

Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
INFORMATICA
Ciclo XXIV

Settore Concorsuale di afferenza: 01/B1
Settore Scientifico disciplinare: INF01

**Semantic Publishing: issues, solutions
and new trends in scholarly publishing
within the Semantic Web era**

Presentata da: Silvio Peroni

Coordinatore Dottorato:
Maurizio Gabbielli

Relatore:
Paolo Ciancarini

Esame finale anno 2012

Abstract

This thesis aims at introducing theories, formalisms and applications for opening up Semantic Publishing to an effective interaction between scholarly documents and their related semantic and formal descriptions. Namely, I investigate and propose solutions for three of the main issues that semantic publishing promises to address: the need of tools for linking document text to a *formal representation of its meaning*, the lack of complete *metadata schemas* for describing documents according to the publishing vocabulary, and the absence of effective *user interfaces* for easily acting on semantic publishing models and theories.

The first part of my work is about *markup theory and technology*. A better comprehension of a document derives from its structural organisation and from the formal semantics defined within it. In digital documents, the way we use to say something about a text is that of markup. Markup has been used for years for decorating documents at all levels of granularity, from the digital document as a whole to its sub-components. However, the most commonly used document formats in publishing (i.e., XML and PDF) were not developed to enable semantic enhancement, although it may be possible in principle to use them for this purpose. Trying to fill the gap between document markup and semantic markup, I have developed an OWL-based markup metalanguage called *EARMARK*. The basic idea is that EARMARK documents are collections of addressable text fragments, and such fragments are associated to OWL assertions that describe structural features as well as semantic properties of (parts of) that content.

Of course, (digital) documents and their content represent the core aspect of Semantic Publishing, since it promotes their discovery and connection to document-related resources and contexts, such as other articles and raw scientific data. Unfortunately, existing Semantic Web vocabularies are too abstract and incomplete to cover all the needs claimed by the actors involved in the publishing process (publishers, editors, authors, etc.). Thus, there is an acute need for new standards (ontologies) that comprehensively cover all the different aspects of the publishing domain. Trying to address these issues, in the central part of my work I propose a suite of orthogonal and complementary OWL 2 DL ontology modules, called *Se-*

mantic Publishing And Referencing (SPAR) ontologies, for describing all the aspects of bibliographic publications as comprehensive machine-readable RDF metadata.

Finally, in the last part of my thesis I deal with the issue of enabling users to use and interact with semantic technologies and semantic data. This aspect is particularly crucial for Semantic Publishing, since its end-users are publishers, researchers, readers, librarians and the like rather than experts in semantic technologies. The use of semantic models and data should be supported by proper interfaces that simplify the work of Semantic Publishing people. In this thesis I illustrate my personal contribution in this direction. I introduce four different tools that I developed to support users when understanding an ontology (*LODE*, *KC-Viz*), when formalising/presenting it (*Graffoo*), and when defining semantic data according to it (*Gaffe*).

Acknowledgements

I have never been able to write acknowledgements. I always feel to forget somebody – by the way, I am quite sure it will happen again. Actually, it is already happening. Let me begin.

First of all, I would like to thanks my family, especially Tiziana for having always and unconditionally endured, supported and encouraged me in everything.

A big thanks to my tutor, prof. Fabio Vitali, who has involved me in his extraordinary research group and who has continuously encouraged me, my ideas and my work – it is really a pleasure working with you. Another thanks to the rest of my *commissione*, my advisor prof. Paolo Ciancarini and prof. Claudio Sacerdoti Coen, for having always been ready to discuss my Ph.D. topics.

I would also like to thank other two people who have been fundamental to my research: prof. Enrico Motta and prof. David Shotton. I am really pleased to have been part of their respective research groups.

Another big thanks goes to all my external referees – namely, dr. Jeni Tennison, prof. Yves Marcoux, prof. Pompeu Casanovas and prof. Daniel Apollon – for their precious comments and advices.

Last but not least, I would like to thank all the other members of my research group, dr. Angelo Di Iorio, Gioele Barabucci and Francesco Poggi, for having supported my work.

Contents

Abstract	iii
Acknowledgements	v
List of Tables	xiii
List of Figures	xv
1 Introduction	1
2 The digital publishing revolution	5
2.1 Towards semantics-aware markup languages	7
2.1.1 Overlapping markup	8
2.1.2 Markup semantics and semantic markup	11
2.2 Metadata schema, vocabularies and ontologies for publishing	14
2.2.1 Dublin Core	14
2.2.2 PRISM	15
2.2.3 BIBO	15
2.2.4 MARC 21	16
2.2.5 FRBR	17
2.2.6 SWAN Citations Ontology	19
2.2.7 SKOS	19
2.3 How to help users: tools and applications for semantic data	20
2.3.1 Ontology documentation	20
2.3.2 Ontology sense-making	22
2.3.3 Visual modelling of ontologies	24
2.3.4 Authoring tools for ontologies	25
2.4 Projects, conferences and initiatives about Semantic Publishing	28
2.4.1 JISC’s Open Citation and Open Bibliography projects	28
2.4.2 JISC’s Lucero project	29

2.4.3	SePublica and Linked Science	30
2.4.4	Beyond Impact, the PDF and Research Communication	30
2.4.5	New Models of Semantic Publishing in Science	31
3	Enhancing markup documents	33
3.1	EARMARK, a Semantic Web approach to metamarkup	34
3.1.1	Ghost classes	36
3.1.2	Shell classes	39
3.1.3	An example and an API	41
3.2	The issue of overlapping markup	43
3.2.1	Range and markup item overlap	44
3.2.2	EARMARK as a standoff notation	46
3.2.3	Looking for authorial changes in Office Documents	48
EARMARK for processing office documents	50	
An evaluation	56	
3.2.4	Overlapping with Microformats and RDFa	58
3.2.5	Wikis: no overlapping where some should be	61
3.3	Structural validation of semantically-defined markup	67
3.3.1	Defining content-models on EARMARK documents	67
3.3.2	Structural patterns	70
Assessing structural patterns on EARMARK documents	71	
Experimental results	74	
3.3.3	Validation of document markup	75
3.4	Dealing with Markup Semantics	76
3.4.1	Searches on heterogeneous digital libraries	80
3.4.2	Validation of “Markup sensibility”	82
4	The Semantic Publishing And Referencing Ontologies	89
4.1	Representing bibliographic information using FaBiO	91
4.1.1	Bibliographic reference metadata encoding using DC Terms . .	92
4.1.2	Bibliographic reference metadata encoding using BIBO	94
4.1.3	Bibliographic reference metadata encoding using FRBR	96
4.1.4	Bibliographic reference metadata encoding using FaBiO	97
Using external models	99	
Extending FRBR within FaBiO	100	
Categorising bibliographic resources with SKOS	101	
4.2	Characterising citations with CiTO	103
4.3	Documents and their bibliographic references	107
4.3.1	Describing the bibliographic reference lists of articles with BiRO109	

An URI for the reference	110
Semantic enhancement of literal elements in references	111
EARMARK ranges for describing references	114
4.3.2 C4O: how much, where and what someone is citing	116
4.4 Characterising document parts with DoCO	120
4.4.1 Building blocks for structuring documents	121
4.4.2 Mixing rhetorical characterisation and structural components .	124
4.5 In the past you were it, now you are not it	127
4.5.1 Using class subsumptions	128
4.5.2 Using property links	128
4.5.3 Using inter-linked classes	130
4.5.4 Using n-ary class modelling	133
4.5.5 A general pattern for roles and statuses	134
Querying a TOC-based model via SPARQL	136
Reusing external classes as categories	137
Constructing second-order inferences	138
4.5.6 Identifying person's roles with PRO	139
4.5.7 Specifying document statuses with PSO	141
4.6 Describing publishing workflows with PWO	145
4.7 How communities uptake SPAR	149
4.7.1 SWAN ontology	149
4.7.2 CiteULike	151
4.7.3 WordPress	151
4.7.4 Linked Education	151
4.7.5 Virtual Observatory	151
4.7.6 Open Citations Corpus	151
4.7.7 WebTracks	152
4.7.8 Società editrice il Mulino	152
4.7.9 Utopia	153

5	Interfaces for the masses	155
5.1	LODE: generating HTML documentation from ontologies	156
5.1.1	What axioms are used to create the documentation	157
5.1.2	Special parameters to call the service	159
	Parameter “owlapi”	161
	Parameter “imported”	161
	Parameter “closure”	161
	Parameter “reasoner”	161
	Parameter “lang”	162
5.1.3	URI fragments	162
5.1.4	Content negotiation via <i>.htaccess</i>	162
5.2	KC-Viz, a tool for visualising and navigating ontologies	163
5.2.1	Key Concept Extraction	165
5.2.2	KC-Viz main features	166
	Description of nodes and arcs	167
	Expansion	167
	Hiding	168
	Refresh visualization	169
	Integration with NeOn	169
	Dashboard	170
	Preferences	172
5.2.3	Empirical evaluation	172
5.3	Graffoo, a framework for visual ontology modelling	176
5.3.1	Introducing classes and properties	178
5.3.2	Defining restrictions and additional class axioms	179
5.3.3	Linking class individuals	180
5.3.4	Defining assertions between ontologies	180
5.4	Gaffe, a flexible and user-friendly authoring tool for semantic data	183
5.4.1	OWiki: ontology-driven generation of templates and forms for semantic wikis	185
	The architecture of OWiki	186
	Using ontologies to model the domain	187
	Using ontologies to model the interface	189
5.4.2	Studying OWiki through a use-case	190
	From ontologies to forms	190
	Forms customization and filling	191
	From semantic data to templates and views	192

6 Conclusions	195
6.1 EARMARK: future works	197
6.2 SPAR: future works	198
6.3 LODE: future works	198
6.4 KC-Viz: future works	199
6.5 Graffoo: future works	199
6.6 Gaffe: future works	199
References	201

List of Tables

3.1	All the versions of a wiki page modified by different authors.	63
3.2	A brief summary of the structural pattern theory of [43] extended with two more patterns: <i>headed container</i> and <i>popup</i>	86
3.3	Testing associations between elements and patterns on the “Paradise Lost” example through an OWL reasoner.	87
4.1	Mappings between TOC entities and PRO classes.	140
4.2	Mappings between TOC entities and PSO classes.	143
5.1	Ontology Engineering Tasks.	173
5.2	Usability scores.	175

List of Figures

2.1	The four FRBR layers, with a specification of roles that people may play in each layer.	18
3.1	A Graffoo (see Section 5.3) representation of the EARMARK ontology.	35
3.2	Three EARMARK examples of overlapping between elements p	45
3.3	Encoding in EARMARK the ODT change-tracking example.	51
3.4	A graph summarizing the results of the first experiment.	57
3.5	A graph summarizing the results of the second experiment.	58
3.6	The abstract model of the EARMARK document solving the micro-formats issue.	61
3.7	The wiki sample versions encoded in a single EARMARK document.	66
3.8	Graffoo diagram that summarises the classes describing the pattern theory.	73
3.9	The EARMARK document, in the form of a graph, of the first three verses of the Paradise Lost by John Milton.	74
3.10	A diagram summarizing the ontology pattern <i>linguistic act</i>	78
4.1	A simple architectural diagram showing the interactions and dependencies between the component ontologies of SPAR.	91
4.2	The main FRBR object properties relating FRBR endeavours (work, expression, manifestation, item), and the related new object properties introduced by FaBiO (fabio:hasManifestation, fabio:hasRepresentation, fabio:hasPortrayal) to provide shortcuts between Work and Manifestation, Work and Item, and Expression and Item, respectively.	102
4.3	The extension to the common SKOS classes and relations implemented in FaBiO.	103

4.4	The diagram shows the CiTO v 2.0 object properties grouped in terms of their characterisation as rhetorical and/or factual, and, for the former, in terms of their connotation (positive, neutral or negative). All the properties shown, except cito:cites and cito:sharesAuthorsWith, are sub-properties of cito:cites itself. The inverse property of <i>cito:cites</i> , namely <i>cito:isCitedBy</i> , and its inverse sub-properties, are not shown. . .	106
4.5	Graffoo diagram summarising the <i>Bibliographic Reference Ontology (BiRO)</i>	110
4.6	Graffoo diagram summarising the <i>Literal Reification</i> pattern.	113
4.7	Graffoo diagram summarising the C4O entities used for counting citations and references.	118
4.8	Graffoo diagram summarising the C4O entities used for describing citation contexts.	119
4.9	Diagram describing the composition and the classes of the <i>Document Components Ontology (DoCO)</i>	126
4.10	A graphical representation of the <i>time-indexed situation</i> ontological pattern.	133
4.11	The Graffoo diagram of the <i>time-indexed owning in context</i> ontological pattern.	135
4.12	Graffoo representation of the Publishing Roles Ontology (PRO).	140
4.13	Graffoo representation of the Publishing Status Ontology (PSO).	142
4.14	Graffoo representation of the Publishing Workflow Ontology (PWO). .	146
4.15	The SWAN ontology ecosystem before (above) and after (below) the harmonisation activity that resulted in the inclusion of FaBiO and CiTO in the SWAN Commons set of ontologies.	150
5.1	The beginning of the Web page generated through LODE starting from the EARMARK Ontology, annotated with OWL assertions in Turtle showing how they are rendered in HTML.	159
5.2	Two possible kinds of descriptions: pure string (for literals) and media object (for resources).	160
5.3	How entities (classes, properties and individuals) are rendered by LODE.	160
5.4	All the possible ways, according to specific needs, for making a request to LODE.	161
5.5	The summarisation made by KC-Viz after its first application on an ontology.	166
5.6	Tooltips that appear hovering nodes and edges.	168
5.7	The menu popped up after clicking on the “Expand” option.	169

5.8	The two options for hiding concepts: “Hide”, applied on the class <i>Metadata</i> , and “Hide others...” used on the class <i>Work</i>	170
5.9	The option “Visualize Class with KC-Viz” to highlight (and eventually add) the class in the current KC-Viz panel.	171
5.10	The dashboard which allows the user to move back and forth through the history of KC-Viz operations, to modify the formatting of the layout, and to save the current display to a file, among other things. .	171
5.11	The preference panel of KC-Viz.	172
5.12	Performances (in seconds) for each task.	175
5.13	The legend for all possible Graffoo objects.	177
5.14	Widgets defining prefixes, classes, object/data properties and property axioms.	179
5.15	Widgets defining restrictions and other class axioms.	180
5.16	Widgets defining individuals and related assertions.	181
5.17	Widgets for defining ontologies and related assertions.	182
5.18	An example page of the Beer OWiki.	188
5.19	A graphical representation of the OWiki domain ontology about beers.	191
5.20	A customized form generated by OWiki.	192

Chapter 1

Introduction

This work is concerned with the increasing relationships between two distinct multidisciplinary research fields, *Semantic Web technologies* and *scholarly publishing*, that in this context converge into one precise research topic: *Semantic Publishing*. In the spirit of the original aim of Semantic Publishing, i.e. the improvement of scientific communication by means of semantic technologies, this thesis proposes theories, formalisms and applications for opening up semantic publishing to an effective interaction between scholarly documents (e.g., journal articles) and their related semantic and formal descriptions. In fact, the main aim of my work is to increase the users' comprehension of documents and to allow document enrichment, discovery and linkage to document-related resources and contexts, such as other articles and raw scientific data¹.

In order to achieve these goals, I investigate and propose solutions for three of the main issues that semantic publishing promises to address, namely: the need of tools for linking document text to a *formal representation of its meaning*, the lack of complete *metadata schemas* for describing documents according to the publishing vocabulary, and absence of effective *user interfaces* for easily acting on semantic publishing models and theories.

Semantic Publishing pioneers' visions, e.g. [164] [167] and [45], go beyond the current interest of publishers and Semantic Web practitioners, i.e., the "mere" addition of semantic data to published articles with the intention of recognising entities (such as persons, places and proteins) or, in general, inclusion of *subject-predicate-object* statements. In fact, their final aim is to be able to represent, semantically, the *scientific discourse* of a document, i.e., how arguments are modelled within the text. This, of course, requires one to identify formal meanings and relations between different part of textual content of a document (e.g., a sentence representing

¹In this way, my purposes conform to the principles that the research data contained in journal articles should be open, asserted in the Brussels Declaration on STM publishing [101].

the main *claim*, supported by other sentences stated as its *evidences*), as well as to indicate the rhetoric functions of clearly bounded parts of the article, such as the Introduction, the Background, the Results, and the Conclusions.

Moreover, in this context, an additional complication is given by the need to be aware of the intrinsic provenance of the conversion into formal statements of natural language texts. This is because multiple semantic interpretations can co-exist for the same text, and these can be defined in alternative, and even contrasting, formal representations. Since different people who may differ in their expertise with formal languages can make such semantic interpretations, a reader may trust one particular conversion more than others, depending on his/her confidence in the translators.

The practice that allows one to enrich natural language documents, associating explicit or implicit semantics to it, is that of *markup*. Markup has been used for years for decorating documents at all levels of granularity, from the digital document as a whole to its sub-components (chapters, paragraphs, sentences, people, places, events, etc.). Multiple and overlapping markup (being either *document markup* or *semantic markup*²) must be able to co-exist within the same textual content so as to allow one to express different – even alternative – semantics on it. The most commonly used document formats in publishing (i.e., XML and PDF) were not developed to enable semantic enhancement, although it may be possible in principle to use them for this purpose.

In this thesis, I investigate the plausible paths for reaching a full and synergistic integration between document markup (usually used for describing the structural components of documents) and semantic markup (used for specifying semantic annotations on textual content). To this end, I propose a markup metalanguage called *EARMARK* that allows one to instantiate the markup of a text document as an independent OWL document separated from the text strings it annotates. Through appropriate OWL characterizations, it can define structures such as trees or graphs and can be used to verify validity constraints, including semantic constraints and co-constraints that are currently not available in most validation languages. When used with a particular instantiation of an ontology developed to create semantics descriptions according to a semiotics model, called *Linguistic Meta-Model* [143], EARMARK allows one to specify formal semantics for any document markup item or text content.

Of course, markup is also used to add metadata to a document or to its sub-parts. It is a firm intention of semantic publishing to adopt effective and complete metadata

²Currently, a distinction is made between *document markup* (or *structural markup*), to enclose fragments of the text within markup elements (e.g., <title>Moby Dick</title>), and *semantic markup*, in which fragments of text become objects of RDF statements (e.g., :document dc:terms:title "Moby Dick"). In my opinion, these are just two sides of the same coin. Both can be used, either interchangeable or coupled, to represent the semantics of the content of a document. I think the reason that document and semantic markup are not used in this interchangeable way, thereby deserving two different names, is because of the different expressive powers of their most famous instances, i.e., XML for document markup and RDF for semantic markup.

schemas. Unfortunately, existing Semantic Web vocabularies are too abstract and incomplete to cover all the needs claimed by the actors involved in the publishing process (publishers, editors, authors, etc.). Thus, there is an acute need for new standards (ontologies) that comprehensively cover all the different aspects of the publishing domain.

To address these needs, I propose a suite of orthogonal and complementary OWL 2 DL ontology modules, the *Semantic Publishing And Referencing (SPAR)* ontologies, that enable one to describe all the aspects of bibliographic publications comprehensively in machine-readable RDF metadata. In the past, several works³ attempted to define general models for describing the whole publishing domain, but they are too abstract, are not interoperable with other models, or are very specific to particularly scenarios, and are thus not fit for purpose. In contrast, I define eight reusable SPAR ontological modules, that can be used either individually or in conjunction, as need dictates, each of them precisely and coherently covering one aspect of the publishing domain using terms with which publishers are familiar. Together, they provide the ability to describe bibliographic entities such as books and journal articles, reference citations, the organization of bibliographic records and references into bibliographies, ordered reference lists and library catalogues, the component parts of documents, and publishing roles, publishing statuses and publishing workflows.

Semantic publishing involves different kinds of users, from Semantic Web practitioners to publishers, some of whom may not have access to ontology modelling tools, or may not be expert in using formal languages, logic and inferences systems. Nevertheless, these users want to *understand* semantic publishing ontologies, to *make sense* of them, to *adopt* them and, finally, to *produce data* according to them. We thus require the development of interfaces that hide the complexity of ontology formalisms, and enable better interaction between non-experts and machine-readable datasets.

My contribution in this area has been the development of applications, that simplify and clarify the interaction with ontologies. First of all, tools are needed to help in ontology *understanding* and *sense-making*⁴. Furthermore, an essential step to open up the effective use of semantic publishing technologies to non-experts requires the development of *customisable user interfaces* that can be adapted quickly to particular contexts. Such interfaces must allow one to add/modify/remove semantic data correctly and intuitively, while hiding the complexity of the underlying ontologies.

³E.g., DC Metadata Elements [64], DC Metadata Terms [63], FRBR [100], PRISM [99] and BIBO [42], that are introduced in detail in Chapter 2.

⁴“Sense-making” is here used to refer to a specific ontology engineering task, where the user is primarily concerned with understanding the contents and overall structure of the ontology, i.e., acquiring an overview of the concepts covered by the ontology and the way they are organized in a taxonomy, so as to comprehend what the ontology can be used for.

An important part of my work has thus been to address these issues. Specifically, I have developed:

- *LODE*, an application that automatically extracts ontological definitions so as to render them in a human-readable HTML page designed for browsing and navigation by means of embedded links;
- *KC-Viz*, a second application that implements a novel approach for visualizing and navigating ontologies, by providing concise overviews of large ontologies, and that also supports a “middle-out” ontology navigation approach, starting from the most information-rich ontological classes;
- *Graffoo*, a tool for presenting the formalisms of ontologies as easy-to-comprehend graphical diagrams; and
- the new version of *Gaffe*, a tool that makes it possible to build customizable editors for semantic data and that allows users to decorate a resource with semantic descriptions (i.e. OWL assertions) according to any scheme expressed through an OWL ontology.

In my thesis, I introduce these three elements – markup, ontologies, and tools – in turn, emphasising their importance and innovativeness within semantic publishing communities and explaining why current available tools fail to address users’ needs in the same effective way.

The thesis is thus structured as follows. Chapter 2 discusses related works and main issues in semantic publishing. Chapter 3 introduces EARMARK, my markup metalanguage that enables semantic enhancement of markup structures and textual content. Chapter 4 illustrates my SPAR ontologies for modelling bibliographic data in the context of digital publishing. Chapter 5 focuses on the tools I have developed to help users when trying to understand ontologies and create semantic metadata using them. Final remarks and ideas for future works are contained in Chapter 6.

Chapter 2

The digital publishing revolution

In this chapter I discuss the most relevant research areas in semantic publishing. I focus particularly on my fields of interest: *markup* models to enable the addition of *semantics* within published scholarly documents, document *metadata schemas* and *ontologies*, and *interfaces* to allow a better comprehension and use of semantic data.

First of all, in order to better understand the general context where my work is set, it is useful to briefly digress on the changes happened to digital publishing during the last decade. Scholarly authoring and publishing are in the middle of a revolution that is exploring the potentiality derived, on the one hand, from the use of Web-related technologies (e.g., transport protocols, markup languages, the Semantic Web) as medium of communication, and, on the other hand, from the adoption of new publishing and editorial processes that seem to aim at converging to a fully-open accessibility of editorial contents and metadata.

The first step of this revolution was possible by means of the Web, which turned publishers to consider the digitalisation process and, consequently, the online publication as new effective ways of bibliographic publication. As predicted in past [136], to date the social and research impact of the online scholarly material is still continuing to grow. One of the main causes of this growth has been the introduction to the Open Access (OA) clause¹. Through it, publishers can either directly – the *gold OA* – or indirectly – the *green OA* [89] – (i.e., asking authors² for choosing to) publish online articles so as to offer their complete and free-of-charge worldwide readability and accessibility (in terms of being reachable).

Originally, the use of OA was considered a bet with low possibilities of success. However, previous works, such as [116] [88] [182], give empirical evidence about the advantages of OA in terms of better visibility, findability and accessibility for research articles. These factors and the development of clear and established strategies [171] [19] for shifting publishers' business model from a non-OA service to an OA publishing process are some of the most important reasons of the success of OA

¹Probably, the first formal document that uses the words “open access” is [28].

²In this case, authors can either publish in their own repositories or in journals' OA archives (obviously paying a conspicuous fee).

ideas – and of the (increasing) growth and consensus in digital online publication. Moreover, innovative publishing approaches have been recently proposed (e.g., the Liquid Publications Project [168]) so as to show how to use Web technologies and Open Access principles to change and improve the current publication process.

Of course, we have covered only a first bit of the long way towards a successful and widely accepted Web-oriented digitalisation and publishing of bibliographic materials. In fact, publishers have not adopted Web standards for their work yet. Rather, they still employ a variety of proprietary XML-based informational models and document type definitions (DTDs). While such independence was reasonable in the pre-Web world of paper publishing, it now appears anachronistic, since publications from different sources and their metadata are incompatible, requiring hand-crafted mappings to convert from one to another. For a large community such as publishers, this lack of standard definitions that could be adopted and reused across the entire industry represents losses in terms of money, time and effort.

In contrast, modern web information management techniques employ standards such as RDF [32] and OWL 2 [127] to encode information in ways that permit computers to query metadata and integrate web-based information from multiple resources in an automated manner. Since the processes of scholarly communication are central to the practice of science, it is essential that publishers now adopt such standards to permit inference over the entire corpus of scholarly communication represented in journals, books and conference proceedings. This requires the availability of appropriate ontologies and tools that are specially tailored to the requirements of authors, publishers, readers, librarians and archivists.

In the past some research institutes and companies that investigated upon publishing topics started to figure out whether and how Web technologies could address the above issues. In retrospect, that moment can be marked as the beginning of what we today call *semantic publishing*.

Semantic publishing is the use of Web and Semantic Web technologies to enhance a published document such as a journal article so as to enable the definition of formal representations of its meaning, facilitate its automatic discovery, enable its linking to semantically related articles, provide access to data within the article in actionable form, and allow integration of data between papers [167] [164]. As confirmed by a number of recent initiatives³, semantic publishing and scholarly citation using Web standards are presently two of the most interesting topics within the scientific publishing domain. I identify some significant research areas in this domain, that include:

1. the *development of markup technologies* that facilitate the creation of complex and semantically-enhanced markup documents, so as to make possible to

³The various initiatives that have been involved research communities around the topics and issues of Semantic Publishing, e.g. the Elsevier Grand Challenge and the SePublica 2011 Workshop, will be introduced in Section 2.4.

- have, simultaneously, a formal semantic description of their structures (e.g., chapters, introduction, paragraphs) as well as of their content;
2. the *development of semantic models* (vocabularies, ontologies) that meet the requirements of scholarly authoring and publishing;
 3. the *development of visualization and documentation tools* that permit such ontologies to be easily understood by users being neither masters nor technicians of particular modelling languages;
 4. the *development of annotation tools* that allow these models to be used by end-users (e.g., publishers, editors, authors) for enhancing documents with relevant semantic assertions;
 5. the *development of new algorithms* that can take advantages of this new semantic layer of annotations, for example when searching over large sets of on-line documents;
 6. the *development of new business models* that arrange effective publishing processes for the creation, use and dissemination of semantic assertions;
 7. the *study and realisation of empirical evaluations* that prove benefits and/or drawbacks of Semantic Publishing for both authors and publishers, such as understanding whether its use increases the impact factor of articles and/or the amount of visits of publishers' Web pages;
 8. the *organisation of events*, such as conferences, workshops, projects, journal issues, for publicising and promoting semantic publishing principles and advantages to a broader audience.

In the rest of this chapter I outline related works that propose solutions for addressing the issues concerning the first four points of the above list, that represent the research areas in which my work is set. I conclude this chapter listing a series of events (i.e., projects, workshops, journal issues, competitions) that characterise development on semantic publishing between 2010 and 2011.

2.1 Towards semantics-aware markup languages

The original definition of markup clearly states that it is used for saying *something* about the content of a document [40]. Understanding what “something” refers to is strictly dependant on the particular *semantics* adopted by the markup vocabulary considered. Of course, this “semantics” can be implicit (as in XML, where the semantics lives either in the mind of the language author), explicit and not formalised

(e.g., defined through a natural language and non-machine-readable description), or explicit and formalised (e.g., defined via logic languages such as prolog and OWL).

Besides having explicit or implicit semantics, overlapping markup structures are needed when different agents associate multiple (even discording) semantics to the same document fragment. Note that having two different interpretations of a particular document passage is possible, in particular within a domain – i.e., Semantic Publishing – where the analysis and formalisation of the scientific discourse are encouraged. Thus, the topic of overlapping markup, that has been discussed and investigated for years, becomes extremely significant in this context.

In the following sections I introduce past studies about both overlapping markup and markup semantics, which are two of the most interesting topics in markup research communities.

2.1.1 Overlapping markup

The need for multiple overlapping structures over documents using markup syntaxes such as XML and SGML is an age-old issue, and a large amount of literature exists about techniques, languages and tools that allow users to create multiple entangled hierarchies over the same content. A good review can be found in [48].

Some of such research proposes to use plain hierarchical markup (i.e., XML) and employ specially tailored elements or attributes to express the semantics of overlapping in an implicit way. The TEI Guidelines [186] present a number of different techniques that use SGML/XML constructs to force multiple hierarchies into a single one, including:

- *milestones*, through which one hierarchy is expressed using the standard hierarchical XML markup and the elements belonging to the other ones are represented through a pair of empty elements representing the start and the end tags, and connected to each other by special attributes;
- *flat milestones*, that represents each of the hierarchy elements as a milestone, i.e., an empty element placed where the start or the end tag should be, all of them contained as children of the same root element;
- *fragmentation*, in which one hierarchy (the primary) is expressed though the standard hierarchical XML markup, and the elements of the secondary hierarchies are fragmented within the primary elements as needed to suit the primary hierarchy and are connected to each other by special attributes;
- *twin documents*, in which each hierarchy is represented by a different document, which contains the same textual content but marks up the elements according to the individual hierarchy;

- *stand-off markup*, which puts all the textual content in a single structure with the possible specification of the shared hierarchy, and puts the remaining elements in other structures (e.g., files) with the positional association of each starting and ending location to the main structure, using, for instance, XPointer [49] locations.

Given the large number of techniques to deal with overlapping structures in XML, in [122] Marinelli et al. present a number of algorithms to convert XML documents with overlapping structures from and to the most common approaches, as well as a prototype implementation.

In [151] Riggs introduces a slightly different technique for fragmentation within XML structures. In this proposal, *floating elements*, i.e., those elements that do not fall in a proper or meaningful hierarchical order, are created using the name of the element followed by an index referring to its semantically-related parent element. For example, the floating element `<name.person[2]>John</name.person[2]>` means that `<name>John</name>` is semantically child of the second occurrence of the element *person*, even though the floating element is not structurally contained by its logical parent.

Other research even proposes to get rid of the theory of trees at the base of XML/SGML altogether, and use different underlying models and newly invented XML-like languages that allow the expression of overlaps through some kind of syntactical flourishing.

For instance, *GODDAG* [176] is a family of graph-theoretical data structures to handle overlapping markup. A GODDAG is a Direct Acyclic Graph whose nodes represent markup elements and text. Arcs are used to explicitly represent containment and father-child relations. Since multiple arcs can be directed to the same node, overlapping structures can be straightforwardly represented in GODDAG. Full GODDAGs cannot be linearised in any form using embedded markup, but *restricted GODDAGs*, a subset thereof, can be and has been linearised into TexMecs [97] [120], a multi-hierarchical markup language that also allows full GODDAGs through appropriate workarounds, such as virtual elements.

LMNL [185] is a general data model based on the idea of *layered text fragments and ranges*, where multiple types of overlap can be modelled using concepts drawn from the mathematical theory of intervals. Multiple serializations of LMNL exist, such as CLIX and LMNL-syntax.

XConcur [160] is a similar solution based on the representation of multiple hierarchies within the same document through *layers*. Strictly related to its predecessor CONCUR as it was included in the SGML, XConcur was developed in conjunction with the validation language XConcur-CL to handle relationships and constraints between multiple hierarchies.

The *variant graph* approach [158] is also based on graph theory. Developed to deal with textual variations – that generate multiple versions of the same document

with multiple overlapping hierarchies – this theory proposes a new data model to represent literary documents and a graph linearisation (based on lists) that scales well even with a large number of versions. Schmidt et al. recently presented an extension of their theory that also allows users to merge multiple variants into one document [157]. In [145] a detailed survey about overlapping approaches was presented, also discussing the MultiX² data model – that uses W3C standard languages such as XInclude to link and fetch text fragments within overlapping structures – and a prototype editor for the creation of multi-structured documents.

In [188] a proposal for using RDF as a standoff notation for overlapping structures of XML documents was proposed. By means of the open-source API *RDF Textual Encoding Framework (RDFTeF)*, Tummarello et al. demonstrate a plausible way for handling overlapping markup within documents and identifying textual content of a document as a set of independent RDF resources that can be linked mutually and with other parent resources.

Besides giving the possibility to define multiple structural markup hierarchies over the same text content, the use of RDF as the language for encoding markup allows one to specify semantic data on textual content as well. But the real main advantage of RDF is the possibility of using particular built-in resources that describe different kinds of containers, either ordered (rdf:Seq) or unordered (rdf:Bag), as defined in the RDF syntax specification [32]. Thus, RDF resources can be used to represent every printable element in the text – words, punctuation, characters, typographical symbols, and so on – while RDF containers can be used to combine such fragments and containers as well.

Although RDF is not sufficient to define a formal vocabulary for structural markup – does a given resource represent an element, an attribute, a comment or a text node? In which way is a resource of a certain type related to others? – the specification of an RDFS [24] or of an OWL [127] layer can successfully address this issue. Hybrid solutions obtained by mixing different models, even when they are built one upon another, may seem elegant but not necessarily the best choice. In fact, there exist well-known interoperability limits between OWL 2 DL and RDF [113] that prevent the correct use of Semantic Web tools and technologies. In particular:

- any markup document made using RDF containers (e.g. to describe what markup items contain and in which order) and OWL ontologies (e.g. to define classes of markup entities and their semantics) results in a set of axioms that end up outside of OWL DL and reach the OWL Full expressive power. This limits the applicability of the most frequently used Semantic Web tools, that are usually built upon the (computationally-tractable) description logic underlying OWL 2 DL;
- the individual analysis of each language may be not applicable when we have to check particular properties that lay between RDF and OWL layers. For

example, verifying the validity of a markup document against a particular schema, which is one of the most common activities with markup, needs a work with both markup item structures (that would be defined in RDF) and logical constraints about classes of markup items (e.g., elements only, attributes only, the element “p”, all the element of a particular namespace, etc., all of them definable in OWL).

Being able to express everything we need directly in OWL addresses both issues quite straightforwardly. The well known absence of containers and sequences in OWL can be overcome by modelling classes in specific ways using specific design patterns such as [35] and [61].

2.1.2 Markup semantics and semantic markup

The advent of the Semantic Web (and social web) has induced a shift of meaning for some terms that are traditionally associated with markup languages. Originally, the act of *marking up* was strictly associated with document markup, where the term “tag” was used to refer to *markup elements*: syntactic items representing the building blocks of a document structure. While, in the original definition, markup “tells us something about [the text or content of a *document*]” [40], in the Semantic Web the term “markup” is sometimes used to identify any data added to a *resource* with the intention to semantically describe it (as well as “metadata” or “resource description”). Because of this recent re-drawing of the markup meaning, the term “tag” has also drastically changed its definition to “a non-hierarchical keyword or term assigned to a piece of information (such as an Internet bookmark, digital image, or computer file)”⁴.

Partially because of this shift of meaning – that brought, as first consequence, the fact of having two different (and often unrelated) visions of the Web: the *Web of documents* and the *Web of data* – the Semantic Web has not considered in detail the issue of *markup semantics* (e.g., what is the meaning of a markup element *title* contained in a document *d*?), concentrating all its efforts in dealing with *semantic markup* (e.g., the resource *r* has the string “Semantic enhancement of document markup” as title) [149].

However, markup semantics is a very well-known and relevant issue for markup languages and consequently for digital libraries. Nowadays, a large amount of content stored in digital libraries is encoded with XML. XML, as any markup (meta)language, provides a machine-readable mechanism for defining document structure, by associating labels to fragments of text and/or other markup. This association has a particular meaning, since each markup element asserts something about its content. What is asserted by the markup is not an issue of the markup

⁴http://en.wikipedia.org/wiki/Tag_%28metadata%29

itself. One of the goals of markup metalanguages is to avoid imposing any particular semantics: they express mere syntactic labels on the text, leaving the implicit semantics of the markup to the interpretation of humans or tools programmed by a human. Of course, a lot of markup languages, such as HTML, TEI and DocBook, are accompanied by natural language descriptions of their markup, but those descriptions are not machine-readable; in other words, there is no formal mechanism to embed markup semantics within markup language schemas.

Previous works [149] [150] [178] pointed out some clear advantages in having a mechanism to define a machine-readable semantics of markup languages: enabling parsers to perform both syntactic and *semantic validation* of document markup; *inferring facts* from documents automatically by means of inference systems and reasoners; simplifying the *federation*, *conversion* and *translation* of documents marked up with different and non-interoperable markup vocabularies; allowing users to *query* upon the structure of the document considering its semantics; creating *visualizations* of documents by considering the semantics of their structure rather than the specific vocabulary in which they are marked up; increasing the accessibility of documents' content, even in the case of *tag abuse* [62], i.e., “using markup languages construction in ways other than intended by the language designer”; promoting a more flexible *software design* for those applications that use markup languages, guaranteeing a better *Maintainability* even when markup language schemas evolve.

For instance, it could be interesting to query documents for specific XML structures (e.g., all data tables in a collection of scientific papers written by a specific author, regardless of the fact that they were marked up with different vocabularies), or verifying semantic constraints of XML elements regardless of their position within the document (e.g., the utterer of each instance of the speech fragments as transcribed in a parliamentary debate document is uniquely assigned to the individual that purportedly made the speech).

Although the Semantic Web might directly address XML semantics in order to gather the above-mentioned advantages, the Semantic Web community has always considered XML only as a serialization language for RDF or OWL, or as a way to encode relational data to be subsequently extracted and expressed in RDF. However, these two usages depart from the original goal of XML, i.e. to provide a mechanism for marking up digital documents (books, papers, messages, etc.). Consequently, it is often the case that e.g. relational data in XML encode both domain and document semantics; in such cases, extracting semantics from markup by means of bulk recipes generates semantic issues, because the dataset and/or ontologies obtained from that extraction will be unreliable (due to the usually conflicting data/text implicit semantics). A case study of this heterogeneity is the translation of FAO FIGIS document management schemata⁵, which generates an ontology describing real world entities as well as documents, provenance, interfaces, versioning data, etc.

⁵<http://www.fao.org/fi/figis/devcon/diXionary/index.html>

There is a large literature about semantics applied to markup. One of the first attempts for describing formal markup semantics is introduced in [177]. The basic idea of Sperberg-McQueen et al. is to point out how users apply markup: through it, they make inferences about the document structures and the text those structures contain. According to them, “the meaning of markup is the set of inferences it licenses”. The general framework they developed to associate semantics to markup and to make inferences on it needs some representation of the markup document, a *sentence skeleton* for each item of the markup language we are considering in order to associate a meaning, and a set of (categorized) predicates and rules for allowing inferences. In this work, all the examples are illustrated using Prolog both for the representation of the nodes and for defining/inferring semantics using predicates and rules.

Focusing on the best-known meta-markup language, XML, in [150] Renear et al. discuss problems characterizing schema languages for XML, from DTD to XMLSchema: those languages only permit a clear definition of the language syntax, and some of them (RelaxNG [37], XMLSchema [75]) allow the declaration of a simple semantics on the datatypes, and little more. Although annotations can be specified for XMLSchema structures, there is no predefined semantics associated to them. Everything else concerning semantics – the meaning of an element, the relationships among items, etc. – is not expressible in a machine-readable format through those schema languages. The Renear et al. propose the BECHAMEL Project as a candidate solution to express markup semantics. As they explain in [149], BECHAMEL allows one to associate semantics with markup by adding new hierarchies to the original structure of the document. Using these additional hierarchies, one can define the meaning of the elements and properties that cannot be expressed using the schema languages alone.

A different approach is used in [169]. Simons et al. developed a framework to associate semantics with any XML document D in a three-step process:

1. defining an OWL ontology O to express all the meanings they want to use;
2. writing a set of rules R in a specific XML language to associate those meanings to a set of elements D ;
3. through a XSLT transformation, processing D using O and R , so obtaining a new semantically-enriched XML document.

Similarly to the previous one, other works, such as [133] [76] [189], propose a general process that, starting from an XMLSchema S , an XML document D (written according to S) and an ontology O (that can be generated starting from S), allows one to convert all the data in D , described by XML elements and attributes, into appropriate RDF instances consistent with O .

The approach introduced in [119] and [121] does not provide a formal machine-readable specification for defining markup semantics, but it is useful when human

interpretation is needed in structuring a document. Marcoux et al. describe *Intertextual Semantics*, a mechanism to associate meaning with markup elements and attributes of a schema as natural language constructs; this happens by associating a pre-text and a post-text with each of them. When the vocabulary of a schema is used correctly, the markup content is combined with the pre-text and post-text descriptions to make a correct natural language text that describes the entire information contained in a document. The difference between the common natural language documentation and Intertextual Semantics is that in the latter the meaning of a markup item is dynamically added when writing a document, and, as a consequence, can be read sequentially in the document editor itself.

Of course, eRDF⁶ and RDFa [2] may be valid choices for associating – and extracting by means of GRDDL [39] applications – formal semantics with arbitrary text fragments, and to markup elements within documents. Although they are very helpful for annotating documents and adding semantic information about markup elements and their content, their use is possible only by adding new attributes or, worse, new elements, therefore changing the document structure. The problem here is that the need of modifying the document structure is not easily suitable for domains, for example within organizations that deal with administrative or juridical documents, which must always preserve their structure as it is.

2.2 Metadata schema, vocabularies and ontologies for publishing

The definition of vocabularies and ontologies that enable the description of document metadata is crucial for the Semantic Publishing. A large amount of these metadata schemas came to light in the nineties, and only years after their related Semantic Web versions were developed, either as RDF/RDFS vocabularies or OWL ontologies.

Several vocabularies and/or models for the publishing domain have been developed in the past. In this section I specifically list those that are usually adopted and currently defined through Semantic Web languages and technologies.

2.2.1 Dublin Core

Born as consequence of a conference held in Dublin, Ohio, USA in 1995 that involved both technicians (librarians, publishers, archivists) and academics (researches, software developers), the current versions of the Dublin Core (DC) Metadata Elements [64] and of the DC Metadata Terms [63] are the most widely used vocabularies for describing and cataloguing resources.

⁶<http://www.egeneva.ch/w3c-RDF-ResourceDescriptionFramework/>

These vocabularies have become particularly important and relevant for sharing metadata about documents among different repositories [111] and digital libraries [125], as well as being used to describe documents in HTML [65], DocBook [194] and other XML formats such as Open Document (OpenOffice document format) [104].

While very useful for the creation of basic metadata for resource discovery, the main limitations of DC is a consequence of the **generic nature of its terms**. For example, using DC Terms one can identify a creator but not an author, a bibliographic resource but not a journal article, an identifier but not an ISSN, and a date but not a publication date.

2.2.2 PRISM

The *Publishing Requirements for Industry Standard Metadata (PRISM)* [99] is a specification defining a rich set of metadata terms for describing published works. It was born from the needs of publishers to address emerging requirements for metadata sharing and aggregation, and it nowadays involves some of the most important publishers and related companies, such as *Adobe Systems*, the *McGraw-Hill Companies*, *Reader's Digest*, *Time Inc.*, the *Nature Publishing Group*, and *U.S. News and World Report*.

The PRISM metadata terms are expressible both in XML, according to a specific DTD, and in RDF [87]. These terms are explicitly recommended for the specification of metadata of documents expressed through markup languages such as DocBook [194]. Moreover, these terms are also included in ontologies describing the publishing domain, such as the *Bibliographic Ontology (BIBO)*⁷ [42], which is discussed below.

While PRISM has a much richer set of terms that describe bibliographic entities than DC, its main limitation is that it is a **flat structure**, lacking hierarchies. This prevents its use for a complete description of the characteristics of bibliographic entities. For example, while the data property *prism:volume* permits the volume number of a bibliographic reference to be represented as a string, PRISM lacks the concept of “Volume” as a distinct class among other bibliographic classes that have a hierarchical partitive relationship to one another (i.e. Journal Article > Issue > Volume > Journal), and whose members can possess other properties, such as having authors and editors.

2.2.3 BIBO

BIBO, i.e. the *Bibliographic Ontology* [42], is an OWL Full ontology that allows one to write descriptions of documents (*bibo:Document* is the core class of that model) for publication on the Semantic Web. It includes both DC terms [63] and PRISM [99] to cover common needs, and it adds other classes and properties to better describe the publishing domain, such as *bibo:AcademicArticle*, *bibo:Journal*, *bibo:Collection*,

⁷BIBO, the Bibliographic Ontology: <http://purl.org/ontology/bibo/>.

bibo:Book, *bibo:Chapter* and *bibo:Issue*. BIBO is a good ontology that is widely used among the bibliographic community.

From a pure computational perspective, BIBO defines the range of the property *bibo:authorList* using either an *rdf:List* or an *rdf:Seq*, therefore making the model non-compliant with OWL 2 DL. This limits the applicability of reasoners and other Semantic Web tools that are usually built upon the (computationally-tractable) description logic underlying OWL 2 DL.

2.2.4 MARC 21

Another relevant work in this field, widely used in the Libraries community and developed before the introduction of the Semantic Web, is the *MARC 21 Format for Bibliographic Data* [118]. Dating from an original start in 1961, MARC 21 is a very complex code for describing bibliographic resources as one of seven different primary types: book, continuing resource, computer file, maps, music, visual materials and mixed materials. To each resource can be associated different kinds of metadata, such as titles, names, subjects, notes, publication data, etc.

In MARC21, each type of metadata is represented by a three-digit code (called a *tag* in the MARC21 specification) that identifies the main metadata category of relevance. Other characters can follow this tag so as to specify additional information. For example, let me introduce a simple bibliographic reference describing [121]:

Yves Marcoux, Elias Rizkallah (2009). Intertextual semantics: A semantics for information design. <http://onlinelibrary.wiley.com/doi/10.1002/asi.21134/full>.

To express these data in MARC21 I would have to use the following tags:

```
100 1#$aMarcoux, Yves
100 1#$aRizkallah, Elias
260 ##$c2009
145 10$aIntertextual semantics: $bA semantics for information
     design
856 40$uhttp://onlinelibrary.wiley.com/doi/10.1002/asi.21134/
     full
```

where “100” is used for identifying a personal name, “260” indicates the year of publication, “145” the title of a work, and “856” the electronic location of that entity.

With the advent of the Semantic Web, MARC21 was formalised as an RDF vocabulary [180] in order to be adopted and used in Semantic Web applications. However, many librarians now regard MARC as too complex and esoteric, and are undergoing a mind shift to more pragmatic open standards.

2.2.5 FRBR

The *Functional Requirements for Bibliographic Record (FRBR)* [100] is a general model, proposed by the International Federation of Library Association (IFLA), for describing documents and their evolution. It works for both physical and digital resources and has proved to be very flexible and powerful. One of the most important aspects of FRBR is the fact that it is not associated with a particular metadata schema or implementation.

The following brief description outlines FRBR's basic concepts and the way they can be applied within a publishing domain. FRBR describes all documents from four different and correlated points of view, those of *Work*, *Expression*, *Manifestation* and *Item*, each of which is a FRBR *Endeavour*. These can be illustrated by consideration of the book *Alice's Adventures in Wonderland* by Lewis Carroll:

- **Work.** A FRBR *Work* is a high-level abstract Platonic concept of the *essence* of a distinct intellectual or artistic creation, for example the ideas in Lewis Carroll's head concerning *Alice's Adventures in Wonderland*, independent of any representation of these ideas in a particular form. A *Work* is realized through one or more *Expressions*;
- **Expression.** A FRBR *Expression* is the realisation of the intellectual or artistic *content* of a *Work*. Thus the original text of *Alice's Adventures in Wonderland* and its Italian translation *Le Avventure di Alice nel Paese delle Meraviglie* refer to different *Expressions* of the same *Work*. An *Expression* is embodied in one or more *Manifestations*;
- **Manifestation.** A FRBR *Manifestation* of a work defines its particular physical or electronic embodiment, for example, the particular *format* in which “Alice's Adventures in Wonderland” is stored: as a printed object or in HTML, that are two quite different *Manifestations*. In publishing, different manifestations of a journal article will all bear the same Digital Object Identifier (DOI), which identifies the *Expression* of the work, not its various *Manifestations*. However, a paperback and a hardback version of a book will bear different International Standard Book Numbers (ISBNs), since these identifiers are assigned at the *Manifestation* level. A *Manifestation* is exemplified by one or more *Items*;
- **Item.** A FRBR *Item* is a particular physical or electronic *copy* of *Alice's Adventures in Wonderland* that a person can own, for example the printed version of that book you have in your bookcase, or the MobiPocket format copy you have downloaded to read on your e-book device. All *Items* that are identical to one another – for example books from the same printing, are exemplars of the same *Manifestation*.

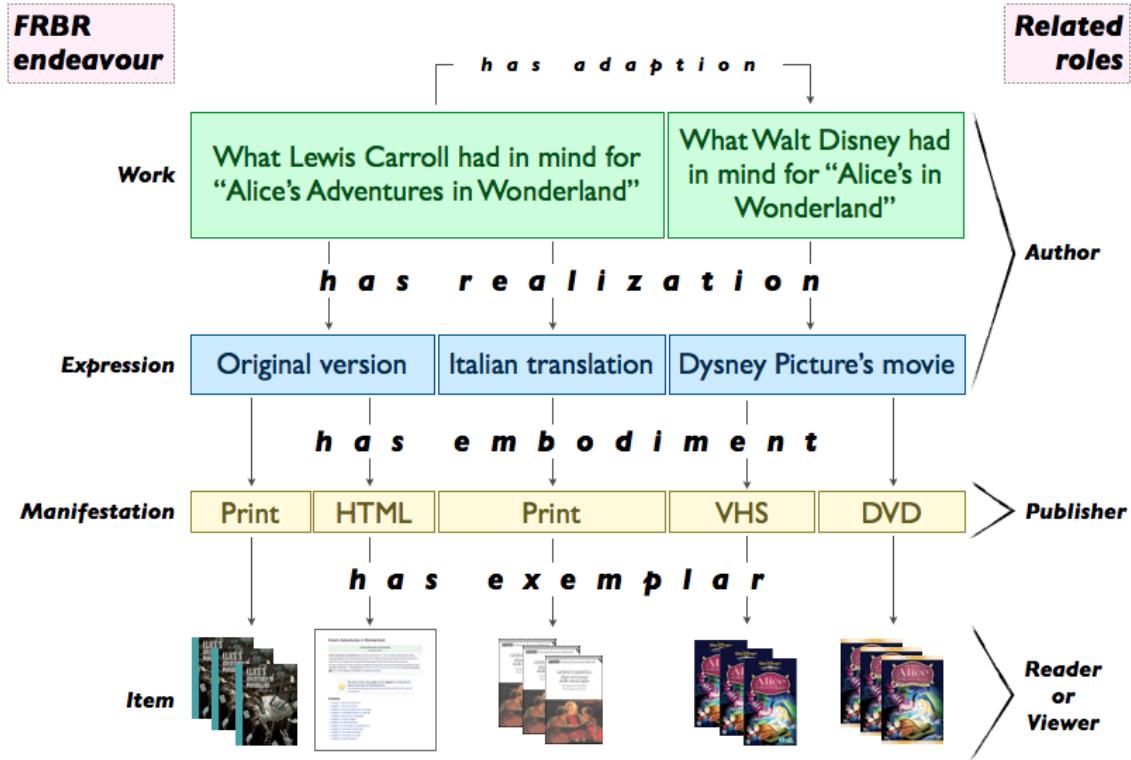


Figure 2.1: The four FRBR layers, with a specification of roles that people may play in each layer.

