. Business semantics management: A case study for competency-centric HRM. *Computers in Industry*, 61(8):760–775, 2010.

- [dCSFGP08] M. del Carmen Suárez-Figueroa and A. Gómez-Pérez. Towards a glossary of activities in the ontology engineering field. In *LREC*. European Language Resources Association, 2008.
- [DD08] P. De Leenheer and C. Debruyne. DOGMA-MESS: A tool for fact-oriented collaborative ontology evolution. In *On the Move to Meaningful Internet Systems 2008: ORM (ORM 2008)*, LNCS, Monterrey, Mexico, 2008. Springer.
- [DdM05] P. De Leenheer and A. de Moor. Context-driven disambiguation in ontology elicitation. In P. Shvaiko and J. Euzenat, editors, *Context and Ontologies: Theory, Practice and Applications*, volume WS-05-01 of *AAAI Technical Report*, pages 17–24, Pittsburg USA, 7 2005. AAAI Press.
- [DDM11] C. Debruyne, P. De Leenheer, and R. Meersman. Empowering enterprise data governance with BSG. In M. A. Musen and Ó. Corcho, editors, *K-CAP*, pages 197–198. ACM, 2011.
- [DdMM07] P. De Leenheer, A. de Moor, and R. Meersman. Context dependency management in ontology engineering: a formal approach. *Journal of Data Semantics*, 8:26–56, 2007. LNCS 4380, Springer.
- [DDS⁺11] C. Debruyne, P. De Leenheer, P. Spyns, G. van Grootel, and S. Christiaens. Publishing open data and services for the flemish research information space. In O. De Troyer, C. Bauzer Medeiros, R. Billen, P. Hallot, A. Simitsis, and H. Van Mingroot, editors, *ER Workshops*, volume 6999 of *LNCS*, pages 389–394. Springer, 2011.
- [Deb10] C. Debruyne. On the social dynamics of ontological commitments. In R. Meersman, T. S. Dillon, and P. Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2010 Workshops*, volume 6428 of *LNCS*, pages 682–686. Springer, 2010.
- [DEMB⁺08] K. Dellschaft, H. Engelbrecht, J. Monte Barreto, S. Rutenbeck, and S. Staab. Cicero: Tracking design rationale in collaborative ontology engineering. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *ESWC*, volume 5021 of *LNCS*, pages 782–786. Springer, 2008.
- [DJM02] J. Demey, M. Jarrar, and R. Meersman. A conceptual markup language that supports interoperability between business rule modeling systems. In R. Meersman and Z. Tari, editors, *CoopIS/DOA/ODBASE*, volume 2519 of *LNCS*, pages 19–35. Springer, 2002.
- [DL09] P. De Leenheer. *On Community-based Ontology Evolution: Foundations for Business Semantics Management*. Phd thesis, Vrije Universiteit Brussel, 2009.
- [DLDP09] P. De Leenheer, C. Debruyne, and J. Peeters. Towards social performance indicators for community-based ontology evolution. In *Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK2008), collocated with the 8th International Semantic Web Conference (ISWC 2009)*. CEUR-WS, October 2009.
- [DLM08] P. De Leenheer and T. Mens. Ontology evolution: State of the art and future directions. In Hepp et al. [HDDoS08], pages 131–176.
- [DM07] P. De Leenheer and R. Meersman. Towards community-based evolution of knowledge-intensive systems. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 4803 of *LNCS*, pages 989–1006. Springer, 2007.
- [DM11] C. Debruyne and R. Meersman. Semantic interoperation of information systems by evolving ontologies through formalized social processes. In J. Eder, M. Bieliková, and A. M. Tjoa, editors, *ADBIS*, volume 6909 of *LNCS*, pages 444–459. Springer, 2011.
- [DM12] C. Debruyne and R. Meersman. GOSPL: A method and tool for fact-oriented hybrid ontology engineering. In T. Morzy, T. Härdter, and R. Wrembel, editors, *ADBIS*, volume 7503 of *LNCS*, pages 153–166. Springer, 2012.