In Fig. 2.1, I summarise these four distinct FRBR layers with particular reference to the publishing domain, using as our example *Alice's Adventures in Wonderland*, and I indicate the most common roles (*Author*, *Publisher* and *Reader/Viewer*) that usually people have with respect to each layer.

Despite the increased expressivity enabled by these layers, the greatest limitation of FRBR with respect to the publishing domain is its lack of terms **that permit publications to be described in normal everyday language** (e.g. “research paper”, “review”, “book chapter”, “newspaper editorial”) rather than using the more abstract and esoteric FRBR-specific terms “work”, “expression”, “manifestation” and “item”.

A further limitation that FRBR has in common to other standards – i.e., DC [64] [63], PRISM [99] – is that it has hitherto been implemented and shared **only as XML or RDF vocabularies**, rather than as OWL DL ontologies, preventing them from being used within applications that employ reasoning based on description logic models.

There now exist two different implementations of the core concepts of FRBR

using standards that permit the encoding of proper formal semantics: the first is authored in 2005 by Richard Newman and Ian Davis in RDFS⁸ and the second, developed from the first, was created in 2010 by Paolo Ciccarese and I in OWL 2 DL⁹.

2.2.6 SWAN Citations Ontology

Another model previously used to define bibliographic resources is the *Citations Ontology*¹⁰ included in the SWAN (Semantic Web Applications in Neuroscience) Ontologies, version 1.2 [35]. In this ontology, the main class *Citation*¹¹ is used as super-class, of which all the other resources (e.g. *JournalArticle*, *WebArticle* and *Book*) are sub-classes.

The main advantage of that ontology is that it is completely developed in OWL 2 DL. Contrary to BIBO, which defines the range of the property *bibo:authorList* using either an *rdf:List* or an *rdf:Seq* therefore making the model non-compliant with OWL 2 DL, the Citation Ontology uses the SWAN Collections Ontology module¹². This is an OWL 2 DL ontology that enable one to handle lists of authors and contributors of a bibliographic object, thus enabling the specification of ordered lists while still keeping the ontology locally consistent.

The main problem of the Citations Ontology is the sparseness of its vocabulary, and the problem of aligning it with other structurally complex models such as FRBR, because, as with BIBO, it collapses all bibliographic entity descriptions within the single class *Citation*.

2.2.7 SKOS

Publishers need to classify the documents they published according to discipline-specific thesauri or classification schemes, for example those belonging to economics¹³ or computer science¹⁴.

The *Simple Knowledge Organization System (SKOS)* [123] is an RDFS model to support the use of knowledge organization systems (KOS) such as thesauri, classification schemes, subject heading lists and taxonomies within the framework of

⁸The FRBR Core in RDFS: <http://vocab.org/frbr/core>.

⁹The FRBR Core in OWL 2 DL: <http://purl.org/spar/frbr>.

¹⁰SWAN Citations Ontology Module: <http://swan.mindinformatics.org/spec/1.2/citations.html>.

¹¹Note that in SWAN the concept “Citation” is used to represent the cited object itself, rather than the performative act of making a citation.

¹²SWAN Collections Ontology Module: <http://swan.mindinformatics.org/spec/1.2/ collections.html>.

¹³The Journal of Economic Literature Classification Scheme: http://www.aeaweb.org/jel/jel_class_system.php.

¹⁴The Association for Computing Machinery (ACM) Computing Classification System 1998: <http://www.acm.org/about/class/1998>.

the Semantic Web. The reception of this language has been particularly positive: a large number of well-known thesauri and classification systems have started to convert their respective specifications into SKOS documents¹⁵, ¹⁶, ¹⁷, ¹⁸. This makes SKOS the *de facto* standard for encoding controlled vocabularies for the Semantic Web.

2.3 How to help users: tools and applications for semantic data

Semantic publishing end-users must be supported when choosing which ontology to adopt according to their needs. Obviously, *understanding* which ontology fits better within a particular domain may be not so trivial, in particular when users are not experts in ontology formalisms, the ontology has not a human-comprehensible *documentation*, or it is so large that it becomes difficult to quickly *make sense* of it. An additional complication is introduced when we need/want to develop new ontologies or to add/modify semantic data according to specific models. To this end, it is crucial to have functional interfaces that support the *creation* of an ontology and the *addition* of semantic data according to it.

In this section, I introduce a series of works that try to address all the issues I introduced above, namely: ontology documentation, ontology sense-making, visual modelling and authoring tools for ontologies.

2.3.1 Ontology documentation

The production of the natural language documentation of ontologies is an important and crucial part of any ontology development process. Such a documentation enable users to comprehend the extent of an ontology without caring about the particular formal language used to define its axioms. At the same time, writing the documentation is an activity that costs an important amount of effort. Thus, in order to help authors of ontologies to document them, applications were developed for the creation of a first draft of the documentation starting from labels (i.e., *rdfs:label*), comments (i.e., *rdfs:comment*), other kinds of annotations (e.g., *dc:description*, *dc:creator*, *dc:date*) and the logical structure of the ontology itself.

*SpecGen*¹⁹ is a Python tool for the generation of ontology specifications, released under the MIT license²⁰. It is available as standalone application and it was used

¹⁵ AGROVOC: <http://aims.fao.org/website/AGROVOC-Thesaurus/sub>.

¹⁶ The Medical Subject Headings (MeSH): <http://www.ncbi.nlm.nih.gov/mesh>.

¹⁷ The Library of Congress Subject Headings (LCSH): <http://id.loc.gov/search/>.

¹⁸ Nuovo Soggettario of the National Central Library in Florence: <http://thes.bnfc.firenze.sbn.it/>.

¹⁹ SpecGen: http://forge.morfeo-project.org/wiki_en/index.php/SpecGen.

²⁰ MIT license: <http://www.opensource.org/licenses/mit-license.php>.

to prepare the HTML documentation of well-known ontologies, such as SIOC²¹ [20]. SpecGen generates the documentation by processing an HTML template and adding the list of ontological classes and properties in specifiable positions within that template. As result, we obtain a new HTML document where the natural language description of the ontology come entirely from the template made by authors, while the software takes care of adding all the information related to the logical structure of the ontology.

Contrarily to SpenGen that needs a base HTML template to work, *VocDoc*²² is a (very little) Ruby script that allows one to produce documentation starting from RDFS vocabularies and OWL ontologies. It is able to produce both HTML documents and LaTeX files containing the description of the ontology/vocabulary.

Like VocDoc, *OWLDoc*²³ is a fully-automatic generator of a set of HTML pages describing the target ontology. It organises the documentation of each ontological entity in different parts: the taxonomy involving the entity, the usage of this entity in the context of the ontology, and all the formal logical axioms related to the entity (in Manchester Syntax [95]). OWLDoc has been developed as plugin of *Protégé*²⁴ [110] and as Web application²⁵.

Oriented to Linked Data applications rather than to ontology documentation, *Paget*²⁶ is a PHP framework that, getting an input URL through a browser, is able to dispatch the request according to the particular mime-type specified by the client. Paget returns RDF entities in four different formats: RDF, HTML, Turtle, JSON. It can be used to describe a set of pure RDF statements (*subject-predicate-object*)²⁷ and, to some extents, to produce an HTML human-comprehensible description from the axioms of an OWL ontology²⁸.

*Neologism*²⁹ [12] is a Web-based editor for the creation of RDFS vocabularies and (very simple) OWL ontologies. Moreover, it implements a publishing system that allows the publication of vocabularies and ontologies on the Web, rendered into natural language HTML pages. Basca et al.'s main goal is to reduce the time needed to create, publish and modify vocabularies for the Semantic Web.

Finally, Parrot³⁰ [184] is a Web service for the generation of HTML+Javascript documentation of OWL ontologies and RIF rules [21]. This service allows one to

²¹The Semantically-Interlinked Online Communities (SIOC) project: <http://sioc-project.org>.

²²VocDoc: <http://kantenwerk.org/vocdoc/>.

²³OWLDoc: <http://code.google.com/p/co-ode-owl-plugins/wiki/OWLDoc>.

²⁴Protégé: <http://protege.stanford.edu/>.

²⁵Ontology browser: <http://code.google.com/p/ontology-browser/>.

²⁶Paget: <http://code.google.com/p/paget>.

²⁷Ian Davis' Linked Data profile, rendered through Paget: <http://iandavis.com/id/me.html>.

²⁸A vocabulary for describing whisky varieties, rendered through Paget: <http://vocab.org/whisky/terms.html>.

²⁹Neologism: <http://neologism.deri.ie>.

³⁰Parrot: <http://ontorule-project.eu/parrot/parrot>.

specify multiple URLs identifying ontologies in order to produce an HTML summary of them “on the fly”, starting from their logical structure and annotations.

2.3.2 Ontology sense-making

The issue of how best to support visualization and navigation of ontologies has attracted much attention in the research community. As Wang and Parsia emphasize [196], “effective presentation of the hierarchies can be a big win for the users”, in particular, but not exclusively, during the early stages of a sense-making process, when a user is trying to build an initial mental model of an ontology, focusing less on specific representational details than on understanding the overall organization of the ontology. In particular, as discussed in [163], there are a number of functionalities that an effective visualization system needs to support, including (but not limited to) the ability to provide high level overviews of the data, to zoom in effectively on specific parts of the data, and to filter out irrelevant details and/or irrelevant parts of the data.

An approach to addressing the issue of providing high level overviews of hierarchical structures focuses on maximizing the amount of information on display, through space-filling solutions, such as those provided by *treemaps* [162]. Treemaps have proved to be a very successful and influential visualization method, used not just to represent conceptual hierarchies but also to visualize information in several mainstream sectors, including news, politics, stock market, sport, etc. However, while treemaps define a clever way to provide concise overviews of very large hierarchical spaces, they are primarily effective when the focus is on leaf nodes and on a particular dimension of visualization, in particular if colour-coding can be used to express different values for the dimension in question.

However, as pointed out in [196], treemaps are not necessarily effective in supporting an understanding of topological structures, which is what is primarily needed in the ontology sense-making context. State of the art ontology engineering toolkits, such as *Protégé* [110] and *TopBraid Composer*³¹, include visualisation systems that use the familiar node-link diagram paradigm to represent entities in an ontology and their taxonomic or domain relationships. In particular, both the *OwlViz visualizer* in Protégé and the “*Graph View*” in TopBraid make it possible for users to navigate the ontology hierarchy by selecting, expanding and hiding nodes. However OwlViz arguably provides more flexibility, allowing the user to customize the expansion radius and supporting different modalities of use, including the option of automatically visualizing in OwlViz the current selection shown in the Protégé Class Browser.

SpaceTree [144], which also follows the node-link diagram paradigm, is able to maximize the number of nodes on display, by assessing how much empty space is available. At the same time it also avoids clutter by utilising informative preview

³¹TopBraid Composer: <http://www.topbraidcomposer.com>.

icons. These include miniatures of a branch, which are able to give the user an idea of the size and shape of an un-expanded subtree at a very high level of abstraction, while minimizing the use of real estate.

Like treemaps, *CropCircles* [196] also uses geometric containment as an alternative to classic node-link displays. However, it tries to address the key weakness of treemaps, by sacrificing space in order to make it easier for users to understand the topological relations in an ontology, including both parent-child and sibling relations. The empirical evaluation comparing the performance of users on topological tasks using treemaps, CropCircles and SpaceTree introduced in [196] showed that, at least for some tasks, users of CropCircles performed significantly better than those using treemaps. However, SpaceTree appears to perform significantly better than either treemaps or CropCircles on node finding tasks.

A number of hybrid solutions also exist, such as Jambalaya [179] and Knoocks [112], which attempt to combine the different strengths of containment-based and node-link approaches in an integrated framework, by providing both alternative visualizations and hybrid, integrated views of the two paradigms.

The group of techniques categorized in [105] as “context + focus and distortion” are based on “the notion of distorting the view of the presented graph in order to combine context and focus. The node on focus is usually the central one and the rest of the nodes are presented around it, reduced in size until they reach a point that they are no longer visible” [105]. These techniques are normally based on hyperbolic views of the data and offer a good trade-off – a part of the ontology is shown in detailed view, while the rest is depicted around. A good exemplar of this class of approaches is HyperTree [172].

Finally, I should also consider the most ubiquitous and least visual class of tools, exemplified by plugins such as the *Class Browser* in Protégé and the *Ontology Navigator* in the *NeOn Toolkit*³² [181]. These follow the classic file system navigation metaphor, where clicking on a folder opens up its sub-folders. This approach is ubiquitous in both file system interfaces and ontology engineering tools and, in the case of ontologies, it allows the user to navigate the ontology hierarchy simply by clicking on the identifier of a class, to display its subclasses, and so on. While superficially a rather basic solution, especially when compared to some of the sophisticated visual metaphors that can be found in the literature, this approach can be surprisingly effective for two reasons:

- it is very familiar to users;
- it makes it possible to display quite a lot of information in a rather small amount of space, in contrast with node-link displays, which can be space-hungry.

³²NeOn Toolkit: <http://www.neon-toolkit.org>.

As a result it is not surprising that these interfaces often perform better in evaluation scenarios than the graphical alternatives. For instance, the evaluation reported in [106] shows that subjects using the Protégé Class Browser fared better than those using alternative visualization plugins in a number of ontology engineering tasks.

2.3.3 Visual modelling of ontologies

Usually, people who want to model ontologies have to be experts in formal languages (e.g., description logic) that must be enough expressive to enable the definition of all the semantic constraints required by particular needs of conceptualisation of a domain. As consequence of that, years ago a new professional role arose to specifically deal with the development of ontologies: the *ontology engineer*.

Nowadays an increasing amount of people of several fields (e.g., biology, medicine, literature, software engineering) is beginning to develop ontologies for their own needs. To this end, they usually adopt visual approaches that enable the development of complex ontologies hiding the intrinsic complexity of the formal language used. Moreover, as side effect, these visual approaches help when *presenting* ontologies to an audience that is not expert in formal languages.

One of the most common of these approaches is that of *semantic networks* [197]. A semantic network is a “graphic notation for representing knowledge in patterns of interconnected nodes and arcs” [174]. Ontology classes and individuals are defined as nodes of a graph (i.e., the visual representation of a semantic network). At the same time, direct and labelled arcs can interlink nodes so as to represent predicates between them: sub-class relations, belongingness to a class, individual assertions, etc. Although this tool is commonly used for representing OWL ontologies, it has been developed within the Artificial Intelligence field, years before the Semantic Web.

Another interesting approach coming from software engineering fields is that of using modified versions of UML [135], opportunely adapted to describe OWL ontologies. In [79], Gasevic et al. propose a new UML profile for the definition of OWL ontologies that is entirely based on the UML class notation. Moreover, they illustrate a process that allows one to produce OWL ontologies by applying an XSLT transformation to the XMI version of an UML document made through an UML editor (such as Poseidon³³).

Brockmans et al. [27] [26] propose another UML profile that enables one to define OWL entities using an extended set of UML-based graphic notations. In particular, the UML class notation is used for the representation of classes and, when opportunely decorated with dedicated stereotypes, class restrictions. Properties are represented as UML n-ary associations, eventually with some stereotypes specified

³³Poseidon for UML: <http://www.gentleware.com/products.html>.

according to property characteristics (e.g., functional, symmetric, reflexive). Individuals belonging to particular OWL classes are represented by using the UML “object:Class” notation, while OWL data types are represented as UML classes.

Two years ago, the industry consortium responsible of UML, i.e. the Object Management Group, released an official UML profile [134] for defining OWL ontologies, called *Ontology Definition Metamodel (ODM)*, which incorporates and harmonises works done previously on this topic. All these UML-like proposals appear to be less intuitive than the semantic network approach, since they ask users to learn at least some basic principles of the UML notations. Of course, the UML-like approach has been very successful in software engineering communities, because it introduced, with a relatively low effort, software engineers to ontology modelling.

2.3.4 Authoring tools for ontologies

Decorating resources (e.g., documents) with semantic data is usually a tedious task without the support of appropriate applications that make this activity practicable, intuitive and relatively quick. Luckily nowadays there exist many tools that help users to deal with metadata enrichment, such as metadata editors and automatic content processing mechanisms.

For instance, *DC-dot*³⁴ retrieves Web pages and automatically proposes related metadata according to Dublin Core Metadata Elements [64] and Terms [63]. Metadata can be edited using the form provided by the system, which is also accompanied by a context-sensitive help. However, DC-dot only provides text areas for Dublin Core resources, and does not allow domain-specific constraints and any other kinds of customizations to the Dublin Core standard.

Another tool specifically designed for Dublin Core data is *Metamaker*³⁵. This editor allows the creation of metadata from the scratch through simple web forms. Metadata can be saved in different formats, e.g. HTML, XHTML, XML, RDF or AGRIS AP³⁶. Unlike DC-dot, Metamarker allows one to use terms from external sources, such as the AGROVOC thesaurus³⁷.

*TKME*³⁸ is an application that allows the creation and modification of metadata, and organises them according to hierarchical structures (i.e., trees). Since TKME does not oblige the use of a particular metadata schema, it allows one to customise its interface according to alternative semantic models.

In contrast to TKME, *Metasaur* [107] provides a general visualisation tool for ontologies describing a particular domain. Through Metasaur, users can create lightweight ontologies according to existing metadata schemas as well as change

³⁴DC-dot: <http://www.ukoln.ac.uk/metadata/dcdot>.

³⁵Metamarker: <http://www.fao.org/aims/tools/metamaker.jsp>.

³⁶AGRIS Application Profiles: <http://aims.fao.org/standards/agmes/application-profiles/agris>.

³⁷AGROVOS thesaurus: <http://aims.fao.org/website/AGROVOC-Thesaurus/sub>.

³⁸TKME: <http://geology.usgs.gov/tools/metadata/tools/doc/tkme.html>.

existing ontologies adding new constraints and restrictions. As drawback, Metasaur has not a flexible user interface. In fact, although it creates automatically a form starting from the ontology in consideration, it does not allow users to customise and personalise in any way the form itself.

Some other approaches address metadata editing through Web applications, such as Wikis. Several approaches to semantic wikis have been developed to bring together the benefits of the free editing philosophy of wikis and ontological data. Semantic wikis can be organized into two main categories according to their connections with the ontologies: “wikis for ontologies” and “ontologies for wikis” [30]. In the first case, the wiki is used as a serialization of the ontology: each concept is mapped into a page and typed links are used to represent object properties.

Such a model – initially proposed in the ontology for MediaWiki articles, called WikiOnt [90] – has been adopted by most semantic wikis. SemanticMediaWiki [193] is undoubtedly the most relevant one. It provides users with an intuitive syntax to embed semantics, i.e. RDF statements, within the markup of a page. *SemanticMediaWiki* allows users to freely edit the content without any limitation. The more the information is correctly encoded the more semantic data are available, but no constraint is imposed over the authoring process. SemanticMediaWiki has been developed with the idea of creating a machine-readable version of Wikipedia, to better exploit that huge amount of information and the competencies and enthusiasm of its community.

The original term *Wikitology* [109] summarized very well the potentialities of such approach. The DBpedia project [6] is also worth mentioning, being the most recent effort in translating the Wikipedia content into RDF. One of the main obstacles to the realization of Wikitology and similar projects is certainly the difficulty in creating semantic content. Although the syntax is very simple, in fact, authors still have to learn some new markup and above all to manually write correct statements. *SemanticForms*³⁹ is an extension of SemanticMediaWiki that addresses such issue by allowing users to create semantic content via pre-defined forms. SemanticForms generates forms from templates, whose fragments and data have been previously typed. The generation process exploits an embedded mapping between each datatype and each type of field (radio-buttons, checkboxes, textareas, etc.). Users do not need to manually write statements anymore as they are only required to fill HTML forms.

The difficulties in generating SemanticMediaWiki data have been mitigated by an ad-hoc importer that allows the creation of multiple articles from an input OWL ontology [192]. This tool uses a PHP API for managing OWL and automatically creates wiki content according to the basic MediaWiki model: each concept is mapped into a page and each property into a typed link. Some checks about the consistency of the ontology and the duplication of non-relevant data are also performed.

Other wikis provides users with mixed interfaces for creating semantic data.

³⁹SemanticForms: http://www.mediawiki.org/wiki/Extension:Semantic_Forms.

MaknaWiki [47] is a JSP wiki-clone that allows users to embed semantic statements or to fill HTML forms for querying and adding data to the ontology represented by the wiki. These forms provide general tools for aided navigation of the semantic data, do not depend on the domain of the wiki and their structure is hard-coded in the system.

Rhizome [173] provides friendly interfaces and textareas where users can write statements directly. It relies on ZML (a textual syntax serialisable into XML), a generic language to express semi-structured data, and an engine to apply rules for intermixing semantics and free texts.

A novel solution is provided by *AceWiki* [8] and *CNL-approach* [44]. AceWiki is a semantic wiki that allows users to write ontological statements by simply writing sentences in the ACE (Attempo Controlled English) language. The system includes a predictive authoring tool that suggests options to the users and autocompletes fields consistently to the ontology represented by the wiki. The same editor can be used to extend the ontology by creating new classes, instances and relations. In [44] De Coi et al. proposed a similar approach for SemanticMediaWiki, as they designed a CNL (Controlled Natural Language) interface able to convert sentences written in multiple languages into semantic data.

The second category of semantic wikis, based on the principle of “ontologies for wikis”, includes all those wikis that are actually built on top of ontological foundations. The idea is to exploit ontologies to create and maintain consistent semantic data within a wiki so that sophisticated analysis, queries and classifications can be performed on its content. *IkeWiki* [156] was one of the first wikis to adopt this approach. Its deployment starts by loading an OWL ontology into the system that is automatically translated into a set of wiki pages and typed links. Multiple interfaces are provided to the users for editing the plain wiki content, adding new metadata or tagging pages. IkeWiki strongly relies on Semantic Web technologies: it even includes a Jena OWL⁴⁰ [31] repository and a SPARQL [78] engine used for navigation, queries and display of the semantic content of the wiki.

Similarly, OntOWiki [7] is a complete ontology editor. It relies on a strong distinction between the ontological back-end of the system and a user-friendly interface. Data are natively stored as OWL/RDF statements that are dynamically rendered into the final HTML wiki pages. OntOWiki provides users with multiple views of the same content: (1) ontological data can be navigated by listing classes, individuals, properties, etc., (2) domain-specific views can be added as plugins (for instance, a MapView of geographical data can be dynamically mashed-up from GoogleMaps) and (3) editing-views are natively available in the system.

SweetWiki [30] implements a user-friendly ontology tool designed for both expert and non-expert users. Two aspects characterize the system: the strong connection with the ontologies and the provision of Ajax-based interfaces for editing content and

⁴⁰Jena: <http://jena.sourceforge.net>.

metadata. SweetWiki defines a “Wiki Object Model”, i.e. an ontology describing the wiki structure. Concepts like “document”, “page”, “link”, “version”, “attachment” are all codified in an OWL file that is accessed and manipulated through the wiki itself. These concepts are made explicit in SweetWiki, although they are usually hard-coded in most semantic wikis. SweetWiki also allows users to import external ontologies and to access and manipulate those ontologies through ad-hoc interfaces (similar to those provided by the above mentioned full ontology editors). Finally, the system provides “assisted social tagging” facilities: users can add metadata to any page and can put pages in relation. These metadata values form a folksonomy that, on the one hand, is freely editable by users and, on the other, is built on top of ontological data. The interface for tagging, in fact, suggests consistent metadata by exploiting SPARQL queries and autocompletion features.

Finally, UFOWiki [138] is another project that aims at integrating wikis, ontologies and forms. UFOWiki is a wiki farm, i.e. a server that allows users to set up and deploy semantic wikis. The overall content is stored in a centralized repository as RDF triples that express both the actual content of each page and its metadata. The same farm deploys multiple wikis, so that they can share (ontological) data in a distributed environment.

2.4 Projects, conferences and initiatives about Semantic Publishing

Between 2010 and 2011, a large number of initiatives arose with the precise intention of advertising Semantic Publishing to a broader audience. Everything, from projects to workshops and journal issues, seems to confirm that semantic publishing and scholarly citation using Web standards are presently two of the most interesting topics within the scientific publishing domain.

In this section I list the most important initiatives about Semantic Publishing in 2010/2011, sponsored by both academia and companies. They represent the first formal starting point of the increasing interest that semantic publishing is engendering in scientific and industrial research.

2.4.1 JISC’s Open Citation and Open Bibliography projects

In mid-2010, JISC funded two sister projects: the *Open Citation project*⁴¹ and the *Open Bibliography project*⁴², held respectively by the University of Oxford and the University of Cambridge. Their broad goal is a study on feasibility, advantages and applications at using RDF datasets and OWL ontologies when describing and publishing bibliographic data and citations.

⁴¹Open Citation project blog: <http://opencitations.wordpress.com>.

⁴²Open Bibliography project blog: <http://openbiblio.net>.

The Open Citation project, in which I took part actively, proposes to increase the effectiveness of scientific publishing and scholarly communication, making available on the Web bibliographic information as RDF data, according to particular ontologies developed for the description of the publishing domain. In particular, it aims at creating a semantic infrastructure that describes articles as bibliographic records and their citations to other related works.

The main outcomes of this project are:

- the development of a suite of ontologies, called *Semantic Publishing And Referencing (SPAR)* ontologies, I developed under the supervision of professor David Shotton when I was at the University of Oxford, and that represents an important part of my work (Chapter 4);
- the development of two tools for ontology documentation, i.e. the *Live OWL Documentation Environment (LODE)*, and visualisation, i.e. the *Graphical Framework for OWL Ontologies (Graffoo)*, that I developed to support users when understanding and documenting ontologies. They are introduced in detail in Section 5.1 and Section 5.3 respectively;
- a corpus of interlinked bibliographic records⁴³ obtained converting the whole set of reference lists contained in all the PubMed Central⁴⁴ Open Access articles into RDF data according to SPAR ontologies. The converted RDF data are published as Linked Open Data.

The Open Bibliographic project aimed at publishing a large corpus of bibliographic data as Linked Open Data, starting from four different sources: the Cambridge University Library⁴⁵, the British Library⁴⁶, the International Union of Crystallography⁴⁷ and PubMed⁴⁸. The key strategies promoted by this project were:

- the transformation of publishers' models so as to natively include the open publication of bibliographic data as Linked Open Data;
- the immediate and continuing engagement of the scholarly community.

2.4.2 JISC's Lucero project

The *Lucero project*⁴⁹ is another JISC project, held by the Open University, which aimed at exploring the use of Linked Data within the academic domain. In particular, it proposes solutions that could take advantages from the Linked Data to connect

⁴³It is available online at <http://opencitations.net>.

⁴⁴PubMed Central: <http://www.ncbi.nlm.nih.gov/pmc/>.

⁴⁵Cambridge University Library: <http://www.lib.cam.ac.uk>.

⁴⁶British Library: <http://www.bl.uk>.

⁴⁷International Union of Crystallography: <http://www.iucr.org>.

⁴⁸PubMed: <http://www.ncbi.nlm.nih.gov/pubmed/>.

⁴⁹Lucero project blog: <http://lucero-project.info>.

educational and research content, so as students and researches could benefit from semantic technologies.

Lucero main aims are:

- to promote the publication as Linked Open Data of bibliographic data through a tool to facilitate the creation and use of semantic data;
- to identify a process in order to integrate the Linked Data publication of bibliographic information as part of the University's workflows;
- to demonstrate the benefits derived from exposing and using educational and research data as Open Linked Data, through the development of applications that improve the access to those data.

2.4.3 SePublica and Linked Science

Two of the most important Semantic Web conferences, i.e. the *extended* and the *international* ones, began to explicitly promote Semantic Publishing through two specific workshops.

The workshop *SePublica*⁵⁰, co-located with the 8th Extended Semantic Web Conference, is the first formal event entirely dedicated to Semantic Publishing. The aim of the workshop was to investigate upon the different aspects of using semantic technologies within the publishing industry. In this half-day workshop were presented seven different papers, one of which was awarded as best workshop paper (Elsevier sponsored 750 dollars for that).

The *Linked Science*⁵¹ workshop, co-located with the 10th International Semantic Web Conference, is a full-day event that involved research and practitioners discussing new ways of publishing, sharing, linking and analysing scientific resources, such as articles, datasets and results. Each of the five workshop sessions relates to a particular topic, from data-based applications to semantic integration of data, to end up in an open meeting to discuss about the topics of the workshop.

2.4.4 Beyond Impact, the PDF and Research Communication

Recently the interest in proposing and adopting new formats for the improvement of research communications appeared to be self-evident. More or less all research and industrial communities agreed that current formats are not enough for covering the

⁵⁰The 1st International Workshop about Semantic Publication (SePublica 2011): <http://sepublica.mywikipaper.org>.

⁵¹The 1st International Workshop on Linked Science (LISC2011): <http://data.linkedscience.org/events/lisc2011>.

needs of a Web-based research communication. The workshop *Beyond the PDF*⁵², organised at the University of California San Diego in January 2011, went to that direction. The scope of this event was to identify a set of requirements, applications and deliverables that can be used to accelerate knowledge sharing and discovery.

Beyond the PDF was not the only event organised with the aim of exploring new research possibilities and directions in scholarly publication. The workshop on *The Future of Research Communication*⁵³, held in Dagstuhl in August 2011, was another gathering where scientist and practitioners coming from different disciplines met each other to discuss about future directions in scholarly publishing. They discussed about all the prominent topics of scientific communication. In particular, they proposed new formats, addressed the changes in media and modes of communication, draw opportune infrastructures and outlined social challenges with the aim of making a sign on the future scholarly communication evolution.

From a broader point of view but always considering the Web as prominent medium of communication, the *Beyond Impact Workshop*⁵⁴ (held in London in May 2011) tried to establish different forms of *impact* – that is a measure of how research outcomes influence and are used by other people – in today's and future ways of publishing. The output of this workshop is a document⁵⁵ that introduces research collaborations and future works to be done in the next years.

2.4.5 New Models of Semantic Publishing in Science

Journals about Semantic Web technologies and digital publishing started to be actively interested in Semantic Publishing topics. An example is a special issue of the *Semantic Web Journal*⁵⁶: *New Models of Semantic Publishing in Science*⁵⁷. The central topic spotted by that issue is about the promotion of emergent forms of Web-based publications, so as to allow a rapid and automatic integration of research information, making it readily available and reproducible.

Towards this goal, the issue call identifies Semantic Web technologies as key tool for providing effective opportunities to new modes of scientific publications and asks for submissions of researches in various related fields: Semantic Publishing, Computer Supported Collaborative Work, Linked Data, eScience and Workflow-driven tools, and Digital Libraries. The editors' hope is to promote and advertise all the important researches there were underway in this field.

⁵²The Beyond the PDF Workshop: <http://sites.google.com/site/beyondthepdf/>.

⁵³The Future of Research Communication Dagstuhl Workshop: <http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=11331>.

⁵⁴The Beyond Impact Workshop: <http://beyond-impact.org/>

⁵⁵Beyond Impact Workshop Report: https://docs.google.com/document/d/1sH3JOW5Luki4i37Ve1mOnI2wNZJbaUOx1T42S_7txQ0/edit?hl=en_GB

⁵⁶Semantic Web Journal: <http://www.semantic-web-journal.net>.

⁵⁷New Models of Semantic Publishing in Science: <http://www.semantic-web-journal.net/blog/special-issue-new-models-semantic-publishing-science>.

Chapter 3

Enhancing markup documents

The semantic enhancement of markup documents is a crucial requirement of Semantic Publishing. It is stated in more than one important work in this field, e.g. [45], that a better comprehension of a document derives also from the formal semantics defined within it. The formal semantic layer, thus, becomes a vehicle to extend the amount of ways through which we practice science. Although a large amount of theoretical studies discuss the different kinds of semantics applicable to a text, it seems they miss a crucial point: the semantics of a document, such as a scientific article, must be intrinsically and explicitly tied with the textual content of the document itself.

The way we use to say something about a text is that of *markup*. In people's mind, there still exists a clear distinction between *document markup* (e.g., XML), that we commonly use to define the structure of a document, and the *semantic markup* (e.g., RDF), usually needed when we want to represent, in a particular formalism, the meaning (or, better, a subjective interpretation) of the natural language text of a document¹. Document markup and semantic markup are, actually, two sides of the same coin. However a large amount of people sees them living in two separated levels. Even if the document markup is used to structure a document in the most cases, it is not denied to having some markup elements that characterise textual fragments as real-world entity, e.g. a person.

Within a text, the element <person>² and the class *foaf:Person*³ are (habitually) used to convey the same meaning, at least from the markup author's perspective. What really differs between them is that the latter define a formal representation of their semantics (e.g., in OWL 2 Direct Semantics [126]), while the former usually does not. As all the XML-like languages, the markup semantics is left to the human

¹The document structure on its own can be seen as a particular kind of semantics. In fact, when we speak about a text as structured in terms of its paragraphs, sections, chapters, etc., what we are doing is to associate a semantic role of particular parts of the text.

²The element *person* as defined in TEI [186]: <http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-person.html>.

³The class *Person* as defined in FOAF [25]: http://xmlns.com/foaf/spec/#term_Person.

interpretation of a natural language definition or, in the worst case, of the markup on its own.

Besides this difference, what XML-like languages really miss is an appropriate expressiveness for the description of multiple and overlapping markup on the same text. Overlapping markup is a crucial feature to express multiple (even discording) semantic interpretations on the same fragment. If we agree that XML markup elements convey semantics, then our field of action is limited by the syntactic rules of XML itself that impose to structure the markup as a tree. Previous works, such as TEI [186] and RDFa [2], propose to go beyond the fixed syntactic limits of XML, in order to have multiple overlapping markup elements and RDF statements within the same text. The problem is that those languages are not enough to address all the possible scenarios. For example, on the one hand, XML documents with TEI-like overlap workarounds present problems when trying to validate them against a particular schema. On the other hand, RDFa uses the document content to define formal statements, but it does not allow one to link a particular piece of text, e.g. a sentence, to a semantic formalisation of it.

Trying to address the aforementioned issues, in this chapter I propose a new markup metalanguage called *EARMARK* (*Extremely Annotational RDF Markup*) defined by means of Semantic Web technologies. Among the lines of other representational frameworks for documents such as [190], EARMARK is a logic-based model that tries to be a nexus between document markup and semantic markup, and aims to reach an intuitively complete equivalence between them. Besides presenting EARMARK, in the following sections I investigate upon issues strictly related to markup expressiveness, i.e., handling multiple and overlapping hierarchies on the same fragment, the validation (against a particular schema) of documents having overlapping markup, and the definition of the semantics of markup elements and textual content.

3.1 EARMARK, a Semantic Web approach to meta-markup

This section discusses a different approach to metamarkup, called EARMARK (Extremely Annotational RDF Markup) [55] [142] [56] [57] [58] [140] [11] [59] based on ontologies and Semantic Web technologies. The basic idea is to model EARMARK documents as collections of addressable text fragments, and to associate such text content with OWL assertions that describe structural features as well as semantic properties of (parts of) that content. As a result EARMARK allows not only documents with single hierarchies (as with XML) but also multiple overlapping hierarchies where the textual content within the markup items belongs to some hierarchies but not to others. Moreover EAMARK makes it possible to add semantic annotations to the content though assertions that may overlap with existing ones.

One of the advantages of using EARMARK is the capability to access and query documents by using well-known and widely supported tools for Semantic Web. EARMARK assertions are simply RDF assertions, while EARMARK documents are modelled through OWL ontologies. The consequence is that query languages (such as SPARQL 1.1 [78]) and actual existing tools (such as *Jena* [31] and *Pellet*⁴ [170]) can be directly used to deal with even incredibly complicated overlapping structures. What is very difficult (or impossible) to do with traditional XML technologies becomes much easier with these technologies under the EARMARK approach.

In the rest of this section I give a brief overview of the EARMARK model and how it can be used for addressing issues of overlapping markup, validation and semantics. The model itself is defined through an OWL document⁵, summarized in Fig. 3.1⁶, specifying classes and relationships. We distinguish between *ghost classes* – that define the general model – and *shell classes* - that are actually used to create EARMARK instances.

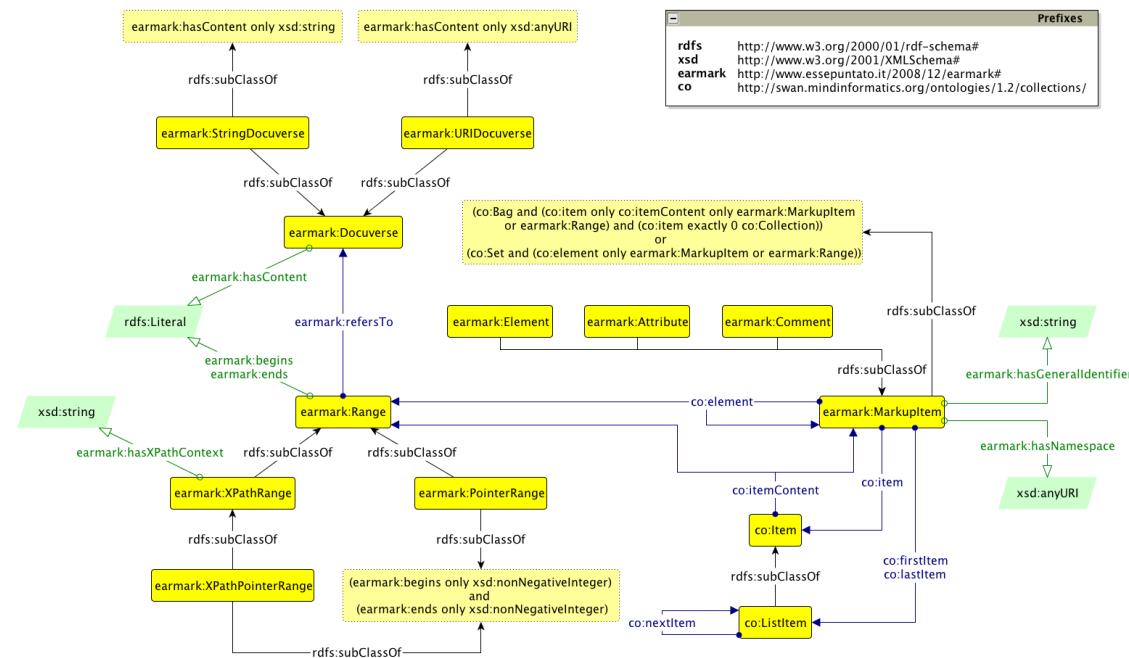


Figure 3.1: A Graffoo (see Section 5.3) representation of the EARMARK ontology.

⁴Pellet: <http://pellet.owlowl.com>.

⁵EARMARK ontology: <http://www.essepuntato.it/2008/12/earmark>.

⁶This and the following diagrams comply with the *Graphic framework for OWL ontologies (Graffoo)*, introduced in Section 5.3. A legend for all Graffoo diagrams can be found in Fig. 5.13 on page 177.

3.1.1 Ghost classes

The ghost classes describe three disjoint base concepts – *docuverses*, *ranges* and *markup items* – through three different and disjoint OWL classes⁷.

The textual content of an EARMARK document is conceptually separated from its annotations, and is referred to through the *Docuverse* class⁸. The individuals of this class represent the object of discourse, i.e. all the containers of text of an EARMARK document.

```
Class: earmark:Docuverse

DatatypeProperty: earmark:hasContent
  Characteristics: FunctionalProperty
  Domain: earmark:Docuverse
  Range: rdfs:Literal
```

Any individual of the *Docuverse* class – commonly called a *docuverse* (lowercase to distinguish it from the class) – specifies its actual content with the property *hasContent*.

We then define the class *Range* for any text lying between two locations of a docuverse. A *range*, i.e., an individual of the class *Range*, is defined by a starting and an ending location (any literal) of a specific docuverse through the properties *begins*, *ends* and *refersTo* respectively.

```
Class: earmark:Range
  EquivalentTo:
    earmark:refersTo some earmark:Docuverse and
    earmark:begins some rdfs:Literal and
    earmark:ends some rdfs:Literal

ObjectProperty: earmark:refersTo
  Characteristics: FunctionalProperty
  Domain: earmark:Range
  Range: earmark:Docuverse

DatatypeProperty: earmark:begins
  Characteristics: FunctionalProperty
```

⁷All our OWL samples are presented using the Manchester Syntax [95] and Turtle [147], which are two of the standard linearisation syntaxes of OWL. The prefixes *rdfs* and *xsd* refer respectively to RDF Schema and XML Schema namespaces, while the prefix *earmark* refers to the EARMARK ontology URI plus “#”. Moreover, we use the prefix *co* to indicate entities taken from an imported ontology made for the SWAN project [35], available at <http://swan.mindinformatics.org/spec/1.2/collections.html>.

⁸This class (and its name) is based on the concept introduced by Ted Nelson in his Xanadu Project [132] to refer to the collection of text fragments that can be interconnected to each other and transcluded into new documents.

```

Domain: earmark:Range
Range: rdfs:Literal

DatatypeProperty: earmark:ends
Characteristics: FunctionalProperty
Domain: earmark:Range
Range: rdfs:Literal

```

There is no restriction on locations used for the *begins* and *ends* properties. That is very useful: it allows us to define ranges that *follow* or *reverse* the text order of the docuverse they refer to. For instance, the string “desserts” can be considered both in document order, with the *begins* location lower than the *ends* location or in the opposite one, forming “stressed”⁹. Thus, the values of properties *begins* and *ends* define the way a range must be read.

The class *MarkupItem* is the superclass defining artefacts to be interpreted as markup (such as elements and attributes).

```

Class: earmark:MarkupItem
SubClassOf:
  (co:Set that co:element only
    (earmark:Range or earmark:MarkupItem)) or
  (co:Bag that co:item only
    (co:itemContent only
      (earmark:Range or earmark:MarkupItem)))

DatatypeProperty: earmark:hasGeneralIdentifier
Characteristics: FunctionalProperty
Domain: earmark:MarkupItem
Range: xsd:string

DatatypeProperty: earmark:hasNamespace
Characteristics: FunctionalProperty
Domain: earmark:MarkupItem
Range: xsd:anyURI

Class: co:Collection

Class: co:Set
SubClassOf: co:Collection

Class: co:Bag
SubClassOf: co:Collection

Class: co>List

```

⁹<http://en.wikipedia.org/wiki/Palindrome#Semordnilaps>

```

SubClassOf: co:Bag

Class: co:Item

Class: co:ListIItem
  SubClassOf: co:Item

ObjectProperty: co:element
  Domain: co:Collection

ObjectProperty: co:item
  SubPropertyOf: co:element
  Domain: co:Bag
  Range: co:Item

ObjectProperty: co:firstItem
  Characteristics: FunctionalProperty
  SubPropertyOf: co:item
  Domain: co>List

ObjectProperty: co:lastItem
  Characteristics: FunctionalProperty
  SubPropertyOf: co:item
  Domain: co>List

ObjectProperty: co:itemContent
  Characteristics: FunctionalProperty
  Domain: co:Item
  Range: not co:Item

ObjectProperty: co:nextItem
  Characteristics: FunctionalProperty
  Domain: co:ListIItem
  Range: co:ListIItem

```

A *markupitem* individual is a collection (co:Set, co:Bag or co>List, where the latter is a subclass of the second one and all of them are subclasses of co:Collection) of individuals belonging to the classes *MarkupItem* and *Range*. Through these collections it is possible to define a markup item as a set, a bag or a list of other markup items, using the properties *element* (for sets), *item* and *itemContent* (for bags and lists). Thus it becomes possible to define elements containing nested elements or text, or attributes containing values, as well as overlapping and complex structures. Note also that handling collections directly in OWL allows us to reason about content models for markup items, which would not be possible if we had used

the corresponding constructs in RDF¹⁰.

A *markupitem* might also have a name, specified in the functional property *hasGeneralIdentifier* (recalling the SGML term to refer to the name of elements [81]), and a namespace specified using the functional property *hasNamespace*. Note that we can have *anonymous* markup items – as it is possible in LMNL [185] and GODDAG [176] – by simply asserting that the item belongs to the class of all those markupitems that do not have a general identifier (i.e., *earmark:hasGeneralIdentifier exactly 0*).

3.1.2 Shell classes

The ghost classes discussed so far give us an abstract picture of the EARMARK framework. We need to specialize our model, defining a concrete description of our classes. These new *shell* subclasses apply specific restrictions to the *ghost* classes.

First of all, the class *Docuverse* is restricted to be either a *StringDocuverse* (the content is specified by a string) or an *URIDocuverse* (the actual content is located at the URI specified).

```
Class: earmark:StringDocuverse
DisjointWith: earmark:URIDocuverse
SubClassOf:
  earmark:Docuverse ,
  earmark:hasContent some xsd:string

Class: earmark:URIDocuverse
SubClassOf:
  earmark:Docuverse ,
  earmark:hasContent some xsd:anyURI
```

Depending on particular scenarios or on the kind of docuverse we are dealing with – it may be plain-text, XML, LaTeX, a picture, etc. – we need to use different kinds of ranges. Therefore, the class *Range* has three different subclasses:

- *PointerRange* defines a range by counting characters. In that case, the value of the properties *begins* and *ends* must be a non-negative integer that identifies unambiguous positions in the character stream, remembering that the value *0* refers to the location immediately before the 1st character, the value *1* refers to the location after the 1st character and before the 2nd one, and so on. By using the *hasKey* OWL property, we also assert that two pointer ranges having equal docuverse, begin and end locations are the same range;

¹⁰A blog post by Paolo Ciccarese explaining why RDF collections cannot be used in OWL contexts: <http://hcklab.blogspot.com/2008/12/moving-towards-swan-collections.html>.

- *XPathRange* defines a range considering the whole docuverse or its particular context specifiable through an XPath expression [15] as value of the property *hasXPathContext*. Note that, by using these ranges, we implicitly admit that the docuverse it refers to must be an XML structure. Moreover, the properties *begins* and *ends* have to be applied on the string value obtained by juxtaposing all the text nodes identified by the XPath. By using the *hasKey* OWL property, we also assert that two xpath ranges having equal docuverse, XPath context, begin and end locations are the same range;
- *XPathPointerRange* is an *XPathRange* in which the value of the properties *begins* and *ends* must be a non-negative integer that identifies unambiguous positions in the character stream as described for the class *PointerRange*.

```

Class: earmark:PointerRange
HasKey: earmark:refersTo earmark:begins earmark:ends
SubClassOf:
  earmark:Range ,
  earmark:begins some xsd:nonNegativeInteger and
  earmark:ends some xsd:nonNegativeInteger

Class: earmark:XPathRange
SubClassOf: earmark:Range
EquivalentTo:
  earmark:hasXPathContext some rdfs:Literal
HasKey:
  earmark:refersTo earmark:begins
  earmark:ends earmark:hasXPathContext

Class: earmark:XPathPointerRange
SubClassOf:
  earmark:XPathRange ,
  earmark:begins some xsd:nonNegativeInteger and
  earmark:ends some xsd:nonNegativeInteger

DatatypeProperty: earmark:hasXPathContext
  Characteristics: FunctionalProperty
  Domain: earmark:XPathRange
  Range: rdfs:Literal

```

MarkupItem is specialized in three disjointed sub-classes: *Element*, *Attribute* and *Comment*, which allow a more precise characterization of markup items.

```

Class: earmark:Element
SubClassOf: earmark:MarkupItem

```

```

Class: earmark:Attribute
  SubClassOf: earmark:MarkupItem

Class: earmark:Comment
  SubClassOf: earmark:MarkupItem

DisjointedClasses: earmark:Element , earmark:Attribute ,
  earmark:Comment

```

3.1.3 An example and an API

In order to understand how EARMARK is used to describe markup hierarchies, let me introduce an XML excerpt, using TEI fragmentation [186] to express overlapping elements upon the string “Fabio says that overlhappens”:

```

<p>
  <agent>Fabio</agent> says that
  <noun xml:id="e1" next="e2">overl</noun>
  <verb>
    h<noun xml:id="e2">ap</noun>pens
  </verb>
</p>

```

Here, the two elements *noun* represent the same element fragmented and overlapping with part of the textual content of *verb*, i.e., the characters “ap”. The EARMARK translation of it is the following (linearised in Turtle [147]):

```

@prefix earmark: <http://www.essepuntato.it/2008/12/earmark
#>.
@prefix co: <http://swan.mindinformatics.org/ontologies/1.2/
collections/>.
@prefix ex: <http://www.example.com/>.

ex:doc earmark:hasContent "Fabio says that overlhappens" .

ex:r0-5 a earmark:PointerRange ; earmark:refersTo ex:doc
; earmark:begins "0"^^xsd:nonNegativeInteger
; earmark:ends "5"^^xsd:nonNegativeInteger .

ex:r5-16 a earmark:PointerRange ; earmark:refersTo ex:doc
; earmark:begins "5"^^xsd:nonNegativeInteger
; earmark:ends "16"^^xsd:nonNegativeInteger .

ex:r16-21 a earmark:PointerRange ; earmark:refersTo ex:doc
; earmark:begins "16"^^xsd:nonNegativeInteger
; earmark:ends "21"^^xsd:nonNegativeInteger .

```

```

ex:r21-28 a earmark:PointerRange ; earmark:refersTo ex:doc
; earmark:begins "21"^^xsd:nonNegativeInteger
; earmark:ends "28"^^xsd:nonNegativeInteger .

ex:r22-24 a earmark:PointerRange ; earmark:refersTo ex:doc
; earmark:begins "22"^^xsd:nonNegativeInteger
; earmark:ends "24"^^xsd:nonNegativeInteger .

ex:p a earmark:Element ; earmark:hasGeneralIdentifier "p"
; co:firstItem [ co:itemContent ex:agent
; co:nextItem [ co:itemContent ex:r5-16
; co:nextItem [ co:itemContent ex:noun
; co:nextItem [ co:itemContent ex:verb ]]]] .

ex:agent a earmark:Element
; earmark:hasGeneralIdentifier "agent"
; co:firstItem [ co:itemContent ex:r0-5 ] .

ex:noun a earmark:Element
; earmark:hasGeneralIdentifier "noun"
; co:firstItem [ co:itemContent ex:r16-21
; co:nextItem [ co:itemContent ex:r22-24 ]] .

ex:verb a earmark:Element
; earmark:hasGeneralIdentifier "verb"
; co:firstItem [ co:itemContent ex:r21-28 ] .