- [dMDM06] A. de Moor, P. De Leenheer, and R. Meersman. DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering. In *Proceedings of the 14th International Conference on Conceptual Structures (ICCS 2006)*, volume 4068 of *LNCS*, pages 189–203. Springer, 2006.
- [DN13] C. Debruyne and N. Nijs. Using a reputation framework to identify community leaders in ontology engineering. In R. Meersman, H. Panetto, T. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. De Leenheer, and D. Dou, editors, *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, volume 8185 of *Lecture Notes in Computer Science*, pages 677–684. Springer, 2013.
- [DNMN05] A. De Nicola, M. Missikoff, and R. Navigli. A proposal for a unified process for ontology building: UPON. In K. V. Andersen, J. K. Debenham, and R. Wagner, editors, *DEXA*, volume 3588 of *LNCS*, pages 655–664. Springer, 2005.
- [DNMN09] A. De Nicola, M. Missikoff, and R. Navigli. A software engineering approach to ontology building. *Information Systems*, 34:258–275, April 2009.
- [DRM10] C. Debruyne, Q. Reul, and R. Meersman. GOSPL: Grounding ontologies with social processes and natural language. In S. Latifi, editor, *ITNG*, pages 1255–1256. IEEE Computer Society, 2010.
- [DTM13] Christophe Debruyne, Trung-Kien Tran, and Robert Meersman. Grounding ontologies with social processes and natural language. *Journal of Data Semantics*, 2(2-3):89–118, 2013.
- [DTMP83] O. De Troyer, R. Meersman, and F. Ponsaert. RIDL user guide. Research report (available from the authors), International Centre for Information Analysis Services, Control Data, 1983.
- [DV13] C. Debruyne and C. Vasquez. Exploiting natural language definitions and (legacy) data for facilitating agreement processes. In D. Winkler, S. Biffl, and J. Bergsmann, editors, *SWQD*, volume 133 of *LNBIP*, pages 244–258. Springer, 2013.
- [ERRJ95] G. Erich, H. Richard, J. Ralph, and V. John. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.
- [Fel98] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [Fen01] D. Fensel. Ontologies: Dynamic networks of formally represented meaning. <http://sw-portal.deri.at/papers/publications/network.pdf>, 2001.
- [FHL⁺96] E. D. Falkenberg, W. Hesse, P. Lindgreen, B. E. Nilsson, J. L. H. Oei, C. Rolland, R. K. Stamper, F. J. M. Van Assche, A. A. Verrijn-Stuart, and K. Voss. Frisco: A framework of information system concepts. Technical report, The IFIP WG 8.1 Task Group FRISCO, 1996.
- [FLGPJ97] M. Fernández-López, A. Gomez-Perez, and N. Juristo. METHONTOLOGY: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, 1997.
- [FM12] E. Franconi and A. Mosca. The formalisation of ORM2 and its encoding in OWL2. Available from <https://www.inf.unibz.it/krdb/pub/TR/KRDB12-2.pdf>, 2012.
- [FMS12] E. Franconi, A. Mosca, and D. Solomakhin. ORM2 encoding into description logics. *2012 International Description Logics workshop (DL-2012), Rome, Italy*, 2012.
- [GF95] M. Grüninger and M. Fox. Methodology for the design and evaluation of ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*, 1995.
- [GG95] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases*. IOS Press, 1995.

- [GGA⁺02] P. Gamallo, M. Gonzalez, A. Agustini, G. Lopes, and V.S. De Lima. Mapping syntactic dependencies onto semantic relations. *ECAI Workshop on Machine Learning and Natural Language Processing for Ontology Engineering*, 2002.
- [GL90] R. Guha and D. Lenat. CYC: A midterm report. *AI Magazine*, 11(3):32–59, 1990.
- [GPdCSF09] A. Gómez-Pérez and M. del Carmen Suárez-Figueroa. Scenarios for building ontology networks within the NeOn methodology. In Y. Gil and N. F. Noy, editors, *K-CAP*, pages 183–184. ACM, 2009.
- [GPFLC03] A. Gómez-Pérez, M. Fernández-López, and Ó. Corcho. *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003. ISBN: 1852335513.
- [Gru95] T. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.
- [GS86] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, July 1986.
- [Gua98] N. Guarino. Formal ontology and information systems. In *International Conference On Formal Ontology In Information Systems FOIS’98*, pages 3–15, Trento, Italy, June 1998. Amsterdam, IOS Press.
- [Hal89] T. A. Halpin. *A Logical Analysis of Information Systems: static aspects of the data-oriented perspective*. PhD thesis, University of Queensland, 1989.
- [Hal05] T. A. Halpin. ORM 2. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2005 Workshops*, volume 3762 of *LNCS*, pages 676–687. Springer, 2005.
- [HDdS08] M. Hepp, P. De Leenheer, A. de Moor, and Y. Sure, editors. *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications*, volume 7 of *Semantic Web And Beyond Computing for Human Experience*. Springer, 2008.
- [Hep07] M. Hepp. Possible ontologies: How reality constrains the development of relevant ontologies. *IEEE Internet Computing*, 11(1):90–96, 2007.
- [Hep08] M. Hepp. Ontologies: State of the art, business potential, and grand challenges. In Hepp et al. [HDdS08], pages 3–22.
- [Hey99] F. Heylighen. Collective intelligence and its implementation on the web: algorithms to develop a collective mental map. *Computational & Mathematical Organization Theory*, 5(3):253–280, 1999.
- [Hey11] F. Heylighen. Self-organization of complex, intelligent systems: an action ontology for transdisciplinary integration. *Integral Review*, 2011.
- [Hey13] Francis Heylighen. Self-organization in communicating groups: The emergence of coordination, shared references and collective intelligence. In Àngels Massip-Bonet and Albert Bastardas-Boada, editors, *Complexity Perspectives on Language, Communication and Society*, Understanding Complex Systems, pages 117–149. Springer Berlin, 2013.
- [HJ02] C. W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45:42–47, February 2002.
- [HJ10] R. Hodrob and M. Jarrar. Mapping ORM into OWL 2. In A. Alnsour and S. Aljawarneh, editors, *ISWSA*, page 9. ACM, 2010.
- [HM08] T. A. Halpin and T. Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann, San Francisco, CA, USA, 2008.
- [Hov93] E.H. Hovy. Automated discourse generation using discourse structure relations. *Artificial intelligence*, 63(1):341–385, 1993.

- [HPSB⁺04] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. Swrl: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21:79, 2004.
- [ISO98] ISO. ISO 9241-11: Ergonomic requirements for office work with visual display terminals (vdts) – part 11: Guidance on usability. Technical report, ISO, 1998.
- [JA10] Keith Jeffery and Anne Asserson. CERIF-CRIS for the European e-Infrastructure. *Data Science Journal*, 9:1–6, 2010.
- [Jar05] M. Jarrar. *Towards Methodological Principles for Ontology Engineering*. PhD thesis, Vrije Universiteit Brussel, Brussels, Belgium, May 2005.
- [Jar06] M. Jarrar. Position paper: towards the notion of gloss, and the adoption of linguistic resources in formal ontology engineering. In L. Carr, D. De Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *WWW*, pages 497–503. ACM, 2006.
- [Jar07a] M. Jarrar. Mapping ORM into the SHOIN/OWL description logic. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2007 Workshops*, volume 4805 of *LNCS*, pages 729–741. Springer, 2007.
- [Jar07b] M. Jarrar. Towards automated reasoning on orm schemes. In C. Parent, K. D. Schewe, V. C. Storey, and B. Thalheim, editors, *ER*, volume 4801 of *LNCS*, pages 181–197. Springer, 2007.
- [JM09] M. Jarrar and R. Meersman. Ontology engineering – the DOGMA approach. In T. S. Dillon, E. Chang, R. Meersman, and K. Sycara, editors, *Advances in Web Semantics I*, volume 4891 of *LNCS*, pages 7–34. Springer Berlin Heidelberg, 2009.
- [JMSV92] M. Jarke, J. Mylopoulos, J. W. Schmidt, and Y. Vassiliou. DAIDA: An environment for evolving information systems. *ACM Trans. Inf. Syst.*, 10(1):1–50, 1992.