```

I designed and implemented a framework for the creation, validation and manipulation of EARMARK documents, such as the above one. The API is hosted by SourceForge¹¹ under the Apache 2.0 license and fully implements the current EARMARK model.

All the code is written in JavaTM and uses well-known libraries in the Semantic Web community such as Jena [31]. The implementation of the EARMARK data structure follows exactly what is defined in the EARMARK ontology and uses ghost and shell classes, encoding OWL properties as methods of these classes.

The JavaTM classes *EARMARKDocument*, *Range* and *MarkupItem* have been written taking into consideration a particular interface, called *EARMARKNode*, directly derived from the *Node* interface of the JavaTM DOM implementation¹². This choice has been made to maintain the EARMARK data structure as close as possible to a well-known and used model for XML documents.

¹¹<http://earmark.sourceforge.net>

¹²<http://java.sun.com/xml>

The API makes it simple to create/load/store/modify EARMARK documents directly in a Java™ framework. Let me take again into consideration the simple example previously introduced. In the followings excerpts I illustrate how to build that document using the API.

Let me start creating a new EARMARK document and a docuverse, containing all the text content of our document:

```
EARMARKDocument ed =
    new EARMARKDocument(URI.create("http://www.example.com"));

String ex = "http://www.example.com/";

Docuverse doc = ed.createStringDocuverse(ex+"doc",
    "Fabio says that overlhappens");
```

The next excerpt shows how to create ranges starting from the above docuverse:

```
Range r0_5 = ed.createPointerRange(ex+"r0_5", doc, 0, 5);
...
```

Finally, I define all the markup items we need to build the structure of the document. Usually, a markup item needs three different values in order to be created: a string representing its general identifier, an identifier for the item and the type for the collection it defines (either set, bag or list). In the following excerpt we show the creation of three different elements, composed in order to define hierarchical relations among them by using the method *appendChild*:

```
Element p = ed.createElement(
    "p", ex+"p", Collection.Type.List);
Element agent = ed.createElement(
    "agent", ex+"agent", Collection.Type.List);

ed.appendChild(p); p.appendChild(agent);
agent.appendChild(r0_5);
...
```

As seen, the API allows one to create EARMARK documents with very simple and straightforward methods. Even rather complicated structures can be created with a few lines of Java code.

3.2 The issue of overlapping markup

There are multiple applications of EARMARK. The most interesting for this section concerns its capability of dealing with overlapping structures in an elegant and straightforward manner. Under EARMARK such structures do not need to be specified through complex workarounds as with XML, but they are explicit and can

be easily described and accessed. Sophisticated searches and content manipulations become very simple when using this ontological model.

Thus, the goal of this section is to demonstrate the soundness and applicability of EARMARK by introducing theoretical aspects about overlapping markup in Section 3.2.1 and Section 3.2.2, and by discussing how some real-case scenarios are addressed (Section 3.2.3, Section 3.2.4 and Section 3.2.5). Notice that throughout the following sections I investigate multiple EARMARK data structures and documents, focussing on the feasibility and potentiality of such an ontological representation.

3.2.1 Range and markup item overlap

The presence of overlap in EARMK is worth discussing more in detail. Different types of overlap exist – according to the subset of items involved – and different strategies are needed to detect them. In particular, there is a clear distinction between *overlapping ranges* and *overlapping markup items*.

By definition, *overlapping ranges* are two ranges that refer to the same docuverse and so that at least one of the locations of the first range is contained in the interval described by the locations of the second range (excluding its terminal points). *Totally overlapping ranges* have the locations of the first range completely contained in the interval of the second range or vice versa, while *partially overlapping ranges* have either exactly one location inside the interval and the other outside or identical terminal points in reversed roles.

Thus, if we consider the ranges *ex:r21-28* and *ex:r24-24* of the example in Section 3.1.3, we can infer, through a reasoner such as Pellet [170], that these two ranges overlap by using the following rule¹³ (expressed in SWRL-like syntax [96]):

```
earmark:begins(x,b1) , earmark:ends(x,e1) ,
earmark:begins(y,b2) , earmark:ends(y,e2) ,
earmark:refersTo(x,d) , earmark:refersTo(y,d) ,
DifferentFrom(x,y) , P
-> overlapping:overlapWith(x,y)
```

where *P* is one of:

- lessThan(b1,e1) , greaterThan(b2,b1) , lessThan(b2,e1)
- lessThan(b1,e1) , greaterThan(e2,b1) , lessThan(e2,e1)
- lessThan(e1,b1) , greaterThan(b2,e1) , lessThan(b2,b1)
- lessThan(e1,b1) , greaterThan(e2,e1) , lessThan(e2,b1)

¹³In the excerpt, the prefix *overlapping* refers to “<http://www.essepuntato.it/2011/05/overlapping/>”.

The case of *overlapping markup items* is slightly more complicated. We define that two markup items A and B **overlap** when at least one of the following sentences holds:

1. [overlap by range] A contains a range that overlaps with another range contained by B ;
2. [overlap by content hierarchy] A and B contain at least a range in common;
3. [overlap by markup hierarchy] A and B contain at least a markup item in common.

The three possible scenarios for such item overlap are summarized in Fig. 3.2¹⁴.

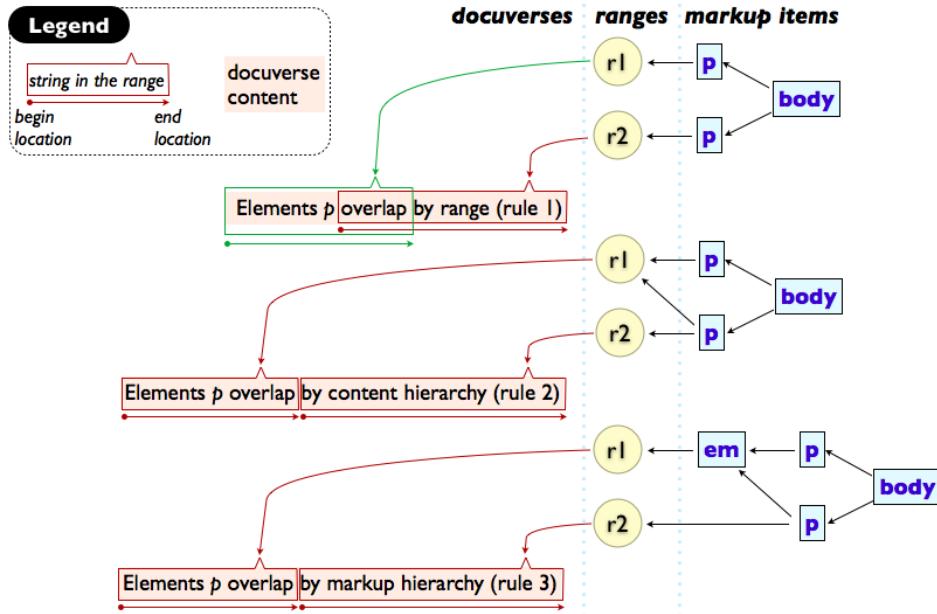


Figure 3.2: Three EARMARK examples of overlapping between elements p .

The EARMARK ontology, in fact, is completed by another ontology¹⁵ that models all overlapping scenarios, either for ranges or markup items, and includes rules for inferring overlaps automatically, through a reasoner.

¹⁴The EARMARK documents describing these three overlapping scenarios and all the other ones presented in the following sections are available at <http://www.essepuntato.it/2011/jasist/examples>.

¹⁵The EARMARK Overlapping Ontology: <http://www.essepuntato.it/2011/05/overlapping>.

3.2.2 EARMARK as a standoff notation

If we ignore for a moment the semantic implications of using EARMARK and concentrate only on its syntactical aspects, it is easy to observe that EARMARK is nothing but yet another standoff notation, where the markup specifications point to, rather than contain, the relevant substructure and text fragments.

Standoff notations, also known in literature as out-of-line notations [186], are hardly new, but never really caught on for a number of reasons, most having to do with their perceived fragility under the circumstances of desynchronized modification to the text. In [80] and [14] we can find a pair of recent and substantially complete analysis of their merits and demerits. In particular, according to [80], “standoff annotation has [...] quite a few disadvantages:

1. very difficult to read for humans
2. the information, although included, is difficult to access using generic methods
3. limited software support as standard parsing or editing software cannot be employed
4. standard document grammars can only be used for the level which contains both markup and textual data
5. new layers require a separate interpretation
6. layers, although separate, often depend on each other”¹⁶.

And yet, although EARMARK *is* in practice a standoff notation, it provides a number of workarounds to most of the above-mentioned issues.

Firstly, since EARMARK is based on OWL and can be linearised in any of the large number of OWL linearisation syntaxes, it follows that 1) readability, 2) access and 3) software support for it are exactly those existing for well-known, widespread and important W3C standards such as RDF and OWL. Being able to employ common RDF and OWL tools such as Jena and SPARQL for EARMARK documents was in fact a major motivation for it.

Issue 4 should be examined beyond the mere validation against document grammars and towards a general evaluation of the level of compliancy of the markup to some formally specified expectations. EARMARK documents, while being subject to no document grammar in the stricter XML sense, allow the specification of any number of constraints, expressed either directly in OWL, or in SWRL [96] or even in SPARQL [78], that trigger or generate validity evaluations. In [58] we tried to show that a large number of requirements, from hierarchical well-formedness in the XML

¹⁶In order to individually address the issues, we edited the original bullets into a numbered list.

sense, to validation requirements in terms of XML DTDs, to adherence to design patterns, can be expressed satisfactorily using these technologies.

Item 5 regards the difficulty of standoff notations to provide inter-layer analysis on XML structures: separate interpretation of markup layers is easy, but identification and validation of overlapping situations is more complex: standoff markup is mainly composed of pointers to content, and does not have any direct way to determine overlap locations without some kind of pointer arithmetics to compute them. Validation of contexts allowing overlaps as describable using rabbit-duck grammars [175] is also not trivial. In this regard EARMARK provides yet again a solution that does not require special tools: although OWL does not allow direct pointer arithmetics, SWRL on the contrary does, as shown in Section 3.2.1 where we described a batch of (SWRL-implementable) rules that do in fact determine overlapping locations on EARMARK documents with good efficiency.

Finally, issue 6 refers to the fact that evolution of separate markup annotation layers need to take place synchronously, lest one of them become misaligned with the new state of the document. This is, in summary, the *fragility of pointers*, which can be considered the fundamental weakness of standoff, as well as of any notation that has markup separate from its content: if a modification occurs to the underlying (probably text-based) source, all standoff pointers that could not be updated at the same time of the change become outdated and possibly wrong. All standoff notations fall prey of this weakness, and there is no way to completely get rid of it.

What is possible is to identify exactly what are the conditions under which such weakness acts, and see if there is a way to reduce the mere frequency of such events. In order for a standoff pointer to become outdated, several conditions must take place at the same time:

- the standoff notation must be used as a storage format, rather than just as a processing format;
- the source document must make sense even without the additional standoff markup (i.e., the standoff notation contains no information that is necessary for at least some types of document modifications);
- the source document must be editable (and, in fact, must be edited) on its own;
- the standoff pointers must rely on positions that change when the source is edited (e.g., character-based locations);
- editing must be done in contexts and with tools that cannot or do not update the standoff pointers;
- there must be no computable way to determine the modifications of the document (e.g. via a *diff* between the old and the new version).

Of course, no standoff notation can rule out that these conditions occur on their documents. But it is worth pointing out that **all six** of them must occur, for standoff pointers to become outdated. EARMARK is not safe from these occurrences either, but, at least for some use cases, one or more of these conditions simply do not apply. For instance, when EARMARK is used as a processing format, with no need to save it on disk (conversion from the source formats, e.g. MS Word, is described in Section 3.2.3 and does not require special storage), the data format described is either in a very specific format (such as MS Word or ODT) that in fact already does handle internally its data changes and requires the overlapping data exactly for this purpose, or is in fact the result of a diff action on successive versions of a document (as in the case of the wiki pages, as introduced in Section 3.2.5).

Finally, EARMARK allows references to relatively stable fragment ids of the documents (by using XPath ranges without specifying explicitly begin and end locations), rather than the extremely fragile character locations, further reducing the chances of outdated pointers.

For this reason, without being able to completely rule out the possibility of standoff pointers to go wrong, we tend to consider it as a significantly little risk, at least for the use case here described.

3.2.3 Looking for authorial changes in Office Documents

Word processors such as Microsoft Word and Open Office Writer provide users with powerful tools for tracking changes, allowing each individual modification by individual authors to be identified, highlighted, and acted upon (e.g. by accepting or discarding them). The intuitiveness of the relevant interfaces actually hides the complexity of the data format and of the algorithms necessary to handle such information.

For instance, the standard ODT format [104] used by Open Office, when saving change tracking information, relies on two specific constructs for insertions and deletions that may overlap with the structural markup. While adding a few words within a paragraph is not in itself complex, as it does not require the breaking of the fundamental structural hierarchy, conversely changes that affect the structure itself (e.g. the split of one paragraph in two by the insertion of a return character, or vice versa the joining of two paragraphs by the elimination of the intermediate return character) require that annotations are associated to the end of a paragraph and the beginning of the next, in an unavoidably overlapping pattern. ODT uses milestones and standoff markup for insertions and deletions respectively, and also relies on standoff markup for annotations about the authorship and date of the change.

For instance, the insertion of a return character and a few characters in a paragraph creates a structure as follows:

```

<text:tracked-changes>
  <text:changed-region text:id="S1">
    <text:insertion>
      <office:change-info>
        <dc:creator>John Smith</dc:creator>
        <dc:date>2009-10-27T18:45:00</dc:date>
      </office:change-info>
    </text:insertion>
  </text:changed-region>
  [... other changes ...]
</text:tracked-changes>
[... content ...]
<text:p>The beginning and
  <text:change-start text:change-id="S1"/></text:p>
<text:p> also<text:change-end text:change-id="S1"/> the end
  .</text:p>

```

The empty elements `<text:change-start>` and `<text:change-end>` are *milestones* marking respectively the beginning and the end of the range that constituted the insertion, while the element `<text:insertion>`, before the beginning of the document content, is *standoff markup* for the metadata about the change (author and date information).

Similarly, a deletion creates a structure as follows:

```

<text:tracked-changes><text:changed-region text:id="S2">
  <text:deletion><office:change-info>
    <dc:creator>John Smith</dc:creator>
    <dc:date>2009-10-27T18:46:00</dc:date>
  </office:change-info><text:p/><text:p/></text:deletion>
  </text:changed-region>
  [... other changes ...]
</text:tracked-changes>
[... content ...]
<text:p>The beginning and
  <text:change text:change-id="S2" />also the end.</text:p>

```

The element `<text:change>` represents a *milestone* of the location where the deletion took place in the content, and the corresponding *standoff markup* annotation `<text:deletion>` contains not only the metadata about the change, but also the text that was deleted.

The OOXML format [103] (the XML-based format used by Microsoft Office 2007), on the other hand, uses a form of *segmentation* to store change-tracking information across all previous elements involved.

```

<w:p>
  <w:pPr><w:rPr>

```

```

<w:ins w:id="0" w:author="John Smith"
        w:date="2009-10-27T18:50:00Z"/>
</w:rPr></w:pPr>
<w:r><w:t>The beginning and </w:t></w:r></w:p>
<w:p>
  <w:ins w:id="1" w:author="John Smith"
         w:date="2009-10-27T18:50:00Z">
    <w:r><w:t>also </w:t></w:r></w:ins>
  <w:r><w:t>the end.</w:t></w:r></w:p>

```

This heavily simplified version of an OOXML document shows two separate changes: the first is the insertion of a return character and the second is the insertion of a word. These modifications are not considered as a single change, and therefore the segments are not connected to each other, but simply created as needed to fit the underlying structure. In fact, change tracking in OOXML is a fairly complex proposition. Although providing more complete coverage of special cases and situations than ODT, dealing with its intricacies is not for the casual programmer.

EARMARK for processing office documents

At this point, it is clear that the use of complex data structures in ODT and OOXML, needed with storing overlaps generated by change-tracking functionalities, make it very difficult to search and manipulate the content when using XML languages and tools. Even very simple edits generate a rather tangled set of overlapping elements.

EARMARK, on the other hand, stores overlapping data in a direct and streamlined manner that does not require tools to rebuild information from the twists of a tree-based XML structure. The information is already available and expressed through consistent RDF and OWL statements. Fig. 3.3 on the facing page graphically shows the corresponding EARMARK document.

The original paragraph content and the new string “also” are now encoded as two *docuverses* over which the ranges r_1 , r_2 and r_3 are defined. The original paragraph is then composed of the (content of) ranges r_1 and r_2 , while the paragraphs resulting after the (text and carriage return) insertion now comprise respectively range r_1 and ranges r_2 , r_3 . Metadata about the author and the modification date are encoded as further RDF statements.

```
Individual: doc1 Types: earmark:StringDocuverse
  Facts: earmark:hasContent "The beginning and the end"
```

```
Individual: doc2 Types: earmark:StringDocuverse
  Facts: earmark:hasContent " also"
```

```
Individual: r1 Types: earmark:PointerRange
```

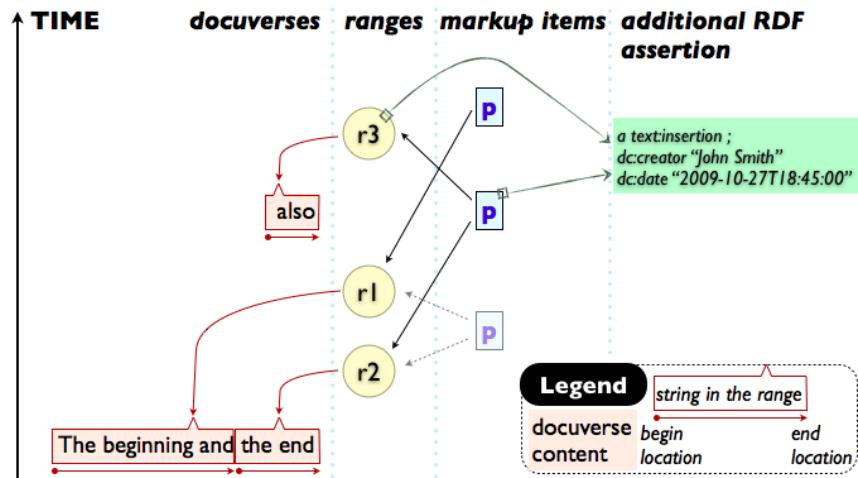


Figure 3.3: Encoding in EARMARK the ODT change-tracking example.

Facts :

```
earmark:refersTo doc1 ,
earmark:begin "0"^^xsd:nonNegativeInteger ,
earmark:end "17"^^xsd:nonNegativeInteger
```

Individual: r2 Types: earmark:PointerRange

Facts :

```
earmark:refersTo doc1 ,
earmark:begin "17"^^xsd:nonNegativeInteger ,
earmark:end "25"^^xsd:nonNegativeInteger
```

Individual: r3 Types: earmark:PointerRange , insJS

Facts :

```
earmark:refersTo doc2 ,
earmark:begin "0"^^xsd:nonNegativeInteger ,
earmark:end "5"^^xsd:nonNegativeInteger
```

Individual: p-b Types: earmark:Element

Facts: co:firstItem p-b-i1

Individual: p-b-i1

Facts: co:itemContent r1 , co:nextItem p-b-i2

Individual:

p-b-i2 Facts: co:itemContent r2

Individual: p-m Types: earmark:Element , insJS

Facts: co:firstItem p-m-i1

```

Individual: p-m-i1
  Facts: co:itemContent r3 , co:nextItem p-m-i2

Individual: p-m-i2
  Facts: co:itemContent r2

Individual: insJS Types: Insertion
  Facts:
    dc:creator "John Smith" ,
    dc:date "2009-10-27T18:45:00"

Individual: p-t Types: earmark:Element
  Facts: co:firstItem p-t-i

Individual: p-t-i
  Facts: co:itemContent r1

```

The advantages of streamlining overlaps become apparent if we consider tasks a little beyond the mere display. For instance, the query for “the textual content of all paragraphs inserted by John Smith” ends up rather entangled if we used XPath [15] on the ODT structure. The process for finding that textual content needs to browse the current version of the document, look for all the *text:change-start/text:change-end* pairs that refer to an insertion made by John Smith involving the creation of a new paragraph (i.e., *text:change-start* is in a first paragraph while its pair, *text:change-end*, is in the following one), that are either currently present in the document body or hidden behind a subsequent deletion made by someone else. Once identified the paragraphs, I need to retrieve the content that originally was contained there, i.e., the text fragments that still are within those boundaries or that may have been deleted in subsequent versions. The following XPath represent an implementation of the above process:

```

for $cr in (//text:changed-region) , $date in ($cr/text:
  insertion//(@office:chg-date-time | dc:date)) return $cr
  [./text:insertion[(./@office:chg-author = 'John Smith'
  and count($cr//text:p) = 2) or (.//dc:creator = 'John
  Smith' and (//text:change-start[@text:change-id = $cr/
  @text:id]/following::text:p intersect //text:change-end[
  @text:change-id = $cr/@text:id]/(ancestor::text:p)))]]
  /root()//((text:change-start[@text:change-id = $cr/@text:id]
  ]/(following::text:p//((text()|(for $tc in (text:change)
  return //text:changed-region[@text:id = $tc/@text:change-
  id and not(text:insertion//(@office:chg-date-time | dc:
  date) > $date)]//text:p[1]//text())))
  except ((for $tc in (text:change) return $tc[count(//text:changed-region[@text:

```

```

id = $tc/@text:change-id and not(text:insertion//(@office:
chg-date-time | dc:date) > $date)]//text:p) = 2]/following
::text()) union (//text:changed-region/text:deletion[./dc
:date <= $date]/text:p//text())))) | (text:change[@text:
change-id = $cr/@text:id]//(following::text()[ancestor::
text:p] | (for $tc in (following::text:change) return //
text:changed-region[@text:id = $tc/@text:change-id and not
(text:insertion//(@office:chg-date-time | dc:date) > $date
)]//text:p[1]]//text())))
except ((for $tc in (following::
text:change) return $tc[count(//text:changed-region[@text:
id = $tc/@text:change-id and not(text:insertion//(@office:
chg-date-time | dc:date) > $date)]//text:p) = 2]/following
::text()) union (//text:changed-region/text:deletion[./dc
:date <= $date]/text:p//text())))) | $cr//text:p[2]]//text()
) except (//(text:change|text:change-end)[@text:change-id
= $cr/@text:id]//following::text:p[not((text:change-end|
text:change-start|text:change)[1]/self::text:change-end)
]/(. | following-sibling::element())//(text() | (for $tc
in (text:change) return //text:changed-region[@text:id =
$tc/@text:change-id]]//text())))
)

```

The XML structure of a MS Word file, using segmentation rather than milestones, does simplify a bit the query, but still presents some radical complexities. The process starts by choosing all those *w:p* elements that were inserted by John Smith, as well as all their previous and contiguous *w:p* elements that were deleted before or inserted after the first ones. In OOXML, each sequence of contiguous *w:p* elements represents implicitly one paragraph. Therefore, I can now take all the text fragments contained in each *w:p* sequence that were inserted before or deleted after the paragraph defined by the sequence itself. The following is the resulting XPath for an OOXML document:

```

for $p in (//w:p[w:pPr//w:ins/@w:author = 'John Smith'])
return for $date in ($p/w:pPr//w:ins/@w:date) return ($p|(
$p/preceding-sibling::w:p[w:pPr//w:del[@w:date <= $date]|w
:pPr//w:ins[@w:date > $date]] except $p/preceding-sibling
::element()[not(self::w:p) or empty(w:pPr//w:del[@w:date
<= $date]|w:pPr//w:ins[@w:date > $date])]/(.|preceding-
sibling::element()))//(w:t|w:delText)[empty(ancestor::w:
ins[@w:date > $date]|ancestor::w:del[@w:date <= $date])])

```

The complexity of both XPath queries is due to the intrinsic complexity of the data structure the query has to work on. Although the interface of OpenOffice or MS Word may provide tools to directly deal with these queries using specific strategies on the internal data structures, applications working directly on the XML structure have very little help in disentangling the mess of the data formats.

On the other hand, since EARMARK documents are actually OWL files it is possible to access and query them with plain Semantic Web tools. Powerful searches can be then performed without using niche-specific tools or complex and long XPath expressions but simply with mainstream technologies such as SPARQL 1.1 [78].

The corresponding SPARQL query for (“the textual content of all paragraphs inserted by John Smith”) can therefore be written as follows:

```
SELECT ?p ?r WHERE {
  ?r a earmark:Range .
  ?p a [ a :Insertion ; dc:creator "John Smith" ]
    ; earmark:hasGeneralIdentifier "p"
    ; earmark:hasNamespace "http://..."
    ; co:item/co:itemContent ?r . }
```

But EARMARK is useful for even more than querying: EARMARK also decreases the costs, in terms of efforts and lines of code, for manipulating documents.

Let me consider the task of generating an intermediate version (i.e., neither the first nor the last one of a version chain), from a document that includes change-tracking information about the whole document history.

The process of rebuilding these versions by working on the XML structure without specific APIs is complex and inefficient at the same time. For example a basic XSLT that returns an XML document defining the desired version requires at least to:

- define templates for all the elements actively involved in the change tracking – e.g., for ODT, *text:changed-region*, *text:change-start*, *text:change-end* and *text:change*, and similarly for OOXML– in order to understand, by looking at their creation date, whether they must be considered or ignored when building the requested version. In particular, we must exclude insertions following and deletions preceding the version we are building;
- define templates for paragraphs, in order to handle cases where the paragraph is the result of an insertion or a deletion of other paragraphs, in order to identify whether it should be considered for the result and, in such case, finding out its real text content and remembering that, in the following versions, such content may have spread out among other paragraphs;
- define templates for handling insertions/deletions for structures such as images, sections, lists, and tables;
- define an identity template for the other elements, in order to visit the entire document.

Even the most basic and incomplete implementation of such XSLT requires hundreds of lines of complex and convoluted code and a large number of *ad hoc* decisions

based on the specificities of whether we start from ODT or OOXML. Notice also that a Java-based implementation (or in any other procedural language) of the same process would be equally or even more complex.

The same result can be achieved on EARMARK documents with a few lines of Java code:

```
public EARMARKDocument getVersion(
    EARMARKDocument d, String creator, String date) {

    private final boolean RECURSIVELY = true;
    EARMARKDocument version = null;
    String query =
        "PREFIX earmark: <http://www.essepuntato.it/2008/12/earmark
         #>" +
        "PREFIX co: <http://swan.mindinformatics.org/ontologies
         /1.2/collections/>" +
        "PREFIX dc: <http://purl.org/dc/elements/1.1/>" +
        "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
        "+ \"SELECT ?root\" +
        \"WHERE {\" +
        "?root a earmark:Element ; dc:creator \"\" + creator +
        "\"; dc:date \"\" + date + "\".\" +
        \"FILTER NOT EXISTS { ?mi a earmark:MarkupItem ;\" +
        \"co:item/co:itemContent ?root . } }\";

    EARMARKElement theRoot = d.findNode(query) ;
    version = new EARMARKDocument(
        URI.create("http://www.example.com/version"));
    version.copyNode(theRoot, RECURSIVELY);
    return version;
}
```

This approach uses the EARMARK Java API presented in Section 3.1.3 and a single SPARQL query, runnable on any SPARQL 1.1 processor such as Jena, to identify the root node of the subtree of the version that is associated with the specified date and creator. Then, it performs a simple recursive deep-first visit in order to clone all the nodes in the tree and to combine them in the output EARMARK document.

This method heavily uses Semantic Web technologies on the structures provided by EARMARK whose characteristics are always explicit and clear. Since *all* versions coexist within the EARMARK document and each version can be encoded *explicitly* as a tree within the overall graph, this operation is straightforward and fast.

An evaluation

One of the most frequent criticisms when proposing a different approach to solving a well-known problem in ICT is that the new solution may simplify the difficulties of the specific problem, but brings with it hidden costs in terms of size of the data structure, computation efforts or conversions restrictions that compensate the advantages. In our case, one of the anonymous reviewers of our paper [55] wondered whether a difference in file size could weigh in on the convenience of adopting EARMARK as opposed to working with the original files.

As such, a discussion of cost functions of EARMARK versus other formats is in order. Yet, a systematic discussion of the relative costs (e.g., in byte size) of some original XML-based data structures versus their EARMARK equivalent is an open-ended undertaking that heavily depends on the original XML data structure and the specific features present in the document, and is badly defined anyway: while XML is a linearisation format immediately expressible in actual bytes, OWL (or, more precisely, RDF, the language in which OWL ontologies are expressed) is an abstract structure that allows a large number of linearisation formats (including XML itself) with corresponding huge differences in the final byte counts.

For these reasons, in order to provide at least an initial test of meaningful concepts, I selected two XML-based data formats (OOXML and ODT), and specifically a set of documents where overlapping tricks were present (i.e., where change-tracking was active). And to bypass the size discussion, I decided to test not byte-lengths (which are not meaningful and easily skewed, e.g., by reducing the string length of the element names or of the class names), but the number of nodes for XML documents and of triples for OWL documents. This comparison is again not particularly appropriate (triples are naturally numerous in OWL ontologies, and it is customary to deal with hundreds of thousands and even millions of assertions in Semantic Web applications), but closer to meaningfulness than mere byte count.

Our comparison was carried on a small set of documents in ODT and OOXML that included change-tracking information. As discussed in the previous sections, change-tracking facilities generate rather complex overlaps even for basic operations on small text fragments, which in turn are expressed as a potentially huge number of standoff and milestone markup within the XML hierarchy. The same documents were individually converted into EARMARK. I then charted how simple edits under change-tracking affect the number of nodes in XML formats and of statements in OWL files¹⁷.

I created seven different versions, named after the Seven Dwarfs for recognisability, by applying very common edits (the insertion of few words, the deletion of some sentences, the split of a paragraph, and so on) on a small document, creating multiple overlaps. Fig. 3.4 on the next page shows the results of our comparison.

¹⁷The full details about each version and each format are also available at <http://www.essepuntato.it/2011/jasist/discussion>.

The overall trend is interesting and comforting: while in simple documents with no overlap the node count of XML is lower than the assertion count of EARMARK triples, the presence of overlaps makes EARMARK and XML formats comparable. The growth of EARMARK statements is in fact very close to the growth of XML nodes when the number of overlaps increases. EARMARK is even more efficient than XML for more complex documents.

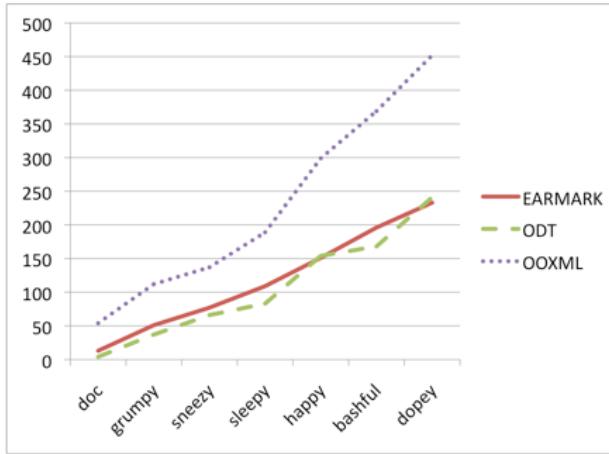


Figure 3.4: A graph summarizing the results of the first experiment.

The measure for each format was done by counting only those nodes and statements instrumental to encode content and (overlapping) structures: I did not take into account neither the presentational information for ODT and OOXML (each file, for instance, includes a very long list of style definitions that are not relevant for the purposes of our analysis) nor namespace declarations (OOXML files, for instance, lists all relevant namespaces for the Office toolkit) nor ignorable white-spaces (that are only added to indent content and improve readability).

Interestingly, EARMARK and ODT show a very similar increase in size, while OOXML is much more verbose and grows faster. The content of the first version, for instance, is encoded using 4 nodes in ODT, 13 statements in EARMARK and 54 nodes in OOXML; the last one contains 241 ODT nodes, 233 EARMARK statements and 452 OOXML nodes. To return to our original enquiry, anyway, it is clear that the weight of EARMARK documents is very good compared to the other ones.

It is also worthy of note the regularity in the growth of EARMARK statements. Regardless of the actual modifications applied to the document, in fact, EARMARK adds about 40 statements for each edit. Both OOXML and ODT, on the contrary, show a more irregular “pace”. The reason is that EARMARK externalizes *all* assertions, so that *all* modifications (either to leaf-nodes or to intermediate nodes in the original XML) are “flattened” onto the documverses and do not depend on the complexity of the structure within which the edit took place.

Fig. 3.5 shows the results of a similar comparison on a different set of documents and edits. We collected seven versions named after the weekdays and created by seven different authors when editing a very simple document. The overall trend does not change and shows that EARMARK and ODT have again a comparable behaviour, far better than OOXML.

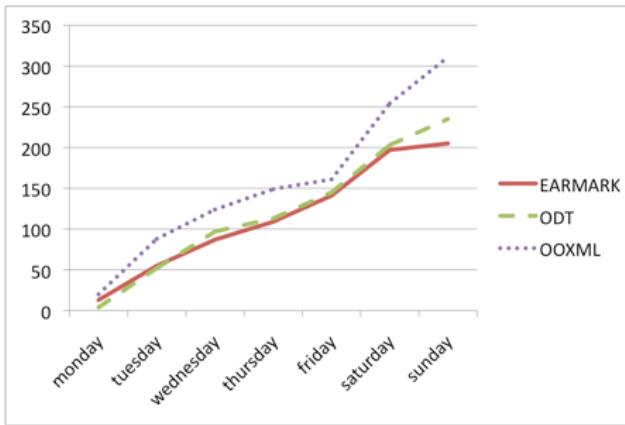


Figure 3.5: A graph summarizing the results of the second experiment.

In conclusion, although preliminary, this study shows clear trends of a very conservative behaviour of EARMARK with respect to document size.

3.2.4 Overlapping with Microformats and RDFA

Microformats [4] add semantic markup to web documents by using common structures of the HTML language itself, in particular the *class* attribute.

The HTML code is annotated using microformats so as to provide new semantic, machine-processable assertions. In the following example, a plain HTML table is enriched with metadata about events¹⁸ and people¹⁹:

```
<body>
  <p class="vevent">
    <span class="summary">WWW 2010 Conference</span>:
    <abbr class="dtstart" title="2010-04-26">April 26</abbr>
    -<abbr class="dtend" title="2010-10-30">30</abbr>,
    <span class="location">Raleigh, NC, USA</span>.</p>
  <table>
    <tr><th>Name</th><th>Role</th></tr>
    <tr class="vcard">
      <td class="fn">Juliana Freire</td>
```

¹⁸HCalendar: <http://microformats.org/wiki/hcalendar>

¹⁹HCard: <http://microformats.org/wiki/hcard>

```

<td class="role">Program Committee Chair</td></tr>
<tr class="vcards">
  <td class="fn">Michal Rappa</td>
  <td class="role">Conference Chair</td></tr>
<tr class="vcards">
  <td class="fn">Paul Jones</td>
  <td class="role">Conference Chair</td></tr>
<tr class="vcards">
  <td class="fn">Soumen Chakrabarti</td>
  <td class="role">Program Committee Chair</td></tr>
</table>
<body>
```

The table was enriched by additional data declaring it to be an event (a conference) and data about the event itself – the url, summary, location – and about four relevant individuals – with their names and roles within the conference – were associated where necessary to the actual content of the table.

So far, so good, and no overlap to speak about. Things change dramatically, though, when the overall structure of the main hierarchy (the HTML table) is at odds with the intrinsic hierarchy of the microformat data, for instance if the people are organized in columns rather than rows. For instance:

```

<table>
  <tr>
    <td>Program Committee Chair</td>
    <td>Conference Chair</td>
    <td>Conference Chair</td>
    <td>Program Committee Chair</td></tr>
  <tr>
    <td>Juliana Freire</td>
    <td>Michael Rappa</td>
    <td>Paul Jones</td>
    <td>Soumen Chakrabarti</td></tr>
</table>
```

Unfortunately, vcards are a hierarchy themselves, and if the hierarchy of vcards is organized differently from the hierarchy of the HTML table, as in the latter case, it is just impossible to define the four vcards for the four people organizing the conference. Thus in plain HTML the choice of one of two possible presentation models for the main hierarchy of content makes it trivial or completely impossible the existence of the second hierarchy.

A possible and partial solution to express vcard hierarchies in the latter example is RDFA [2], a W3C recommendation. It describes a mechanism to embed RDF statements into HTML documents by using some HTML attributes (*href*, *rel*,

rev, content) in combination with other *ad hoc* attributes (*property, about, typeof*) proposed in the recommendation itself.

```
<table
  xmlns:vc="http://www.w3.org/2006/vcard/ns#"
  xmlns:my="http://www.essepuntato.it/2010/05/myVCard#">
  <tr>
    <td about="my:pcc" typeof="vc:Role">
      Program Committee Chair</td>
    <td about="my:cc" typeof="vc:Role">
      Conference Chair</td>
    <td about="my:cc" property="vc:hasName">
      Conference Chair</td>
    <td about="my:pcc" property="vc:hasName">
      Program Committee Chair</td></tr>
  <tr>
    <td about="my:jf" rel="vc:role" resource="my:pcc">
      <span about="my:jf" property="vc:fn">
        Juliana Freire</span></td>
    <td about="my:mr" rel="vc:role" resource="my:cc">
      <span about="my:mr" property="vc:fn">
        Michael Rappa</span></td>
    <td about="my:pj" rel="vc:role" resource="my:cc">
      <span about="my:pj" property="vc:fn">
        Paul Jones</span></td>
    <td about="my:sc" rel="vc:role" resource="my:pcc">
      <span about="my:sc" property="vc:fn">
        Soumen Chakrabarti</span></td></tr>
  </table>
```

Since all attributes live in the context of elements, the price to pay is that to assert everything we want to assert we often need to add some structurally unnecessary elements to the current markup hierarchy of a document, needed only to add the RDF statements (e.g., the *span* elements emphasized above). Even if that does not represent a significant problem for strict Semantic Web theorists, document architects and markup expert see this as a kludge and an inelegant compromise.

Converting the Web document with annotations into an EARMARK document allowing both semantic and structural annotations to coexist can solve these issues. Through EARMARK, I can explicitly express both markup structures and *vcard* assertions. Fig. 3.6 on the next page shows how the *vcard* example can be modelled (once again we show a graphical representation for the sake of clarity).

The textual content of the original table cells is now encoded in two different docuverses, one for the header (with roles) and one for the body (with names of committee members). Ranges *r1, r2, ..., r8* are then created to distinguish each role and name. Two independent and coexisting hierarchies are then built on top of the same

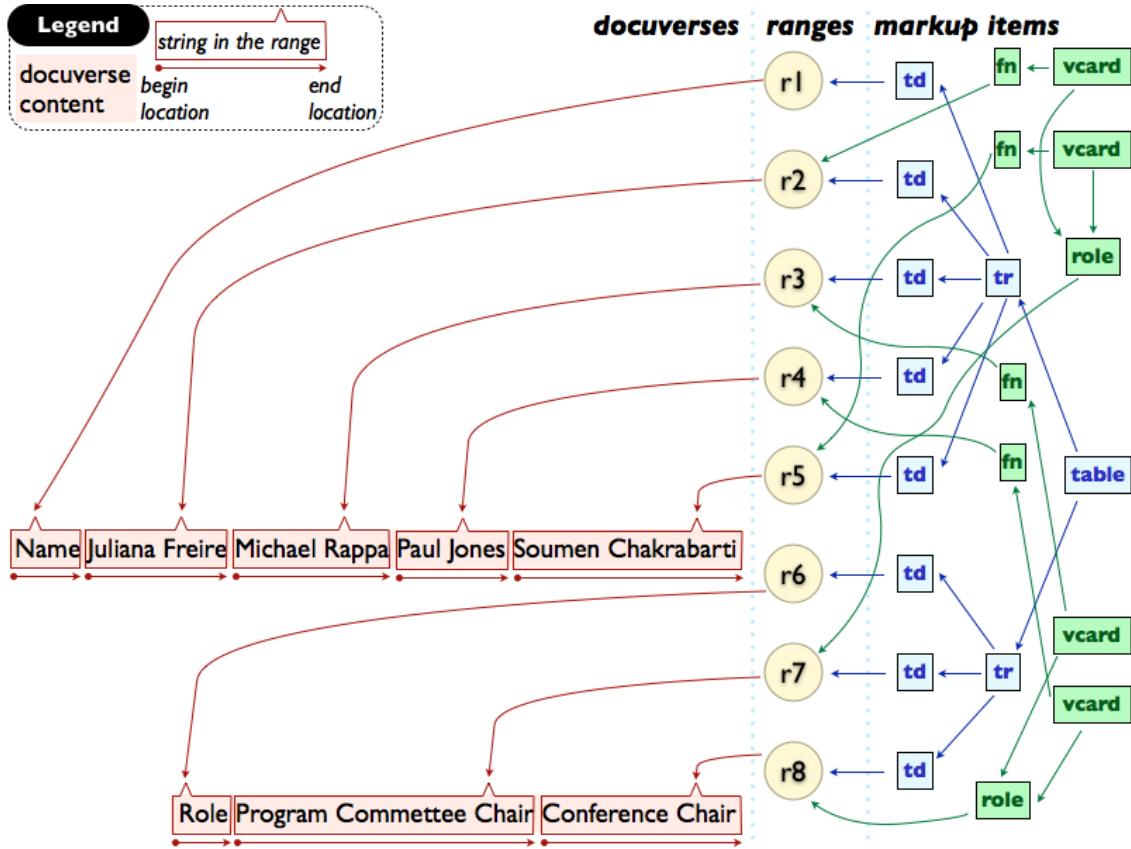


Figure 3.6: The abstract model of the EARMARK document solving the microformats issue.

set of ranges: the HTML table that includes one cell for each range (in blue) and the *Vcards* about each person (in green) that include only the relevant ranges and overlap the previous one. Notice also that the *Vcards* are defined in such a way that does not interfere with the structural features of the table. The full linearisation in OWL of this example can be found at <http://www.essepuntato.it/2011/jasist/examples>.

3.2.5 Wikis: no overlapping where some should be

The strength of wikis lies in their allowing users to modify content at any time. The mechanisms of change-tracking and rollback that are characteristics of all wikis, in fact, promote users' contributions and make "malicious attacks" pointless in the long run, since previous versions can be easily restored.

A number of tools exist that automatically discover "wiki vandalism" and provide users with powerful interfaces to surf changes, *diff* subsequent versions and revert content. For instance, Huggle²⁰ is an application dealing with vandalism in

²⁰Huggle: <http://en.wikipedia.org/wiki/Wikipedia:Huggle>.

Wikipedia, based on a proxy architecture and .NET technologies. A straightforward interface allows users to access any version of a page, highlights contributions of a specific user and reverts the content to old versions.

Even client-side tools – meant to be installed as browsers extensions or bookmarklets – exist to extend the rollback mechanisms of Wikipedia, giving users more flexibility and control over (vandalistic) changes. For instance, Lupin²¹ is a set of javascript scripts that check a wiki page against a list of forbidden terms so that authors can identify undesirable modifications and restore previous (good) versions without a continuous control over the full content of the page; yet again, Twinkle²² provides users powerful rollback functions and includes a full library of batch deletion functions, automatic reporting of vandals, and users notification functions.

These tools are successful in highlighting vandalism and in identifying versions created by malicious users. However, although it is possible to revert the page to any previous version, all changes (even acceptable ones) that were subsequent to the malicious version cannot be automatically inherited by the restored page.

For instance, let me consider versions V1, V2, and V3 of a wiki page, where versions V1 contains a baseline (acceptable) content, V2 is identified as a partial vandalism and is agreed to be removed, but V3 contains (possibly, in a completely different section than the target of the malicious attack) relevant and useful content that was added before the vandalistic version V2 was declared as such. The task of removing the modifications of version V2 while maintaining (whatever is possible of) version V3 is a difficult, error-prone and time-consuming task if done manually, yet there is no tool we are aware of that automatically filters contributions from multiple versions and merges them into a new one (or, equivalently, removes only selected intermediate versions).

Yet, it is possible to characterize the interdependencies between subsequent changes to a document in a theoretical way. Literature has existed for a long time on exactly these themes (see for instance [67] [66]). Although a detailed discussion of abstract models of interconnected changes is out of scope for this paper – details and authoritative references can be found in the above mentioned works – what is relevant in this discussion is that they happen to assume a hierarchical form that is frequently at odds with the hierarchical structure of the content of the document, and as such most issues derive from the data structures in which content is stored and from the model for manipulating these structures. For instance, the fact that in the wiki perspective each version is an independent unit that shares no content (even unchanged content) with the other versions prevents considering multiple versions as overlapping structures coexisting on the same document. If we were able to make these hierarchies explicit we would be able to create models and tools to manipulate these documents in a more powerful way and to exploit the existing interconnections between the overlapping hierarchies.

²¹Lupin, the Anti-vandal tool: http://en.wikipedia.org/wiki/User:Lupin/Anti-vandal_tool.

²²Twinkle: <http://en.wikipedia.org/wiki/Wikipedia:Twinkle>.

EARMARK can be used to improve wiki reversion mechanisms and overcome the limitations discussed above: the automatic filtering and merging of contributions from multiple versions of the same page is now still a *manual* process, but it can be fully automatized if the overlapping structures buried in the *whole* history of the page become explicit.

The role of EARMARK is to make those structures explicit and available for more sophisticated content manipulation. In order to understand to what extent EARMARK structures can be derived from wikis and how they can be exploited by the final users, we use as our example the wiki platform MediaWiki²³, i.e., the wiki engine of Wikipedia.

MediaWiki offers sophisticated functionalities for creating *diffs* of wiki content. Users can compare any two revisions in the page history and highlight changes in a friendly interface that shows modifications with a word-level granularity. *Diff* pages contain metadata about each compared version (when the version was created, who was the author or which IP address an anonymous author was connected from, etc.) and a two-column table showing the changes side-by-side. Changes are detected *a posteriori* by comparing two arbitrary versions, not even requiring them to be temporally contiguous.

The output of the MediaWiki diff engine has regularities that can be exploited to automatically build the overlapping structures of the *diff* and to express them in EARMARK. Let us consider a fictitious example summarized in Table 3.1, where an initial text is revised three times by different authors.

Table 3.1: All the versions of a wiki page modified by different authors.

Version	V1	V2	V3	V4
Author	151.61.3.122	Angelo Di Iorio	Silvio Peroni	Fabio Vitali
Content	Bob was farming carrots and tomatoes	Bob was farming carrots, tomatoes and beans	Bob was farming carrots, tomatoes and green beans. They were all tasteful.	Bob was farming carrots, tomatoes and green beans. [new paragraph] They were all tastyful.

To display the differences between V1 and V2, Mediawiki creates a page whose HTML code is as follows²⁴:

²³MediaWiki: <http://www.mediawiki.org>.

²⁴For the sake of clarity we removed all markup irrelevant to our discussion.

```

<table class="diff"><tbody>
  <tr valign="top">
    <td class="diff-otitle">
      <div id="mw-diff-otitle1">
        <a href=".oldid=413">Revision as of 15:46, 8
          November 2009</a></div>
      <div id="mw-diff-otitle2"><a>151.61.3.122</a></div>
    </td>
    <td class="diff-ntitle">
      <div id="mw-diff-ntitle1">
        <a href=".oldid=414">Revision as of 15:47, 8
          November 2009</a></div>
      <div id="mw-diff-ntitle2">Angelo Di Iorio</div>
    </td></tr>
  <tr>
    <td class="diff-marker">-</td>
    <td class="diff-deletedline">
      <div>Bob was farming carrots <del class="diffchange
        diffchange-inline">and </del>tomatoes.</div></td>
    <td class="diff-marker">+</td>
    <td class="diff-addedline">
      <div>Bob was farming carrots
        <ins class="diffchange diffchange-inline">
          , </ins> tomatoes
        <ins class="diffchange diffchange-inline">
          and beans</ins>. </div></td></tr>
  </tbody></table>

```

This is an HTML table of two rows, the first showing metadata (date and author of the modification) and the second the actual modifications. The first cell of the second row contains all the unmodified text and a *del* element for each inline fragment that was deleted. The second cell contains all the unmodified text and an *ins* element for each inline fragment that was inserted. Thus, these cells share *exactly* the same unmodified part(s) of the two compared versions.

When the structure itself is modified, rather than merely the text, the source code of the MediaWiki *diff* is slightly different. Thus the diff between V3 and V4 (which splits a paragraph in two) is as follows:

```

<tr>
  <td class="diff-marker">-</td>
  <td class="diff-deletedline">
    <div>Bob was farming carrots, tomatoes and green beans.
      They were all tasteful.</div></td>
  <td class="diff-marker">+</td>
  <td class="diff-addedline">

```

```

<div>Bob was farming carrots, tomatoes and green beans.&
nbsp;</div></td></tr>
<tr>
  <td colspan="2">&nbsp;</td><td class="diff-marker">+</td>
  <td class="diff-addedline"><div>&nbsp;</div></td></tr>
<tr>
  <td colspan="2">&nbsp;</td><td class="diff-marker">+</td>
  <td class="diff-addedline">
    <div>They were all tasteful.</div></td></tr>

```

The *diff* output is neither complete nor sophisticated, and of course it is a completely different task to re-plan such algorithm (but for a first idea of natural changes in recognising differences of XML documents, see [51]). Thus, limitations of that algorithm are inevitably shared by any EARMARK representation. Yet, this output is sufficiently rich to allow us to extract the overlapping information we need. For instance, the insertion of a non-breakable-space or a carriage-return generates rows according to specific rules that can be easily detected to capture the actual change by the author.

Fig. 3.7 on the next page shows the above example rebuilt in EARMARK. All versions are encoded in the same document by creating overlapping assertions over the docuverses. Metadata and RDF statements are layered on top of those assertions and create a rich knowledge base about the history of the documents and, in particular, about the history of each fragment.

Due to the complexity of the example we labelled arrows with numbers indicating the position of each range within each markup item. Consider for instance version *V4*: it is composed of two DIV elements, the first one containing the concatenation of “Bob was farming carrots” + “,” + “tomatoes” + “and” + “green” + “beans” + “.”, and the second one contains the string “They are all tasteful”.

Implementing a wiki content filtering mechanism on top of such a structure is rather simple. For instance, the removal of all the contributions of “Angelo Di Iorio”, that leaves untouched all the content written (previously and subsequently) by “Silvio Peroni” and “Fabio Vitali”, can be performed straightforwardly. Three steps are enough to apply such an intermediate content reversion:

1. the identification of the fragments written by “Angelo Di Iorio”, which is a straightforward SPARQL query on the embedded statements;
2. the creation of a new version where references to those fragments are removed and references to fragments no longer in the document are correctly fixed;
3. the translation of that document into an actual MediaWiki page through the serialization process described in [142].

Of course, an automatic process may generate ambiguities or even errors in the resulting content (some parts may become dangling, wrong or unclear after removing

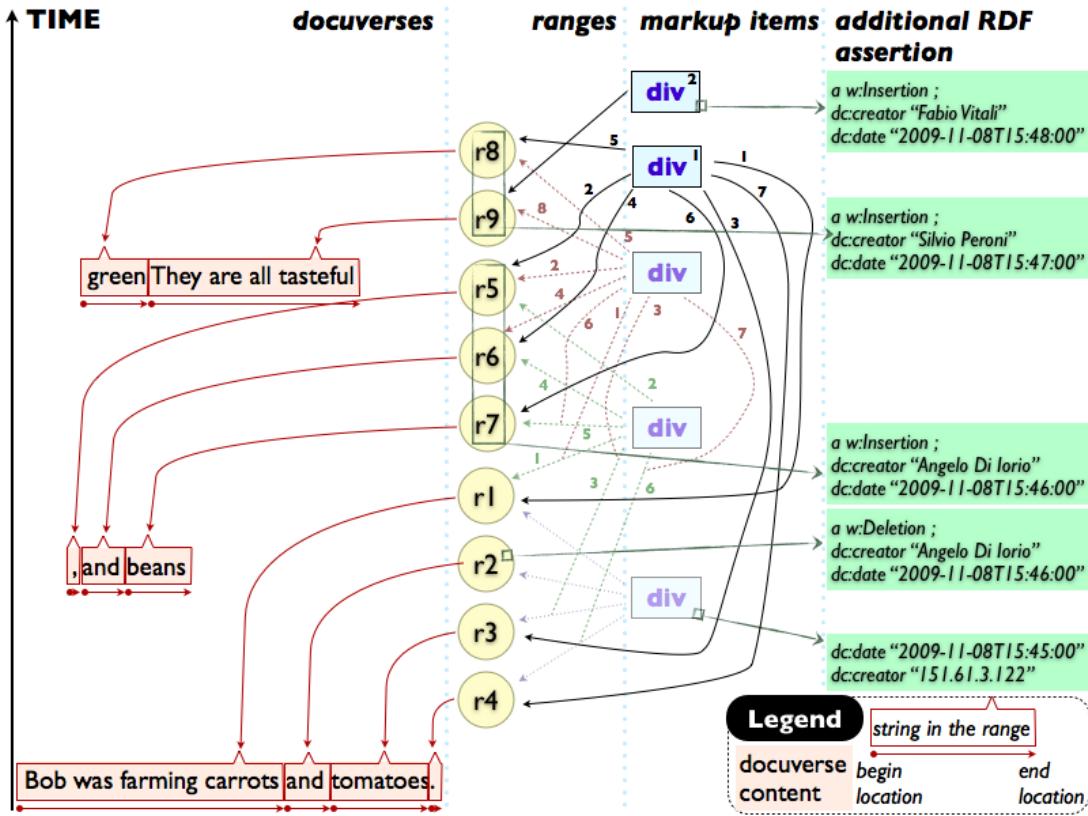


Figure 3.7: The wiki sample versions encoded in a single EARMARK document.

text fragments elsewhere); grammar discrepancies might also be generated by the same approach. Linguistic and semantic problems, however, become a problem once the technical issues of managing independent yet successive edits are solved. What is important is that all the information about overlaps and dependencies among fragments are available in EARMARK and can easily be searched, filtered and manipulated. Besides, foreseeing a manual intervention for checking and polishing automatically-filtered content is perfectly in line with the wiki philosophy, so that the wiki community itself can use the reversion tools wisely to revise the content and adjust any intervening minor nuisances or imperfections. Such checks would still be far simpler and faster than the manual process of partially reverting versions as we have today.

3.3 Structural validation of semantically-defined markup

One of the most important issues addressed by document markup concerns the possibility to express and verify specific syntactic properties:

- the *well-formedness*, that depends to the syntax specified for the particular markup language considered – e.g., in XML-based languages the begin and end tags which delimit elements must correctly nest without overlapping;
- the *validity* against a *vocabulary*, that (formally) restricts the set of values we can use for naming elements and attributes – definable by using schema languages such as XML Schema and RelaxNG for XML;
- the *validity* against a *content model*, that defines contexts in which a particular named element can or cannot be²⁵ – definable by using schema languages as well. Particularly interesting in this context is the validity of a document against a set of *patterns*, i.e., a set of recurring rules and content models.

In this section I discuss how most correctness properties typical of the structural markup, such as the *validity* against a schema, can be expressed through OWL ontologies and verified upon EARMARK documents at a semantic level by means of reasoners, such as Pellet [170], even when dealing with multi-hierarchical structures. In order to support this claim, I also introduce two different running examples, the former based on a simple markup schema and the latter on a specific meta-level theory for document structures based on structural patterns [43].

3.3.1 Defining content-models on EARMARK documents

The assessment of ontological properties in the semantic domain is in a way comparable and can be made to act as validation in the XML domain, that is the process of verification of whether relevant markup items of a well-formed XML document satisfy the constraints embodied in the relevant components of a schema. In the world of XML, several schema languages have been introduced such as DTD [23], XML Schema [75] and RelaxNG [37]. They have different expressive power but they share the same objectives and basic principles: the validator checks the structural properties of a well-formed document by verifying all the *syntactical* constraints expressed in the schema.

Moving from a syntactic perspective to a *semantic* one – as proposed by EARMARK – opens new perspectives for a general approach to assessment as well. A

²⁵More formally, for XML-based languages, a *content model* of a markup element is “a simple grammar governing the allowed types of the child elements and the order in which they are allowed to appear” [23].

key point of such approach is the translation of many markup properties from a syntactical to an ontological level. In the case of XML schema validation, for instance, this means expressing (a) schema definitions as ontology classes and properties and (b) schema documents as ontology instances and assertions, that express hierarchies as semantic relations. Starting from an ontological *TBox* representing the schema and an *ABox* representing the document, we can then conclude that the document is valid according to the schema if and only if the *ABox* is consistent with the *TBox*.

My goal is to design a framework and to implement tools that verify if an EARMARK document is compliant to any property P over its syntax, structure and semantics. The same approach can thus be used for common validation as well as for all the other above-mentioned constraints we want to verify on our documents. My approach – that is meant to be instantiated for each specific case – can be summarized as follows:

1. define an ontology fragment O that details the particular property P we want to verify;
2. associate the EARMARK document instances to O , so as to obtain an ABox for O ;
3. use a reasoner to prove whether the ABox is consistent (i.e., that property P is held) or not (and P is not held).

In order to illustrate how to assess properties on EARMARK documents defining specific ontologies, we first take into consideration simple syntactical property definitions, such as those specifiable for the content models of markup items using schema languages such as DTD, XML schema or RelaxNG.

For instance, let us consider the following seven-sentences informal description of a schema:

1. it is only possible to use elements (e.g., no attributes are allowed anywhere);
2. no element is associated to a namespace;
3. all elements must specify a general identifier;
4. each element contains the other nodes in a specific order;
5. no element with a general identifier different from “phrase”, “noun” and “verb” can be used;
6. each element with general identifier “phrase” can contain, in any order, only an unlimited number of elements with general identifier “noun” or “verb”, and cannot contain text;

7. each element with general identifier “noun” or “verb” can contain only text and no other elements.

Using a RelaxNG Compact-like syntax [38], opportunely extended to allow us to define more than one root element for our documents in the structure *start* (when applied to XML documents, RelaxNG only allows one to define one root), the previous informal schema may be formally expressed as follows:

```
start = (e.phrase | e.noun | e.verb)*
e.phrase = element phrase { (e.noun | e.verb)* }
e.noun = element noun { text }
e.verb = element verb { text }
```

As I have previously introduced, in order to understand whether an EARMARK document is written according to the previous sentences and, consequently, to the previous schema, I have to develop an OWL ontology that implements them. In this case, we can obtain implicit associations between EARMARK document instances and the above property constraints to assess by extending the EARMARK ontology itself.

First of all, I can limit the use of elements (sentence 1) simply defining all markup items equivalent to elements only, as shown in the following excerpt:

```
Class: earmark:MarkupItem
EquivalentTo:
  earmark:Element and
  not(earmark:Attribute or earmark:Comment)
```

This sentence also results in asserting, by inference, that *Attribute* and *Comment* are subclass of the OWL class *Nothing*, which represents the empty set: no individuals can belong to it.

Then, I express sentences 2 to 4 by adding three subclass relations to the *Element* class:

```
Class: earmark:Element
SubClassOf:
  earmark:hasNamespace exactly 0 , co>List ,
  earmark:hasGeneralIdentifier some xsd:string
```

To express sentences 5 to 7, we create a new object property to describe the parent-child relations among EARMARK nodes (i.e., markup items and ranges):

```
ObjectProperty: earmark:hasChild
SubPropertyChain:
  co:item o co:itemContent
```

Since it is defined by a property chain between the properties *item* and *itemContent*, I can say that if an Element individual *e* has an item *i* that refers to a particular EARMARK node *m* – that is a typical parent-child relation concerning

EARMARK elements expressed by bags or lists – then we can infer that e has m as child.

Then, using this property, I can easily cover the remaining sentences (4-7) by adding another subclass relation to the *Element* class:

```
(earmark:hasGeneralIdentifier value 'phrase' and earmark:
    hasChild only
    (earmark:hasGeneralIdentifier some {'noun', 'verb'})) or
((earmark:hasGeneralIdentifier some {'noun', 'verb'}) and
    earmark:hasChild only earmark:Range)
```

That's it: the above assertions are sufficient to assess whether an EARMARK document is valid against the schema presented²⁶. For instance, when I try to verify, through a reasoner, whether the EARMARK document in Section 3.1.3 is consistent with the model introduced in this section, I will receive an inconsistency exception because the element p of the sample document is not allowed.

Property definitions through OWL ontologies can be used to define schemas for EARMARK documents. Yet, rather than discussing validation, that is addressed in other works, such as [68] [152] [198], in the following sections we will concentrate, as an extensive example, on the assessment of the properties connected to *structural patterns*, such as the ones discussed in [43] and [50].

3.3.2 Structural patterns

The idea of using patterns to produce reusable and high-quality assets is not new in the literature. Software engineers [69], architects (as Alexander who first introduced this term [3]) and designers very often use – or rather *reuse* – patterns to handle problems that recur over and over. Patterns have also been studied to modularize and customize web ontologies [146]. They guarantee the flexibility and maintainability of concepts and solutions in several heterogeneous scenarios.

My research group has been investigating patterns for XML documents for some time [43] [50]. The overall goal of our research is to understand how the structure of digital documents can be segmented into atomic components, which can be manipulated independently and re-flowed in different contexts. Instead of defining a large number of complex and diversified structures, a small number of *structures/patterns* has been identified, sufficient to express what most users need. The idea is that ten patterns, shown in Table 3.2 on page 86, are enough to capture the most relevant document structures.

The two main characterizing aspects of such pattern theory are:

- *orthogonality* – each pattern has a specific goal and fits a specific context. The orthogonality between patterns makes it possible to associate a single pattern

²⁶The model is available at <http://www.essepuntato.it/2011/03/schemaexample>.

to each of the most common situations in document design. Then, whenever a designer has a particular need he/she has to only select the corresponding pattern and to apply it;

- *assemblability* – each pattern can be used only in some locations (within other patterns). Although this may seem a limitation, such strictness improves the expressiveness and non-ambiguity of patterns. By limiting the possible choices, patterns prevent the creation of uncontrolled and misleading content structures. This characteristic still allows the presence of overlapping items – for example, a block that contains two different inlines that overlap upon the same segment continues to be a valid structure in terms of patterns because its content model is not violated, even though the presence of overlapping descendants.

These patterns allow authors to create unambiguous, manageable and well-structured documents. The regularity of pattern-based documents makes it possible to perform easily complex operations even when knowing very little about the documents' vocabulary. Designers can implement more reliable and efficient tools, can make hypotheses regarding the meanings of document fragments, can identify singularities and can study global properties of sets of documents.

There are two main methods to check if (and how) a document uses patterns or can be normalized into a new pattern-based resource. A procedural approach requires *ad hoc* tools – written in a procedural programming language and running on a software platform – that are difficult to write, test, maintain, and extend. A declarative approach, on the other hand, guarantees more flexibility, extensibility and portability. The ontological model adopted by EARMARK is particularly suitable for such a context: verifying that a document meets the requirements given by patterns, in fact, only requires verification using a reasoner of some properties of an OWL ontology.

In the next section I will propose an example of this approach, made even more complex by the presence of an overlapping structure that makes property verification through XML technologies much more complex, and possibly even impossible.

Assessing structural patterns on EARMARK documents

The first step to verify patterns-related properties of EARMARK documents is obviously to build an ontology that describes such patterns and their relationships. We developed the ontology through a set of hierarchical class/sub-class relations, summarized in Fig. 3.8 on page 73.

We have three levels of abstraction modelling different aspects of our theory:

- the *top-level* classes describe the general features of a pattern. In particular, they characterize their content model expressing the possibility for the element

to contain text (*Textual*) or not (*Structural*) and to contain other elements (*Hierarchical*) or not (*Leaf*);

- the *middle-level* classes – i.e., *Marker*, *Bucket*, *Flat* and *Mixed* – model all the combinations of top-level classes and express their disjointness;
- the *bottom-level* classes include all the “instanceable” classes, i.e., all those classes which document elements can explicitly belong to.

This ontology allows me to verify whether or not an EARMARK document follows the structural patterns. In particular, it allows me to verify if a given association between elements and patterns (that assigns one pattern to each element) is valid. That, again, means checking whether the ontological association of each element to a particular pattern is consistent.

For assessing patterns on an EARMARK document D , we need to apply the following steps:

1. for each element in D , associate an instanceable pattern to the element;
2. for each element in D , assert the element belongs to the class defined by the restriction `pattern:isContainedBy exactly 0 pattern:Pattern`²⁷ whether it is not contained by any other element – where `isContainedBy` is the inverse of the property `contains` shown in Fig. 3.8 on the facing page;
3. launch a reasoner to check if the pattern ontology with these added assertions is consistent (all the pattern constraints hold) or not (there are some errors when assigning patterns to elements).

Before presenting experimental results, let me show some excerpts from the pattern ontology developed, just to sketch out how we formally define those structural properties. First of all, I need to express containment relations among patterns through specific OWL object properties, as shown as follows:

```
ObjectProperty: pattern:contains
  Domain: pattern:Hierarchical
  Range: pattern:Pattern
ObjectProperty: pattern:isContainedBy
  InverseOf: pattern:contains
```

Expressing EARMARK markup item containments in terms of these two properties is straightforward by using particular rules to infer new assertions:

```
earmark:MarkupItem(x), co:Set(x),
co:element(x,y), earmark:MarkupItem(y)
-> pattern:contains(x,y)
```

²⁷The prefix `pattern` refers to “<http://www.essepuntato.it/2008/12/pattern#>”.

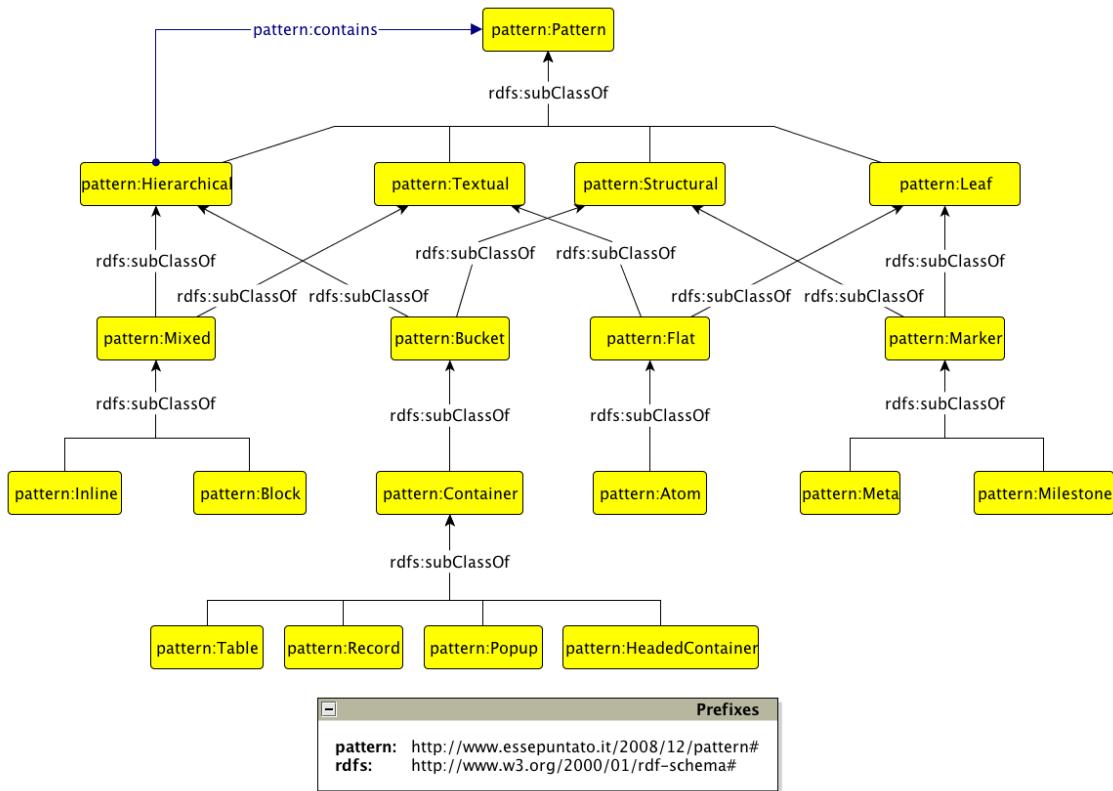


Figure 3.8: Graffoo diagram that summarises the classes describing the pattern theory.

```

earmark:MarkupItem(x), co:item(x,y),
co:itemContent(y,z), earmark:MarkupItem(z)
-> pattern:contains(x,z)

```

Taking into consideration the above properties and assertions, I illustrate, as example, the definition of *inline* and *block* patterns, discussing the main differences among them. I implement them through two OWL classes:

```

Class: pattern:Inline
SubClassOf:
  pattern:Mixed and pattern:contains only
    (pattern:Inline or pattern:Milestone or pattern:Popup),
    pattern:isContainedBy some pattern:Block
Class: pattern:Block
SubClassOf:
  pattern:Mixed and pattern:contains only
    (pattern:Inline or pattern:Milestone or pattern:Popup)
DisjointWith: pattern:Inline

```

As shown, content models for these classes are defined by adding restrictions in form of superclasses, such as:

```
pattern:Mixed and pattern:contains only
  (pattern:Inline or pattern:Milestone or pattern:Popup)
```

All *Textual* individuals are inferred starting from markup items containing some ranges:

```
earmark:MarkupItem(x), co:Set(x),
co:element(x,y), earmark:Range(y)
-> pattern:Textual(x)

earmark:MarkupItem(x), co:item(x,y),
co:itemContent(y,z), earmark:Range(z)
-> pattern:Textual(x)
```

The entire pattern ontology definition is available online²⁸ and uses SWRL [96] as the rule language.

Experimental results

In the following I discuss an explicative example on the first three verses of the *Paradise Lost* by John Milton, which are often quoted as a standard example for enjambement in poetry:

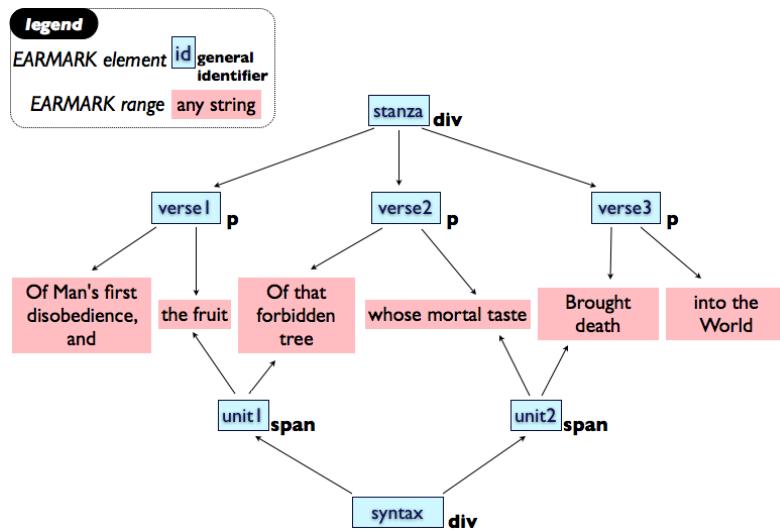


Figure 3.9: The EARMARK document, in the form of a graph, of the first three verses of the *Paradise Lost* by John Milton.

²⁸The Pattern Ontology: <http://www.essepuntato.it/2008/12/pattern>.

```
Of Man's first disobedience, and the fruit
Of that forbidden tree whose mortal taste
Brought death into the World
```

I want to describe two different hierarchies, one for the verses and another one for the syntactical units giving rise to the enjambments, and we use an HTML-like syntax for the general identifiers. Fig. 3.9 on the preceding page shows the graph representation of a corresponding EARMARK document²⁹.

Table 3.3 on page 87 summarizes our experiments running a reasoner to verify patterns on the previous example³⁰. I tried eight different combinations of patterns and – for each combination – I checked whether it generated a consistent ontology.

The fourth column of the table shows the output of the reasoner. The answer “yes” indicates that a specific combination does not violate the constraints of the relevant patterns. That combination is then valid and can be used for identifying structural roles of document’s elements. Negative results are very relevant because the reasoner is able to identify that requirements are not respected, and why. For instance, the reasoner detects that an element *div* cannot be an inline, being the root of a hierarchy.

Although this example is very simple (and focuses only on a few patterns), it should give readers the idea of how a reasoner and the ontological descriptions of a set of constraints can give validity results on rules expressed over an EARMARK document, and regardless of whether the document is in fact a single hierarchy (i.e., an XML document) or actually represents a multiplicity of hierarchies hard or impossible to express in XML. The fact that here I deal with patterns is just an example, and does not affect the generality and applicability of this approach: additional classes and additional SWRL rules would allow us to test other properties of the same documents.

3.3.3 Validation of document markup

Two particular topics are relevant to what we examine in previous sections: the ontological representation of digital document properties such as schema validity and other non-ontological approaches for validating complex overlapping structures.

After generating a well-defined document structure mapped into ontological assertions through EARMARK, we would like to verify whether it satisfies some particular properties or not – for example, validation against a document schema. Document schemas can also be good starting points for developing or reengineering ontologies, as pointed out in [68]. Proposing a translation mechanism from an XML document into a set of RDF assertions that describe it, Ferdinand et al.’s main aim is to define a way to translate an XML Schema document into an OWL ontology

²⁹<http://www.essepuntato.it/2010/04/ParadiseLost>

³⁰<http://www.essepuntato.it/2010/04/ParadiseLost/test>

in order to obtain a complete TBox+ABox from a document with markup and its schema.

Similar to the previous work but following a different approach, [152] introduces a framework for producing an ABox starting from an XML document, passing through an existing OWL ontology and the XML Schema document used by the original document. The *JXML2OWL*³¹ framework thus generates such an Abox via some XSLT processing, made possible through handmade associations between the XML Schema declarations and the OWL classes. The result of this process is a set of OWL instances related to the OWL ontology used for the transformation. Proving the consistency of these instances with respect to the ontology means verifying the validity of the original document against the referred schema.

Another study that works similarly is [198]. After introducing the main differences between XML Schema and ontologies in general, Yang et al. propose an ontology-based approach for representing mappings between that document schema language and ontologies. The goals of this work are round-trip translations from an XML document into ontology instances and the realization of an ontology-mediated transformation between different XML documents, in order to allow seamless data exchange between heterogeneous XML data source.

Considering non-ontological approaches, the *rabbit/duck grammars* mechanism [175] is a good technique for the validation of documents with overlaps. The basic idea of this approach is simple: to define different document schemas whose intersection describes the markup language to be validated. In each schema, each markup element is associated to one out of four classes – *normal*, *milestones*, *transparent* and *opaque* – that define how elements have to be considered in the context of the schema taken into consideration. Given a complete set of rabbit/duck grammars, each element must belong to the class *normal* in at least one schema of the set. Then, a document is considered valid against the set of rabbit/duck grammars if and only if it is valid against each schema in the set.

3.4 Dealing with Markup Semantics

Complementary to existing Semantic Web research work, which typically aims at studying uses and applications of *semantic markup* (i.e., defining relations among resources), in this section I address the issue of *markup semantics*: the formal definition of meanings of markup elements, besides the syntactical structure of a markup document [149].

EARMARK is suitable for expressing markup semantics straightforwardly. However, I want to associate coherent semantics to markup items following precise and theoretically-founded principles, which makes our application interoperable across different vocabularies used e.g. in digital libraries.

³¹JXML2OWL: <http://jxml2owl.projects.semwebcentral.org>.

As a matter of fact, different existing vocabularies tackle the representation of terms vs. meanings vs. things in general, and this is not only true for XML markup languages, but also for semantic web ontologies such as SKOS, FRBR, CIDOC, OWL-WordNet, LIR, LMF, etc. Unfortunately, each of them has a particular approach depending on the original requirements they were designed for (thesauri encoding, media item representation, standardizing digital library vocabularies, lexicon or (multi-)linguality representation, etc.), so that aligning all or part of them for a specific use is a difficult operation, specially when we consider the domain of document structures, where arbitrary representations lead to different realizations for the user, to lack of interoperability, and lock markup semantics into islands.

A viable solution to get around this problem is to align existing vocabularies to more general and comprehensive theories. The main benefit of using shared principles and well-grounded studies – e.g., patterns (e.g., documental [43], ontological [146]), linguistic theories (e.g., Pierce’s semiotic triangle [139], Saussure’s semiology [155], Jakobson’s communication model [102], Searle’s linguistic acts [161]), NLP approaches (e.g., Guthrie et al.’s [86]) – is that they enable the interoperability between different vocabularies.

In this work I have adopted the *linguistic act*³² (*LA*) ontology, based on the *Linguistic Meta-Model (LMM)* [143]. It provides a semiotic-cognitive representation of linguistic knowledge. The general idea beyond it is to handle the representation of different knowledge sources developed according to different (and even implicit) semiotic theories, putting each of them in the context of the *semiotic triangle* [139] and some related semiotic notions, as shown in Fig. 3.10 on the next page.

The *linguistic act* is defined through an OWL ontology that implements the basic ideas of semiotics:

- *References*: any individual, set of individuals, or fact from the world we are describing. They can have interpretations (meanings) and can be denoted by information entities. For example: *Fabio*, *the set of Fabio’s relatives*, or *the fact that Fabio is a professor*;
- *Meanings*: any (meta-level) object that explains something, or is intended by something, such as linguistic definitions, topic descriptions, lexical entries, thesaurus concepts, logical concepts or relations, etc. They can be “interpretants” for information entities, and “conceptualizations” for individuals and facts. For example, concepts such as *person*, *paragraph*, *having a role*;
- *Information entities*: any symbol that has a meaning, or denotes one or more references. They can be natural language terms, sentences or texts, symbols in formal languages, icons, or whatever can be used as a vehicle for communication – for example: the string “*Fabio*”, the markup elements *p*, *agent*, *noun*

³²Linguistic Act Ontology: <http://ontologydesignpatterns.org/cp/owl/semiotics.owl>.

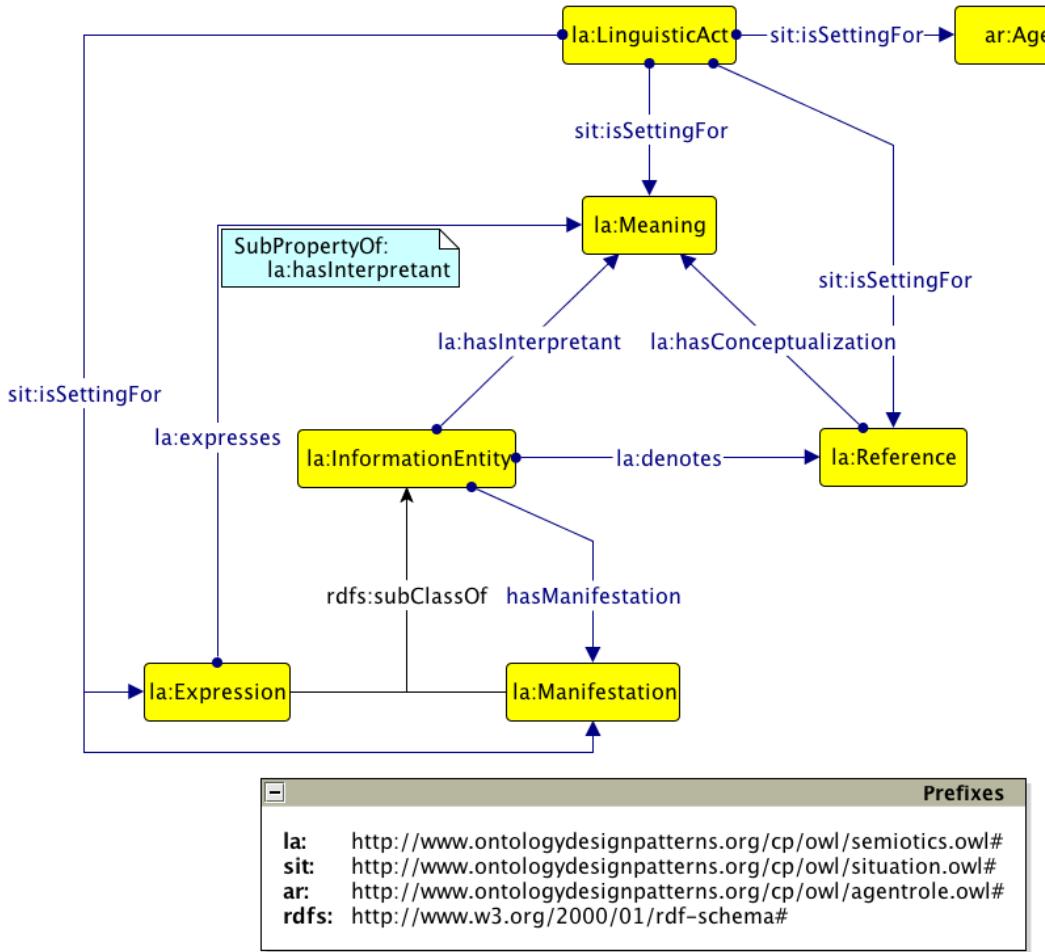


Figure 3.10: A diagram summarizing the ontology pattern *linguistic act*.

and *verb*. They have at least one meaning and can denote references. Moreover, each information entity can be an *expression* (e.g., the string “Fabio”) realized in one or more *manifestations* (e.g., the string “Fabio” contained in a particular XML file stored on somebody’s hard drive) having the same interpretation.

- Linguistic acts: any communicative situation including information entities, agents, meanings, references, and a possible spatio-temporal context (i.e. when and/or where the act has been performed). For example, dialogs, taggings, writings.

Considering these premises, EARMARK markup items are specific kinds of expressions expressing a particular meaning, usually assigned implicitly by the author of a schema or a markup, which are used to denote local objects (e.g., their content,

according to the definition of a markup object) and/or social entities (e.g., persons, places, communication events, etc.).

For example, in the XML example introduced in Section 3.1.3 there are different semantic blocks: firstly, the element *agent* expresses the meaning of “agent” (i.e., as the resource defined by DBpedia³³) and denotes a specific person (i.e., the person, using FOAF, is known as “Fabio Vitali”), while the element *p* must be interpreted as a paragraph (i.e. a specific document structure according to the DoCO ontology, that will be introduced in Section 4.4) and denotes the string “Fabio says that overl-happens” (rather than the corresponding concept). This in a way differs from the XML syntactical structure in which the element *p* contains the elements *agent*, *noun* and *verb* – that themselves express/denote/contain the other meanings/references.

In LA-EARMARK, it is possible to describe both the rigid syntactic structure, as described in Section 3.1.3, as well as its semantic connotation:

```

@prefix ar: <http://www.ontologydesignpatterns.org/cp/owl/
  agentrole.owl#> .
@prefix la: <http://www.ontologydesignpatterns.org/cp/owl/
  semiotics.owl#> .
@prefix sit: <http://www.ontologydesignpatterns.org/cp/owl/
  situation.owl#> .
@prefix doco: <http://purl.org/spar/doxo/> .
@prefix dbpr: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:r0-28 a earmark:PointerRange
; earmark:refersTo ex:doc
; earmark:begins "0"^^xsd:integer
; earmark:ends "28"^^xsd:integer .

ex:p la:expresses doco:Paragraph
; la:denotes ex:r0-28 .

ex:agent la:expresses dbpr:Agent , doco:TextChunk
; la:denotes ex:fv , ex:r0-5 .

ex:fv a foaf:Person
; foaf:givenName "Fabio"
; foaf:familyName "Vitali" .

ex:markupAuthor a ar:Agent
; ar:hasRole [ a ar:Role
; rdfs:label "markup author" ] .

```

³³DBpedia “agent” resource: <http://dbpedia.org/resource/Agent>.

```

[] a la:LinguisticAct
; rdfs:comment "marking a paragraph up"
; sit:isSettingFor ex:p , ex:r0-28
, doco:Paragraph , ex:markupAuthor .

[] a la:LinguisticAct ; rdfs:comment "marking text up"
; sit:isSettingFor ex:agent , ex:r0-5
, doco:TextChunk , ex:markupAuthor .

[] a la:LinguisticAct
; rdfs:comment "markup element as instance"
; sit:isSettingFor ex:agent , ex:fv , dbpr:Agent .
...

```

The example introduced explains how it is possible to describe markup hierarchies – and therefore their semantics – upon those markup items. In the next sub-sections I show the advantages of using LA-EARMARK in two different use cases, previously highlighted in [149]: querying documents marked up with the same implicit semantics but marked up with different vocabularies that share the same implicit semantics and the semantic validation of markup items.

3.4.1 Searches on heterogeneous digital libraries

Digital libraries about journal research articles use to actually store their documents' content using specific XML formats, e.g. the common TEI [186], DocBook [194], or other less common vocabularies developed expressly for a specific collection. Clearly, the more digital libraries we consider, the more non-interoperable formats we will find, although they express, more or less, the same kinds of documents and, consequently, document semantics. Paragraph, sections (implicitly or explicitly labelled as abstract, introduction, results, discussion, related works, conclusions, acknowledgements, bibliography, etc.), figures, tables, formulas are a little but important part of the elements that we will find in the markup of journal papers, regardless of the actual vocabulary used.

In this scenario of heterogeneous formats expressing homogeneous content, looking throughout a number of digital libraries for particular document fragments, such as “All the tables that are part of the results sections of articles written by Silvio Peroni”, can be approached only by addressing each digital library with a query specific of the vocabulary used, and then merging the results. Obviously, the implicit (shared) semantics of the query must be implemented in each digital library in a (different) explicit way, for example by using tools for mapping the query into each specific markup structure. This means requiring a particular *ad hoc* and non-interoperable mechanism for each format of each digital library.

Expressing semantics of elements in a journal article by considering a shared model may help for increasing interoperability, but it is not enough, because the different formats will still be a substantial problem. For example, being a *section* presenting *results* in a particular research article may be expressed differently depending on the format used: `<div class="section.results">`, `<section id="results">`, `<sec class="results">`, `<results>`, etc.

Expressing journal articles in LA-EARMARK – obtained, for instance, by translating the original XML documents via GRDDL [39] – allows one to specify the semantics of markup elements according to some formal model, without attention to the specific markup vocabulary³⁴:

```
ex:div a earmark:Element
; earmark:hasGeneralIdentifier "div"
; co:firstItem [ co:itemContent ex:classAttr ]
; la:expresses doco:Section , deo:Results .

ex:results a earmark:Element
; earmark:hasGeneralIdentifier "results"
; la:expresses doco:Section , deo:Results .
```

As shown in the previous excerpt, both `ex:div` and `ex:results` elements express the same semantics even if their names differ: they are syntactically different (their content models differ), but semantically equivalent.

Enabling digital libraries to express each LA-EARMARK document as a named graph, with all the document metadata referring to it, allows one to query more than one digital library at the same time by using a single SPARQL 1.1 query [78]. For instance, a plausible SPARQL query for the above-mentioned request – “All the tables that are part of results sections of the article written by Silvio Peroni” – is:

```
SELECT ?table WHERE {
GRAPH ?doc {
?table a earmark:Element
; la:expresses doco:Table
; (^co:itemContent/^co:item)+
[ a earmark:Element
; la:expresses
doco:Section , deo:Results ] } .
?doc dcterms:creator "Silvio Peroni" }
```

³⁴The prefix `deo` refers to the *Discourse Element Ontology (DEO)*, ontology for the characterisation of the major rhetorical elements of a document (e.g., a research article), such as the introduction part, the evaluation section, the conclusions and so on. It is available at <http://purl.org/spar/deo>.

3.4.2 Validation of “Markup sensibility”

Sometimes it is not possible to understand whether a particular markup element that is valid at the syntactic and structural level is also valid at the semantic level, i.e., the level that Bauman described as *markup sensibility*: “Does a construct make sense, e.g., a proposition or an assertion?” [13]. A clear example of this difficulty can be found with heavily interlinked documents that make systematic references to precise concepts in their content.

For instance, Akoma Ntoso [10] [9] is an open legal XML standard for parliamentary, legislative and judiciary documents, promoted by the Kenya Unit of the United Nations Department for Economics and Social Affairs (UN/DESA) in 2004. Originally meant for African Countries, it is now promoted also in Latin America, Asia and various European countries. Akoma Ntoso describes structures for legal documents using a vocabulary of common structures based on XML, references to legal documents across countries using a common naming convention based on URIs, and a systematic set of legal metadata values using an ontologically sound approach compatible with OWL and GRDDL.

This markup language is defined by means of a very complex XML Schema document, which defines the vocabulary and the content models of markup items. Although that schema is enough to guarantee the validity of a document from a pure syntactical point of view, there are semantic connections that are useful to verify but cannot be simply using a schema language. Let us introduce an Akoma Ntoso excerpt to clarify the point:

```
<akomaNtoso>
  <meta>
    ...
    <references source="#fv">
      <TLCPerson id="fv"
        href="/ontology/it/person/FabioVitali">
      <TLCPerson id="smith"
        href="/ontology/uk/person/JohnSmith">
      <TLCRole id="mineconomy"
        href="/ontology/role/government/MinisterOfEconomy">
      ...
    </references>
    ...
  </meta>
  <body>
    ...
    <speech id="sp1" by="#smith" as="#mineconomy">
      <p>Honorable Members of the Parliament, ... </p>
    </speech>
    ...
  </body>
</akomaNtoso>
```

```
</body>
</akomaNtoso>
```

The elements *TLCPerson* and *TLCRole*, introduced within the metadata block (element *meta*) of the document, are used for specifying the presence, in the document in which it is defined, of two particular ontological entities, respectively a person and a role, according to a specific underlying ontology. Wherever these elements are referred to by a markup element by means of its identifier (as expressed in the attribute *id*), what really is referred to are the ontological individuals that are specified by the attribute *href*. For instance, within the body of the document, the element *speech* is used to mark up the transcription of a speech performed by the person *John Smith* (attribute *by*) who is temporarily playing the particular role of Minister of the Economy (attribute *as*). Moreover, the attribution of all the metadata concerning the speech transcription is an editorial activity, rather than authorial, made specifically by an agent identified through the attribute *source* of the element *reference*. For self-containment, the attributes *by* and *as* do not refer directly to the ontological concepts associated to John Smith and the Minister of the Economy, but to an intermediate jumping station, i.e., the elements *TLCPerson* and *TLCRole* in the metadata block.

Although it is a fundamental requirement of the language, the syntactic validation through XML Schema of the document does not provide sufficient information to understand whether an Akoma Ntoso document is really correct and coherent, because it cannot prove the sensibleness of markup. In the preceding example, we also need to check:

- the validity of the elements *TLCPerson* and *TLCRole* as reflection of the consistence of people and role individuals within an underlying ontology, particularly by checking whether each individual can really be a person (or a role) without provoking an inconsistency with other classes the individual may belong to;
- the validity of the element *speech* as markup denoting a particular speech event that involves only and at least a person as speaker. Moreover, because it reflects a speech, it must contain some text.
- the fact that the person John Smith was, at the moment of the speech, either the Minister of Economy or acting as an authorized delegate through a track of explicit delegations starting from the current minister.

The XML Schema language is not able to express these kinds of constraints. Naive or inexpert metadata authors could very well generate documents that are syntactically and structurally valid, possibly even apparently correct from a semantic point of view, but fundamentally incoherent. For instance, a common misconception is to confuse persons and roles, as in the following (syntactically valid but ontologically incorrect) example:

```
<speech id="sp1" by="#mineconomy">
  <p>Honorable Members of
    the Parliament, ...
</speech>
```

The LA-EARMARK translation of the above fragment, that also includes its semantic description, is the following³⁵:

```
@prefix akomantoso: </ontology/entity/> .

</ontology/uk/person/JohnSmith>
  a akomantoso:Person .

[] a la:LinguisticAct
  ; sit:isSettingFor
    <smith> , akomantoso:Person
    , </ontology/uk/person/JohnSmith> .

<sp_1> a earmark:Element
  ; earmark:hasGeneralIdentifier "speech"
  ; la:expresses akomantoso:Speech
  ; la:denotes _:aSpeechEvent , _:p .

_:aSpeechEvent a akomantoso:SpeechEvent
  ; akomantoso:hasSpeaker
  </ontology/uk/person/JohnSmith> .

_:p a earmark:Element
  ; earmark:hasGeneralIdentifier "p" .

[] a la:LinguisticAct
  ; sit:isSettingFor
    <sp_1> , _:aSpeechEvent
    , </ontology/uk/person/JohnSmith>
    , akomantoso:Speech .

[] a la:LinguisticAct
  ; sit:isSettingFor <sp_1> , _:p
  , akomantoso:Speech .
```

LA-EARMARK allows one to check the sensibility of markup precisely, by defining semantic constraints as ontological axioms, taking into account both classes and properties defined in LA-EARMARK and in the underlying ontology behind Akoma Ntoso. Inasmuch as such semantic constraints can be defined as axioms adhering to

³⁵The prefix *akomantoso* is associated to the minimal *glue* ontology within the XML document itself that connects markup structures to legal concepts according to the model explained in [10].

or in contrast with axioms of the underlying ontologies, they can be directly applied to reasonings even in open world frameworks such as OWL.

For example, a plausible ontological constraint (written in Manchester Syntax) for all the markup elements *speech* is:

```
(earmark:Element that earmark:hasGeneralIdentifier
  value "speech")
SubClassOf
(sit:hasSetting only
  (la:LinguisticAct that
    sit:isSettingFor exactly 1
      (earmark:Element and la:InformationEntity)
    and
    sit:isSettingFor exactly 1
      (earmark:Range and la:Reference)
    and
    sit:isSettingFor value akomantoso:Speech)
  or
  (la:LinguisticAct that
    sit:isSettingFor exactly 1
      (earmark:Element and la:InformationEntity)
    and
    sit:isSettingFor exactly 1
      ((akomantoso:SpeechEvent and
        la:Reference) that
        akomantoso:hasSpeaker some
        akomantoso:Person)
    and
    sit:isSettingFor
      value akomantoso:Speech))
```

This specification would be able to capture ontological errors in the actual Akoma Ntoso document such as the one presented previously, where the author of the speech is specified as a role rather than a person.

Table 3.2: A brief summary of the structural pattern theory of [43] extended with two more patterns: *headed container* and *popup*.

Pattern	Description	Example (HTML)	May contain
Meta	an empty element whose meaning depends on its presence in a document (rather than on its position) and its attributes	meta	EMPTY
Milestone	an empty element whose meaning depends on its position	br	EMPTY
Atom	a unit of unstructured information	title	text
Block	a block of text mixed with unordered and repeatable inline elements, with the same content model	p	text, milestone, inline, popup
Inline	a block of text mixed with unordered and repeatable inline elements, with the same content model	i	text, milestone, inline, popup
Container	a sequence of heterogeneous, unordered, optional and repeatable elements	body	meta, atom, block, record, container, table
Headed Container	a sequence of heterogeneous, unordered, optional and repeatable elements headed by a sequence of blocks	section that contains h1	meta, atom, block, record, container, table
Record	a set of optional, heterogeneous and non-repeatable elements	html	meta, atom, block, record, container, table
Table	a sequence of homogeneous elements	ul	meta, atom, block, record, container, table
Popup	a sequence of heterogeneous, unordered, optional and repeatable elements	math	meta, atom, block, record, container, table

Table 3.3: Testing associations between elements and patterns on the “Paradise Lost” example through an OWL reasoner.

div	p	span	Is the ontology still consistent?
container	block	atom	yes
container	block	inline	no: a container (syntax) cannot contain inline
block	block	inline	no: a block (stanza) cannot contain at any level another block
block	inline	inline	yes
inline	inline	inline	no: inlines (both stanza and syntax) cannot be the root of a hierarchy
block	inline	inline for unit1 and atom for unit2	no: elements with the same general identifier (unit1 and unit2) must have the same pattern
atom	atom	atom	no: atoms (both stanza and syntax) cannot contain other elements
table	atom	atom	yes
record	atom	atom	no: records (stanza and syntax) can contain only elements with different general identifiers

Chapter 4

The Semantic Publishing And Referencing Ontologies

One of the main research areas in semantic publishing is the *development of semantic models* (vocabularies and ontologies) that meet the requirements of scholarly authoring and publishing. As introduced in Chapter 2, in the past several works have proposed metadata schemas, vocabularies and ontologies to describe the publishing domain. However those models show some limitations. Some of them (e.g., Dublin Core Metadata Terms [63]) define bibliographic objects by means of abstract concepts that do not fully comply with the vocabulary used by publishers. Others (e.g., the Bibliographic Ontology [42]) have been developed to describe parts of the publishing domain, but lack in describing specific topics (e.g., characterisation of bibliographic citations, definition of agent's publishing roles, description of publishing workflows) and are not interoperable with other models (e.g., FRBR [100]).

It appears clear that the development of a set of models that aim at describing the main part of the publishing domain must pass through the adoption of established methodologies for ontology modularisation and development [148] [166]. These activities may be eventually supported by the use of statistical clustering techniques, e.g. [36], though I did not explicitly used them in this work. Moreover, the following principles¹ should be taken into account:

- to have an extensive interaction with publishers and members of academic communities to clarify their requirements;
- any area of interest of the publishing domain (bibliographic description of documents, characterisation of citations, person's roles, etc.) should be covered by separate yet interoperable ontologies;
- to permit maximum reusability of each ontology module, logical constraints, for example domain and range constraints on properties, should be added only where they are strictly required;

¹All these principles are derived from my personal experience in developing ontologies for a specific domain (i.e., publishing) and for specific end-users (primarily, publishers and authors).

- where well-known and widely shared vocabularies covering parts of the domain already existed, these should be properly imported and re-used;
- alongside the development of the ontologies themselves, it was important to use or opportunely develop tools that assist people both to understand and to use each ontology with minimum effort, without having to know the specific technical language in which the ontology is implemented.

In this chapter, I describe the principles and architecture of eight ontologies central to the task of semantic publishing: **SPAR**, the *Semantic Publishing and Referencing Ontologies*², a suite of orthogonal and complementary OWL 2 DL ontology modules. They together permit the creation of comprehensive machine-readable RDF metadata for all aspects of semantic publishing and referencing: documents description, types of citations and their related contexts, bibliographic references, document parts and status, agents' roles and workflow processes, etc.

The main characteristics of SPAR, that mark it out as distinct from previous contributions, are firstly the creation of ontologies of sufficient expressivity to meet the requirements of academic authors and publishers, and secondly the development of accompanying presentation technologies, *LODE* (introduced in Section 5.1) and *Graffoo* (presented in Section 5.3), that enable these ontologies to be readily understood by potential users such as academic researchers, publishers and librarians who, while expert in their own domains, lack skill in ontology modelling and knowledge formalisation.

The starting point for SPAR was version 1.6 of *CiTO*, the *Citation Typing Ontology*, described in [165]. This was both preliminary and incomplete, and yet contained within the single ontology both terms for handling both bibliographic document descriptions and also properties to enable the characterisation of citations, as well as terms to permit recording of the number of citations to a given article, both within the citing paper and globally.

A simple architectural diagram of the eight SPAR ontologies is shown in Fig. 4.1 on the facing page. As the diagram indicates, the eight principal SPAR ontologies are supported by three other OWL 2 DL ontologies that the SPAR ontologies import as required – *FRBR* in OWL 2 DL, *DEO*, the *Discourse Elements Ontology*³, and the *Error Ontology*⁴. They are also supported by the external *FOAF Essentials*⁵ and *SWAN Collections*⁶ ontologies, by three *Ontology Design Patterns* ontology

²The SPAR (Semantic Publishing and Referencing) Ontologies: <http://purl.org/spar>.

³DEO, the Discourse Elements Ontology: <http://purl.org/spar/deo>.

⁴The Error Ontology: <http://www.essepuntato.it/2009/10/error>.

⁵FOAF essentials in OWL: <http://purl.org/swan/2.0/foaf-essential>.

⁶CO, the Collections Ontology: <http://swan.mindinformatics.org/ontologies/1.2/collections.owl>.

modules (*Time-indexed situation*⁷, *Sequence*⁸, *Participation*⁹), and by the *Patterns Ontology*¹⁰ for document structures.

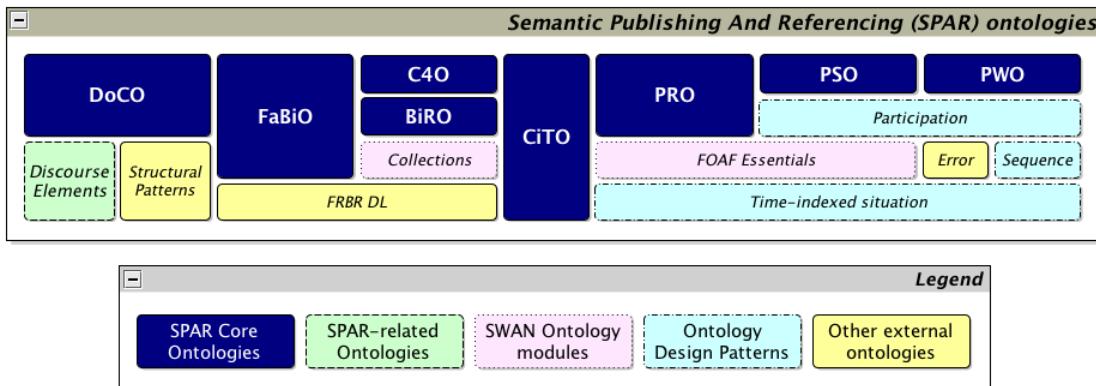


Figure 4.1: A simple architectural diagram showing the interactions and dependencies between the component ontologies of SPAR.

The characteristics and benefits of all the SPAR ontologies are outlined in the following sections, to provide a comprehensive picture of the scope of SPAR. Where appropriate, I also will show how to integrate SPAR semantic data with documents defined through EARMARK (introduced in Section 3.1).