- [KA06] S. Karapiperis and D. Apostolou. Consensus building in collaborative ontology engineering processes. *Knowledge Creation Diffusion Utilization*, 1(3):199–216, 2006.
- [Kee07] M. C. Keet. Mapping the object-role modeling language ORM2 into description logic language DLR_{ifd}. *CoRR*, abs/cs/0702089, 2007.
- [Kot08] K. Kotis. On supporting HCOME-3O ontology argumentation using semantic wiki technology. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, volume 5333 of *LNCS*, pages 193–199. Springer, 2008.
- [Kow78] R. A. Kowalski. Logic for data description. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 77–103. Plenum Press, New York, N. Y., 1978.
- [KR70] W. Kunz and H. Rittel. Issues as elements of information systems. Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.
- [KV03] K. Kotis and G. A. Vouros. Human centered ontology management with HCONE. In *Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems*, volume 71, 2003.
- [KV06] K. Kotis and G. A. Vouros. Human-centered ontology engineering: The HCOME methodology. *Knowledge Information Systems*, 10(1):109–131, 2006.
- [KVA04] K. Kotis, G. A. Vouros, and J. P. Alonso. HCOME: A tool-supported methodology for engineering living ontologies. In Bussler et al. [BTF05], pages 155–166.
- [LCD08] A. Liapis, S. Christiaens, and P. De Leenheer. Collaboration across the enterprise - an ontology-based approach for enterprise interoperability. In J. Cordeiro and J. Filipe, editors, *ICEIS (4)*, pages 255–262, 2008.

- [Len95] D. Lenat. Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38:33–38, November 1995.
- [Lew93] J. R. Lewis. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use (technical report 54.786). Technical report, IBM, 1993.
- [Lew02] J. R. Lewis. Psychometric evaluation of the pssuq using data from five years of usability studies. *International Journal of Human-Computer Interaction*, 14(3-4):463–488, 2002.
- [Lew12] J. R. Lewis. Usability testing. In *Handbook of Human Factors and Ergonomics*, pages 1267–1312. John Wiley, 4 edition, 2012.
- [MD10] R. Meersman and C. Debruyne. Hybrid ontologies and social semantics. In *Proceedings of 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST 2010)*, pages 92–97. IEEE Press, 2010.
- [Mee82] R. Meersman. The RIDL conceptual language. Research report (available from the authors), International Centre for Information Analysis Services, Control Data, 1982.
- [Mee99a] R. Meersman. Semantic ontology tools in IS design. In Z. W. Ras and A. Skowron, editors, *ISMIS*, volume 1609 of *LNCS*, pages 30–45. Springer, 1999.
- [Mee99b] R. Meersman. The use of lexicons and other computer-linguistic tools in semantics, design and cooperation of database systems. In Y. Zhang, M. Rusinkiewicz, and Y. Kambayashi, editors, *The Proceedings of the Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS99)*, pages 1–14. Springer, 1999.
- [Mee01a] R. Meersman. Ontologies and databases: More than a fleeting resemblance. In A. d’Atri and M. Missikoff, editors, *OES/SEO 2001 Rome Workshop*. Luiss Publications, 2001.
- [Mee01b] R. Meersman. Reusing certain database design principles, methods and design techniques for ontology theory, construction and methodology. Technical report, VUB Starlab, 2001.
- [MGH⁺09] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language: Profiles. W3C Recommendation, W3C, October 2009.
- [MHS07] B. Motik, I. Horrocks, and U. Sattler. Adding integrity constraints to OWL. In C. Goldreich, A. Kalyanpur, and B. Parsia, editors, *OWLED*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
- [MHS09] B. Motik, I. Horrocks, and U. Sattler. Bridging the gap between OWL and relational databases. *Journal of Web Semantics*, 7(2):74–89, 2009.
- [MT88] W. C. Mann and S. A. Thompson. Rhetorical structure theory: A theory of text organization. *Text*, 8:243–281, 1988.
- [NGM00] N. F. Noy, W. Gross, and M. A. Musen. Knowledge-acquisition interfaces for domain experts: An empirical evaluation of Protégé-2000. *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, July, 5-7*, 2000.