4.1 Representing bibliographic information using FaBiO

The existing well-known and commonly used vocabularies described in Section 2.2 are either poor in concepts or are flat, preventing their use for accurately describing publishing reality, I will illustrate this by considering the representation of a typical bibliographic reference first using Dublin Core, then BIBO and finally FRBR. I will then show how this information can be accurately described using FaBiO, which incorporates elements of all three of these vocabularies. Consider the following typical bibliographic reference describing [121]:

Yves Marcoux, Elias Rizkallah (2009). Intertextual semantics: A semantics for information design. *Journal of the American Society for Information Science and Technology*, 60 (9): 1895-1906. September 2009.

⁷ *Time-indexed situation* pattern: <http://www.ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl>.

⁸ *Sequence* pattern: <http://www.ontologydesignpatterns.org/cp/owl/sequence.owl>.

⁹ *Participation* pattern: <http://www.ontologydesignpatterns.org/cp/owl/participation.owl>.

¹⁰ The Patterns Ontology: <http://www.essepuntato.it/2008/12/pattern>.

John Wiley & Sons, Inc. DOI: 10.1002/asi.21134. First published online (HTML and PDF) 21 August 2009.

From the previous description we can extract the following information:

1. the document is an academic research article – deducible from the journal in which it is published;
2. Yves Marcoux and Elias Rizkallah are the authors of the article;
3. the article was published in 2009;
4. the article is entitled “Intertextual semantics: A semantics for information design”;
5. it was published in the 9th Issue of the 60th volume of the *Journal of the American Society for Information Science and Technology*;
6. the DOI of the article is “10.1002/asi.21134”;
7. the article was first published online on the 21st of August, 2009, in two different formats (HTML and PDF);
8. the journal issue within which the printed version of the article was published bears the publication date September 2009;
9. the page range of the article within the printed version is “1895-1906”;
10. the publisher of the journal is John Wiley & Sons, Inc.

4.1.1 Bibliographic reference metadata encoding using DC Terms

In the following RDF encoding example¹¹, we attempt to describe all these facts using only terms from the DC Terms vocabulary [63]:

```

@prefix : <http://www.example.com/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix text: <http://purl.org/NET/mediatypes/text/> .
@prefix application: <http://purl.org/NET/mediatypes/
application/> .

:intertextual-semantics a dcterms:BibliographicResource

```

¹¹This and the following RDF encodings are written in Turtle [147].

```

; dcterms:creator :marcoux , :rizkallah
; dcterms:title "Intertextual semantics: A semantics for
  information design"
; dcterms:issued "2009"^^xsd:gYear
; dcterms:issued "09-2009"^^xsd:gYearMonth
; dcterms:identifier "doi:10.1002/asi.21134"
; dcterms:extent [ a dcterms:SizeOrDuration
  ; dcterms:description "1895-1906" ]
; dcterms:hasFormat :html , :pdf
; dcterms:isPartOf [ dcterms:identifier "9"
  ; dcterms:description "Issue"
; dcterms:isPartOf [ dcterms:identifier "60"
  ; dcterms:description "Volume"
; dcterms:isPartOf [ dcterms:title "Journal of the
  American Society for Information Science and
  Technology"
; dcterms:format text:html
; dcterms:publisher :wiley-and-sons ] ] ] .

:html a dcterms:BibliographicResource
; dcterms:issued "21-08-2009"^^xsd:date .

:pdf a dcterms:BibliographicResource
; dcterms:format application:pdf
; dcterms:issued "21-08-2009"^^xsd:date .

:marcoux a dcterms:Agent
; dcterms:description "Marcoux Yves" .

:rizkallah a dcterms:Agent
; dcterms:description "Rizkallah Elias" .

:wiley-and-sons a dcterms:Agent
; dcterms:description "John Wiley & Sons, Inc." .

```

There are some obscure points that emerge from the preceding formalisation:

- There is *no clear characterisation* of the entities involved. We are able to speak about a general “bibliographic resource” (*dcterms:BibliographicResource*) and an “agent” (*dcterms:Agent*), but not about a journal article, a journal, a volume, or an issue of a journal, nor about persons, authors, etc.
- Some of the statements are *too generic*. E.g., the property *dcterms:issued* that is used to represent the various dates associated with the publication of this article, is itself employed in conjunction with three different date for-

mats, i.e. "21-08-2009"^^ xsd:date, "09-2009"^^ xsd:gYearMonth, and "2009"^^ xsd:gYear.

- Some of the statements *hide the semantics within the textual content* of the statement. E.g., the statement *dcterms:identifier* “*doi:10.1002/asi.21134*” implicitly says that the character string “*10.1002/asi.21134*” is a Digital Object Identifier, i.e. a special type of identifier used to identify journal articles. Similarly “*1895-1906*” implicitly says that the *printed version (only)* of the article goes from starting page “*1895*” to ending page “*1906*”. While these implied facts are understandable to human readers, they are not available to computational agents processing such metadata.
- The relations between the various formats of the article are not clear. For example, the manner in which the resource “*:intertextual-semantics*” relates to the resources “*:html*” and “*:pdf*” is not specified. Do the latter represent the content of the former in different formats, or there is something more?

4.1.2 Bibliographic reference metadata encoding using BIBO

Some of these points are addressed by BIBO [42]. BIBO is the first OWL ontology specifically designed to address the domain under discussion, and expands the DC Terms vocabulary with terms specific for bibliographic metadata, particularly relating to legal documents, and for various types of event. It also includes PRISM [87] and FOAF [25] terms.

In the following RDF encoding example, the information given in the bibliographic reference cited above is encoded using BIBO:

```

@prefix bibo: <http://purl.org/ontology/bibo/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:intertextual-semantics a bibo:AcademicArticle
; bibo:authorList ( :marcoux :rizkallah )
; dcterms:title "Intertextual semantics: A semantics for
    information design"
; dcterms:issued "2009"^^xsd:gYear
; dcterms:issued "09-2009"^^xsd:gYearMonth
; bibo:doi "10.1002/asi.21134"
; bibo:pageStart "1895"
; bibo:pageEnd "1906"
; dcterms:hasFormat :html , :pdf
; dcterms:isPartOf [ a bibo:Issue
; bibo:issue "9"
; bibo:volume "60"
; dcterms:isPartOf [ a bibo:Journal

```

```

; dcterms:title "Journal of the American Society for
    Information Science and Technology"
; dcterms:publisher :wiley-and-sons ] ] .

:html a bibo:AcademicArticle
; dcterms:format text:html
; dcterms:issued "21-08-2009"^^xsd:date .

:pdf a bibo:AcademicArticle
; dcterms:format application:pdf
; dcterms:issued "21-08-2009"^^xsd:date .

:marcoux a foaf:Person
; foaf:givenName "Yves"
; foaf:familyName "Marcoux" .

:rizkallah a foaf:Person
; foaf:givenName "Elias"
; foaf:familyName "Rizkallah" .

:wiley-and-sons a foaf:Organization
; foaf:name "John Wiley & Sons, Inc." .

```

As this example shows, BIBO resolves many of the semantic ambiguities present in the DC version – the DOI is specifiable through the specific data property *bibo:doi*, the article is identified as a *bibo:AcademicArticle*, the authors and the publisher are respectively *foaf:Persons* and *foaf:Organization*, etc. However, other ambiguities are still unresolved. The relations between the various formats are still not clear, and the date properties continue to be too generic. In addition, new issues emerge:

- BIBO specifies that the property for listing authors (*bibo:authorList*) must have, as its range, either an *rdf>List* or an *rdf:Seq*. Since these RDF classes are not supported by OWL 2, this has the disadvantage of making that model non-compliant with the decidable and computable OWL 2 DL, thus preventing OWL 2 DL reasoners from inferring new axioms from a current knowledge base encoded using BIBO¹².
- BIBO can record a volume number through the data property *bibo:volume*, but, although it has the classes *bibo:AcademicArticle*, *bibo:Issue* and *bibo:Journal*, it lacks the concept of “Volume” as a distinct class among other bibliographic classes that have a hierarchical partitive relationship to one another (i.e. Journal Article > Issue > Volume > Journal).

¹²For a longer and clearer justification of why RDF collections and containers are not usable and interpreted correctly by OWL 2 DL, please consult <http://hcklab.blogspot.com/2008/12/moving-towards-swan-collections.html>.

- Furthermore, because it lacks the layered structure of FRBR, it does not have the flexibility to distinguish between concepts at these various levels, for example an academic paper (a FRBR Work) and the various possible Expressions of that paper as a journal article, a conference paper or a book chapter. The class *bibo:AcademicArticle* is in fact a conflation of the concepts “academic paper” and “journal article”.

4.1.3 Bibliographic reference metadata encoding using FRBR

It is possible to handle the third of the issues raised above by adopting the more structured FRBR model [100], as expressed in the FRBR Core ontology, together with DC terms for textual statements (i.e. those statements having a literal string as their object). This is illustrated in the following example:

```
@prefix frbr: <http://purl.org/vocab/frbr/core#> .

:intertextual-semantics a frbr:Work
; frbr:creator :marcoux , :rizzallah
; dcterms:title "Intertextual semantics: A semantics for
    information design"
; frbr:realization :version-of-record .

:version-of-record a frbr:Expression
; dcterms:issued "2009"^^xsd:gYear
; dcterms:identifier "doi:10.1002/asi.21134"
; frbr:embodiment :printed , :html , :pdf
; frbr:partOf [ a frbr:Expression
    ; dcterms:identifier "9"
    ; dcterms:description "Issue"
    ; frbr:embodiment :printed-issue
    ; frbr:partOf [ a frbr:Expression
        ; dcterms:identifier "60"
        ; dcterms:description "Volume"
        frbr:partOf [ a frbr:Expression
            ; dcterms:title "Journal of the American Society for
                Information Science and Technology" ] ] ] .

:printed-issue a frbr:Manifestation
; frbr:producer :wiley-and-sons
; dcterms:issued "09-2009"^^xsd:gYearMonth
; frbr:part :printed .

:printed a frbr:Manifestation
; frbr:producer :wiley-and-sons
; dcterms:issued "09-2009"^^xsd:gYearMonth
```

```

; dcterms:extent [ a dcterms:SizeOrDuration
; dcterms:description "1895-1906" ] .

:html a frbr:Manifestation
; frbr:producer :wiley-and-sons
; dcterms:format text:html
; dcterms:issued "21-08-2009"^^xsd:date .

:pdf a frbr:Manifestation
; frbr:producer :wiley-and-sons
; dcterms:format application:pdf
; dcterms:issued "21-08-2009"^^xsd:date .

:marcoux a frbr:Person
; dcterms:description "Yves Marcoux" .

:rizkallah a frbr:Person
; dcterms:description "Elias Rizkallah" .

:wiley-and-sons a frbr:CorporateBody
; dcterms:description "John Wiley & Sons, Inc." .

```

Although it is possible to use FRBR in this manner to give a structured and unambiguous description of all the bibliographic entities, this example makes clear the severe limitations of FRBR, due to the lack of terms in the FRBR Core ontology to permit publications to be described in normal everyday language.

4.1.4 Bibliographic reference metadata encoding using FaBiO

FaBiO, the *FRBR-aligned Bibliographic Ontology*, was developed precisely to address all the issues revealed by the previous examples, while re-using the previous fundamental work in this domain (so as not to re-invent the wheel). In particular, DC Terms, PRISM, FRBR and SKOS terms that are all included in FaBiO.

Considering again the previous bibliographic reference example, a possible FaBiO formalisation is:

```

@prefix fabio: <http://purl.org/spar/fabio> .
@prefix prism: <http://prismstandard.org/namespaces/basic
/2.0/> .
@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 dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

```

```

:intertextual-semantics a fabio:ResearchPaper
; dcterms:creator :marcoux , :rizkallah
; dcterms:title "Intertextual semantics: A semantics for
    information design"
; frbr:realization :version-of-record .

:version-of-record a fabio:JournalArticle
; fabio:hasPublicationYear "2009"^^xsd:gYear
; prism:doi "10.1002/asi.21134"
; frbr:embodiment :printed , :html , :pdf
; frbr:partOf [ a fabio:JournalIssue
    ; prism:issueIdentifier "9"
    ; frbr:embodiment :printed-issue
    ; frbr:partOf [ a fabio:JournalVolume
        ; prism:volume "60"
        frbr:partOf [ a fabio:Journal
            ; dcterms:title "Journal of the American Society for
                Information Science and Technology" ] ] ] .

:printed-issue a fabio:Paperback
; dcterms:publisher :wiley-and-sons
; prism:publicationDate "09-2009"^^xsd:gYearMonth
; frbr:part :printed .

:printed a fabio:PrintObject
; dcterms:publisher :wiley-and-sons
; prism:publicationDate "09-2009"^^xsd:gYearMonth
; prism:startingPage "1895"
; prism:endingPage "1906" .

:html a fabio:Web page
; dcterms:publisher :wiley-and-sons
; dcterms:format text:html
; prism:publicationDate "21-08-2009"^^xsd:date .

:pdf a fabio:DigitalManifestation
; dcterms:publisher :wiley-and-sons
; dcterms:format application:pdf
; prism:publicationDate "21-08-2009"^^xsd:date .

:marcoux a foaf:Person
; foaf:givenName "Yves"
; foaf:familyName "Marcoux" .

:rizkallah a foaf:Person

```

```

; foaf:givenName "Elias"
; foaf:familyName "Rizkallah" .

:wiley-and-sons a foaf:Organization
; foaf:name "John Wiley & Sons, Inc." .

```

With FaBiO, it thus becomes possible:

- To write semantic descriptions of a wide variety of bibliographic objects, including research articles, journal articles and journal volumes, using terms that closely resemble the language used in everyday speech by academics and publishers¹³;
- To employ FRBR categories to define clear separations between each part of the publishing process, involving different people (authors, publishers, readers) depending on which aspect of the bibliographic entity we are considering: the high-level conceptualisation of the research paper, the version of record of that paper forming a journal article, the publication of that article in various formats, and the individual physical or electronic exemplars of the published article that people may read and own.
- To include with ease elements from other vocabularies for describing particular entities involved in a publishing process that are not specified by FaBiO itself, such as those from FOAF for persons and organizations.

Other advantages of FaBiO are outlined in the following sections.

Using external models

As already mentioned, FaBiO was developed with the minimum of restrictions to its classes and to the domains and ranges of its properties. This flexibility has the great advantage of allowing FaBiO to be used together with other models for describing scenarios that may be needed in some situations. We have already seen how FOAF can be used to describe agents. Another common requirement is to specify the order of components in a list, for example authors in an author list. Unlike the use of *bibo:authorList*, which breaks compliance as explained above, this can be achieved in a manner that is compliant with the decidable and computable OWL 2 DL by combining FaBiO with the Collections Ontology (CO)¹⁴, an OWL 2 DL ontology specifically designed for defining orders among items, in the following way:

¹³This has come about through many meetings with different academics and publishers that we have undertaken in order to understand their working practices and requirements.

¹⁴CO, the Collections Ontology: <http://purl.org/co>.

```

@prefix co: <http://purl.org/co/> .

:intertextual-semantics a fabio:ResearchPaper
; dcterms:creator :listOfAuthors .

:listOfAuthors a co>List
; co:firstItem [ co:itemContent :marcoux
; co:nextItem [ co:itemContent :rizzallah ] ] .

:marcoux a foaf:Person
; foaf:givenName "Yves"
; foaf:familyName "Marcoux" .

:rizzallah a foaf:Person
; foaf:givenName "Elias"
; foaf:familyName "Rizzallah" .

```

In this way we can still keep the model in OWL 2 DL. Additionally, because the ranges of *dcterms:creator* and other properties within FaBiO have intentionally been left unspecified, FaBiO guarantees a level of interoperation with other models without incurring in any undesirable collateral effects, such as ontology inconsistencies or the generation of undesired inferences.

Extending FRBR within FaBiO

One of the explicit requests from publishers and end-users was to be able to create shortcuts between FRBR endeavours (work, expression, manifestation, item) that were not part of the original FRBR model. Let me introduce an example to illustrate this needs, by slightly changing the bibliographic reference we introduce previously:

Yves Marcoux, Elias Rizzallah (2009). Intertextual semantics: A semantics for information design. <http://onlinelibrary.wiley.com/doi/10.1002/asi.21134/full>.

In this reference, we have just a FRBR work – the paper by Marcoux and Rizzallah – and the URL for a specific FRBR item that portrays that work – the HTML version of the paper on the publishers’ website. If I wished to link these concepts using the FRBR OWL ontology terms I have employed so far, I would be obliged to specify each intermediate FRBR endeavour, namely the expression and manifestation of that paper, even if we were not interested in doing that:

```

@prefix wiley: <http://onlinelibrary.wiley.com/doi/> .

:intertextual-semantics a frbr:Work
; frbr:creator :marcoux , :rizzallah

```

```

; dcterms:title "Intertextual semantics: A semantics for
    information design"
; frbr:realization [ a frbr:Expression
    ; frbr:embodiment [ a frbr:Manifestation
        ; frbr:exemplar wiley:10.1002/asi.21134/full ] ] .

```

In order to avoid this kind of verbosity, it is possible to use the new FaBiO properties shown in Fig. 4.2 on the next page¹⁵ to link a work directly to its manifestations or to its items (*fabio:hasPortrayal*), or to link an expression directly to its items (*fabio:hasRepresentation*).

Obviously, these added properties allows us to treat even the previous case quite easily and in a less verbose way:

```
@prefix wiley: <http://onlinelibrary.wiley.com/doi/> .
```

```

:intertextual-semantics a frbr:Work
; frbr:creator :marcoux , :rizkallah
; dcterms:title "Intertextual semantics: A semantics for
    information design"
; fabio:hasPortrayal wiley:10.1002/asi.21134/full .

```

Categorising bibliographic resources with SKOS

One of the most important needs for a publisher is to categorising each bibliographic entity it produces by adding free-text keywords and/or specific terms structured according to recognised classification systems and/or thesauri developed for certain academic disciplines. While through FaBiO the definition of keywords is possible using the PRISM property *prism:keyword*, terms from thesauri, structured vocabularies and classification systems are described using SKOS [123].

To facilitate this, FaBiO extends some classes and properties of SKOS as shown in Fig. 4.3 on page 103.

As shown, any FRBR endeavour can be associated (*fabio:hasSubjectTerm*) with one or more descriptive terms (*fabio:SubjectTerm*, a sub-class of *skos:Concept*) found in a specific dictionary (*fabio:TermDictionary*, a sub-class of *skos:ConceptScheme*) that is relevant to (*fabio:hasDiscipline*) particular disciplines (*fabio:SubjectDiscipline*, also a sub-class of *skos:Concept*) describing a field of knowledge or human activity such as Computer Science, Biology, Economics, Cookery or Swimming. At the same time, the subject disciplines can be grouped by an opportune vocabulary (i.e., *fabio:DisciplineDictionary*).

For instance, the previous example can be enriched in the following way:

¹⁵This and the following diagrams comply with the *Graphic framework for OWL ontologies (Graffoo)*, introduced in Section 5.3. A legend for all Graffoo diagrams can be found in Fig. 5.13 on page 177.

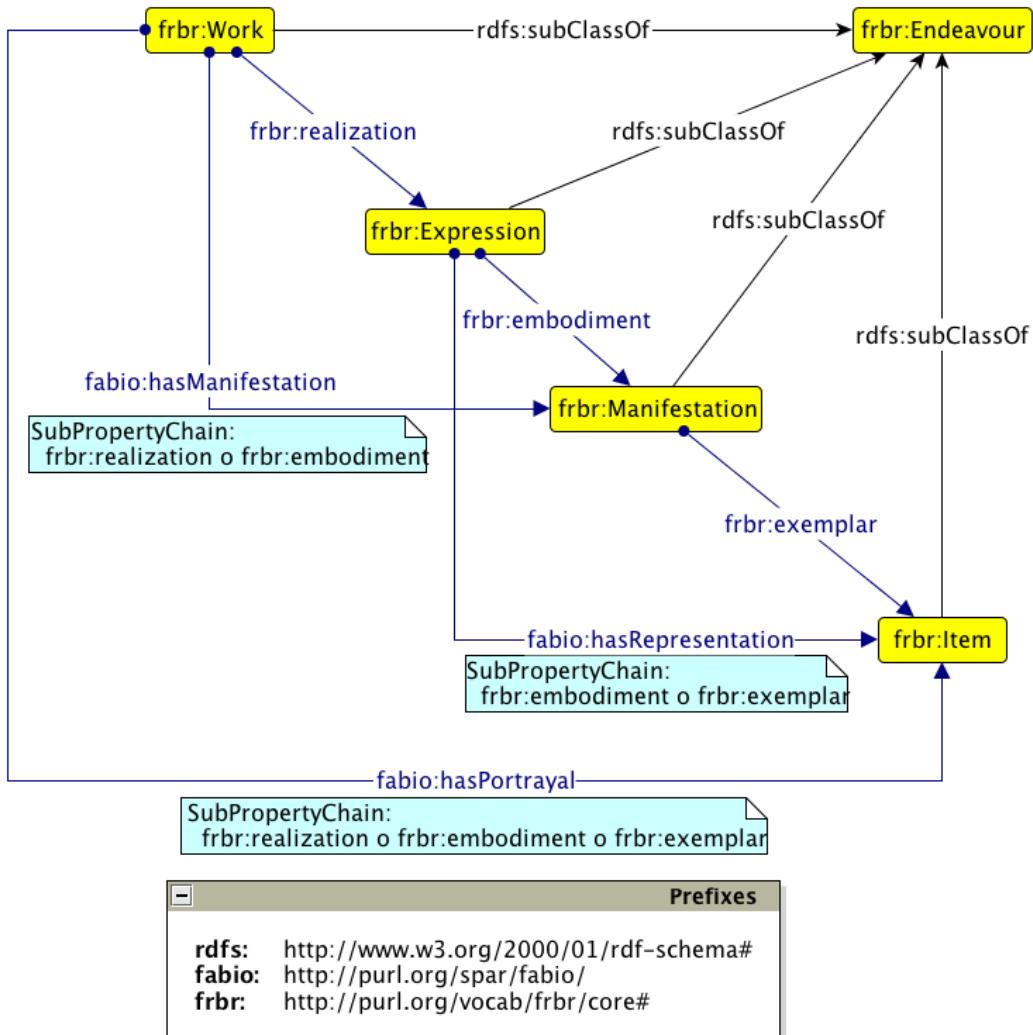


Figure 4.2: The main FRBR object properties relating FRBR endeavours (work, expression, manifestation, item), and the related new object properties introduced by FaBiO (fabio:hasManifestation, fabio:hasRepresentation, fabio:hasPortrayal) to provide shortcuts between Work and Manifestation, Work and Item, and Expression and Item, respectively.

```

@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix acm: <http://www.acm.org/class/1998/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .

:intertextual-semantics a fabio:ResearchPaper
; fabio:hasSubjectTerm acm:markup-languages , acm:semantics
; prism:keywords "semantics of markup" , "semiotic"

```

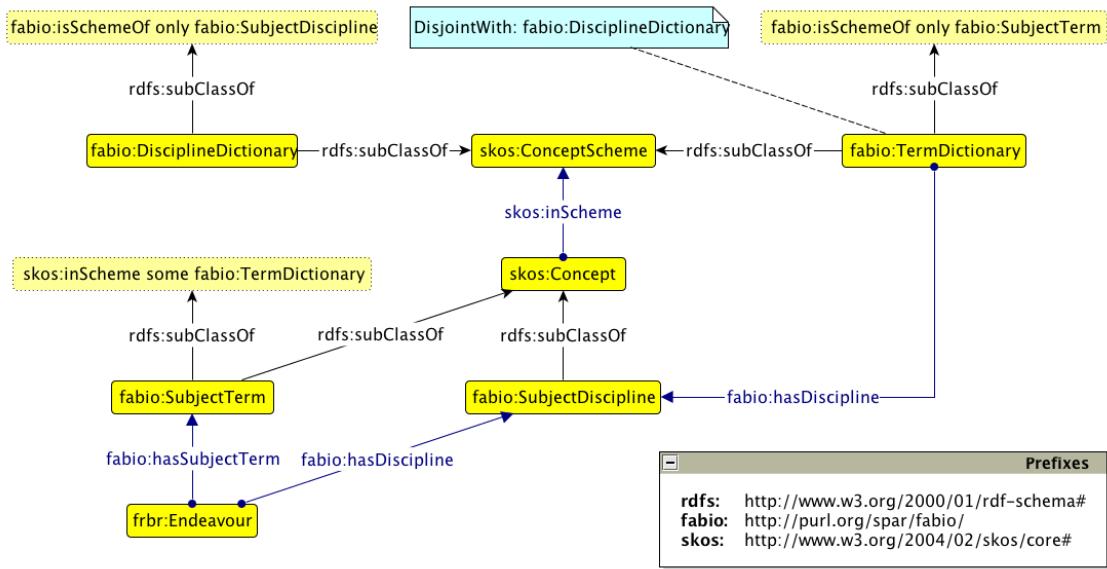


Figure 4.3: The extension to the common SKOS classes and relations implemented in FaBiO.

```
application" , "xml" .  
  
<http://www.acm.org/class/1998> a fabio:TermDictionary  
; skos:prefLabel "The 1998 ACM Computing Classification  
System"  
; fabio:hasDiscipline dbpedia:Computer_science .  
  
acm:markup-languages a fabio:SubjectTerm  
; skos:prefLabel "Markup languages"  
; skos:inScheme <http://www.acm.org/class/1998>  
; skos:broader acm:document-preparation .  
  
acm:semantics a fabio:SubjectTerm  
; skos:prefLabel "Semantics"  
; skos:inScheme <http://www.acm.org/class/1998>  
; skos:broader acm:formal-definitions-and-theory .  
...  
...
```

4.2 Characterising citations with CiTO

Bibliographic citation, i.e., the act of referring from a citing entity to the cited one, is one of the most important activities of an author in the production of any bibliographic work, since the acknowledgement of sources that this activity represents

stands at the very core of the scholarly enterprise. The network of citations created by combining citation information from many academic articles and books is a source of rich information for scholars, and can be used by publisher to create new and interesting ways of browsing their data, as well as for calculating metrics reflecting the relative importance of a journal (e.g. the *impact factor*) or an author (e.g. the *h-index*).

The reasons that an author cites other publications are varied. Usually, it is because the author has gained assistance of some sort, perhaps in the form of background information, ideas, methods or data, from the previously published cited works, and wishes to acknowledge this. More rarely, citations may be made to review, critique or refute previous works. Most citations are direct and explicit (as in the reference list of a journal article). However, they can also be indirect (for example, by means of a citation to a more recent paper by the same research group on the same topic), or implicit (as in artistic quotations or parodies, or *in extremis* in cases of plagiarism).

Classical scholarship has well-developed methods for citing individual sections, paragraphs or verses of referenced works. In addition, it was not uncommon for the citing author to reproduce entire sections of the cited work in his own document, so that the reader could understand exactly the relationship of the earlier document to the present one, since the author could not be sure the reader would have ready access to the works of the cited authority (for example, Aristotle). In contrast, modern scientific practice takes the other extreme – a citation is made to the previously published paper as a whole, with little or no indication given as to why this paper has been cited or what portions of it are relevant to the discussion in hand, except what the reader can glean from the citation context.

Of course, previously developed models for describing bibliographic objects allow for the existence of citations among bibliographic entities to be recorded. For instance, taking again the previous example about the article “Intertextual semantics: A semantics for information design”, using BIBO [42] it is possible to declare citations as follows:

```
@prefix bibo: <http://purl.org/ontology/bibo/> .

:intertextual-semantics bibo:cites :towards-a-semantics
, :meaning-and-interpretation
, :design-everyday-things
, :exploring-intertextual-semantics ...
```

Alternatively, it is possible to use the Discourse Relationships Module¹⁶ in SWAN v1.2 [35] in the same way:

¹⁶The Discourse Relationships Ontology: http://swan.mindinformatics.org/spec/1.2/discourse_relationships.html.

```
@prefix disrel: <http://swan.mindinformatics.org/ontologies
/1.2/discourse-relationships/> .

:intertextual-semantics disrel:cites :towards-a-semantics
, :meaning-and-interpretation
, :design-everyday-things
, :exploring-intertextual-semantics ...
```

However, while the *cites* properties in these two ontologies, as well as the more generic property *dcterms:relation* in DC Terms, permit the bald **existence** of a citation to be recorded, they do not permit the author to invest the citation with any specific factual and/or rhetorical meanings that would describe the **reasons** the author had in mind when creating citations to some particular documents rather than to others.

CiTO, the *Citation Typing Ontology*¹⁷ version 2.0, seeks to improve upon this situation by making it possible for authors (or others) to capture their intent when citing a particular publication, permitting them to create metadata describing their citations, quite distinct from metadata describing the cited works themselves. CiTO thus permits the motivations of an author when referring to another document to be captured. For instance, the previous example may be rewritten using CiTO as follows:

```
@prefix cito: <http://purl.org/spar/cito/> .

:intertextual-semantics
cito:disputes :towards-a-semantics
; cito:citesAsRelated :meaning-and-interpretation
; cito:agreesWith :design-everyday-things
; cito:extends :exploring-intertextual-semantics ...
```

The current version of CiTO, version 2.0, contains and extends the citation-specific object properties that were originally contained in CiTO version 1.6 [165], to the exclusion of all other original classes and properties within CiTO v1.6, which, as part of the modularization we have undertaken, have been moved into FaBiO (Section 4.1) or into C4O (Section 4.3.2) and PSO (Section 4.5.7).

CiTO now contains just two main object properties, *cito:cites* and its inverse *cito:isCitedBy*, each of which as thirty-two sub-properties, plus one additional generic object property, *cito:shareAuthorsWith*, that may be used even outside a citation act.

As shown in Fig. 4.4 on the following page, all these properties (and, consequently, their inverses) may be classified as rhetorical and/or factual, with the rhetorical properties being grouped in three sets depending on their connotation: *positive*, *informative* (or *neutral*) or *negative*.

When developing CiTO v2.0 from CiTO v1.6, we intentionally removed the domain and range constraints from its object properties, so that this ontology could be

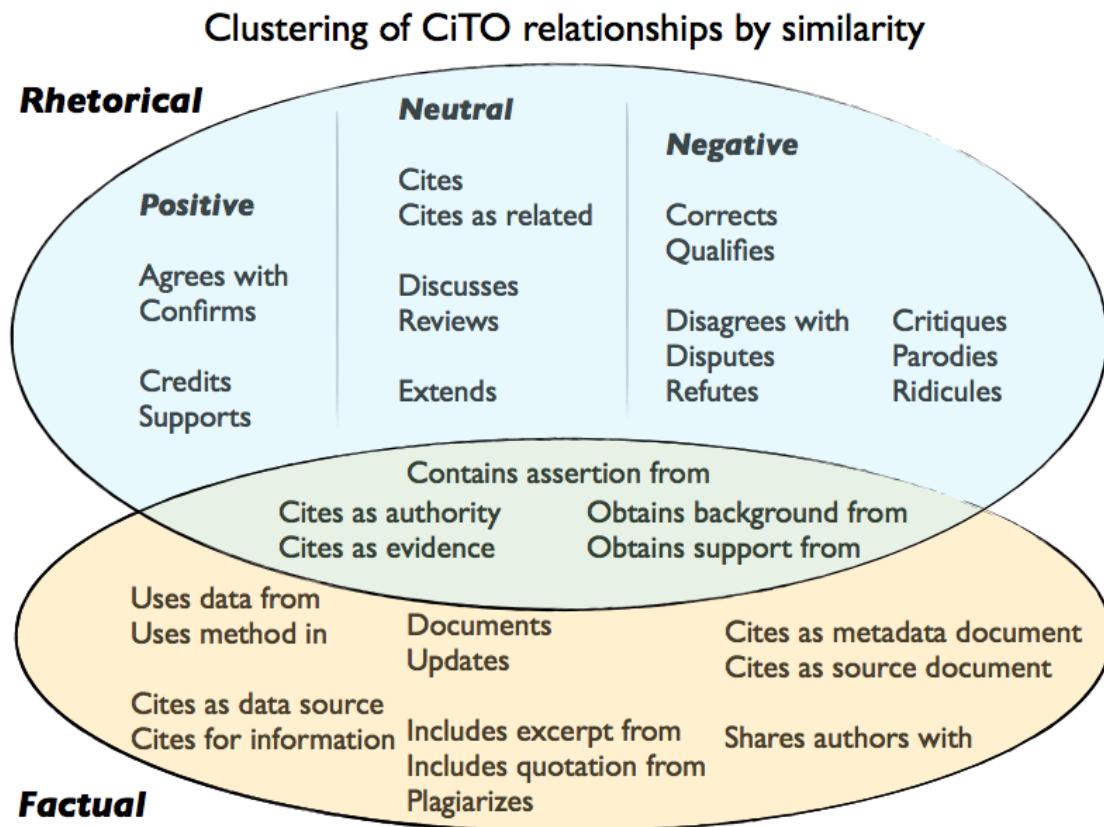


Figure 4.4: The diagram shows the CiTO v 2.0 object properties grouped in terms of their characterisation as rhetorical and/or factual, and, for the former, in terms of their connotation (positive, neutral or negative). All the properties shown, except *cito:cites* and *cito:sharesAuthorsWith*, are sub-properties of *cito:cites* itself. The inverse property of *cito:cites*, namely *cito:isCitedBy*, and its inverse sub-properties, are not shown.

easily integrated with other models. Obviously, it can be successfully used in conjunction with FaBiO, so that descriptions of a bibliographic entity and its citations can be mixed within a single RDF graph:

```
:intertextual-semantics a fabio:ResearchPaper
; dcterms:creator :marcoux , :rizkallah
; dcterms:title "Intertextual semantics: A semantics for
    information design"
; cito:disputes :towards-a-semantics ...

:towards-a-semantics a fabio:ResearchPaper
; dcterms:creator :renear , :dubin , :sperberg-mcqueen
; dcterms:title "Towards a semantics for XML markup"
```

```
; cito:isDisputedBy :intertextual-semantics ...
```

4.3 Documents and their bibliographic references

The word “citation” is often subject of misinterpretations and misuses. The main problem is that it is used to identify objects having different purposes, at least in scientific literature. For instance, we often identify as “citation” the *act of citing* another work, a *bibliographic reference* put at the end of a paper (usually in a list), as well as particular *pointers* (e.g., “[3]”) denoting that bibliographic reference.

In order to expand more on this topic, let me take into account the following text from the article “Intertextual semantics: A semantics for information design” [121] used in the previous examples:

Related Works

...

Renear, Dubin, and Sperberg-McQueen (2002, pp. 121–122) proposed a formal semantic approach for structured documents. The basic premise is that natural-language descriptions are insufficient and must be supplemented by a separate, formal apparatus. Our approach (IS) does not share that premise; in a way, we aim at operationalizing natural-language descriptions in such a way that they can support document creation and other operations on documents. We do not replace natural language; we frame it with mechanisms that make it supportive of interactions between humans and documents.

...

References

...

Renear, A., Dubin, D., & Sperberg-McQueen, C.M. (2002). Towards a semantics for XML markup. In E. Munson (Chair), Proceedings of the ACM Symposium on Document Engineering, (pp. 119–126). New York: ACM Press.

The above excerpt contains a particular paragraph from the section “Related Works” of the paper and a list item from the final “References” section. We can clearly identify six different kinds of objects that relate to the introduced citation, all of them having different purposes:

1. the *citing article*, i.e. the article that contains such a text;
2. the *cited article*, i.e. the article that is got involved by the text;
3. the *in-text reference pointer* referring to a particular bibliographic reference (usually contained in the section “References”), e.g. the text “Renear, Dubin, and Sperberg-McQueen (2002, pp. 121-122)”. In scientific literature it can be presented in other forms – e.g., as an in-square-brackets number (i.e., “[3]”), as an in-square-brackets string with the first letters of each (at most three) authors’ surname plus the last two digits of the publication year (i.e., “[RDS02]”), or as an in-round-brackets string with the first author’s surname followed by “et al.” and the publication year (i.e., “(Renear et al., 2002)”);
4. the *citation context*, the point of the paper (sentence, paragraph, section, chapter, etc.) in which the in-text reference pointer appears;
5. the *bibliographic reference*, e.g. the list item at the end of the above excerpt that briefly summarises some metadata of a particular paper. It is explicitly denoted (by an in-text reference pointer) somewhere in the text;
6. the *act of citing*, i.e. a statement that connects two different articles (more precisely, the *citing document* and the *cited document*) for a particular reason, as described in Section 4.2.

Ontologies that want to describe such elements should be provided with appropriate entities (classes and properties) in order not to allow (or at least decrease) the degree of ambiguity when modelling citing acts in documents.

Having a clear and unambiguous description of elements taking part in citations is particularly relevant for those applications that extract citation contexts in an automatic and semi-automatic ways. For instance, the *Citation-Sensitive In-Browser Summariser (CSIBS)* [195] is a tool for presenting a preview of possible excerpts from the cited document that are relevant to a particular in-text reference pointer in the citing document. One of its promised future works is to enable RDF-based descriptions of these elements.

In Section 4.2 I introduced the SPAR ontology for the description of factual and rhetorical aspects of citations. In this section I present two models focussed on the remaining aspects of citing acts: the *Bibliographic Reference Ontology (BiRO)* and the *Citation Counting and Context Characterisation Ontology (C4O)*. They are two SPAR ontologies developed for describing bibliographic lists, bibliographic references, in-text reference pointers, citation contexts and mechanism for counting locally (within an article) or globally (by means of particular platforms, such as Google Scholar¹⁸) document citations.

¹⁸Google Scholar: <http://scholar.google.com>.

4.3.1 Describing the bibliographic reference lists of articles with BiRO

According to [164], the sixth rule that digital publishers should follow to participate actively to Semantic Publishing is to make available at least reference lists of articles in a machine-readable form. In principle, reference lists are the platform from where we should start for building citation networks. In order to reach that goal, ontologies modelling article references and reference lists are needed. Besides offering flexible mechanisms for the description of references, these ontologies should also allow one to link the reference to the particular semantic representation of the document it *references*.

This is an important point to understand: the reference in the reference list of a particular article **is not** the cited document. Rather, it is a compact description considered (usually) sufficient to make readers aware of what document is cited. Therefore, having references expressed in a machine-readable form should allow machines to make the inference step, i.e. to link automatically the article containing the reference (i.e., the citing document) to the article referenced by the reference itself (i.e., the cited document).

I developed the *Bibliographic Reference Ontology*¹⁹ (*BiRO*) so as to offer a standard model for the description of reference lists and references according to (machine-readable) Semantic Web standards. In particular, BiRO is an ontology structured according to the FRBR model Section 2.2.5 to define bibliographic records (as subclasses of FRBR Work) in relation to bibliographic references (as subclasses of FRBR Expression), and their compilation into bibliographic collections and ordered bibliographic lists, respectively (as shown in Fig. 4.5 on the next page).

An individual bibliographic reference, such as one in the reference list of a published journal article, may exhibit varying degrees of incompleteness, depending on the formatting rules of the journal. For example, it may lack the title of the cited article, the full names of the listed authors, or indeed a full listing of all the authors. It will also lack other information that one would hope to find in the complete bibliographic description for that article.

BiRO provides a logical system for relating such an incomplete bibliographic reference:

- to the full bibliographic record for that cited article, which, in addition to any author and title fields missing from the reference, may also be expected to include the name of the publisher, and the ISSN or ISBN of the publication;
- to collections of bibliographic records, such as library catalogues; and
- to ordered bibliographic lists, such as reference lists.

¹⁹BiRO, the Bibliographic Reference Ontology: <http://purl.org/spar/biro>.

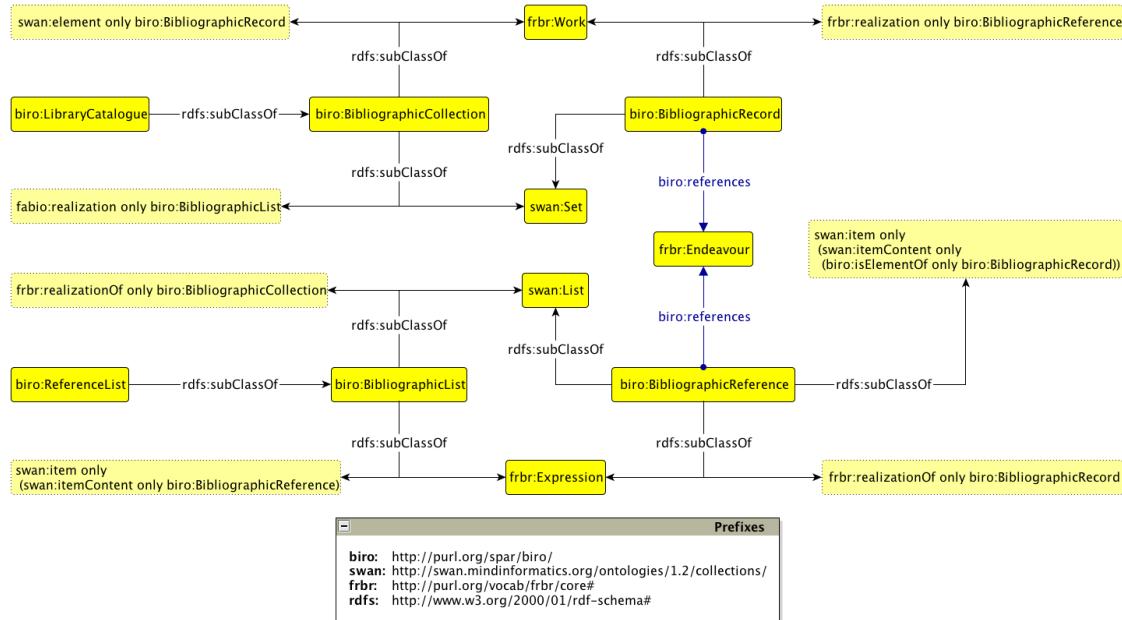


Figure 4.5: Graffoo diagram summarising the *Bibliographic Reference Ontology* (*BiRO*).

In order to understand how to use BiRO to describe reference lists, let me take into account again the above introduced reference referring to [149]:

Renear, A., Dubin, D., & Sperberg-McQueen, C.M. (2002). Towards a semantics for XML markup. In E. Munson (Chair), Proceedings of the ACM Symposium on Document Engineering, (pp. 119–126). New York: ACM Press.

An URI for the reference

A first, very quick, way for defining a simple machine-readable representation of that reference using BiRO is the following one²⁰:

```
:version-of-record frbr:part :reference-list .  
  
:reference-list a biro:ReferenceList  
; swan:firstItem [ swan:itemContent :barwise83  
; swan:nextItem [ swan:itemContent :black37 ...  
; swan:nextItem [
```

²⁰The prefix *swan* refers to entities defined in the old version of the Collection Ontology (SWAN Collection Ontology 1.2), currently imported in BiRO. The SWAN Collection Ontology is available at <http://swan.mindinformatics.org/ontologies/1.2/collections.owl>.

```

swan:itemContent :renear02 ... ] ... ] ] .

:renear02 a biro:BibliographicReference
; biro:references :towards-a-semantics
; dcterms:bibliographicCitation "Renear, A., Dubin, D. &
Sperberg-McQueen, C.M. (2002). Towards a semantics for
XML markup. In E. Mudson (Chair), Proceedings of the ACM
Symposium on Document Engineering, (pp. 119-126). New
York: ACM Press." .

```

Obviously the improvement given by this formal description is not so meaningful. What I did is just to assign an URL to the reference list and to each of its references. The semantics beyond the string representing the reference is still obscure. For instance, at this stage I do not express that the strings “Renear”, “A.”, “2002”, “Towards a semantics for XML markup” are, respectively, the surname of one of the authors, the first letter of his name, the year of publication and the title of the article.

Semantic enhancement of literal elements in references

A way to enable the semantic enhancement of strings is to use literals as subjects of statements and assertions, which are not allowed by Semantic Web standards such as RDF and OWL. Recently, within the Semantic Web community, this topic, i.e. whether and how to allow literals to be subjects of RDF statements²¹, has been actively discussed. This discussion failed to provide a unique and clear indication of how to proceed in that regard.

Although one of the suggestions coming out of the discussion was to explicitly include the proposal in a (future) specification of RDF, this topic is not actually new, particularly in ontology modelling. The need to describe “typical” literals (specially strings) as individuals of a particular class has been addressed by a lot of models in past, such as Common Tag²² (through the class *Tag*), SIOC [20] (through the classes *Category* and *Tag*), SKOS-XL [123] (through the class *Label*), and LMM [143] (through the class *Expression*). After considering the above-mentioned models and other related and inspiring ones, I have developed – in collaboration with Aldo Gangemi and Fabio Vitali – a pattern called *literal reification* to address this issue. It allows one to express literal values as proper ontological individuals so as to use them as subject/object of any assertion within OWL models.

Extending the pattern *region*²³ [71], the pattern *literal reification*²⁴ [74] promotes

²¹Literals as subjects: http://www.w3.org/2001/sw/wiki/Literals_as_Subjects.

²²Common Tag: <http://www.commonTag.org>.

²³Region pattern: <http://ontologydesignpatterns.org/cp/owl/region.owl>. The prefix *region* refers to entities defined in it.

²⁴Literal reification pattern: <http://www.essepuntato.it/2010/06/literalreification>. The prefix *litre* refers to entities defined in it.

any literal as “first class object” in OWL by reifying it as a proper individual of the class *litre:Literal*. Individuals of this class express literal values through the functional data property *litre:hasLiteralValue* and can be connected to other individuals that share the same literal value by using the property *litre:hasSameLiteralValueAs*. Moreover, a literal may refer to, and may be referred by any OWL individual through *litre:isLiteralOf* and *litre:hasLiteral* respectively.

Note that the pattern defines also a SWRL rule [96] that allows one to infer the (not explicitly asserted) literal value of a particular literal individual when it is connected to another literal individual via *litre:hasSameLiteralValueAs*:

```
litre:hasSameLiteralValueAs(x,y) , litre:hasLiteralValue(y,v)
-> litre:hasLiteralValue(x,v)
```

This pattern allows one to use each reified literal as subject or object of any assertion, and it is able to address scenarios described, for example, by the following competency questions:

- What is the context in which entities refer to a particular literal value?
- What is the meaning of a particular value considering the context in which it is used?

Plausible scenarios of its application include:

- modelling domains concerning descriptive tags, in which each tag may have more than one meaning depending on the context in which it is used;
- extending quickly the capabilities of a model by adding the possibility to make assertions on values, previously referred through data properties, without modifying it.

As briefly introduced above and as also shown in Fig. 4.6 on the facing page, the pattern *literal reification* is composed by a class, a data property and three object properties, described as follows:

- class *litre:Literal*. It describes reified literals, where the literal value they represent is specified through the property *litre:hasLiteralValue*. Each individual of this class must always have a specified value;
- data property *litre:hasLiteralValue*. It is used to specify the literal value that an individual of *litre:Literal* represents;
- object property *litre:hasSameLiteralValueAs*. It relates the reified literal to another one that has the same literal value;
- object property *litre:hasLiteral*. It connects individuals of any class to a reified literal;

- object property *litre:isLiteralOf*. It connects the reified literal to the individuals that are using it.

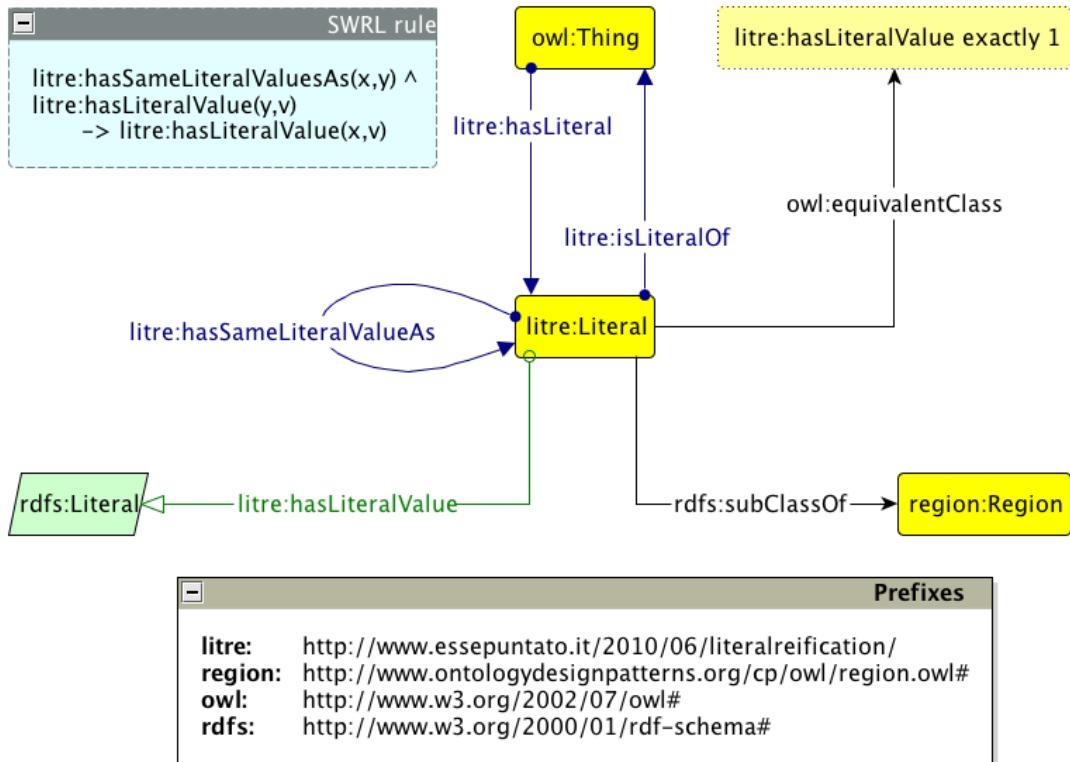


Figure 4.6: Graffoo diagram summarising the *Literal Reification* pattern.

By means of this pattern and of the OWL 2 capabilities in meta-modelling, it becomes possible to link specific strings in the references and to enhance them through semantic assertions according to specific vocabularies, as shown in the following excerpt:

```

:renear02 a biro:BibliographicReference
; biro:references :towards-a-semantics
; swan:firstItem [ swan:itemContent :first-author-name
; swan:nextItem [ swan:itemContent :second-author-name ...
; swan:nextItem [ swan:itemContent :publication-year
; swan:nextItem [ swan:itemContent
:paper-title ... ] ] ] ] .

:first-author-name a litre:Literal , foaf:name
; litre:isLiteralOf :renear # it is the URL identifying the
person

```

```

; litre:hasLiteralValue "Renear, A."^^xsd:string .

:second-author-name ...

:publication-year a litre:Literal , fabio:hasPublicationYear
; litre:isLiteralOf :towards-a-semantics
; litre:hasLiteralValue "2002"^^xsd:gYear .

:paper-title a litre:Literal , dcterms:title
; litre:isLiteralOf :towards-a-semantics
; litre:hasLiteralValue "Towards a semantics for XML markup
."^^xsd:string .

...

```

As shown above, now the bibliographic reference in consideration is described as a list of literals, each of them having a particular semantic connotation.

EARMARK ranges for describing references

Another approach to deal with the semantic enhancement of bibliographic references is to use LA-EARMARK ranges for associating appropriate semantic statements to textual fragments, as illustrated in Section 3.4. For instance, let me consider the article by *Marcoux et al.* “Intertextual semantics: A semantics for information design” [121] implemented as an EARMARK document. In that case, I will have a particular docuverse containing the text of the reference taken into account previously, for example:

```

:renear02-reference a earmark:StringDocuverse
; earmark:hasContent "Renear, A., Dubin, D. & Sperberg-
McQueen, C.M. (2002). Towards a semantics for XML markup
. In E. Mudson (Chair), Proceedings of the ACM Symposium
on Document Engineering, (pp. 119-126). New York: ACM
Press." .

```

From this docuverse, I can define ranges for each string I want to use to describe the bibliographic reference according to BiRO. These ranges, that cover the same literal values used in the example in the previous section, can be defined as follows:

```

:renear02 a biro:BibliographicReference
; biro:references :towards-a-semantics
; swan:firstItem [ swan:itemContent :first-author-surname
; swan:nextItem [ swan:itemContent
:first-author-namefirstletter ...
; swan:nextItem [ swan:itemContent :publication-year
; swan:nextItem [ swan:itemContent

```

```

:paper-title ... ] ] ] .

:first-author-surname a earmark:PointerRange # the string "
Renear"
; earmark:refersTo :renear02-reference
; earmark:begins "0"^^xsd:nonNegativeInteger
; earmark:ends "6"^^xsd:nonNegativeInteger .

:first-author-namefirstletter a earmark:PointerRange # the
string "A"
; earmark:refersTo :renear02-reference
; earmark:begins "8"^^xsd:nonNegativeInteger
; earmark:ends "9"^^xsd:nonNegativeInteger .

...
:publisher a earmark:PointerRange # the string "2002"
; earmark:refersTo :renear02-reference
; earmark:begins "48"^^xsd:nonNegativeInteger
; earmark:ends "52"^^xsd:nonNegativeInteger .

:paper-title a earmark:PointerRange # the string "Towards a
semantics..."
; earmark:refersTo :renear02-reference
; earmark:begins "55"^^xsd:nonNegativeInteger
; earmark:ends "89"^^xsd:nonNegativeInteger .

...

```

Now, using the Linguistic Acts ontology introduced in Section 3.4, it is possible to link EARMARK ranges to their formal meaning and to their particular references, i.e. literals. For instance, considering the range `:first-author-namefirstletter`, I can say that:

1. this range denotes a particular literal (i.e., “Allen”) that is owned by the first author;
2. this range express a particular meaning, i.e. the fact of having a first name;
3. this meaning is a conceptualisation of the literal considered introduced in the first point of this list.

Thus, according to LA-EARMARK, I will have:

```

:first-author-namefirstletter a la:Expression # string "A"
; la:expresses foaf:givenName

```

```

; la:denotes :renear-given-name .

:renear-given-name a litre:Literal
; litre:hasLiteralValue "Allen"
; litre:isLiteralOf :renear
: la:hasConceptualization foaf:givenName .

[] a la:LinguisticAct
; sit:isSettingFor
<http://www.essepuntato.it/me> # myself, as author of
this interpretation
, :renear # as the person having a certain name
, :first-author-namefirstletter # The letter identifying
the name
, :renear-given-name # The full version of the name
, foaf:givenName . # The meaning associated to such a
string

```

4.3.2 C4O: how much, where and what someone is citing

Besides defining reference lists and bibliographic references in a machine-readable form, I also focus on how these references are used in the citing paper. In particular, I need entities that describe:

- in-text reference pointers within the citing paper;
- links to the bibliographic references denoted by in-text reference pointers;
- how much a particular document is locally cited by the citing document – i.e., the total amount of in-text reference pointers within the citing paper denoting the same bibliographic reference;
- how much an article is globally cited (according to particular bibliographic citation service);
- the contexts involved in a citation – i.e., the part P_{citing} of the citing article containing a particular in-text reference pointer and the part P_{cited} of the cited article that is relevant to P_{citing} .

The *Citation Counting and Context Characterization Ontology*²⁵ (*C4O*) has been developed to allow the description of the above entities. This ontology enables the characterisation of bibliographic citations in terms of their presence in an article by means of the following classes (shown in Fig. 4.7 on page 118):

²⁵C4O, the Citation Counting and Context Characterization Ontology: <http://purl.org/spar/c4o>. The prefix *c4o* refers to entities defined in it.

- class *c4o:InTextReferencePointer*. An in-text reference pointer is a textual device denoting (property *c4o:denotes*) a single bibliographic reference that is embedded in the text of a document within the context of a particular sentence;
- class *c4o:InTextReferencePointerList*. A list containing (through the chain *swan:item* and *swan:itemContent*) only in-text reference pointers denoting the specific bibliographic references to which the list pertains (property *c4o:pertains*). Such a list cannot contain more than one item containing the same in-text reference pointer;
- class *c4o:SingleReferencePointerList*. Defined as subclass of the previous one, it is an in-text reference pointer list that pertains to exactly one bibliographic reference;
- class *c4o:GlobalCitationCount*. The number of times a work has been cited globally (property *c4o:hasGlobalCountValue*), as determined from a particular bibliographic information source (property *c4o:hasGlobalCountSource*) on a particular date (property *c4o:hasGlobalCountDate*).

C4O provides the ontological structures to permit one to record the number of in-text citations (property *c4o:hasInTextCitationFrequency*, i.e. the number of in-text reference pointers to a single reference in the reference list of the citing article), and also the number of citations a cited entity has received globally (property *c4o:hasGlobalCitationFrequency*), as determined by a bibliographic information resource such as Google Scholar²⁶, Scopus²⁷ or Web of Knowledge²⁸ on a particular date.

Taking into account the example in Section 4.3.1, I can write a set of assertions according to C4O that describes how many times a reference is used within the citing article and how much the cited article is globally cited (according to Google Scholar):

```
:renear02 a biro:BibliographicReference
; biro:references :towards-a-semantics
; c4o:hasInTextCitationFrequency "1"^^xsd:nonNegativeInteger .

:towards-a-semantics c4o:hasGlobalCitationFrequency [
  a c4o:GlobalCitationCount
; c4o:hasGlobalCountDate "2011-12-02"^^xsd:date
; c4o:hasGlobalCountSource [
```

²⁶Google Scholar: <http://scholar.google.it>.

²⁷Scopus: <http://www.info.sciverse.com/scopus/>.

²⁸Web of Knowledge: <http://apps.isiknowledge.com>.

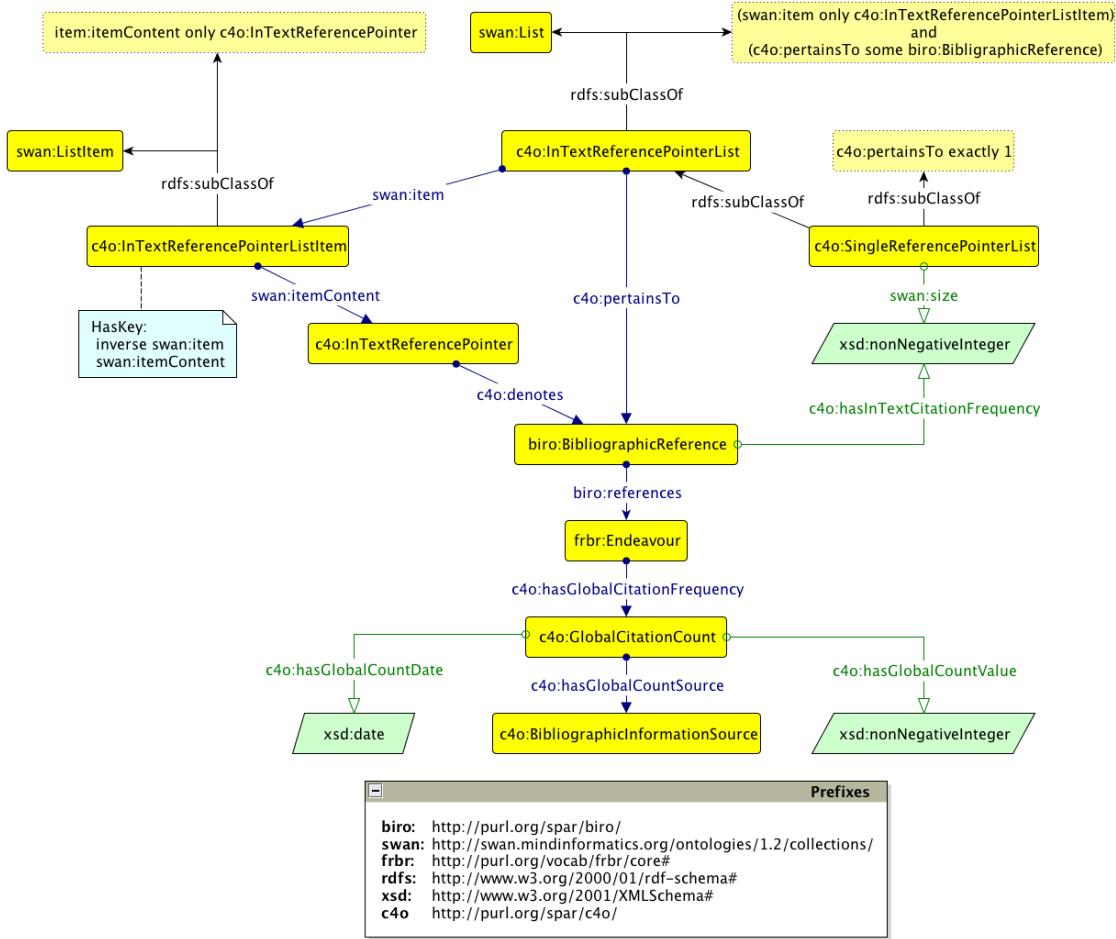


Figure 4.7: Graffoo diagram summarising the C4O entities used for counting citations and references.

```

a c4o:BibliographicInformationSource
; foaf:homepage <http://scholar.google.com> ]
; c4o:hasGlobalCountValue "5"^^xsd:nonNegativeInteger ] .

```

Moreover, C4O enables ontological descriptions of the context where an in-text reference pointer appears in the citing document (modelled as shown in Fig. 4.8 on the facing page), and permits one to relate that context to relevant textual passages in the cited document.

Considering the previous bibliographic reference example, a possible C4O formalisation of the contexts involved by that citing act is:

```

:version-of-record frbr:part :in-text-renear02 .

:in-text-renear02 a c4o:InTextReferencePointer

```

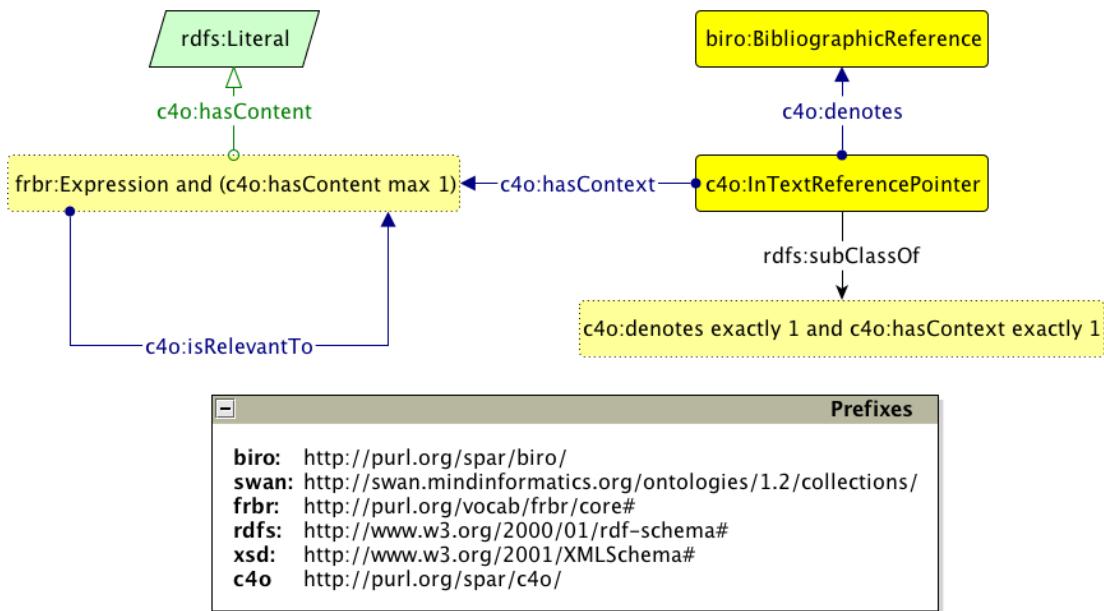


Figure 4.8: Graffoo diagram summarising the C4O entities used for describing citation contexts.

```

; c4o:denotes :renear02
; c4o:hasContext :paragraph-in-version-of-record .

:paragraph-in-version-of-record a frbr:Expression
; c4o:hasContent "Renear, Dubin, and Sperberg-McQueen
  (2002, pp. 121-122) proposed a formal semantic approach
  for structured documents. The basic ..." .

:sentence-in-towards-a-semantics a frbr:Expression
; frbr:partOf :towards-a-semantics
; c4o:hasContent "The source of those problems is that even
  though SGML/XML is thought of as providing access to a
  document's meaningful structure, current SGML/XML
  methods cannot represent the fundamental semantic
  relationships amongst document components and features
  in a systematic machine-processable way."
; c4o:isRelevantTo :paragraph-in-version-of-record .

```

4.4 Characterising document parts with DoCO

A large amount of literature exists about models and theories for the description of structural, rhetorical and argumentative functions of texts through the adoption of Semantic Web technologies, as summarised in [159]. The description of these alternative document layers is crucial for Semantic Publishing. As remarked in [45], to effectively improve the users' comprehension of documents, a formalisation of the document *discourse* (e.g., the *scientific discourse* in scholarly articles) should be explicitly represented in machine-readable forms within the document itself.

Issues related to the rhetorical and the argumentative layers in documents have been discussed for years, e.g. [108] and [82], even in fields different from Computer Science, such as Philosophy and Publishing. For example, in his fundamental work [187], Stephen Toulmin introduces a model for the explanation of arguments (including scientific arguments). In this model, each argument is composed of statements belonging to one of the following six roles:

- **Claim.** A fact that must be established – “That is a scientific article”.
- **Evidence.** Another fact that represents a foundation for the claim – “That article has been submitted to a scientific conference”.
- **Warrant.** A statement bridging from the evidence to the claim – “An article submitted to a scientific conference is a scientific article.”
- **Backing (optional).** Sort of credentials that certifies the warrant – the Call for Papers of the particular conference where the article was submitted.
- **Rebuttal (optional).** Restrictions that may be applied to the claim – “Unless conference reviewers will reject it, considering it non-scientific at all”.
- **Qualifier (optional).** It expresses the degree of certainty concerning the claim via words or phrases such as “certainly”, “possible”, “probably”, etc.

Similarly, in the field of publishing there exist specific constraints authors have to follow when writing a paper. For example, some scientific journals, such as the Journal of Web Semantics²⁹, impose to their articles to follow a particular rhetorical segmentation, in order to explicitly identify meaningful parts from a scientific point of view – for example, *introduction*, *background*, *evaluation*, *materials*, *methods*, *conclusion*, etc. Although these parts usually (but do not necessarily) correspond to structural entities of the article, such as sections, they carry a specific semantics that characterises all the text they contain. From this perspective, this text means more than “being within a section”.

²⁹Journal of Web Semantics, Guide for Authors: http://www.elsevier.com/wps/find/journaldescription.cws_home/671322/authorinstructions.

During the development of the SPAR ontologies, these aspects were analysed carefully. Particularly, I investigated about previous works that tried to address the description of structural and rhetorical components of a document. For the rhetorical part, I found out three models that deal with document segmentation: the *Ontology of Rhetorical Blocks (ORB)* [33], the *SALT Rhetorical Ontology* [83] [84] and the *Medium-Grained structure* [46]. These models offer, the formers, a coarse-grained description (header, introduction, methods, claims, etc.) and, the latter, a medium-grained description (hypothesis, objects of study, direct representation of measurements, etc.) of the rhetorical components of a document.

Although all those models are effectively used, they do not deal with all the compositional aspects of a document. Besides not allowing one to express all the rhetorical functions SPAR needs, those models do not enable rich descriptions of the document structure. One of the requirements of publishers is to have a model that enables the description of the several sub-parts of a document according to its structural components and their rhetorical characterisations. To this end, I developed the *Document Components Ontology*³⁰ (*DoCO*). It provides a structured vocabulary of document components, both structural (e.g. block, inline, container), rhetorical (e.g. introduction, discussion, acknowledgements, reference list, figure, appendix) and mixed (e.g., paragraph, section, chapter), enabling these components, and documents composed of them, to be described in RDF.

Namely, DoCO imports the *Patterns Ontology* presented in Section 3.3.2 and the *Discourse Elements Ontology*³¹ (*DEO*) to describe, respectively, structural and rhetorical components of documents. Moreover, the latter ontology uses seven rhetorical block elements (*background*, *conclusion*, *contribution*, *discussion*, *evaluation*, *motivation* and *scenario*) abstracted from the *SALT Rhetorical Ontology*³². In the following subsections we analyse in detail the structural and rhetorical functions expressible through DoCO entities.

4.4.1 Building blocks for structuring documents

A brief introduction of the theory about structural patterns for documents was illustrated in Section 3.3.2 and in previous works [43] [50]. In this section I list the instanceable patterns again, giving more precise definitions supported by HTML examples³³.

³⁰DoCO, the Document Components Ontology: <http://purl.org/spar/doxo>. The prefix *doxo* refers to entities defined in it.

³¹DEO, the Discourse Elements Ontology: <http://purl.org/spar/deo>. The prefix *deo* refers to entities defined in it.

³²SRO, the SALT Rhetorical Ontology: <http://salt.semanticauthoring.org/ontologies/sro.rdfs>.

³³Notice that it is possible to create valid HTML documents that are not compliant with the presented structural pattern theory. For that reason, in the examples that follow I use HTML elements considering their (informal) semantics as a strong requirement to make a *correct* document. For sure, there are other markup formats that fit the structural pattern theory better than HTML,

The first patterns I introduce are *milestone* and *meta*. They are defined as empty elements that can have zero or more attributes. Moreover:

- the distinctive characteristic of the pattern *milestone* is the *position* it assumes in the document. This pattern typically describes elements that change the aspect of a document depending on where they are put. Moreover, this pattern is usually followed by elements that are used to define the actual content of a document. In HTML, the element *img* is a perfect example of this pattern;
- the main feature of the pattern *meta* is its *existence*, independently from the position that it has within the document. All the elements following this pattern are commonly used to define metadata about the document itself or part of it, independently of where they are. In HTML, the elements *meta* and *link* are good examples that comply with this pattern.

The pattern *atom* defines elements that can contain text only. Like meta, this pattern is commonly used for the description of metadata or, at the most, text that is not considered part of the document body. In HTML, the element *title* (inside the element *head*) is an example of this pattern. The following HTML code summarises the patterns introduced so far:

```
<html>
  <head>
    <title>S.'s home</title>
    <link href="layout.css"
          rel="stylesheet" type="text/css" />
    <meta http-equiv="Content-Type"
          content="text/html; charset=UTF-8" />
  </head>
  <body>
    
  </body>
</html>
```

The next two patterns I illustrate, i.e. *inline* and *block*, are followed by elements that are commonly used for the specification of the document content. Both of them can contain text and have the same content model that enable the definition of hierarchical structures: they can contain other inline and milestone elements and items that comply with the special container *popup* – I will introduce the latter in few paragraphs. Elements compliant with the patterns *inline* and *block* differ for two aspects:

- although inline elements can contain other inlines, block elements cannot contain other blocks;

such as AkomaNtoso [10]. However, I defend the choice of using HTML since it is well-known and easily understandable by readers.

- inline elements cannot be used as root element of documents, but must always be contained by block elements.

In HTML, there are many elements that comply with these two patterns. For example, *p* and *h1* follow the pattern block, while *em* and *a* comply with the pattern inline.

Finally, the pattern *container* concerns the structural organization of a document. All the elements following this pattern do not contain any non-empty text. However, they can contain elements compliant with the following patterns: meta, atom, block and all the subtypes of container but *popup*.

While the content model of container elements (e.g., HTML *body*) admit to contain all optional and repeatable elements, particular restrictions are applied to the subtypes of the pattern container in terms of element repeatability. In particular:

- the pattern *table* contains homogeneous and repeatable elements. In HTML, elements that comply with this pattern are *ul* and *table*;
- the pattern *record* contains no repeatable elements (e.g., HTML element *html*);
- the pattern *hierarchical container* contains always a sort of header at the beginning that must be formed by block elements only. In HTML, the element *section* (when containing always an *h1* as first child) is a good example of this pattern;
- the pattern *popup* is a special container that can be contained by block and inline elements only. It is often used for the inclusion of complex quotations or other complex structures. In HTML, the element *math* is one of the most representative of this pattern.

The following HTML code summarises latter set of patterns:

```
<html>
  <head><title>The formula</title></head>
  <body>
    <section>
      <h1>The <em>magic</em> mathematical formula</h1>
      <p>In this section I would like to introduce two things
         :</p>
      <ul>
        <li><p>the magic mathematical formula;</p></li>
        <li><p>the <a href="http://dev.w3.org/html5">website
           </a> that inspired me.</p></li>
      </ul>
      <p>And now the <em>magic</em> mathematical formula,
         that is
      <math>
```

```

<mi>x</mi>
<mo>=</mo>
<mfrac>
  <mrow>
    <mo form="prefix">-</mo>
    <mi>3</mi>
    <mo>*</mo>
    <msqrt>
      <msup>
        <mi>y</mi>
        <mn>2</mn>
      </msup>
    </msqrt>
  </mrow>
  <mrow>
    <mn>2</mn>
  </mrow>
</mfrac>
</math>.
Isn't it terrific?</p>
</section>
</body>
</html>

```

The ontology introduced in Section 3.3.2 implements the whole theory I presented so far. As remarked in that section, a document compliant with this theory appears to be more unambiguous, manageable and well-structured according to defined and shared principles of document engineering.

I chose to use this theory as one of the building blocks of DoCO (by importing the related ontology) for two reasons. On the one hand, it becomes then possible to understand whether a document described in terms of DoCO entities is valid against the pattern theory. I just need to check whether the ABox describing that document is consistent or not. On the other hand, I give prescriptive recipes to follow in case one wants to model a new document according to the pattern theory from the scratch.

4.4.2 Mixing rhetorical characterisation and structural components

Documents such as a scientific research articles are characterised by precise rhetorical organisations, sometimes in a way that is partially independent from their structural components. As stated previously, there exist models – e.g., [33] [82] [83] [84] [46] – that try to describe rhetorical characterisations of documents from different perspectives. Although these ontologies can be used for the description of rhetorical aspects

of documents, some of them lack in linking explicitly and correctly pure structural behaviours to rhetorical aspects. Probably, one of the principal causes of this lack should be attributed to the intrinsic complexity of defining some components as purely rhetorical or purely structural.

In order to clarify this point, let me consider as example a well-known document component: the *paragraph*. The structural behaviour of a document component can be described by the syntactic structures that it enables and do not relate to its rhetoric nature. From this perspective, a paragraph cannot be considered a pure structural component – i.e. a component carries only a syntactic function – since it *de facto* carries a meaning through its natural language sentences. Thus paragraphs have more than a syntactic attitude. At the same time, the aforementioned models for the rhetorical characterisation of documents do not include the concept *paragraph* as part of them. Are thus paragraphs neither structural components nor rhetorical elements?

Of course, the truth must lie somewhere in the middle. Let me write down two definitions that take active part to this discussion. On the one hand, the definition of *rhetoric* as “the art of discourse, an art that aims to improve the facility of speakers or writers who attempt to inform, persuade, or motivate particular audiences in specific situations”³⁴. On the other hand, the definition of *paragraph* as “a self-contained unit of a discourse in writing dealing with a particular point or idea”³⁵. From these definitions I come to a particular conclusion: the fact that a paragraph is a unit of *discourse* implies that it must have a rhetorical connotation, since the rhetoric is the art of *discourse*. Thus, a textual fragment of a document is a paragraph when it is more than a mere syntactic structure: it should express some ideas and should carry some meanings.

On the other hand, document markup languages such as HTML and DocBook define a paragraph as a pure structural component, without any reference to its rhetoric function:

- “A paragraph is typically a run of phrasing content that forms a block of text with one or more sentences” [92];
- “Paragraphs in DocBook may contain almost all inlines and most block elements” [194].³⁶

Here the term “block of text” and the verb “contains” emphasise the structural connotation of the paragraph, which is amplified by our direct experience as readers. Experience that implicitly tells us that a particular textual fragment shown in a book or in an HTML page is a paragraph rather than a chapter or a table.

³⁴Wikipedia article “Rhetoric”: <http://en.wikipedia.org/wiki/Rhetoric>.

³⁵Wikipedia article “Paragraph”: <http://en.wikipedia.org/wiki/Paragraph>.

³⁶The words *inline* and *block* in these list items do not refer to the structural pattern theory introduced previously, although some sort of overlapping exist.

The *Document Components Ontology (DoCO)*, shown in Fig. 4.9, has been developed so as to fill the gap between the pure structural characterisation of document elements and their the pure rhetorical connotation. Besides including the Pattern Ontology (describing structural components) and Discourse Element Ontology (describing rhetorical components), DoCO also defines other hybrid classes describing elements that are structural and rhetorical at the same time. For instance:

- class *doco:Paragraph*. It is a discourse element based on the pattern block, and contains some sentences;
- class *doco:Sentence*. It is a discourse element based on the pattern inline;
- class *doco:Chapter*. It is a discourse element based on the pattern container, and it is part of the body-matter of a document;
- class *doco:BodyMatter*. It is a discourse element based on the pattern container;
- etc.

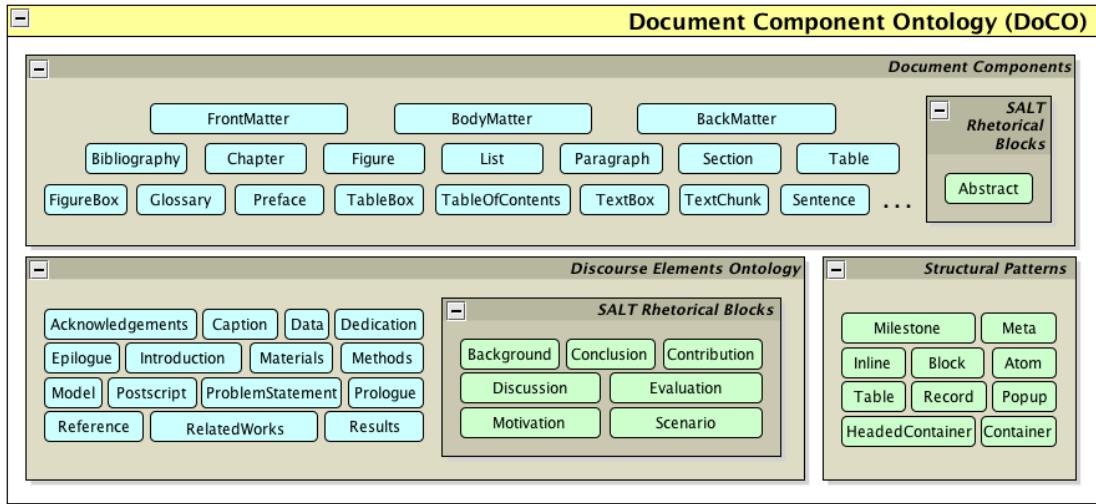


Figure 4.9: Diagram describing the composition and the classes of the *Document Components Ontology (DoCO)*.

The following excerpt shows how to use DoCO to describe structural and rhetorical aspects of the text of the example in Section 4.3:

```
:version-of-record a pattern:Container
; pattern:contains
:front-matter , :body-matter , :back-matter .
```

```

...
:body-matter a doco:BodyMatter
; pattern:contains :introduction , :related-works , ...

:related-works a doco:Section , deo:RelatedWork
; pattern:contains :first-paragraph , ...
, :paragraph-in-version-of-record , ...

:paragraph-in-version-of-record a doco:Paragraph
; pattern:contains :sentence1 , :sentence2 , ...

:sentence1 a doco:Sentence
; pattern:contains :in-text-renear02 .

:in-text-renear02 a deo:Reference , pattern:Inline .

...
:back-matter a doco:BackMatter
; pattern:contains [ a doco:Section
; pattern:contains :reference-list ] .

:reference-list a doco>ListOfReferences
; pattern:contains
:barwise83 ,:black37 ... , :renear02 , ...

:renear02 a deo:BibliographicReference , pattern:Inline .

...

```

Moreover, as shown for BiRO (Section 4.3.1) and as illustrated in Section 3.4.1, DoCO can be used in combination with LA-EARMARK to enhance the document markup with axioms related to its structural and rhetorical aspects.

4.5 In the past you were it, now you are not it

When modelling a domain using ontologies, we may need to describe scenarios in which an *owner* hold some *owned entity* during a specific temporal interval and/or within a specific context. For instance, in the publishing domain, we may want to describe when a *status* (e.g., *under review*, *accepted for publication*, *draft*) is gained or lost by a document, to which institution a particular document's author is affiliated, or which *roles* are held by people at a particular time. All these scenarios involve

the following specific objects as part of the discourse: the *owner*, the *owned entity*, and the *time* and the *context* in which the act of owning holds.

Most ontologies are unable to model such scenarios effectively, for different reasons. Three techniques have been used in attempts to address this modelling issue – *class subsumptions*, *property links*, *inter-linked classes* and *n-ary class modelling* – but each falls short in some aspect, as we will now explain.

4.5.1 Using class subsumptions

To clarify this design technique and the issues that arise from it, consider the agent/role relations as described in the Portal Ontology³⁷ of the AKT Reference Ontology³⁸. This ontology defines the class *Student* as a person (class *portal:Person*) who studies at (property *portal:studies-at*) some educational organization (class *portal:EducationalOrganization*), as defined as follows (in Manchester Syntax [95]):

```
Class: portal:Student
SubClassOf:
    portal:Person that
        portal:studies-at some portal:EducationalOrganization
```

The statement that a student is a person is fine. But a clear distinction needs to be made between the classes *Person* and *Student*. The fact of being a person is *independent of the passage of time* – Silvio Peroni is a living person; Kurt Vonnegut is a person who has died. His death does not prevent us describing him as a person. However, the fact of being a student is strictly *time-dependant* – Silvio Peroni is a graduate student now, but (hopefully) he will not be one for much longer! This subsumption model shows a clear design issue: classes defining time-dependent characteristics (namely, *anti-rigid* classes according to [85]) have been placed in the same *is-a* hierarchy³⁹ as classes defining time-independent characteristics (namely, *rigid* classes according to [85]). As suggested in [85], I believe they should be part of two separated hierarchies and conclude that descriptions of time-dependant entities cannot be satisfactorily achieved by using subsumption.

4.5.2 Using property links

An alternative solution that properly takes into account time-dependant characteristics is the use of properties for defining the *owned entity* (e.g., *status*), while continuing to express the *owner* as an individual of a particular class (e.g. a document). For instance, consider a person (the *owner*) as author (the *owned entity*) of a particular document. Much ontologies for describing bibliographic resources,

³⁷The OWL formalisation of the Portal Ontology is available at <http://www.aktors.org/ontology/portal>. The prefix *portal* refers to entities defined in it.

³⁸AKT Reference Ontology: <http://www.aktors.org/publications/ontology/>.

³⁹That, in OWL, refers to the hierarchy defined through *rdfs:subClassOf* relations.

such as DCTerms [63], FRBR [100] and BIBO [42], use object properties to model this situation. The idea is to link the document, i.e. the object of discourse, to the people who are its authors by the use of a 'role' property, as shown in the following RDF excerpt (using Turtle syntax [147])⁴⁰:

```
# Using DCTerms
:earmark-paper a dcterms:BibliographicResource
; dcterms:creator :peroni , :vitali .

:kce-paper a dcterms:BibliographicResource
; dcterms:creator :peroni , :motta , :daquin .

# Using FRBR (Expression layer)
:earmark-paper a frbr:Expression
; frbr:realizer :peroni , :vitali .

:kce-paper a frbr:Expression
; frbr:realizer :peroni , :motta , :daquin .

# Using BIBO
:earmark-paper a bibo:Article
; bibo:authorList ( :peroni , :vitali ) .

:kce-paper a bibo:Article
; bibo:authorList ( :peroni , :motta , :daquin ) .
```

This design approach has at least two problems. The first occurs when the requirements of the ontology are not fully known at the time of development. For instance, the number of roles that can exist for people involved in the publishing domain (author, editor, publisher, etc.) may increase with time as new technologies are developed, leading, for example, to the creation of a new role, such as *linked-data manager*. Clearly, an ontology describing publishing roles cannot take into account all the possible roles that may be invented in the future. Consequently, the TBox of the ontology will require frequent extensions to include new properties on a case by case basis, requiring maintenance and increasing the risk of creating ontology inconsistencies.

A possible solution that allows better *extensibility* of a model is to use data properties rather than object properties. For instance, the W3C specification [98] of the ontology⁴¹ for describing vCard objects in RDF prescribes the use of a (very general) data property, *vcard:role*, for describing an individual's roles. On the one hand, this permits easy extensions to the ontology, since we can add arbitrary literals

⁴⁰The following examples refer to [142] and [141].

⁴¹An Ontology for vCards: <http://www.w3.org/2006/vcard/ns#>. The prefix *vcard* refers to entities defined in it.

to represent new roles. However, the lack of a clear and defined vocabulary for roles can cause ambiguity problems. For example, the literals “Ph.D.” and “PhD student” are clearly referring to the same role, although they are different literals and end up being formally different within the model.

A second problem, that affects the scenario whether the properties used are object properties or data properties, is to discern in which context an *owner/owned entity* association holds (e.g. to which journal does a person’s role of editor relate). Consider the problem of an author having different institutional affiliations in the context of different publications. Using the *Semantic Web Conference Ontology*⁴² [124], we can define affiliations for authors as follows:

```
# For the EARMARK paper
:peroni swrc:affiliation :cs-unibo .
:vitali swrc:affiliation :cs-unibo .

# For the KCE paper
:peroni swrc:affiliation :kmi .
:motta swrc:affiliation :kmi .
:daquin swrc:affiliation :kmi .

# Institution descriptions
:cs-unibo a foaf:Organization
; dcterms:description "Department of Computer Science,
University of Bologna, Bologna, Italy" .

:kmi a foaf:Organization
; dcterms:description "Knowledge Media Institute, Open
University, Milton Keynes, United Kingdom" .
```

This specification, although straightforward, does not differentiate between associations. It is still not possible to determine the affiliation of an author in the context of a particular paper – e.g., “give me the institutional affiliation of the person *:peroni* as author of the paper *:earmark-paper*”. There is no way of specifying formally which of a person’s affiliations is associated with which document.

4.5.3 Using inter-linked classes

Another possible way to resolve the problem introduced in Section 4.5.1 is to link the *owner* class and the *owned entity* class through object properties. Here, *agent-role* relations, such as the above-mentioned *Person/Student* relationship, should involve two classes that have no subclass subsumptions (declared or inferable) linked by a

⁴²The Semantic Web Conference Ontology: <http://data.semanticweb.org/ns/swc/ontology>. The prefixes *swc* and *swrc* refer to entities defined in it.

specific property such as *holdsRole*. For instance, the *Semantic Web Conference Ontology* implements such a design principle through two different classes, *foaf:Person* and *swc:Role*, and the object property *swc:holdsRole* linking them.

In this way the extensibility of the ontology is guaranteed, reducing the possibility of undesirable inferential side effects. Adding new roles simply involves adding new individuals to the class *swc:Role*. That requires no modification to the TBox of the ontology, making it a very good design principle. However, this solution still is unable to describe that a person holds a particular role during a specific time period. To demonstrate this issue, consider the following description:

the person *:peroni* was a student from October 2005 to March 2008 as an undergraduate, and from January 2009 until now as a PhD candidate, at the University of Bologna. He was an intern at the Open University from April 2008 to September 2008, and was an intern at the University of Oxford from June 15, 2010 to December 15, 2010.

The SWC ontology, used together with FOAF [25], permits only a partial description of this scenario, as shown in the following excerpt:

```
:student a swc:Role .
:intern a swc:Role .

:peroni a foaf:Person
; swc:holdsRole :undergraduateStudent , :graduateStudent ,
:intern .
```

Obviously, the previous data cannot answer the question “Was *:peroni* a graduate student in June 2008?”, because it lacks information about time. Of course, they can be added using specific models, such as the Time ontology⁴³ [93], as shown as follows:

```
:atTime a owl:ObjectProperty
; rdfs:domain swc:Role
; rdfs:range time:TemporalEntity .

:undergraduateStudent :atTime
[ a time:TemporalEntity
; time:hasBeginning [ a time:Instant
; time:inDateTime [ a time:DateTimeDescription
; time:month "10" ; time:year "2005" ] ]
; time:hasEnd [ a time:Instant
; time:inDateTime [ a time:DateTimeDescription
; time:month "03" ; time:year "2008" ] ] ] .
```

⁴³The Time Ontology: <http://www.w3.org/2006/time>. The prefix *time* refers to entities defined in it.

```

:graduateStudent :atTime
[ a time:TemporalEntity
; time:hasBeginning [ a time:Instant
; time:inDateTime [ a time:DateTimeDescription
; time:month "01" ; time:year "2009" ] ] ] .

:intern :atTime
[ a time:TemporalEntity
; time:hasBeginning [ a time:Instant
; time:inDateTime [ a time:DateTimeDescription
; time:month "04" ; time:year "2008" ] ]
; time:hasEnd [ a time:Instant
; time:inDateTime [ a time:DateTimeDescription
; time:month "09" ; time:year "2008" ] ] ] .

:intern :atTime
[ a time:TemporalEntity
; time:hasBeginning [ a time:Instant
; time:inDateTime [ a time:DateTimeDescription
; time:day "15" ; time:month "06"
; time:year "2010" ] ]
; time:hasEnd [ a time:Instant
; time:inDateTime [ a time:DateTimeDescription
; time:day "15" ; time:month "12"
; time:year "2010" ] ] ] .

```

The problem here is that the information about time is associated with the roles themselves, rather than with the person holding those roles. This will create problems once we add another person with the same roles, since it will become impossible correctly to relate the particular times certain roles are held to the people holding them. Two co-authors of mine (now professors), namely David Shotton (resource `:shotton`) and Fabio Vitali (resource `:vitali`), were once undergraduate students, but in different periods from `:peroni`:

```

:shotton a foaf:Person
; swc:holdsRole :undergraduateStudent .

:vitali a foaf:Person
; swc:holdsRole :undergraduateStudent .

```

Things becomes even more complicated if we need to describe the *context* within which the agent-role relation holds, for example, by specifying in which institution `:peroni` was an intern on a given date.

4.5.4 Using n-ary class modelling

Obviously, OWL ontologies and RDF-based models are not able to handle time periods and contexts directly, but need workarounds such as reification or, more generally, *n-ary descriptions*, in order to express the range of possible scenarios mentioned in the previous sections. Some ontological patterns [146] [91] were previously developed to address these issues. For example, through the *time-indexed situation* pattern⁴⁴ [73], shown in Fig. 4.10, it becomes possible to link a subject to a time-dependant description of a situation⁴⁵.

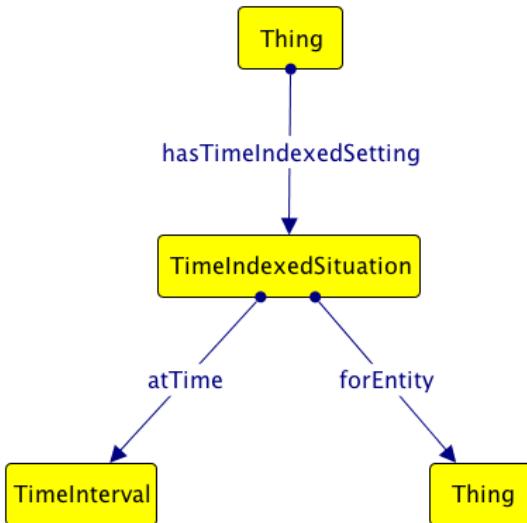


Figure 4.10: A graphical representation of the *time-indexed situation* ontological pattern.

Using this pattern, the scenario presented in Section 4.5.3 can be correctly defined as follows:

```

# University of Bologna
:unibo a foaf:Organization .

# University of Oxford
:oxac a foaf:Organization .

# Open University
:ouac a foaf:Organization .
  
```

⁴⁴Time-indexed situation pattern: <http://ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl>. The prefixes *tisit:*, *sit:* and *ti:* refer to entities defined in it.

⁴⁵In this context, a *situation* is defined as a view on a set of entities. It can be seen as a “relational context”, reifying a relation.

```

:peroni tisit:hasTimeIndexedSetting
:peroniAsUndergraduateStudentInUnibo , :
    peroniAsInternInOUAc
, :peroniAsGraduateStudentInUnibo , :peroniAsInternInOxAc .

:peroniAsUndergraduateStudentAtUnibo a tisit:
    TimeIndexedSituation
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate "2005-10"^^xsd:gYearMonth
    ; ti:hasIntervalEndDate "2008-03"^^xsd:gYearMonth ]
; tisit:forEntity :unibo
; tisit:forEntity :undergraduateStudent .

:peroniAsInternInOUAc a tisit:TimeIndexedSituation
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate "2008-04"^^xsd:gYearMonth
    ; ti:hasIntervalEndDate "2008-09"^^xsd:gYearMonth ]
; tisit:forEntity :ouac
; tisit:forEntity :intern .

:peroniAsGraduateStudentInUnibo a tisit:TimeIndexedSituation
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate "2005-10"^^xsd:gYearMonth ]
; tisit:forEntity :unibo
; tisit:forEntity :graduateStudent .

:peroniAsInternInOxAc a tisit:TimeIndexedSituation
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate "2010-06-15"^^xsd:date
    ; ti:hasIntervalEndDate "2010-12-15"^^xsd:date ]
; tisit:forEntity :oxac
; tisit:forEntity :intern .

```

Although this pattern can be used for formally describe the scenario in Section 4.5.3, it is still too abstract, both as a model and in terms of its terminology. It can be applied in billion of different contexts, of course, but is not specific enough for our particular requirement – that of describing time- and context-dependant relationships between an agent and an owned entity, or, more specifically, defining roles and statuses within the publishing domain.

4.5.5 A general pattern for roles and statuses

What emerges from the preceding discussion is the needs for a model to describe time-dependant and contextualised entities. In particular, we identified four different

things involved in these kinds of scenarios:

1. the entity *owner* of something, e.g. a person or a document possessing a role or a status;
2. the entity that is *owned* by someone, e.g. a role or a status;
3. the *time period* during which the owner *owns* that entity, e.g. from April 2008 to September 2008;
4. the particular *context* that characterises the act of *ownership*, e.g. being a member of an institution or the editor of a particular journal.

In Section 4.5.4 we introduced a very useful ontological pattern that is able to describe this scenario at a very abstract level. Using that as a starting point, we now wish to define a new ontological pattern called ***t*ime-indexed *o*wning in *c*ontext (TOC)**, summarised in Fig. 4.11 and available as OWL implementation ontology⁴⁶.

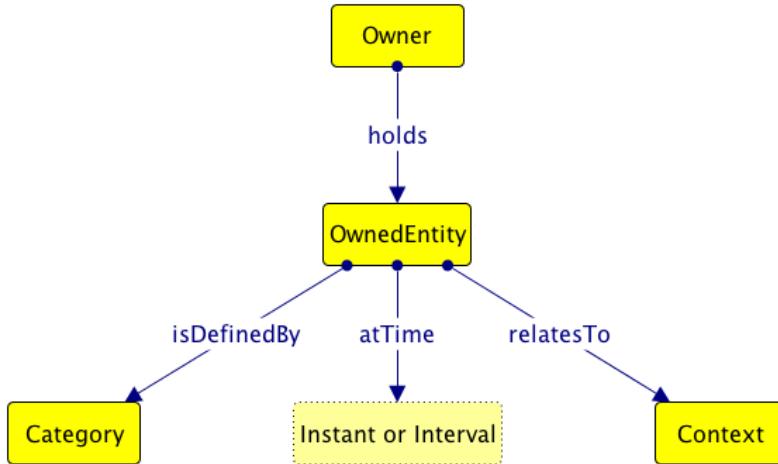


Figure 4.11: The Graffoo diagram of the *time-indexed owning in context* ontological pattern.

Five different classes and four object properties compose this pattern:

- the class *Owner* refers to the agent, e.g. a person, holding (property *holds*) something, such as a role or a status, here defined by *OwnedEntity*;
- the class *Category* identifies the kind of *OwnedEntity* the *Owner* is holding, e.g. the role “student”;

⁴⁶The *time-indexed owning in context* ontological pattern: <http://www.essepuntato.it/2011/11/toc>. The prefix *toc:* refers to entities defined in it.

- the classes *Instant* and *Interval* are used, respectively, to specify specific temporal instants and time periods;
- the class *Context* refers to the specific environment or situation within which the fact that the *Owner* holds the *OwnedEntity* is relevant. For example, the University of Bologna is the institution related to a person, such as *:peroni* in the previous Turtle excerpt, who holds the role of graduate student;
- finally, *OwnedEntity* is the hub that links together the *Category* of the owned entity (property *isDefinedBy*) with the *Owner*, the time (property *atTime*) and the context (*relatesTo*).

Using the TOC ontology, the excerpt introduced in Section 4.5.4 can be re-written as follows:

```
:peroni toc:holds :peroniAsUndergraduateStudentInUnibo , ...

:peroniAsUndergraduateStudentAtUnibo a toc:OwnedEntity
; toc:atTime [ a ti:TimeInterval
; ti:hasIntervalStartDate "2005-10"^^xsd:gYearMonth
; ti:hasIntervalEndDate "2008-03"^^xsd:gYearMonth ]
; toc:relatesTo :unibo
; toc:isDefinedBy toc:undergraduate-student .
...
```

where *toc:undergraduate-student* is an individual within the class *toc:Category*. In the following sections we introduce general use cases and benefits of TOC.

Querying a TOC-based model via SPARQL

In principle, the TOC pattern allows correct answers to be returned in response to a large number of SPARQL queries [78], from the simplest to the most complicated ones. In this section, we present three queries, of increasing difficulty, as examples. For instance, we can ask for all the categories (e.g., kinds of roles) held by a person, such as *:peroni*:

```
SELECT DISTINCT ?cat WHERE {
:peroni a foaf:Person ; toc:holds/toc:isDefinedBy ?cat . }
```

This query can be refined to consider, for instance, only the categories that are defined in a particular context, e.g. the University of Bologna (entity *:unibo*):

```
SELECT DISTINCT ?cat WHERE {
:peroni a foaf:Person toc:holds [ toc:OwnedEntity
; toc:isDefinedBy ?cat
; toc:relatesTo :unibo ]}
```

This will return both the undergraduate and the graduate student roles of *:peroni*. We can further filter the previous results to return just those roles that are applicable in a particular date, such as “24 August, 2011”:

```
SELECT DISTINCT ?cat WHERE {
  :peroni a foaf:Person toc:holds [ a toc:OwnedEntity
    ; toc:isDefinedBy ?cat
    ; toc:relatesTo :unibo
    ; toc:atTime [ a ti:TimeInterval
      ; ti:hasIntervalStartDate ?start
      ; ti:hasIntervalEndDate ?end ]
  FILTER(
    xsd:dateTime(?start) <=
    "2011-08-24T00:00:00Z"^^xsd:dateTime &&
    xsd:dateTime(?end) >
    "2011-08-25T00:00:00Z"^^xsd:dateTime) }
```

This will return just the role of graduate student. If the condition *toc:relatesTo :unibo* was omitted, the query would return both *:peroni*’s role as a graduate student at the University of Bologna, and his concurrent role on that date as an intern at the University of Oxford. More complicated and domain-specific queries are introduced in Section 4.5.6.

Reusing external classes as categories

If required, it is possible, by means of the meta-modelling features of OWL 2, to define classes of external ontologies as individuals of the class *toc:Category*. In this way, we can use them interchangeably either as instances, when we want to associate them to some *toc:Owner*, or as classes when we want to understand hierarchical relationships between them. In addition to opening up the TOC ontology for reuse, this may be very useful for inferring new data for specific categories, even when, in a query, we use their more abstract generalisations (i.e., superclasses).

Suppose for instance, one has a dataset defined according to TOC that includes some entities from the AKT Portal Ontology, introduced in Section 4.5.1, such as:

```
portal:Affiliated-Person a toc:Category .
portal:Student a toc:Category .
portal:PhD-Student a toc:Category .
portal:Employee a toc:Category .
...

# The following statements are defined in the Portal Ontology
portal:Student rdfs:subClassOf portal:Affiliated-Person .
portal:PhD-Student rdfs:subClassOf portal:Student .
portal:Employee rdfs:subClassOf portal:Affiliated-Person .
...
```

In this way, it is possible to query the dataset through SPARQL, asking for all the people affiliated with the University of Bologna (the entity `:unibo`), independently from the roles they may hold as a student, a Ph.D. student, an employee or other subclass of `portal:Affiliated-Person`:

```
SELECT DISTINCT ?person WHERE {
  ?person toc:holds [ a toc:OwnedEntity
    ; toc:isDefinedBy ?aff
    ; toc:relatesTo :unibo ] .
}

  SELECT ?aff WHERE {
    {
      ?aff a toc:Category .
      FILTER(?aff = portal:Affiliated-Person) }
    UNION
    { ?aff rdfs:subClassOf+ portal:Affiliated-Person }
  }
}
```

As highlighted in the previous example, TOC makes it possible and useful to reuse specific parts of ontologies describing categories in form of classes, thus taking advantages of the OWL 2 punning.

Constructing second-order inferences

Of course, it is sometimes desirable to reuse ontologies that specify categories (e.g., roles) through properties rather than classes, as introduced in Section 4.5.2. Consider, for example, the BIBO ontology [42], that associates agent roles with documents through particular sub-properties (`bibo:translator`, `bibo:director`, `bibo:editor`) of the general property `dcterms:contributor`. Using the BIBO ontology with TOC, these object properties can be used as individuals of the class `toc:Category`, again by means of OWL 2 punning. Moreover, it is possible, when needed, to construct second-order inferences using `toc:Category` individuals as properties:

```
CONSTRUCT { ?document ?property ?person } WHERE {
  ?person a foaf:Person ; toc:holds [ a toc:OwnedEntity
    ; toc:isDefinedBy ?property
    ; toc:relatesTo ?document ] }

  SELECT ?property WHERE {
    {
      ?property a toc:Category .
      FILTER(?aff = dcterms:contributor) }
    UNION
    { ?property rdfs:subPropertyOf+ dcterms:contributor }
```

```

    }
}
}
```

As shown in the previous query, through a model that combines TOC and an ontology defining categories as property links, such as BIBO, it becomes feasible to infer second-order logical statements. More generally, TOC can be used as intermediate model for the conversion of *Owner / Owned Entity* data from one ontology into another, independent of the particular design technique used by each ontology (i.e., class subsumptions, property links, inter-linked classes or n-ary relationships). This gives enormous descriptive power and usefulness.