- [NM01] N. F. Noy and D. L. McGuinness. *Ontology development 101: A guide to creating your first ontology*, volume 8. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.
- [NT95] I. Nonaka and H. Takeuchi. *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, USA, 1995.
- [NTDCM08] N. F. Noy, T. Tudorache, S. De Coronado, and M. A. Musen. Developing biomedical ontologies collaboratively. *AMIA Annual Symposium Proceedings*, 2008:520, 2008.
- [NW01a] A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 335–367. Elsevier and MIT Press, 2001.
- [NW01b] A. Nonnengart and C. Weidenbach. Computing small clause normal forms. *Handbook of automated reasoning*, 1:335–367, 2001.

- [OMG09] OMG. Semantics of business vocabulary and business rules, v1.0. <http://omg.org/spec/SBVR/1.0/>, July 2009.
- [OR23] C.K. Ogden and I.A. Richards. *The meaning of meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Harcourt, Brace, 1923.
- [Par07] T. Parr. *The Definitive ANTLR Reference: Building Domain-Specific Languages*. Pragmatic Programmers. Pragmatic Bookshelf, first edition, May 2007.
- [Pei35] C.S. Peirce. *Collected papers of Charles Sanders Peirce*. Belknap Press, 1935.
- [PLC⁺08] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *Journal of Data Semantics*, 10:133–173, 2008.
- [PS08] E. Prud’hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, W3C, January 2008.
- [PSST04] H. S. Pinto, S. Staab, Y. Sure, and C. Tempich. OntoEdit empowering SWAP: a case study in supporting distributed, loosely-controlled and evolving engineering of ontologies (DILIGENT). In Bussler et al. [BDFS04], pages 16–30.
- [PST04] H. S. Pinto, S. Staab, and C. Tempich. DILIGENT: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In R. L. de Mántaras and L. Saitta, editors, *ECAI*, pages 393–397. IOS Press, 2004.
- [PTS09] H. S. Pinto, C. Tempich, and S. Staab. Ontology engineering and evolution in a distributed world using DILIGENT. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 153–176. Springer, 2009.
- [RB06] R. Rifaieh and N. Benharkat. From Ontology Phobia to Contextual Ontology use in Enterprise Information Systems, March 2006.
- [Rei82] R. Reiter. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling (Intervale)*, pages 191–233, 1982.
- [Rei88] R. Reiter. On integrity constraints. In M. Y. Vardi, editor, *TARK*, pages 97–111. Morgan Kaufmann, 1988.
- [SAS03] Y. Sure, J. Angele, and S. Staab. OntoEdit: Multifaceted inferencing for ontology engineering. *Journal of Data Semantics*, 1:128–152, 2003.
- [Sat03] U. Sattler. Description logics for ontologies. *Conceptual Structures for Knowledge Creation and Communication*, pages 96–116, 2003.
- [Sch05] R. Schwitter. A controlled natural language layer for the semantic web. In S. Zhang and R. Jarvis, editors, *Australian Conference on Artificial Intelligence*, volume 3809 of *LNCS*, pages 425–434. Springer, 2005.
- [SEA⁺02] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In I. Horrocks and J. A. Hendler, editors, *International Semantic Web Conference*, volume 2342 of *LNCS*, pages 221–235. Springer, 2002.
- [SHH⁺09] S. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. Thibodeau, S. Auer, J. Sequeda, and A. Ezzat. A survey of current approaches for mapping of relational databases to RDF. Technical report, W3C, 01 2009.
- [SHKG09] M. Smith, I. Horrocks, M. Krötzsch, and B. Glimm. OWL 2 Web Ontology Language: Conformance. W3C Recommendation, W3C, 27 October 2009.
- [SMJ02] P. Spyrs, R. Meersman, and M. Jarrar. Data modelling versus ontology engineering. *SIGMOD Record Special Issue*, 31 (4):12–17, 2002.
- [SS10] K. Siorpae and E. Simperl. Human intelligence in the process of semantic content creation. *World Wide Web*, 13(1-2):33–59, 2010.