4.5.6 Identifying person's roles with PRO

The need to define publishing roles in SPAR was crucial for the completeness of this suite of ontologies. The problems associated with the adoption of external ontologies to handle this particular requirement have been discussed above. None of them were fully able to satisfy the modelling requirements imposed by SPAR, particularly need for ease of *extendibility* and for the simultaneous representation of *time periods* and *contexts*.

Using TOC as the basis, we implemented *PRO*, the *Publishing Roles Ontology*⁴⁷, an OWL 2 ontology. This ontology, shown in Fig. 4.12 on the following page, permits characterization of the roles of agents – people, corporate bodies and computational agents – in the publication process. It permits one to specify the role an agent has in relation to a particular bibliographic entity (e.g., author, editor, reviewer) or to a specific institution (e.g., publisher, librarian), and the period during which each role is held.

Using PRO, that implements TOC as illustrated by the mapping in Table 4.1 on the next page, it becomes possible to describe all the scenarios discussed in Section 4.5.2 and Section 4.5.3, as shown as follows:

```
:peroni pro:holdsRoleInTime
[ a pro:RoleInTime ### as author of two different papers
  ; pro:withRole pro:author
  ; pro:relatesToDocument :earmark-paper , :kce-paper ]
, [ a pro:RoleInTime ### as affiliate of UniBo CS
  Department
  ; pro:withRole pro:affiliate
  ; pro:relatesToDocument :earmark-paper
  ; pro:relatesToOrganization :cs-unibo
  ; ti:atTime
    [ a ti:TimeInterval
```

⁴⁷PRO, the Publishing Roles Ontology: <http://purl.org/spar/pro>. The prefix *pro* refers to entities defined in it.

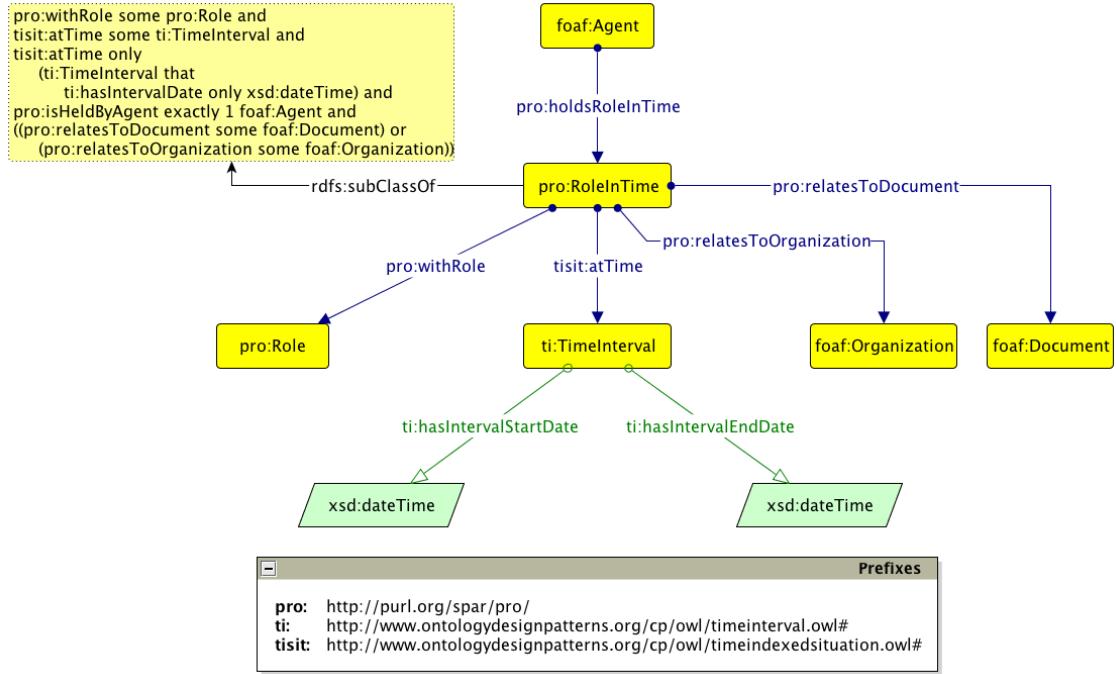


Figure 4.12: Graffoo representation of the Publishing Roles Ontology (PRO).

Table 4.1: Mappings between TOC entities and PRO classes.

TOC entity	PRO class	Description
Owner	foaf:Agent	A class defining any kind of agents, such as a person, a group, an organization or a software agent.
OwnedEntity	pro:RoleInTime	A particular situation that describes a role some agent may have during a particular time interval.
Category	pro:Role	A role an agent may have. Currently, 31 roles are defined in the PRO ontology as individuals of this class.
Instant or Interval	ti:TimeInterval	Two (starting and ending) points in time that define a particular period.
Context	foaf:Document foaf:Organization	Classes defining, respectively, any kind of bibliographic work or publishing organization.

```

; ti:hasIntervalStartDate "2005-10-01T00:00:00Z"^^xsd:
  :dateTime
; ti:hasIntervalEndDate "2008-03-19T12:00:00Z"^^xsd:
  :dateTime ] ]
, [ a ti:TimeInterval
  ; ti:hasIntervalStartDate "2009-01-01T00:00:00Z"^^xsd:
  :dateTime ]
, [ a pro:RoleInTime ### as affiliate of OU KMi
  ; pro:withRole pro:affiliate
  ; pro:relatesToDocument :kce-paper
  ; pro:relatesToOrganization :kmi
  ; ti:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate "2008-04-01T00:00:00Z"^^xsd:
    :dateTime
    ; ti:hasIntervalEndDate "2008-09-30T23:59:59Z"^^xsd:
    :dateTime ] ] ] .

```

As seen, through PRO we can model very rich scenarios, and thus answer complex queries, such as the previously introduced “give me the institutional affiliation of the person `:peroni` as author of the paper `:earmark-paper`”:

```

SELECT ?aff WHERE {
:peroni pro:holdsRoleInTime
  [ a pro:RoleInTime
    ; pro:with :author
    ; pro:relatesToDocument :earmark-paper ]
, [ a pro:RoleInTime
    ; pro:withRole :affiliate
    ; pro:relatesToDocument :earmark-paper
    ; pro:relatesToOrganization ?aff ]
}

```

4.5.7 Specifying document statuses with PSO

The second subdomain of publishing that speaks about *owner-owned entity* relations, again bound together with time and context, that had to be handled in SPAR is that of the *status* of documents. In this case, it is a document (the *owner*), rather than an agent, that holds a particular status (the *owned entity*) at a certain *time*, as direct consequence of particular events (the *context*). For instance, a document holds the status of being under review until all reviewers write their comments about it and send them to the editors, who then make a decision to accept or reject the paper. After that decision has been made, the status ‘under review’ is no longer valid: the document loses that status, and this should be formally describable using an appropriate ontology. Moreover, it can sometimes be useful to link documents to the decisions or events that cause the acquisition or loss of a particular status.

Pre-existing ontologies describing the status of documents use the methods outlined in Section 4.5.2 (e.g., BIBO [42] and the *Project Documents Ontology*⁴⁸ [191]) and in Section 4.5.3 (e.g., the *Document Status Ontology*⁴⁹). As discussed in those sections, property links and inter-linked classes approaches prevent proper descriptions of scenarios that involve the temporal duration of a document status. With the exception of the Document Status Ontology, that implements a mechanism for describing status changes as events, the other ontologies fail to permit definition of these contextual data, or do so only partially.

In order to address these issues in a more satisfactory manner, we developed *PSO*, the *Publishing Status Ontology*⁵⁰. This ontology (shown in Fig. 4.13) characterises the publication status of a document or other publication entity at each of the various stages in the publishing process (e.g. draft, submitted, under review, rejected for publication, accepted for publication, version of record, peer reviewed, open access). As with PRO, PSO was developed following the pattern TOC, as shown by the mapping in Table 4.2 on the next page.

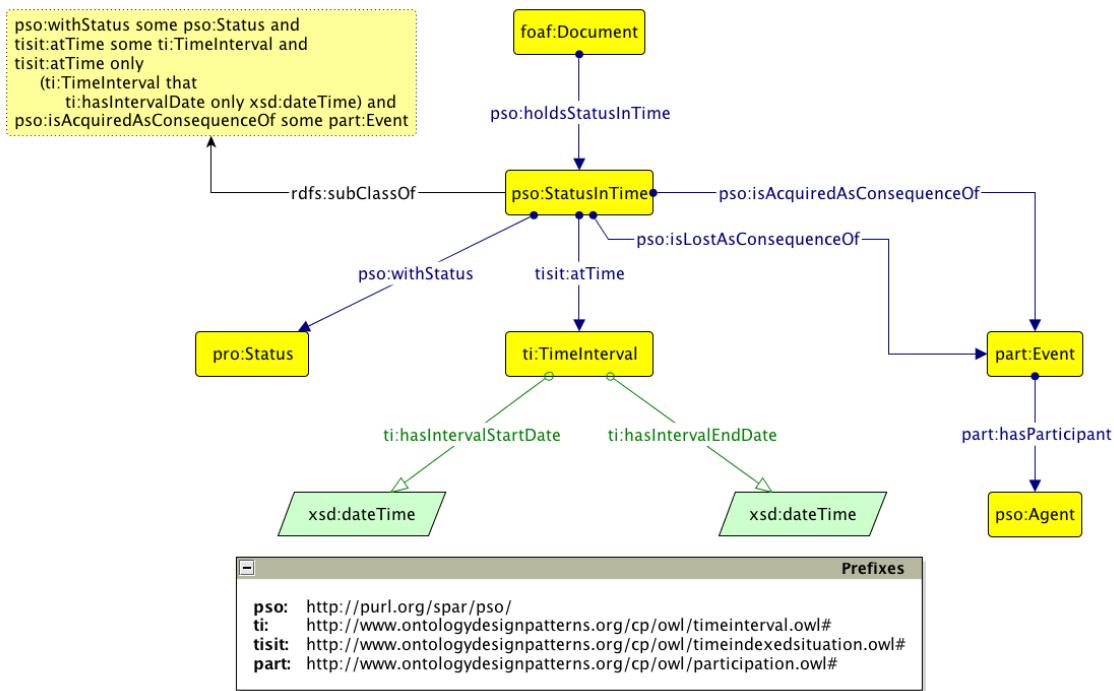


Figure 4.13: Graffoo representation of the Publishing Status Ontology (PSO).

⁴⁸Project Documents Ontology:<http://ontologies.smile.deri.ie/pdo#>.

⁴⁹Document Status Ontology: <http://ontologi.es/status#>.

⁵⁰PSO, the Publishing Status Ontology: <http://purl.org/spar/pso>. The prefix *pso* and *part* refer to entities defined in it.

Table 4.2: Mappings between TOC entities and PSO classes.

TOC entity	PSO class	Description
Owner	foaf:Document	A class defining any kind of bibliographic work.
OwnedEntity	pso:StatusInTime	A particular situation that describes a status or condition a document may have at a particular time as consequence of one or more events.
Category	pso:Status	A status or condition that a document may have. Currently, 26 statuses are defined in the ontology as individuals of this class.
Instant or Interval	ti:TimeInterval	Two (starting and ending) points in time that define a particular period.
Context	part:Event	An event during a publishing process, such as writing a draft, submitting a preprint for publication, or publishing a paper, that changes the status of a document.

Using PSO, it becomes possible to describe the statuses of documents and how they change over time. For instance, consider the following natural language description:

The paper *:earmark-paper* was submitted to DocEng 2009 on 24 April 2009 at 13:18. At noon on 26 April 2009, when the authors received acknowledgement of safe receipt of the paper from the conference editorial committee, the paper was considered “under review”. This status lasted until, on 27 May 2009 at 17:38, the editorial committee notifies the authors that the review process had been completed and that the paper had been accepted. At that moment, the status “under review” ended, and a new status, “accepted for publication”, began.

PSO can be used to represent this description **formally**, as follows:

```
:earmark-paper pso:holdsStatusInTime
  :submitted , :under-review
  , :reviewed-and-accepted-for-publication .

:submitted a pso:StatusInTime
```

```

; pso:withStatus pso:submitted
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate
        "2009-04-24T13:18:21Z"^^xsd:dateTime ]
; pso:isAcquiredAsConsequenceOf :author-submits-paper .

:author-submits-paper a part:Event
; dcterms:description "An author submitted the paper
through the online conference submission system." .

:under-review a pso:StatusInTime
; pso:withStatus pso:under-review
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate
        "2009-04-26T12:00:00Z"^^xsd:dateTime
    ; ti:hasIntervalEndDate
        "2009-05-27T17:38:01Z"^^xsd:dateTime ]
; pso:isAcquiredAsConsequenceOf :reviewers-working
; pso:isLostAsConsequenceOf :reviewers-finish .

:reviewers-working a part:Event
; dcterms:description "Anonymous reviewers are working on
the paper." .

:reviewed-and-accepted-for-publication a pso:StatusInTime
; pso:withStatus pso:accepted-for-publication , pso:
    reviewed
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate
        "2009-05-27T17:38:01Z"^^xsd:dateTime ]
; pso:isAcquiredAsConsequenceOf
    :reviewers-finish , :committee-accepts
    , :committee-notifies-to-authors .

:reviewers-finish a part:Event
; dcterms:description
    "Reviewers finish to review the paper." .

:committee-accepts a part:Event
; dcterms:description "Conference committee accepted the
paper according to reviewers' comments." .

:committee-notifies-to-authors a part:Event
; dcterms:description "A notification of acceptance has
been sent to authors." .

```

4.6 Describing publishing workflows with PWO

Keeping track of publication processes is a crucial task for publishers. This activity allows them to produce statistics on their goods (e.g., books, authors, editors) and to understand whether and how their production changes during time. Organisers of particular events such as academic conference have similar needs. Tracking the number of submission in the current edition of the conference, the number of accepted papers, the review process and the like are important statistics that can be actually used to prevent emerging drawbacks in future edition of the conference itself.

Some communities started to publish⁵¹ data describing those events as RDF statements in the Linked Data, in order to allow software agents and applications to check and reason on them and to infer new information. However, the description of processes, for instance the peer-review process or the publishing process, is something that is not currently handled – although sources of related raw data exist. Having also these kinds of data published would increase the transparency of aforementioned processes and allow their use for statistical analysis. Of course, a model for describing these data is needed. Moreover it should be easy to be integrated and adapted according to needs and constraints of different domains (publishing, academic conferences, research funding, etc.).

In order to accommodate these needs, I developed the *Publishing Workflow Ontology*⁵² (*PWO*). This ontology permits one to describe the logical steps in a workflow, for example the process of publication of a document. Each step may involve one or more events that take part to a particular phase of the workflow (e.g. authors are writing the article, the article is under review, reviewer suggested to revise the article, the article is in printing, the article has been published, etc.).

As shown in Fig. 4.14 on the following page, PWO is based on two main classes, which are:

- class *pwo:Workflow*. It represents a sequence of connected tasks (i.e., steps) undertaken by agents. A workflow may be seen as an abstract model of real work;
- class *pwo:Step*. It is an atomic unit of a workflow, it is characterized by a starting time and an ending time, and it is associated with one or more events. A workflow step usually involves some input information, material or energy needed to complete the step, and some output information, material or energy produced by that step. In the case of a publishing workflow, a step typically results in the creation of a publication entity, usually by the modification of

⁵¹Semantic Web Dog Food: <http://data.semanticweb.org>.

⁵²PWO, the Publishing Workflow Ontology: <http://purl.org/spar/pwo>. The prefix *pwo* refers to entities defined in it.

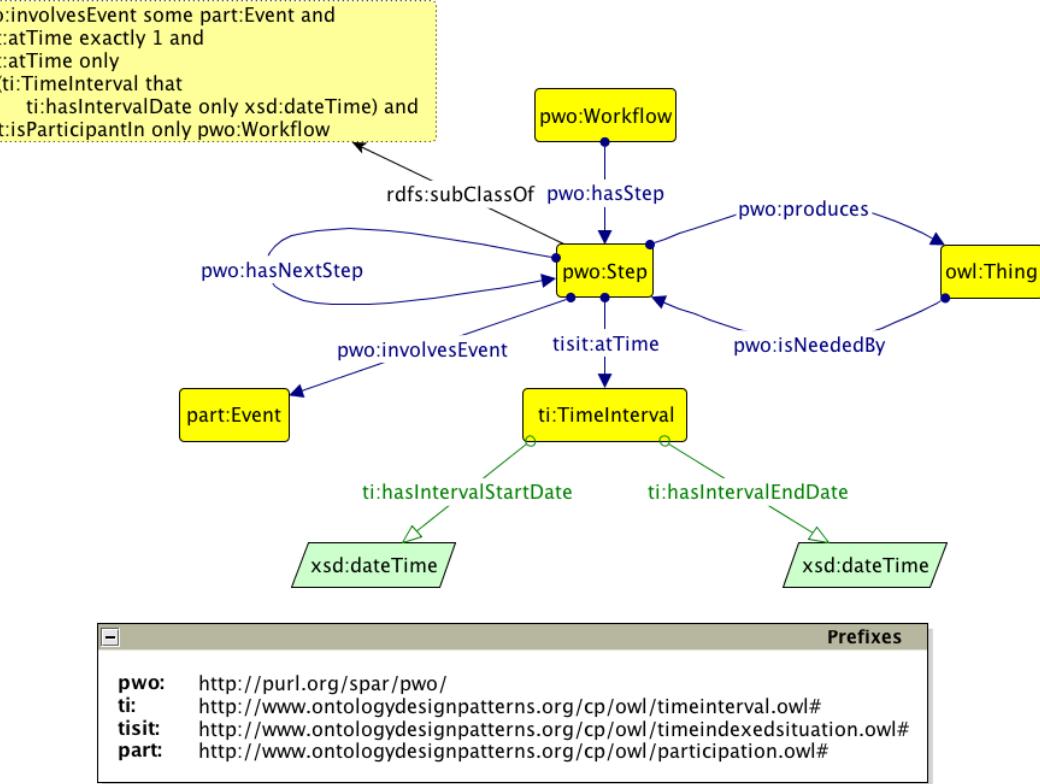


Figure 4.14: Graffoo representation of the Publishing Workflow Ontology (PWO).

another pre-existing publication entity, e.g. the creation of an edited paper from a rough draft, or of an HTML representation from an XML document.

The following excerpt presents the PWO description of the workflow describing the publication of the article (i.e., the resource *:earmark-paper*) introduced in the example in Section 4.5:

```

:workflow a pwo:Workflow
; pwo:hasFirstStep :stepOne
; pwo:hasStep :stepTwo , :stepThree , :stepFour .

:stepOne a pwo:Step # Authors write the paper
; pwo:involvesEvent [ a part:Event
  dcterms:description "Authors write the paper" ]
; tisit:atTime [ a ti:TimeInterval
; ti:hasIntervalStartDate
  "2009-02-14T00:00:00Z"^^xsd:dateTime
; ti:hasIntervalEndDate
  "2009-03-25T00:00:00Z"^^xsd:dateTime ]

```

```
; pwo:produces :earmark-paper
; pwo:hasNextStep :stepTwo .

:stepTwo a pwo:Step # Paper submitted
; pwo:involvesEvent :author-submits-paper
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate
        "2009-04-24T13:18:21Z"^^xsd:dateTime
    ; ti:hasIntervalEndDate
        "2009-04-24T13:18:21Z"^^xsd:dateTime ]
; pwo:needs :earmark-paper
; pwo:produces :submitted # New status in time for the
    paper
; pwo:hasNextStep :stepThree .

:stepThree a pwo:Step # Paper reviewed
; pwo:involvesEvent :reviewers-working , :reviewers-finish
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate
        "2009-04-26T12:00:00Z"^^xsd:dateTime
    ; ti:hasIntervalEndDate
        "2009-05-26T12:00:00Z"^^xsd:dateTime ]
; pwo:needs :earmark-paper
; pwo:produces :review1 , :review2 , review3 .
; pwo:hasNextStep :stepFour .

:review1 a fabio:Comment # Review 1
; frbr:realizationOf [ a fabio:Review ]
; cito:reviews :earmark-paper
; pro:isDocumentContextFor [ a pro:RoleInTime
    ; pro:withRole pro:author
    ; pro:isRoleHeldBy [ a foaf:Person # First anonymous
        reviewer
    ; pro:hasRoleInTime [ a pro:RoleInTime
        ; pro:withRole pro:reviewer
        ; pro:relatesToDocument :earmark-paper ] ] ] .

:review2 a fabio:Comment ...

:stepFour a pwo:Step # Notification of acceptance
; pwo:involvesEvent
:committee-accepts , :committee-notifies-to-authors
; tisit:atTime [ a ti:TimeInterval
    ; ti:hasIntervalStartDate
        "2009-04-26T12:00:00Z"^^xsd:dateTime
```

```

; ti:hasIntervalEndDate
  "2009-05-27T17:38:01Z"^^xsd:dateTime ]
; pwo:needs :review1 , :review2 , :review3
; pwo:produces
  :reviewed-and-accepted-for-publication
, :acceptance-notification .

:acceptance-notification a fabio:Email # The e-mail notifying
the acceptance
; frbr:realizationOf [ a fabio:Opinion ]
; pro:isDocumentContextFor [ a pro:RoleInTime
; pro:withRole pro:author
; pro:isRoleHeldBy [ a foaf:Group # The committee ] .

```

PWO was implemented according to three particular ontology pattern:

- the *time-indexed situation pattern* [73] to describe workflow steps as entities that involve a duration and that are characterised by events and objects (needed for and produced by the step);
- the *sequence pattern*⁵³ [72] to define the order in which steps appear within a workflow;
- the *participation pattern*⁵⁴ [70] to describe events (and eventually agents involved) taking part in steps.

In order to be consistent with real descriptions, PWO implements some strong constraints on steps by means of a particular model: the *Error Ontology*⁵⁵. This ontology is a unit test that allows one to produce an inconsistent model if a particular (and incorrect) situation happens. It works by means of a data property, *error:hasError*, that denies its usage for any resource, as shown as follows (in Manchester Syntax [95]):

```

DataProperty: error:hasError
  Domain: error:hasError exactly 0
  Range: xsd:string

```

A resource that asserts to have an error makes the ontology inconsistent since its domain is defined as “all those resources that do not have any *error:hasError* assertion”.

⁵³The sequence pattern: <http://www.ontologydesignpatterns.org/cp/owl/sequence.owl>. The prefix *seq* refers to entities defined in it.

⁵⁴The participation pattern: <http://www.ontologydesignpatterns.org/cp/owl/participation.owl>. The prefix *part* refers to entities defined in it.

⁵⁵The Error Ontology: <http://www.essepuntato.it/2009/10/error>. The prefix *error* refers to entities defined in it.

By means of the Error Ontology, I can oblige all the PWO workflow descriptions by not having steps that need (property *pwo:needs*) something that is actually produced (property *pwo:produces*) by a following step. The following excerpt shows the implementation of this constraint through a SWRL rule [96]:

```
step(?step1) , step(?step2) , needs(?step1,?resource) ,
produces(?step2,?resource) , sequence:precedes(?step1,?step2)
-> error:hasError(?step1,"A step cannot need a resource
that will be produced by a following step"^^xsd:string)
```

4.7 How communities uptake SPAR

The SPAR ontologies are now being used or are being considered for adoption in a variety of academic and publishing environments. The adoption of these models by different communities can be ascribed, at least in part, to our adoption of the following development strategies:

- Frequent ongoing interactions between the authors of SPAR, and publishers, service developers and other end-users, that have allowed us to understand their various needs and interests.
- The minimization of the constraints applied to the ontological entities, so that the ontologies can be applied in a wide variety of situations.

The following sections briefly described how SPAR ontologies are now being used in various communities.

4.7.1 SWAN ontology

The most recent version (v2.0) of the *SWAN ontology ecosystem* [35], introduced above in Section 2.2.6, has recently been harmonised to include FaBiO and work seamlessly with CiTO [34]. David Shotton and I undertook this harmonisation collaboratively, while developing version 1.6 of CiTO [165] into CiTO v2.0 and FaBiO v1.0, and by Paolo Ciccarese and Tim Clark (Harvard University), authors of the SWAN Ontologies. The resulting combined CiTO/FaBiO + SWAN model is specified in OWL 2 DL, is fully modular, and inherently supports agent-based searching and mash-ups.

The principles adopted for its activity, which resulted in the harmonisation described in Fig. 4.15 on the next page, involved:

- the renaming classes (concepts) or properties (relationships) in one or both set of ontologies to avoid apparent overlap;

- the re-definition of classes or properties to resolve actual overlap between concepts; and
- the deprecation of elements of individual ontologies, or even whole ontologies, in favour of others that more effectively serve the domain of knowledge under consideration, having greater granularity or a more effective structure.

In summary, the SWAN Citations ontology module was deprecated in favour of FaBiO, certain classes in the SWAN Discourse Relationship were renamed and redefined, the property *discourse-relationships:cites* in that module was deprecated, and CiTO was linked to that module by making *cito:cites* a sub-property of *discourse-relationships:refersTo*. Full details are given in [34].

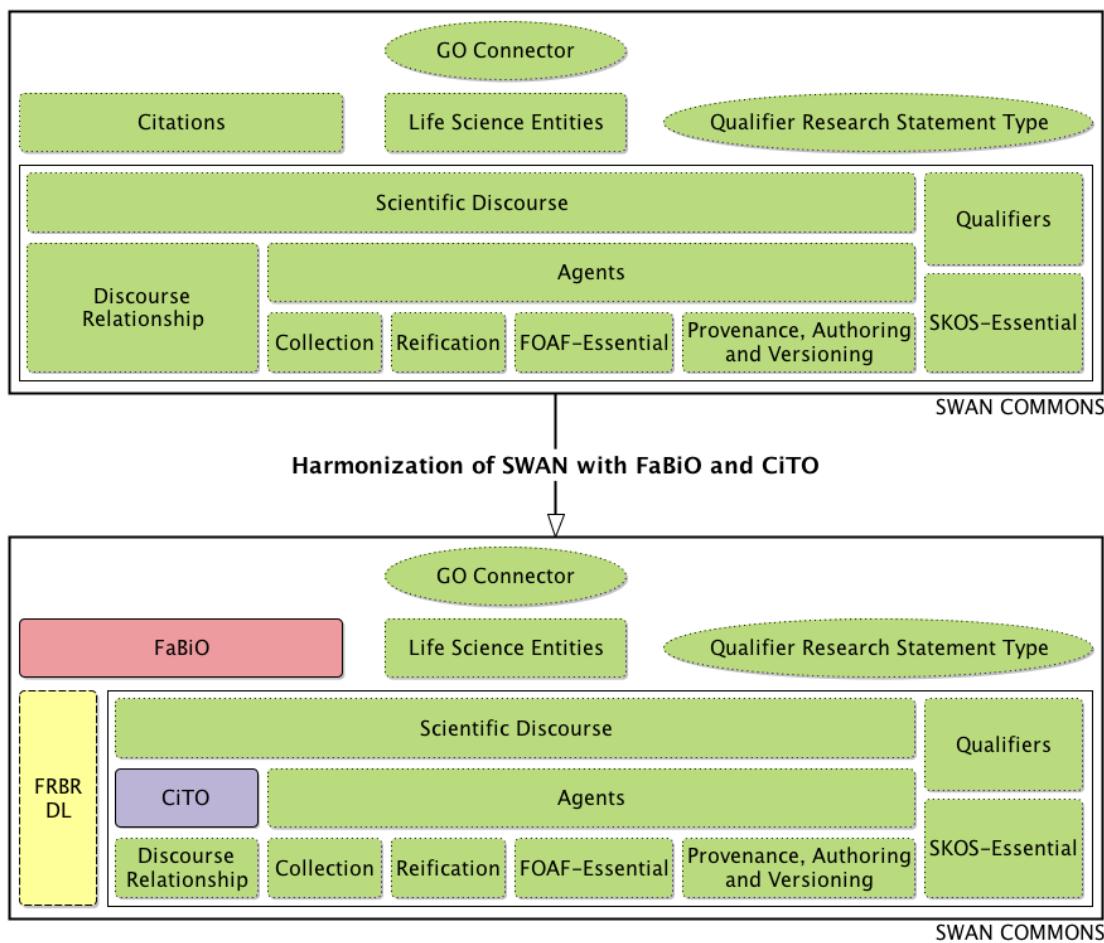


Figure 4.15: The SWAN ontology ecosystem before (above) and after (below) the harmonisation activity that resulted in the inclusion of FaBiO and CiTO in the SWAN Commons set of ontologies.

4.7.2 CiteULike

Egon Willighagen of Uppsala University has pioneered the use CiTO⁵⁶ to characterize bibliographic citations within *CiteULike*⁵⁷, the free service for managing and discovering scholarly references. A user can add a CiTO relationship between articles via the CiteULike interface, provided that both the citing and the cited articles are in the user's library.

4.7.3 WordPress

In a blog post⁵⁸, Martin Fenner describes a plug-in for *WordPress* called *Link-to-Link*⁵⁹, that makes it easy to add citation typing to references within a blog post, using a sub-set of the most commonly used CiTO relationships presented in a convenient drop-down menu.

4.7.4 Linked Education

The open platform *Linked Education*⁶⁰, which aims at sharing and promoting the use of Linked Data for educational purposes, months ago added CiTO to its listing and recently all the other SPAR ontologies⁶¹ of RDF schemas and vocabularies suitable for use in educational contexts, for example to describe educational resources.

4.7.5 Virtual Observatory

In a recent paper [1], Accomazzi and Dave report the adoption of the FaBiO and CiTO ontologies as part of their efforts to create a semantic knowledge base allowing easier integration and linking of the body of heterogeneous astronomical resources into what they term a Virtual Observatory.

4.7.6 Open Citations Corpus

The Open Citations Corpus⁶² is a database of approximately 6.3 million biomedical literature citations, harvested from the reference lists of all open access articles in

⁵⁶<http://chem-bla-ics.blogspot.com/2010/10/citeulike-cito-use-case-1-wordles.html>.

⁵⁷CiteULike: <http://www.citeulike.org/>.

⁵⁸Blog post by Martin Fenner entitled "How to use citation typing ontology (CiTO) in your blog post": <http://blogs.plos.org/mfenner/2011/02/14/how-to-use-citation-typing-ontology-cito-in-your-blog-posts/>.

⁵⁹Link to link: <http://wordpress.org/extend/plugins/link-to-link/>.

⁶⁰Linked Education: <http://linkededucation.org/>.

⁶¹Linked Education – Schemas and vocabularies: <http://linkededucation.wordpress.com/data-models/schemas/>.

⁶²The Open Citations Corpus: <http://opencitations.net/>.

PubMed Central. These contain references to approx. 3.4 million papers, which represent ~20% of all PubMed-listed papers published between 1950 and 2010, including all the most highly cited papers in every biomedical field. The Open Citations Corpus website allows one to browse these bibliographic records and citations, to select an individual article, and to visualize its citation network in a variety of displays. Details of each selected reference, and the data and diagrams for its citation network, may be downloaded in a variety of formats, while the entire Open Citations Corpus can be downloaded in several formats including RDF and BibJSON. SPAR ontologies have been used to encode this information in RDF. Further information is given on the Open Citations Blog⁶³.

4.7.7 WebTracks

WebTracks⁶⁴ is an open source project funded by the JISC Managing Research Data Programme⁶⁵ that is developing a peer-to-peer protocol to enable web-scale link tracking.

Established techniques such as OAI-PMH and the emerging Linked Web of Data provide tools to publish data for linking. WebTracks focuses on actually making these connections, particularly between research datasets and related publications.

It provides a mechanism for informing the target of a hyperlink that a link has been made to that target, so it can reciprocally link back – for example, by including the correct DOI of a published paper in the metadata of a previously published dataset to which the paper refers. WebTracks creates semantically annotated links between data resources using CiTO yielding a graph of citation and provenance to enable web-scaled data management by exposing links between related objects.

4.7.8 Società editrice il Mulino

The Italian scholarly publishing house *Società editrice il Mulino*⁶⁶ is collaborating with the Department of Computer Science of the University of Bologna, to explore how best to benefit from Semantic Web technologies for the digital publication and sharing of bibliographic objects such as books and articles, and their related metadata.

This has led to the recent prototyping of an application called *Folksauro* (the name comes from the concatenation of the words *folksonomy* and *thesaurus*). Using FaBiO and DoCO as its main ontologies, and one or more discipline-specific thesauri developed in SKOS, Folksauro allows a user to associate terms from the

⁶³The Open Citations Blog: <http://opencitations.wordpress.com/>.

⁶⁴WebTracks: <http://webtracks.jiscinvolve.org/wp/about/>.

⁶⁵JISC Managing Research Data Programme: <http://www.jisc.ac.uk/whatwedo/programmes/mrd.aspx>.

⁶⁶Il Mulino: <http://www.mulino.it>.

thesauri and/or free-text keywords with the whole document, and/or with its sub-parts (chapters, sections, paragraphs, etc.), by means of an intuitive interface that hides the complexity of models and languages used.

4.7.9 Utopia

Utopia Documents⁶⁷ [5] is a novel PDF reader that semantically integrates visualization and data-analysis tools with published research articles. It brings PDF documents to life by linking to live resources on the web and by turning static data into live interactive content, and is now being regularly used by the editors of the Biochemical Journal⁶⁸ to transform static document features into objects that can be linked, annotated, visualized and analysed interactively.

Utopia has a mechanism that deconstructs a PDF document into its constituent parts, which are then annotated using DoCO. This is useful for a number of things: generating bibliometric metadata, improving “mouse selection” in multi-column documents, and identifying the correct flow of text in the document, allowing annoying intruding text such as running headers, footers and captions to be excluded. This in turn is useful for text and data mining algorithms, which can now be targeted, for example, at “all the main text excluding intruders” or “just the text in the figure captions”. The Utopia team are planning later this year to release a free web service that takes a PDF document, deconstructs it, and returns DoCO-annotated XHTML.

In addition, the Utopia team are presently developing an Open Citations plugin that pulls bibliographic citation data live from the Open Citations Corpus and uses it to display the citation network for the paper manifested by the PDF, or for any of the articles references in that paper’s reference list.

⁶⁷Utopia Documents: <http://getutopia.com>.

⁶⁸<http://www.biochemj.org/bj/424/3/>.

Chapter 5

Interfaces for the masses

Any strategy that guarantees the broad adoption of Semantic Web technologies must address the development of applications for improving the human-interaction with semantic models and data. Several amounts of research have been done on models, theoretical approaches, development of tools to infer new information from data and ontologies. However, the Semantic Web will be really integrated with the everyday-Web only when it will be effectively available and accessible to every Web-users not only to Semantic Web practitioners. This point is even more crucial for Semantic Publishing since its end-users are definitely publishers, researchers, librarians and readers rather than experts in semantic technologies. Semantic Web/Publishing communities need to invest time and efforts in the development of proper user-friendly interfaces that act as intermediate between semantic models and end-users.

Of course a good amount of work has been done in the past in this direction. For instance, ontology development editors were implemented (e.g., Protégé¹ [110] and the NeOn Toolkit [181]), Web search engines to look for semantic resources were launched (e.g., Sindice² [137] and Watson³ [41]), and semantic desktop applications were released (e.g., SemNotes⁴ [60]). However, what the Semantic Publishing community urgently needs are tools that assist people who are not expert in semantic technologies in dealing with and publishing semantic data.

Usually, this activity is composed by the following steps:

1. once found ontologies suitable for the particular domain of interest, people need (or want) to *understand* these models spending the minimum amount of effort;
2. then, people *develop* new models when existing vocabularies/ontologies are not able to fully describe the domain in consideration. The development process

¹Protégé: <http://protege.stanford.edu>.

²Sindice: <http://sindice.com>.

³Watson: <http://watson.kmi.open.ac.uk>.

⁴SemNotes: <http://smile.deri.ie/projects/semn>.

should plan the interaction with domain experts and end-users so as to produce a model that address the domain in consideration as best it can;

3. finally, once agreed on the model to use, people have to *add* data according to that model and, eventually, to *modify* those data in the future.

Each of these four operations – understanding, developing, adding and modifying – should be supported by proper interfaces that simplify the work of people who usually are not expert in ontology-related formalisms and Semantic Web technologies.

In this chapter, I describe my personal contribution in this direction. I introduce four different tools I developed so as to help users when dealing with Semantic Web technologies. Namely:

- the *Live OWL Documentation Environment (LODE)* is a Web service that creates HTML documentation of ontologies starting from their sources;
- the *Key Concept Visualiser (KC-Viz)* is a plugin of the NeOn Toolkit that helps users in ontology sense-making tasks. It allows one to visualise and browse big ontologies starting from their most representative classes (i.e., the *key concepts* [141]);
- the *Graphical Framework For OWL Ontologies (Graffoo)* allows one to create graphical formalisations of OWL ontologies according to a palette of widgets designed to be understood easily and learned quickly;
- the *Generator of Advanced Forms and Friendly Editor (Gaffe)* is an application that takes, as input, an ontology describing a domain and another ontology mapping domain entities to form widgets, and makes forms to add and modify data according to the domain ontology.

5.1 LODE: generating HTML documentation from ontologies

Usually, the first activity performed when someone wants to understand the extent of a particular ontology is to look for its human-readable documentation. A large number of ontologies, especially those used in the Linked Data, have a very good and comprehensive Web page describing theoretical backgrounds and developed entities.

Problems arise when we look at underdeveloped models, since natural language documentation is usually published only when an ontology becomes stable. This approach is justifiable: writing a proper documentation costs a big effort and changing it every time the ontology is modified can be problematic.

An additional complication is also given by the availability of several “stable” ontologies that do not have any document describing them. Thus, the only way to get a sense of them is to use an ontology editor so as to explore their logical axioms. This approach may create problems to a person approaching the ontology world for the very first time. First, he/she has to download and install the ontology editor, if it is not already present on his/her machine. Second, he/she must learn the editor, in order to use it. Finally he/she can try to get a sense of the ontology loading it in the editor.

Obviously the accomplishment of the three previous steps is very time-consuming. In order to address this issue, standalone tools and Web applications have been developed, as illustrated in Section 2.3.1. However they lack proper and quick mechanisms for the conversion of all the ontology axioms into a human-readable documentation.

So as to address this issue, I developed the *Live OWL Documentation Environment*⁵ (*LODE*). It is a service that automatically extracts classes, object properties, data properties, named individuals, annotation properties, meta-modelling (punning), general axioms, SWRL rules and namespace declarations from any well-formed OWL or OWL 2 ontology, and renders them as ordered lists, together with their textual definitions, in a human-readable HTML page designed for browsing and navigation by means of embedded links.

LODE is basically based on an XSLT stylesheet that takes RDF/XML linearisation of an ontology produced through the OWLAPI⁶ [94] as input and converts it into an HTML representation. If the target ontology is already linearised in that form, it is possible to call the service directly specifying its URL (i.e., “<http://www.essepuntato.it/lode/>”) followed by the complete URL of the ontology, for instance:

```
http://www.essepuntato.it/lode/http://www.essepuntato.it
/2008/12/earmark
```

In the following subsections I introduce the most important features of LODE.

5.1.1 What axioms are used to create the documentation

Primarily, LODE uses the most common annotation properties used for the description of entities, in particular⁷: *dc:contributor*, *dc:creator*, *dc:date*, *dc:description*, *dc:rights*, *dc:title*, *dcterms:contributor*, *dcterms:creator*, *dcterms:date*, *dcterms:description*, *dcterms:rights*, *dcterms:title*, *owl:versionInfo*, *rdfs:comment*, *rdfs:isDefinedBy*, *rdfs:label*. LODE adopts the following rules when transforming those annotations in HTML documentation:

⁵LODE, the Live OWL Documentation Environment: <http://lode.sourceforge.net>.

⁶OWLAPI: <http://owlapi.sourceforge.net>.

⁷The prefixes *dc*, *dcterms*, *owl* and *rdfs* in the following list respectively refers to “<http://purl.org/dc/elements/1.1/>”, “<http://purl.org/dc/terms/>”, “<http://www.w3.org/2002/07/owl#>” and “<http://www.w3.org/2000/01/rdf-schema#>”.

- in presence of Dublin Core annotations defined according to both DC Metadata Elements [64] and DC Metadata Terms [63], the formers have precedence;
- dates (i.e., *dc:date* and *dcterms:date*) written according to the related XML Schema datatype (i.e., *yyyy-mm-dd*) are automatically transformed in *dd/mm/yyyy*⁸;
- agents (i.e., *dc:creator*, *dc:contributor*, *dcterms:creator* and *dcterms:contributor*) are rendered either as strings or as clickable URL according to their type, i.e. literal and resource respectively;
- descriptions (i.e., *dc:description* and *dcterms:description*) are rendered either as strings or as media objects according to their type, i.e. literal and resource respectively;
- comments (i.e., *rdfs:comment*) and descriptions (i.e., *dc:description* and *dcterms:description*) represent respectively abstracts and detailed descriptions of entities;
- labels (i.e., *rdfs:label*) are used to refer to all the entities of the ontology instead of their URLs;
- all the entity names are always coupled with descriptive strings (i.e., “c”, “op”, “dp”, “ap” and “ni”) according to their types (i.e., class, object property, data property, annotation property and named individual).

In Fig. 5.1 on the facing page and Fig. 5.2 on page 160 is shown how these annotations are rendered – I use the EARMARK Ontology introduced in Section 3.1 in the rendering examples.

Beside the annotations, LODE converts all the other axioms of the ontology into Manchester Syntax definitions [95], as shown in Fig. 5.3 on page 160. I prefer to use this syntax rather than others since it is the most human-comprehensible syntax for ontological axioms.

Ontological axioms are rendered in particular grey boxes, one for each entity declared in the ontology. The axioms taken into account by LODE refer to: super-class and super-property, equivalent class and property, disjoint class and property, property domain and range, property chain, keys, object/data property assertion, type, imported ontology, generic axiom and SWRL rule. Moreover, LODE automatically enriches those definitions adding information about sub-classes, domain/range properties of classes, sub-properties and entity meta-modelling.

⁸I am currently working on an extensible template-based mechanism to enable the specification of date formats according to user’s needs (e.g., *dd MonthName yyyy*).

The screenshot shows the EARMARK Ontology page. At the top, there's a logo for 'Powered by LODER' and the title 'EARMARK Ontology'. Below the title, there's a box containing the URL <http://www.essepuntato.it/2008/12/earmark>. The page lists several pieces of metadata:

- Date:** 24/02/2011 → `dc:title "EARMARK Ontology" ; dc:date "2011-02-24" ;`
- Current version:** 1.8.1 → `owl:versionInfo "1.8.1" ;`
- Author:** Silvio Peroni → `dc:creator "Silvio Peroni" ; dc:contributor "Angelo Di Iorio", "Fabio Vitali" ;`
- Contributor:** Angelo Di Iorio, Fabio Vitali → `owl:imports <http://www.essepuntato.it/2010/05/ghost> ; dc:rights "This ontology is distributed..." .`
- Imported ontologies:** <http://www.essepuntato.it/2010/05/ghost> (visualize it with LODE)
- Other visualizations:** [Manchester Ontology Browser](#)

At the bottom, a note states: 'This ontology is distributed under a Creative Commons Attribution License - http://creativecommons.org/licenses/by/3.0'

Abstract

Extremely Annotational RDF Markup (EARMARK) is a meta-syntax for non-embedded markup that can be used for stand-off annotations of textual content with fully W3C-compliant technologies.

This document represents the EARMARK ontology definition of the shell classes i.e., all those classes that must be used to define all the elements, attributes, comments, text nodes and hierarchies among them.

Table of contents

1. [Introduction](#)
2. [Classes](#)
3. [Object properties](#)
4. [Data properties](#)
5. [Annotation properties](#)
6. [General axioms](#)
7. [Namespace declarations](#)

<http://www.essepuntato.it/2008/12/earmark>
rdfs:comment "Extremely Annotational RDF Markup
(EARMARK) is a meta-syNTAX..." .

Figure 5.1: The beginning of the Web page generated through LODE starting from the EARMARK Ontology, annotated with OWL assertions in Turtle showing how they are rendered in HTML.

5.1.2 Special parameters to call the service

LODE can be invoked with a number of optional parameters so as to limit or extend the final documentation produced. For instance, it is possible to take into account all the entities in the ontology closure and/or the inferred axioms. The following pseudo-URL describes how to call LODE:

`http://www.essepuntato.it/lode/optional-parameters/ontology-url`

In particular:

- **www.essepuntato.it/lode** is the URL to call the service;
- **ontology-url** is the full “`http://...`” URL of the OWL ontology that will be processed by the service. It must be always the last item of the pseudo-URL, and may be preceded by one or more (slash-separated) parameters.

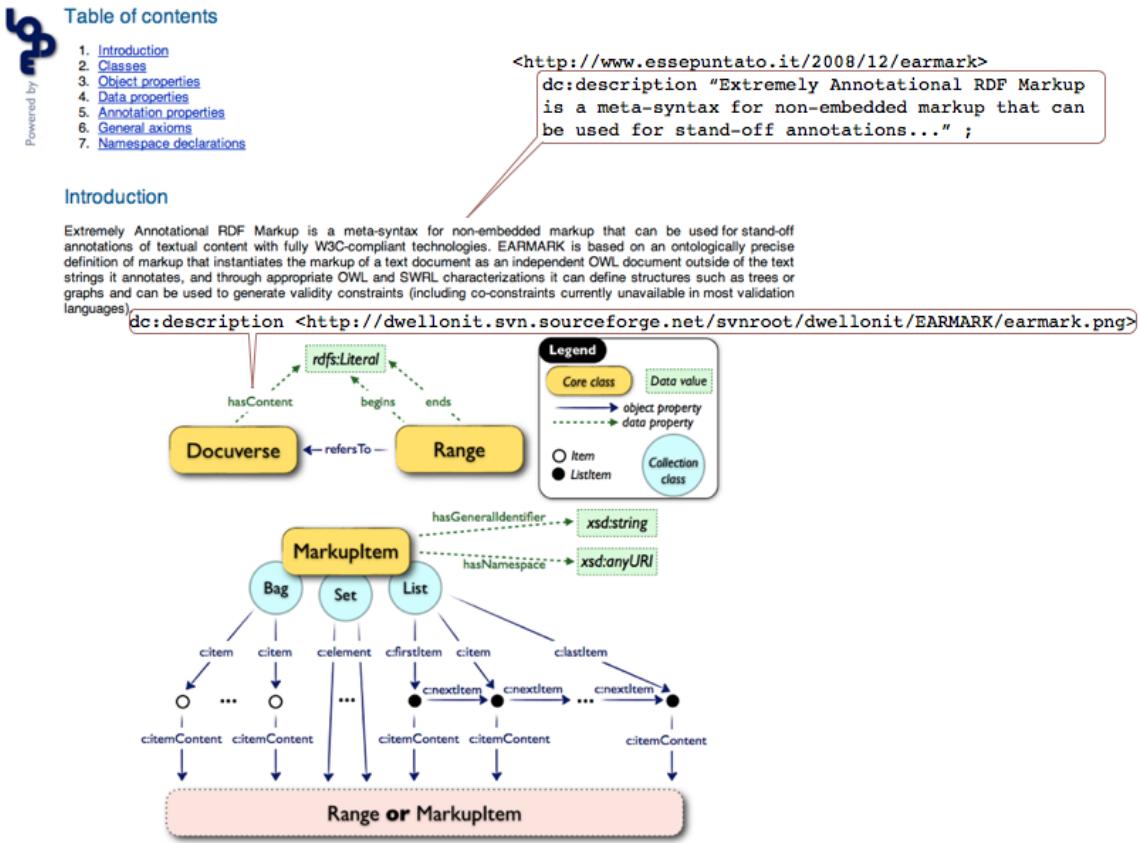


Figure 5.2: Two possible kinds of descriptions: pure string (for literals) and media object (for resources).

earmark:Range
rdfs:label "range" ;
rdfs:comment "An entity referring to any ...".

range
 IRI: <http://www.essepuntato.it/2008/12/earmark#Range>
 An entity referring to any text of a docuverse lying between two locations.
 is equivalent to
 • **(refers to^{op} some docuverse)** and **(begins at^{dp} some rdfs:Literal)** and **(ends at^{dp} some rdfs:Literal)**
 has sub-classes
[pointer range](#), [xpath range](#)
 is in domain of
[begins at^{dp}](#), [ends at^{dp}](#), [refers to^{op}](#)

additional “dc:description” annotations will be added here

Figure 5.3: How entities (classes, properties and individuals) are rendered by LODE.

Fig. 5.4 on the next page illustrates the alternative ways to build the URL to call LODE and the related modules used. The optional slash-separated parameters are described in the following sub-sections.

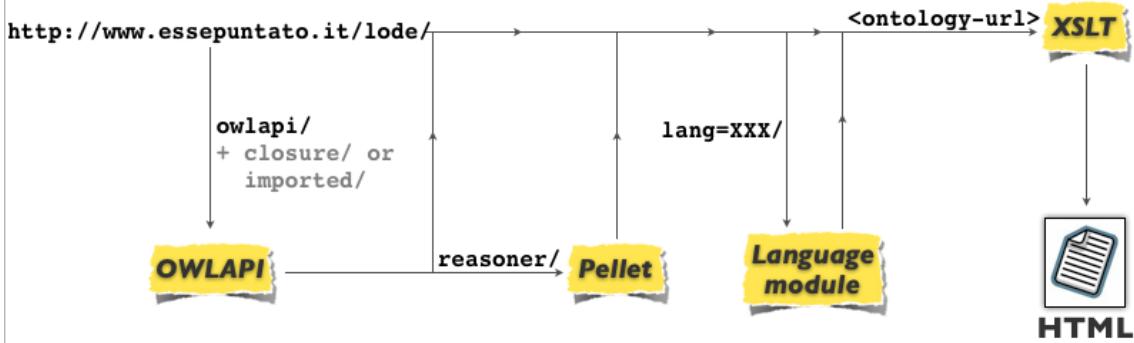


Figure 5.4: All the possible ways, according to specific needs, for making a request to LODE.

Parameter “owlapi”

When this optional parameter is specified, the ontology defined in *ontology-url* will be pre-processed via OWLAPI [94], in order to linearise it in the RDF/XML format accepted by LODE. This parameter is always strongly recommended: it allows LODE to process ontologies implemented in all the formats supported by the OWLAPI.

Parameter “imported”

When this optional parameter is specified, the axioms in the imported ontologies of *ontology-url* are added to the HTML description of the ontology. This parameter implicitly specifies the *owlapi* parameter.

Parameter “closure”

When this optional parameter is specified, the transitive closure given by considering the imported ontologies of *ontology-url* is added to the HTML description of the ontology. This parameter implicitly specifies the *owlapi* parameter. If both the parameters *closure* and *imported* are specified (in any order), *imported* will be preferred.

Parameter “reasoner”

When this optional parameter is specified, the inferred axioms of *ontology-url* (through the Pellet reasoner [170]) will be added to the HTML description of the ontology. This parameter implicitly specifies the *owlapi* parameter. Note that, depending on the nature of the ontology to process, this computationally intensive function can be very time-consuming.