- [SSS03] S. Staab, R. Studer, and Y. Sure. Knowledge processes and meta processes in ontology-based knowledge management. In C. W. Holsapple, editor, *Handbook on Knowledge Management. International Handbooks on Information Systems*, pages 47–68. Springer, 2003.
- [SSS09] Y. Sure, S. Staab, and R. Studer. Ontology engineering methodology. In P. Bernus, J. Blazewicz, J. Schmidt, Günter, M. J. Shaw, S. Staab, and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 135–152. Springer Berlin Heidelberg, 2009.
- [SSSS01] S. Staab, H. P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34, January 2001.
- [ST06] E. Simperl and C. Tempich. Ontology engineering: A reality check. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 4275 of *LNCS*, pages 836–854. Springer, 2006.
- [TD12] T. K. Tran and C. Debruyne. Towards using OWL integrity constraints in ontology engineering. In P. Herrero, H. Panetto, R. Meersman, and T. S. Dillon, editors, *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, volume 7567 of *LNCS*, pages 282–285. Springer, 2012.
- [TNTM08] T. Tudorache, N. F. Noy, S. Tu, and M. Musen. Supporting collaborative ontology development in Protégé. In A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayanan, editors, *International Semantic Web Conference*, volume 5318 of *LNCS*, pages 17–32. Springer, 2008.
- [TPS06] C. Tempich, H. S. Pinto, and S. Staab. Ontology engineering revisited: An iterative case study. In Y. Sure and J. Domingue, editors, *ESWC*, volume 4011 of *LNCS*, pages 110–124. Springer, 2006.
- [TSBM10] J. Tao, E. Sirin, J. Bao, and D. L. McGuinness. Integrity constraints in OWL. In M. Fox and D. Poole, editors, *AAAI*. AAAI Press, 2010.
- [TSPSS05] C. Tempich, H. Sofia Pinto, Y. Sure, and S. Staab. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (DILIGENT). In A. Gómez-Pérez and J. Euzenat, editors, *ESWC*, volume 3532 of *LNCS*, pages 241–256. Springer, 2005.
- [TTM07] D. Trog, Y. Tang, and R. Meersman. Towards ontological commitments with Ω -RIDL markup language. In A. Paschke and Y. Biletskiy, editors, *RuleML*, volume 4824 of *LNCS*, pages 92–106. Springer, 2007.
- [TVN08] T. Tudorache, J. Vendetti, and N. F. Noy. Web-Protégé: A lightweight OWL ontology editor for the web. In C. Dolbear, A. Ruttenberg, and U. Sattler, editors, *OWLED*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
- [UG96] M. Uschold and M. Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.
- [UK95] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, 1995.
- [Usc96] M. Uschold. Building ontologies: towards a unified methodology. In *16th Annual Conference of the British Computer Society Specialist Group on Expert Systems*, pages 16–18, 1996.
- [vBvDHK⁺03] J.F.A.K. van Benthem, van Ditmarsch H.P., J. Ketting, J.S. Lodder, and W.P.M. Meyer-Viol. *Logica voor informatica*. Addison-Wesley, 3rd edition, 2003.
- [VDBM04] P. Verheyden, J. De Bo, and R. Meersman. Semantically unlocking database content through ontology-based mediation. In Bussler et al. [BTF05], pages 109–126.

- [VKCL07] G. A. Vouros, K. Kotis, C. Chalkiopoulos, and N. Lelli. The HCOME-3O framework for supporting the collaborative engineering of evolving ontologies. In L. Chen, P. Cudré-Mauroux, P. Haase, A. Hotho, and E. Ong, editors, *ESOE*, volume 292 of *CEUR Workshop Proceedings*, pages 95–107. CEUR-WS.org, 2007.
- [VPST05] D. Vrandecic, H. S. Pinto, Y. Sure, and C. Tempich. The DILIGENT knowledge processes. *Journal of Knowledge Management*, 9(5):85–96, 2005.
- [Win90] J. Wintraecken. *The NIAM Information Analysis Method: Theory and Practice*. Kluwer Academic Publishers, 1990.