Parameter “lang”

When this optional parameter is specified, the selected language will be used as preferred language instead of English when showing annotations of *ontology-url*. It must be followed by an “=” and the abbreviation of the language to use. E.g.: “lang=en” for English, “lang=it” for Italian, “lang=fr” for French, etc.

5.1.3 URI fragments

LODE offers intuitive mechanisms to refer to particular ontological entities within the HTML documentation, according to the URL of the entity in consideration. The following extension of the pseudo-URL introduced in Section 5.1.2 defines how to refer to a particular entity of an ontology:

```
http://www.essepuntato.it/lode/optional-parameters/ontology-url #entity
```

For instance, to generate the documentation of FaBiO (Section 4.1) and then jumping directly to the point where the resource “<http://purl.org/spar/fabio/Article>” is described, I need to invoke LODE as follows:

```
http://www.essepuntato.it/lode/http://purl.org/spar/fabio#
http://purl.org/spar/fabio/Article
```

This request can be simplified if I look for descriptions of entities defined as *fragment* of the ontology URL, such as the entity *Element* of the EARMARK ontology – i.e., “<http://www.essepuntato.it/2008/12/earmark#Element>”. In this particular case, I can use either the entire entity URL as illustrated previously or the entity local name only, as shown as follows:

```
http://www.essepuntato.it/lode/http://www.essepuntato.it
/2008/12/earmark#Element
```

5.1.4 Content negotiation via *.htaccess*

LODE can be freely used by third parties, as described in its documentation. In particular, it may be very useful in conjunction with content negotiation mechanisms to display a human-readable version of an OWL ontology when the user accesses the ontology using a web browser, or to deliver the OWL ontology file itself when the user accesses the ontology using an ontology development tool such as *Protégé* [110] or the *NeOn Toolkit* [181]. For instance, an implementation of such a content negotiation is given in [17] by using the *.htaccess* file:

```

AddType application/rdf+xml .rdf

# Rewrite engine setup
RewriteEngine On

# Rewrite rule to serve HTML content
RewriteCond %{HTTP_ACCEPT} !application/rdf\+xml.*(text/html |
    application/xhtml\+xml)
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/.*
RewriteRule ^ontology$ http://www.essepuntato.it/lode/http://
    www.mydomain.com/ontology [R=303,L]

# Rewrite rule to serve RDF/XML content if requested
RewriteCond %{HTTP_ACCEPT} application/rdf\+xml
RewriteRule ^ontology$ ontology.owl [R=303]

# Choose the default response
RewriteRule ^ontology$ ontology.owl [R=303]

```

LODE can be seen in action by opening, in a Web browser, any of ontologies presented in this thesis. For instance, the URL “<http://purl.org/spar/fabio>” resolves, by content negotiation, to display the LODE HTML version of the FaBiO ontology with the URL “<http://www.essepuntato.it/lode/http://purl.org/spar/fabio>”. As shown previously, a similar syntax can be used to display the LODE visualization of any other OWL ontology.

5.2 KC-Viz, a tool for visualising and navigating ontologies

Sometimes the HTML documentation may not be enough to properly understand an ontology. This is particularly true for very large ontologies that obviously make a person lost when he/she tries to understand their overall structure. Having clear natural language documentation may not adequately support users. In this case, using abstraction mechanisms that try to take a first salient picture of the overall organisation of the ontology can be extremely useful. Starting from this summary, that may contain for instance the most representative concepts of the ontology, one can start to *make sense* of the ontology itself.

When I speak about “making sense” of an ontology, I refer to a specific ontology engineering task, where the user is primarily concerned with understanding the contents and overall structure of the ontology, i.e., acquiring an overview of the

concepts covered by the ontology and the way they are organized in a taxonomy. Thus, the sense-making process includes:

- understanding the overall size⁹ and shape¹⁰ of the ontology;
- identifying the main components of the ontology and the typical exemplars of these components. Informative exemplars can also help the user to predict the siblings of the class (i.e., the exemplar) in question, thus playing a summarisation role not just with respect to its subtree, but also with respect to its siblings.

Of course, users need to be supported by interfaces and effective summarisation techniques when trying to make sense of large ontologies. This because, once an ontology is large enough, it is not possible to show its entire structure in the limited space provided by a computer screen and therefore a difficult trade-off needs to be addressed. On the one hand the information on display needs to be coarse-grained enough to provide an overview of the ontology, thus ensuring the user can maintain an overall mental model of the ontology.

In this scenario, an exploration process needs to be supported, where the user can effectively home in on parts of the ontology, thus changing the level of analysis, while at the same time not losing track of the overall organization of the ontology.

However, a problem affecting all the approaches discussed in Section 2.3.2 is that all of them essentially use geometric techniques to providing abstraction. In contrast with these approaches, human experts are able to provide effective overviews of an ontology, simply by highlighting the key areas covered by the ontology and the classes that best describe these areas. In particular, the work reported in [141], that I briefly introduce in Section 5.2.1, provides empirical evidence that there is a significant degree of agreement among experts in identifying the main concepts in an ontology. It also shows that the algorithm presented for *key concept extraction* (*KCE*) is able to retrieve a summarisation of the ontology maintaining the same level of agreement with the experts, as they have among themselves [141].

In this section, I introduce KC-Viz, the *key concept visualiser* [131] [128] [129] [130], a tool for ontology visualisation and browsing I developed in collaboration with professor Enrico Motta and his research group (Open University, UK). Building on its ability to abstract out from large ontologies through the KCE algorithm, KC-Viz provides a rich set of navigation and visualization mechanisms, including flexible

⁹Given a node in the ontology, its *size* is the total number of its direct and indirect subclasses.

¹⁰Given a node in the ontology, its *shape* is an indication of the organization of the subclasses. For instance, an ontology (or part of it) can have a horizontal (i.e., many subclasses and few levels of depth), or a vertical (i.e., many inheritance levels and only a few subclasses at each level) shape [183]. Understanding the shape of an ontology (or part of it) also means to understand whether it is balanced, indicating that all parts of the (sub-)ontology in question have been developed to a similar extent, or unbalanced, possibly indicating that some parts of the (sub-)ontology are less developed than others.

zooming into and hiding of specific parts of an ontology, history browsing, saving and loading of customized ontology views, as well as essential interface customization support, such as graphical zooming, font manipulation, tree layout customization, and other functionalities. KC-Viz is a core plugin of the NeOn Toolkit [181].

In the following sections I give a general view of the KCE algorithm principles and of the main features implemented in KC-Viz. Moreover, I also report on additional findings gathered through questionnaires, which offer a number of other insights.

5.2.1 Key Concept Extraction

Informally, *key concepts* can be seen as the best descriptors of an ontology, i.e., information-rich concepts, which are most effective in summarizing what an ontology is about. In [141] are considered a number of criteria to identify the key concepts in an ontology, introduced as follows.

Natural Category. KCE uses the notion of *natural category* [153]¹¹, to identify concepts that are information-rich in a psycho-linguistic sense. This notion is approximated by means of two operational measures: *name simplicity*, which favours concepts that are labelled with simple names, and *basic level*, which measures how “central” a concept is in the taxonomy of an ontology.

Density. The notion of *density* highlights concepts that are information-rich in a formal knowledge representation sense, i.e., they have been richly characterized with properties and taxonomic relationships. The density is decomposed in two sub-criteria, *global* and *local* density. While the global measures are normalised with respect to all the concepts in the ontology, the local ones consider the relative density of a concept with respect to its surrounding concepts. The aim here is to ensure that “locally significant” concepts get a high score, even though they may not rank too highly with respect to global measures.

Coverage. The notion of *coverage* is used to ensure that no important part of the ontology is neglected, by maximizing the coverage of the ontology with respect to its taxonomic relationships (*rdfs:subClassOf*).

Popularity. The notion of *popularity*, drawn from lexical statistics, is introduced as a criterion to identify concepts that are likely to be most familiar to users. Similarly to the density criterion, the popularity is decomposed in two sub-criteria, global and local popularity.

Each of these seven criteria produces a score for each concept in the ontology and the final score assigned to a concept is a weighted sum of the scores resulting

¹¹Eleanor Rosch has been one of the pioneers of the *prototype theory*, i.e. a cognitive approach for categorisation where some members of a particular category are considered more “central” than others. Rosch’s works on categorisation have been the basis of significant following researches, such as the notion of *conceptual spaces* by Gardenfors [77] and of *conceptual metaphors* by Lakoff [115] [114]. In this work I have used an operative definition of Rosch’s *natural category* derived from her notion of *basic level*.

from individual criteria. As described in [141], which provides a detailed account of our algorithm and a formal definition of the criteria it employs, the approach has been shown to produce ontology summaries that correlate significantly with those produced by human experts.

It is important to emphasize that in the current online version of KCE¹² and in KC-Viz the popularity criterion is not longer used, because of its computational cost on large ontologies. However, on the basis of the analytical studies described in [117], the weights associated with the other criteria has been parameterised, to produce a vastly more efficient version, while at the same time maintaining the same level of compliance with respect to the available human-generated benchmarks as the version of the algorithm presented in [141].

5.2.2 KC-Viz main features

Running KC-Viz on an ontology (e.g., FaBiO) for the first time produce an initial visualisation of the network of classes, which includes concepts at different levels in the class hierarchy. The visualisation in Fig. 5.5 includes 26 concepts because I have set the size of our ontology summary to 25 and the algorithm has automatically added the most generic concept, *owl:Thing*, to ensure that the visualization displays a connected graph.

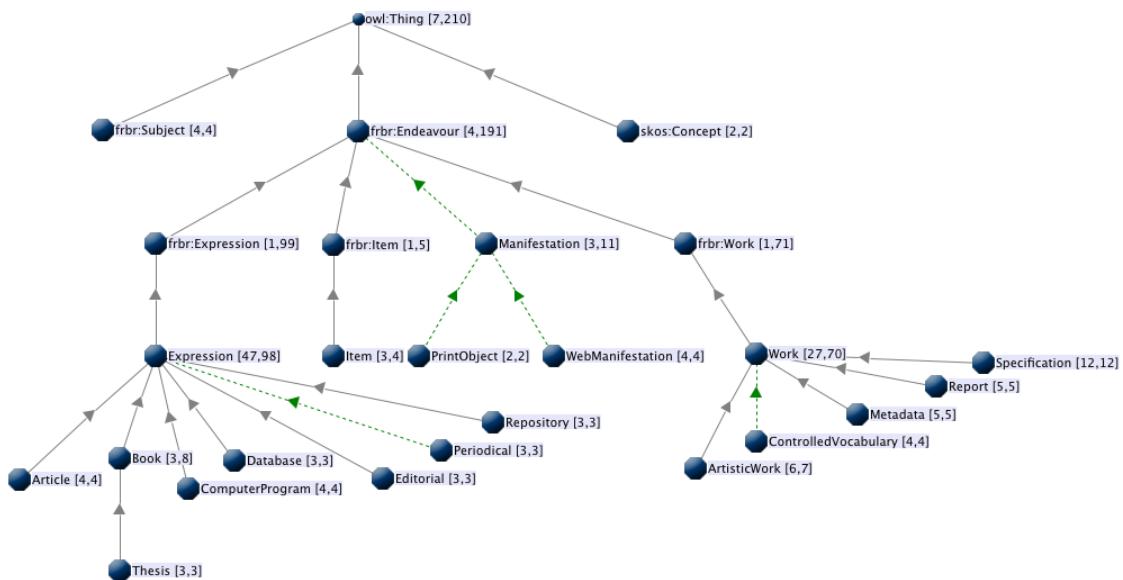


Figure 5.5: The summarisation made by KC-Viz after its first application on an ontology.

¹²KCE Live: <http://www.essepuntato.it/kce>.

If we wish to display more or less succinct graphs, we can do so by changing the size of the ontology summary. The solid grey arrows in Fig. 5.5 on the preceding page indicate direct *rdfs:subClassOf* links, while the dotted green arrows indicate indirect *rdfs:subClassOf* links.

Yet starting from the first visualisation, KC-Viz provides the size of the tree under a particular class, which is indicated by a pair of integers, referring to the number of direct and indirect subclasses. For instance, Fig. 5.5 on the facing page tells us that class *Endeavour* has 4 direct subclasses and 191 total subclasses (direct + indirect). Although more exploration is obviously needed to get a thorough understanding of the contents of the FaBiO ontology, it can be argued that as a first step, the visualisation shown in Fig. 5.5 on the preceding page already provides a rather effective starting point for the ontology sense-making process.

Description of nodes and arcs

By hovering the mouse over an element (node or edge) of the tree, as shown in Fig. 5.6 on the following page, a tooltip is popped up with some information about the element itself: the chain of *rdfs:subClassOf* relations or of domain/range links for edges, while information about number of (shown and total) sub- or super- classes for nodes. For the latter elements, there exists also a section, called “Reason”, which indicates how a node (i.e., a class) fares with respect to the criteria used to determine key concepts. KC-Viz uses a 0-5 scale of labelled values (0 very poor, 1 poor, 2 fair, 3 good, 4 strong, 5 very strong). Moreover, when hovering simultaneously on different arcs, all their descriptions are merged in a single tooltip.

Expansion

If we click right on a class displayed in KC-Viz, in this case *Work*, we obtain a menu that includes options for inspecting, expanding, and hiding a class. If we select “Expand”, a menu pops up, which provides a rich set of options for exploring the subtree under class *Work*, as shown in Fig. 5.7 on page 169. In particular, the following four options for customizing the expansion algorithm are presented to the user:

- whether to explore following taxonomic relations, other relations (through domain and range), or any combination of these;
- whether or not to make use of the ontology summarisation algorithm, which in this case will be applied only to the subtree of class *Work*;
- whether or not to limit the range of the expansion – e.g., by expanding only to 1 or 2 levels;

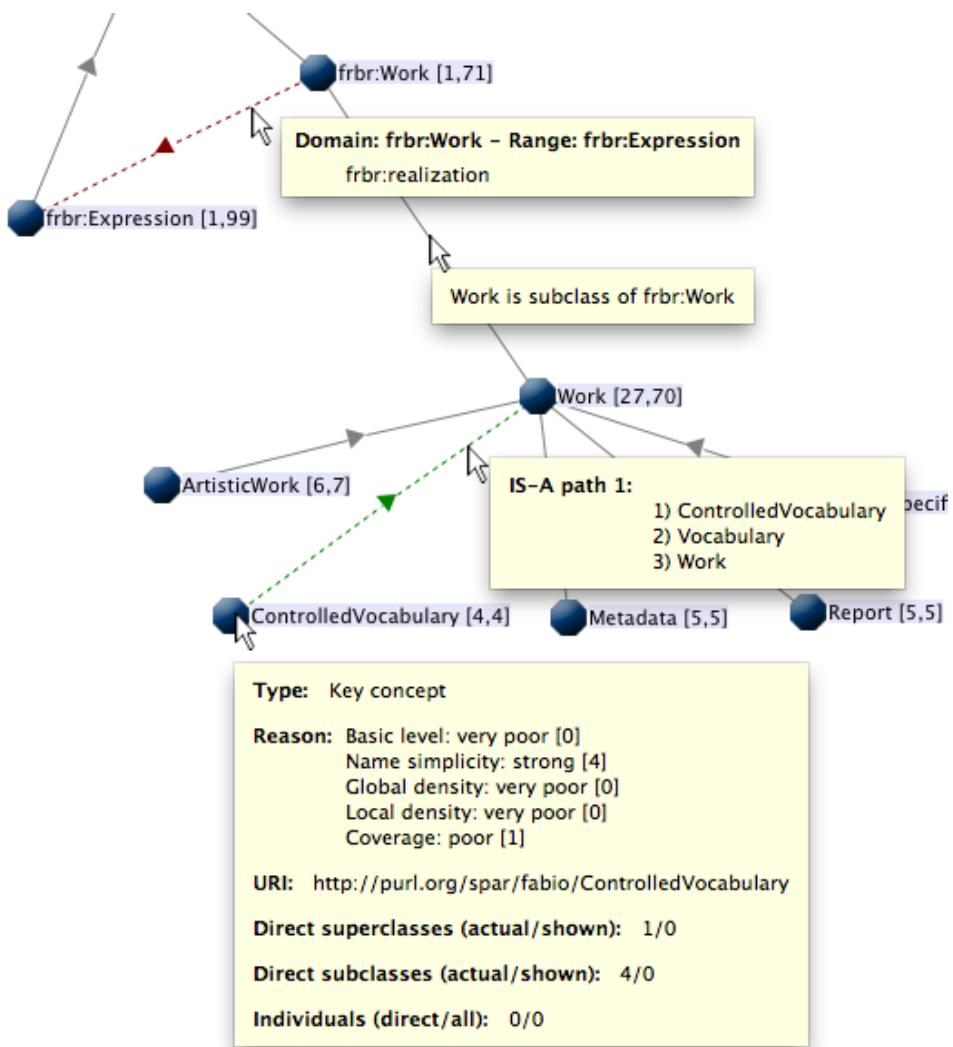


Figure 5.6: Tooltips that appear hovering nodes and edges.

- whether to display the resulting visualization in a new window (“Hide”), or whether to add the resulting nodes to the current windows¹³.

Hiding

After right-clicking on a node, two options, “Hide” and “Hide others” implement a flexible mechanism for hiding nodes, as shown in Fig. 5.8 on page 170. If we select

¹³In the latter case, some degree of control is given to the user with respect to the redrawing algorithm, by allowing her to decide whether or not to limit the freedom of the graph layout algorithm to rearrange existing nodes. This is particularly useful in those situations where expansion is meant to add only a few nodes, and the user does not want the layout to be unnecessarily modified – e.g., because she has already manually rearranged the nodes according to her own preferences.

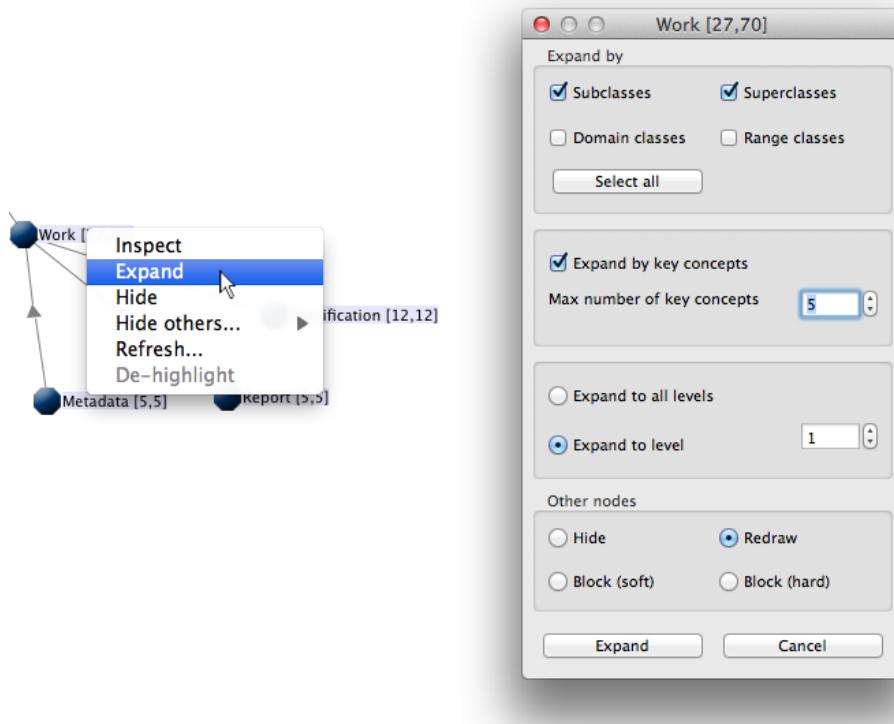


Figure 5.7: The menu popped up after clicking on the “Expand” option.

“Hide”, a menu pops up, which provides a rich set of options for hiding the class selected and its subtree. By selecting “Hide others”, one can choose to hide all the classes but the selected one and, optionally, its subclasses.

Refresh visualization

The pop-up obtained after right-clicking inside the KC-Viz view has a new option called “Refresh”, which allows the user to re-sync the visualization with respect to changes in the model, which may have occurred since the visualization was produced.

Integration with NeOn

KC-Viz is integrated with the core components of the NeOn Toolkit, including the Entity Properties View and Ontology Navigator. This means that it is possible to click on nodes in KC-Viz and highlight them in these components (option “Inspect”), as well as clicking on items shown in the Ontology Navigator and adding them to the visualization in KC-Viz, as shown in Fig. 5.9 on page 171.

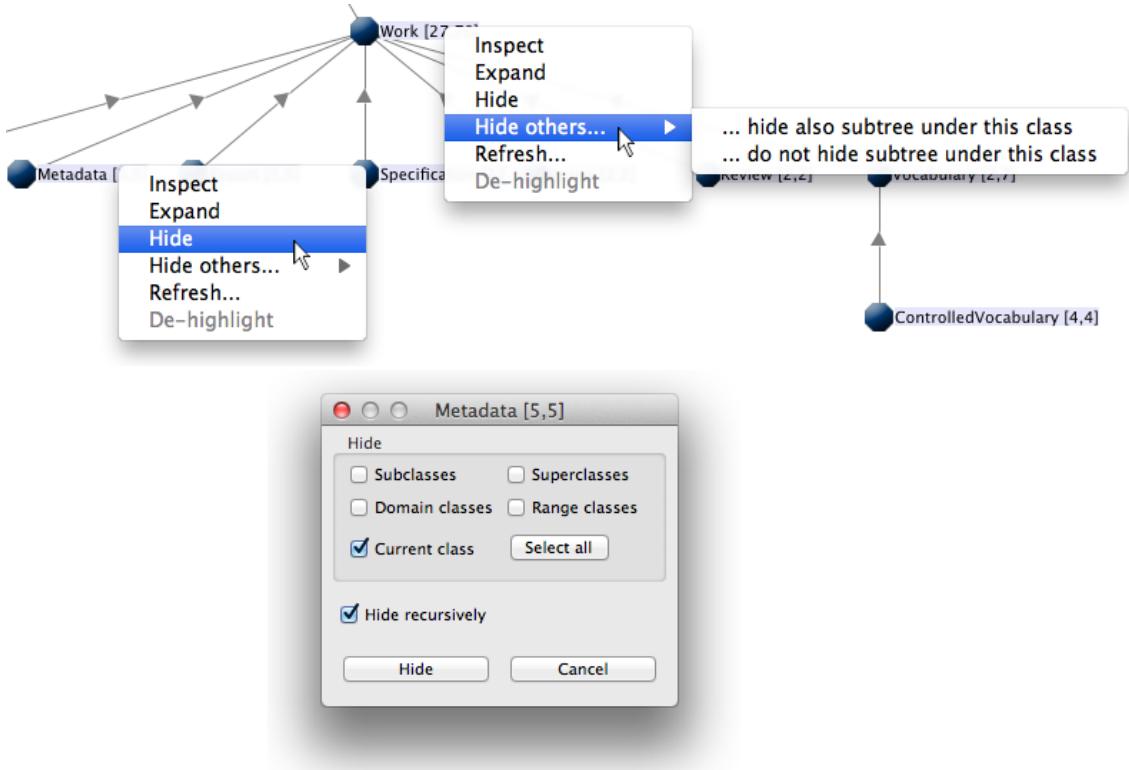


Figure 5.8: The two options for hiding concepts: “Hide”, applied on the class *Metadata*, and “Hide others...” used on the class *Work*.

Dashboard

A dashboard containing buttons for acting on the current KC-Viz panel is positioned immediately above the visualisation panel, as shown in Fig. 5.10 on the facing page.

In the dashboard there are history buttons that allow the user to move back and forth through the history of KC-Viz operations. Each operation, or move, can be distinguished in macro (extraction and hiding) and micro (re-layout, axis adjustment, node movement) moves. It is important to notice that micro moves can be cancelled/re-applied only if they are not preceded/followed by macro moves in the operation history. For instance, considering the sequence “Mac1 mic1 mic2 Mac2 mic3...” and supposing to be visualizing in KC-Viz the window after mic2 and before Mac2, a user can decide:

- to cancel mic2;
- to cancel Mac1;
- to re-apply Mac2.

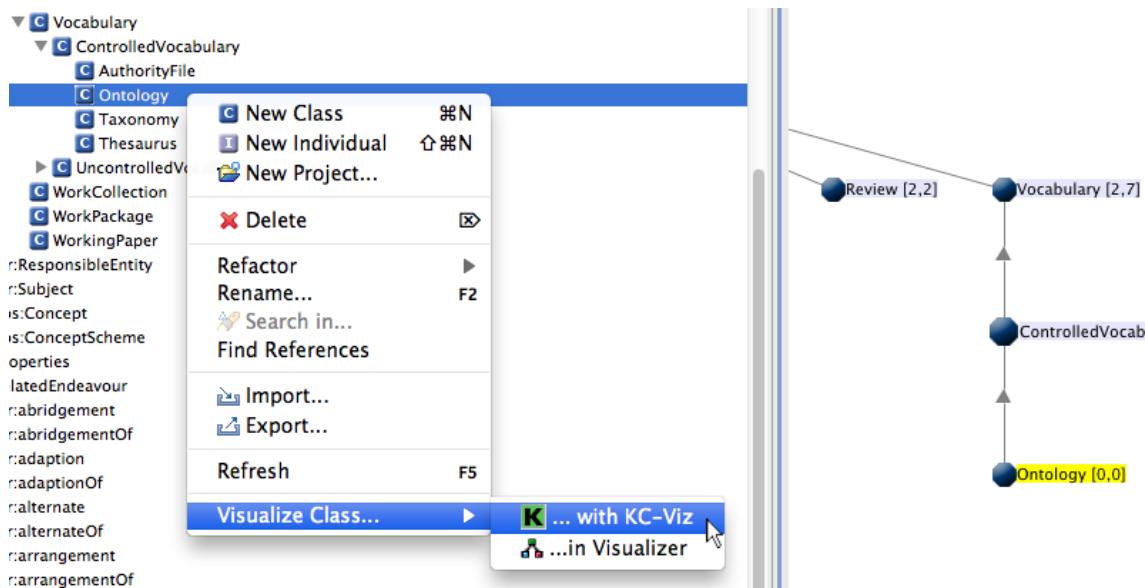


Figure 5.9: The option “Visualize Class with KC-Viz” to highlight (and eventually add) the class in the current KC-Viz panel.

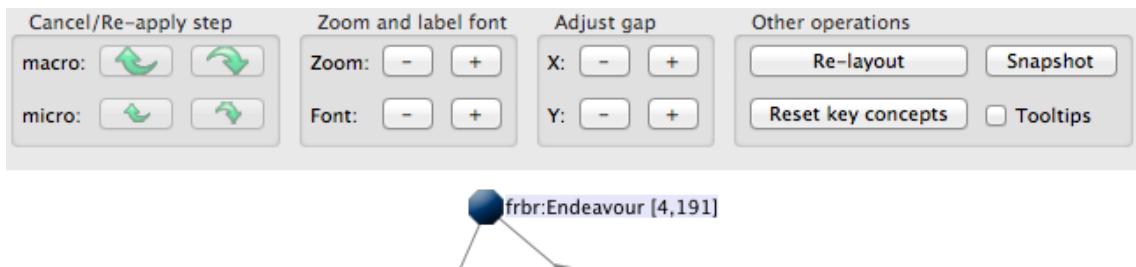


Figure 5.10: The dashboard which allows the user to move back and forth through the history of KC-Viz operations, to modify the formatting of the layout, and to save the current display to a file, among other things.

Moreover, KC-Viz provides an essential interface customization support, such as graphical zooming, font manipulation and tree layout customization. For the latter, after clicking on the button “Re-layout” in the KC-Viz toolbar, a sub-menu is opened, which asks the user whether he/she wants to redraw the current visualization using a top-bottom orientation or a left-right orientation.

Finally, through the button “Snapshot” inside the dashboard, it is possible to save the current display to a file or to load/delete a previously stored one.

Preferences

KC-Viz has a preferences panel, shown in Fig. 5.11, which allows the user to set defaults for the most common operations and also enables him/her to switch to a more efficient (but sub-optimal¹⁴) algorithm when dealing with very large ontologies. Moreover, all the weights of the formulas used in the KCE algorithm are customizable by the user. In particular, the preference panel allows users to change the relative weights of the different criteria used to calculate the overall score for each class in an ontology. Customisation options are also provided for all the criteria.

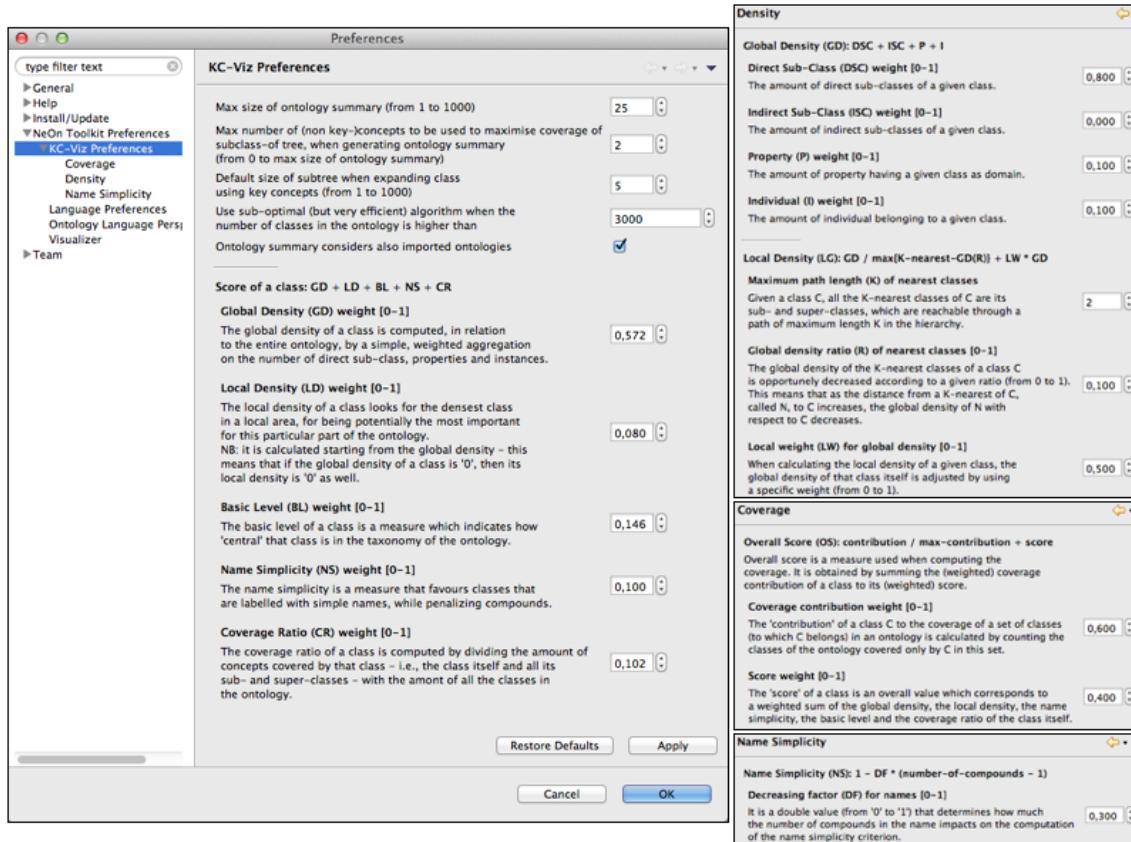


Figure 5.11: The preference panel of KC-Viz.

5.2.3 Empirical evaluation

In order to gather initial data about the performance of KC-Viz, [128] introduces a preliminary empirical evaluation, which required 21 subjects to perform four ontology engineering tasks (max. 15 minutes per task), involving ontology exploration.

¹⁴When this preference is enabled the set of key concepts returned by the algorithm may not guarantee the best possible *coverage* of the ontology.

The tasks given to the subjects are shown in Table 5.1. This set of tasks was designed to ensure coverage of different exploration strategies, which are typically required in the context of a sense-making activity.

Table 5.1: Ontology Engineering Tasks.

T1	Which class has the highest number of direct subclasses in the ontology?
T2	What is the most developed (i.e., has the biggest subtree) subclass of class <i>Quantity</i> found in the ontology at a concrete level of granularity (i.e., do not consider abstract classes which have the term ‘quantity’ in their id)?
T3	Find three subclasses of <i>Agent</i> , at the most abstract level possible (under <i>Agent</i> of course), which are situated at the same level in the hierarchy as each other, and are also subclasses of <i>CorpuscularObject</i> .
T4	We have two individual entities (a particular copy of the book <i>War&Peace</i> and a particular 5p coin). Find the most specific classes in the ontology, to which they belong, say P1 and P2, and then identify the most specific class in the ontology, say C1, which is a superclass of both P1 and P2 – i.e., the lowest common superclass of both P1 and P2.

The 21 subjects were randomly allocated to three different groups, labelled A, B, and C, where each group used a particular configuration of ontology engineering tools. In particular members of group A carried out the tasks using the NeOn Toolkit v2.5, without any visualization support. More precisely, they were only allowed to use the search functionality, the *Ontology Navigator* and the *Entity Properties View*. The role of this group was to provide a baseline to the experiment, providing us with some data on how effectively people can tackle ontology exploration tasks, without any visualization support.

The members of Group C were asked to solve the tasks using KC-Viz, together with the search functionality provided by the NeOn Toolkit. To ensure a separation between groups A and C, members of the latter group were explicitly forbidden from using the Ontology Navigator for exploration, although they were allowed to use it as an interface between the search facility in the NeOn Toolkit and KC-Viz.

Finally, the members of Group B carried out the tasks using the Protégé 4 environment, v4.1.0, in particular using the search functionality, the class browser and the *OwlViz* plugin. This configuration was chosen for three reasons:

1. to compare KC-Viz to a robust tool, widely used in concrete projects by members of the ontology engineering community, to maximize the value of the

experiment to the community;

2. while OwlViz uses the same node-link paradigm as KC-Viz, its design is rather different from KC-Viz; and
3. having considered the visualizers available in other state of the art ontology engineering tools, such as the NeOn Toolkit (*Kaon Visualizer*) and TopBraid (*Graph View*), OwlViz appears to provide a more user friendly and flexible functionality than the comparable ones available in TopBraid and the NeOn Toolkit.

Before the experiment, every subject filled a questionnaire, answering questions about his/her expertise in ontology engineering, knowledge representation languages, and with various ontology engineering tools, including (but not limited to) NeOn and Protégé. At the end of the experiment, every subject filled free text questions on the post-task questionnaire to provide views on the perceived strengths and weaknesses of the tool used by each subject. Additionally, subjects who did not use KC-Viz provided feedback following a demo. Note that none of the subjects had much direct experience with the ontology used for the tasks (i.e., the SUMO ontology¹⁵).

Out of 84 tasks in total ($4 * 21$), 71 were completed within the 15 minutes time limit, while 13 tasks were not completed, a 15.47% percentage failure. The 13 failures were distributed as follows: 5 in group A (NTK), 6 in group B (OwlViz), and 2 in group C (KC-Viz).

Fig. 5.12 on the next page shows the average time (in seconds) taken by each group in each task, as well as the total averages across groups and tasks¹⁶. As shown in the table, on each of the four tasks the fastest mean performance was with KC-Viz, whose overall mean performance was about 13 minutes faster than OWLViz, which in turn was about two minutes faster than NTK. Although not significant, the difference in total time taken across the four tasks with the three different tools appeared to be approaching significance, $F(2, 20) = 2.655$, $p = 0.098$.

The difference in performance across the three tools on Task 1, was statistically significant $F(2, 20) = 9.568$, $p < 0.01$. A Tukey HSD pairwise comparison revealed a significant difference between both KC-Viz and NTK ($p < 0.01$) and KC-Viz and OwlViz ($p < 0.01$). Although mean performance was faster for KC-Viz across the board, performance differences on the other three tasks did not reach statistical significance. Nevertheless these results suggest advantages for KC-Viz in supporting users in such realistic browsing and visualization tasks.

¹⁵The SUMO ontology: <http://www.ontologyportal.org/SUMO.owl>.

¹⁶For tasks not completed within the time limit, we consider a 15 minutes performance. This could be modified to consider ‘penalties’, such as a 5 minutes penalty for a non-completed task. However, adding such penalties does not lead to meaningful changes in the interpretation of the data, other than increasing the performance gap between the KC-Viz group and the others.

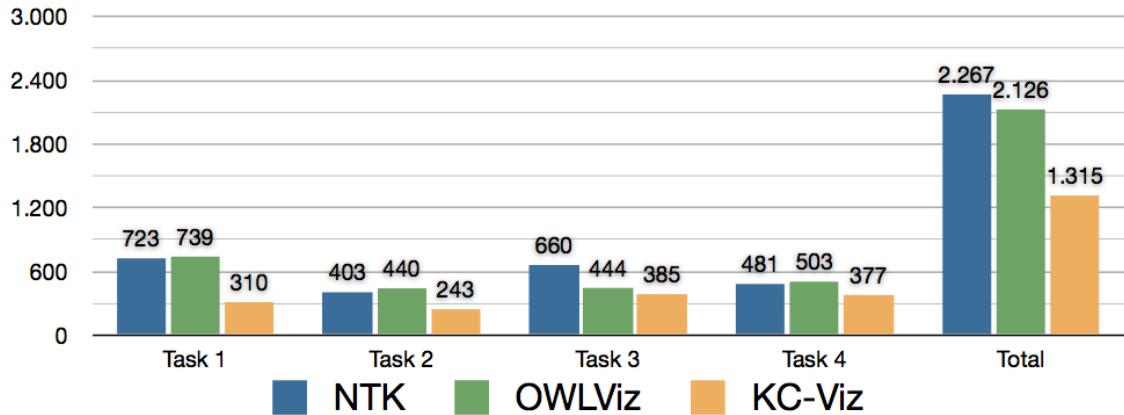


Figure 5.12: Performances (in seconds) for each task.

From questionnaire data, usability scores were calculated using the SUS formula [29] for each of the three conditions – see Table 5.2. The mean usability score was slightly higher for KC-Viz, though very similar across the three tools and not statistically significant.

Table 5.2: Usability scores.

NTK mean	OWLViz mean	KC-Viz mean
26,9	25,7	27,1

The free text questions on the post-task questionnaire were analysed through a grounded theory approach [18]. This approach was used to build categories of comments that either expressed positive feedback, offered criticism, or suggested improvements. Categories were discarded when they only contained comments from a single subject.

The three main categories of positive comments concerned:

- the flexible support provided by KC-Viz to manipulate the visual displays;
- the abstraction power enabled by the KCE algorithm;
- the value of the subtree summaries provided by KC-Viz.

These results are encouraging in the sense that they provide some initial indication that there is probably a direct causal link between the use of key concepts as an abstraction mechanism and the good performance of KC-Viz on the evaluation tasks, even though these were not designed specifically to map directly to KC-Viz features.

Of the three main categories of negative comments described in [128], only one still remains unsolved in the current version of KC-Viz: the lack of integration between KC-Viz and reasoning/query support in the NeOn Toolkit. Obviously, future versions of the tool will be opportunely extended to address this and other minor issues.

5.3 Graffoo, a framework for visual ontology modelling

Twenty years ago the only way Web users had to publish Web pages was to write HTML documents through text editors and, then, uploading them on the Web by means of apposite transfer protocols (e.g., FTP). Obviously people might not be self-sufficient to accomplish this task. A first step forward was done few years after with the introduction of WYSIWYG editors for HTML documents (e.g., Microsoft Frontpage). However, although these editors solved the issue of writing HTML pages through the adoption of user-friendly interfaces, a problem still persisted: where and how uploading those documents.

Everything changed, and in better, when the paradigm of creating and publishing HTML documents started to involve Web-based application rather than desktop applications. The introduction of blog platforms, wikis, CMSs and other social tools has generated an exponential increase of people writing for and publishing on the Web. The winning strategy has been characterised by two main points:

- to hide the complexity of the HTML markup behind Web interfaces;
- to reduce the entire publishing process to pressing a button.

Currently, some aspects of the Semantic Web, such as the creation of OWL ontologies, still continue to stay confined inside unpopular community areas. With that I do not want to say that these aspects are not important at all. Rather, I want to highlight that people cannot deal with some important semantic technologies unless having proper skills. Like the above-mentioned example about the creation and publication of HTML documents, Web users will start to write and publish OWL ontologies only when:

- there will exist proper interfaces that hide the intrinsic complexity of the OWL framework behind user-friendly GUI, and
- the publication process of OWL ontologies will be automatized and reduced to the press of a button.

In this section I describe a work that make a step forward towards the needs pointed out in the first of the above points. Along the lines of what I presented in Section 2.3.3, here I introduce the *Graphic framework for OWL ontologies*¹⁷ (*Graffoo*). It is a tool that can be used to present OWL ontologies, or sub-parts of them, as clear and easy-to-understand diagrams.

All the objects that can be used in a Graffoo diagram are shown in Fig. 5.13. These have been developed using the standard library of *yEd*¹⁸, a free diagram editor running on Windows, Mac and Linux. The *graphml* format version of those objects is also available¹⁹ and can be loaded as proper section in the *yEd* palette.

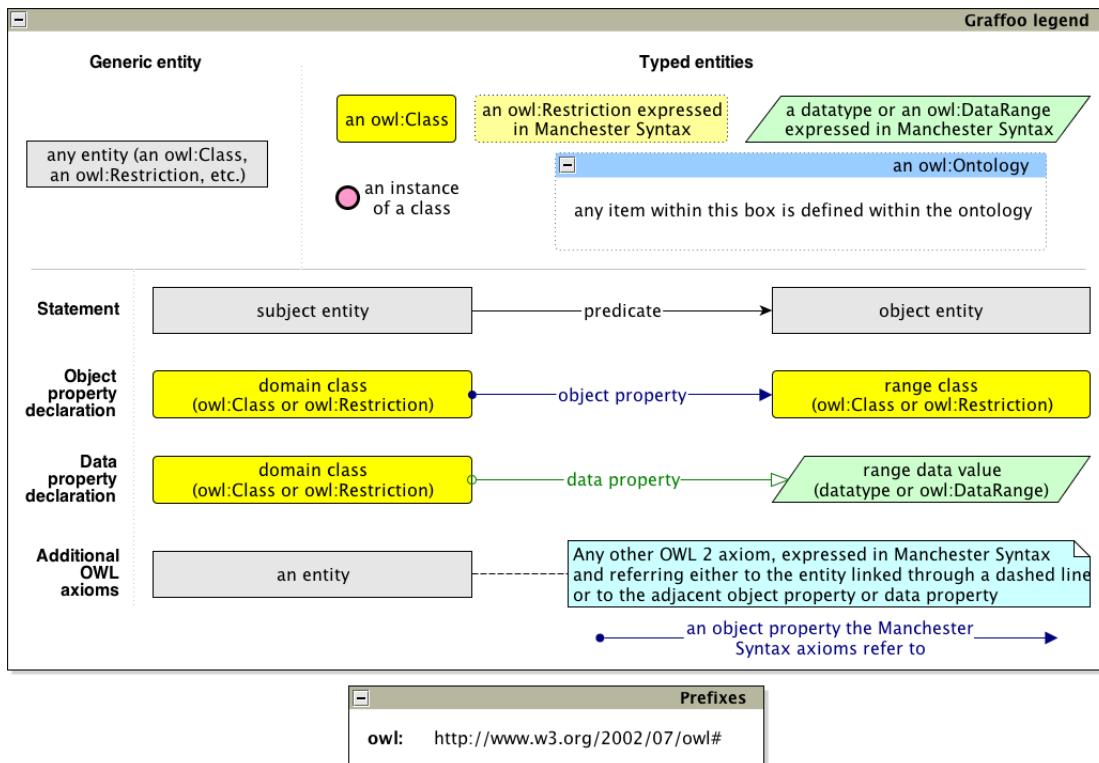


Figure 5.13: The legend for all possible Graffoo objects.

From some preliminary informal studies done, it seems that the Graffoo representations of OWL ontologies can be comprehended without the viewer having to understand all the details of OWL 2 or of any of its linearisations (Turtle, RDF/XML, Manchester Syntax, OWL/XML). In particular, when I was developing the SPAR ontologies, I used Graffoo every time I needed to illustrate an ontology to

¹⁷Graffoo, the Graphic framework for OWL ontologies: <http://www.essepuntato.it/graffoo>.

¹⁸The yEd diagram editor: http://www.yworks.com/en/products_yed_about.html.

¹⁹<http://www.essepuntato.it/graffoo/sources>.

users who were not expert in Semantic Web technologies, but who were interested in understanding SPAR ontologies.

In the following sections, I illustrate how to use Graffoo widgets to formalise ontologies. As example, I use some excerpts from the EARMARK ontology introduced in Section 3.1.

5.3.1 Introducing classes and properties

In Graffoo all the entities can be defined either as an URI surrounded by angular brackets or as a CURIE with a prefix. All the prefixes can be defined within a particular box (entitled “Prefixes”) as a list of prefix-URI couples.

The classes are drawn as yellow boxes, while green rhomboids describe datatypes, and both widgets have solid black borders. Object and data properties are created linking classes, restrictions (introduced in the following section) and datatype through particular arrows. On the one hand, the object property declarations are defined through blue solid lines, where the filled circle at the beginning identifies the domain while the filled arrow at the end indicates the range. On the other hand, the data property declarations are drawn as green solid lines, where the empty circle at the beginning identifies the domain while the empty arrow at the end indicates the range. Moreover, it is possible to associate additional axioms to properties putting a light-blue box close to (or upon) the property it refers to.

For instance, the following excerpt is the Manchester Syntax linearisation of the diagram in Fig. 5.14 on the facing page:

```

Prefix: earmark: <http://www.essepuntato.it/2008/12/earmark#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>

Class: earmark:Range

Class: earmark:Docuverse

ObjectProperty: earmark:refersTo
  Domain: earmark:Range
  Range: earmark:Docuverse
  Characteristics: Functional

DataProperty: earmark:begins
  Domain: earmark:Range
  Range: rdfs:Literal

DataProperty: earmark:ends
  Domain: earmark:Range
  Range: rdfs:Literal

```

```

DataProperty: earmark:hasContent
Domain: earmark:Docuverse
Range: rdfs:Literal
Characteristics: Functional

```

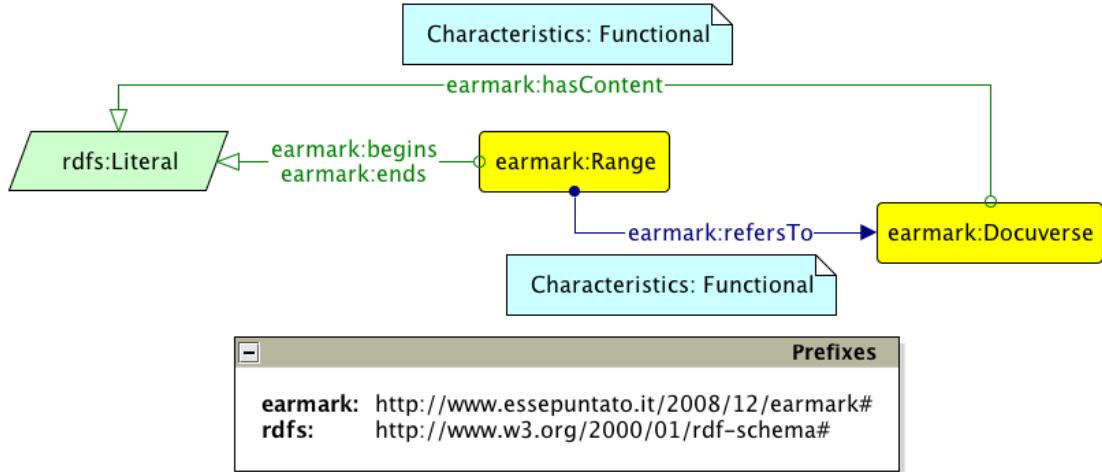


Figure 5.14: Widgets defining prefixes, classes, object/data properties and property axioms.

5.3.2 Defining restrictions and additional class axioms

The widget that defines restrictions is a light-yellow box having dotted border. Classes and restrictions can be linked to create assertions (e.g., *rdfs:subClassOf* relations) through black solid arrows, labelled opportunely according to the property involved in the assertion in consideration. Moreover, I can associate additional axioms (in Manchester Syntax) to classes using a light-blue box linked through a dashed line.

For instance, the following excerpt is the Manchester Syntax linearisation of the diagram in Fig. 5.15 on the next page:

```

Class: earmark:PointerRange
SubClassOf:
    earmark:Range ,
    (earmark:begins only xsd:nonNegativeInteger) and
    (earmark:ends only xsd:nonNegativeInteger)
DisjointWith: earmark:XPathRange
HasKey: earmark:begins earmark:ends earmark:refersTo

Class: earmark:XPathRange
SubClassOf: earmark:Range

```

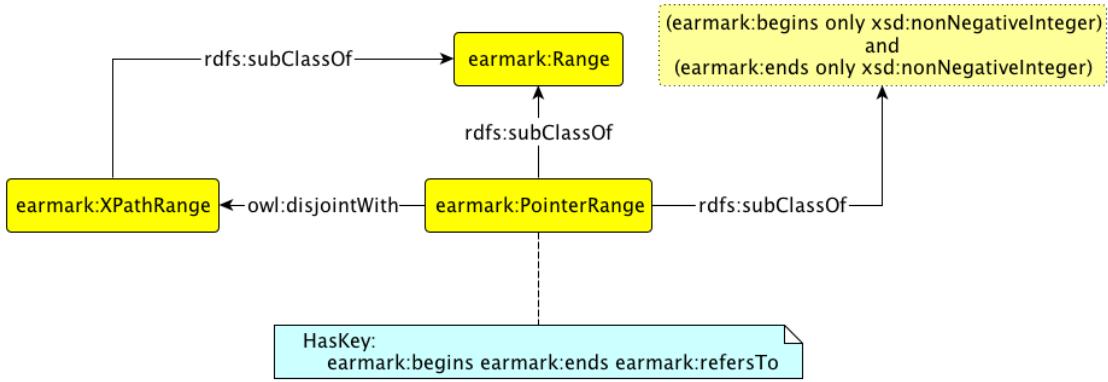


Figure 5.15: Widgets defining restrictions and other class axioms.

5.3.3 Linking class individuals

Graffoo allows one to define also ABox of ontologies, by creating individuals having a particular type. The widget used to define those individuals is a pink circle with solid black border. Note that this widget specifies the URL/CURIE of the related individual around it, rather than in the middle of it. As for all the other resources, individuals can be linked to create assertions through (labelled) black solid arrows.

For instance, the following excerpt is the Manchester Syntax linearisation of the diagram in Fig. 5.16 on the facing page:

```

NamedIndividual: ex:r0-5
  Types: earmark:PointerRange
  Facts:
    earmark:begins "0"^^xsd:nonNegativeInteger ,
    earmark:ends "5"^^xsd:nonNegativeInteger ,
    earmark:refersTo ex:doc

NamedIndividual: ex:doc
  Types: earmark:StringDocuverse
  Facts:
    earmark:hasContent "Fabio says that overlappens"^^xsd:
      string
  
```

5.3.4 Defining assertions between ontologies

Graffoo includes a specific widget – a transparent box with a light-blue heading and a dotted black border – for the specification of ontologies. The ontology URIs are put in the heading of this widget.

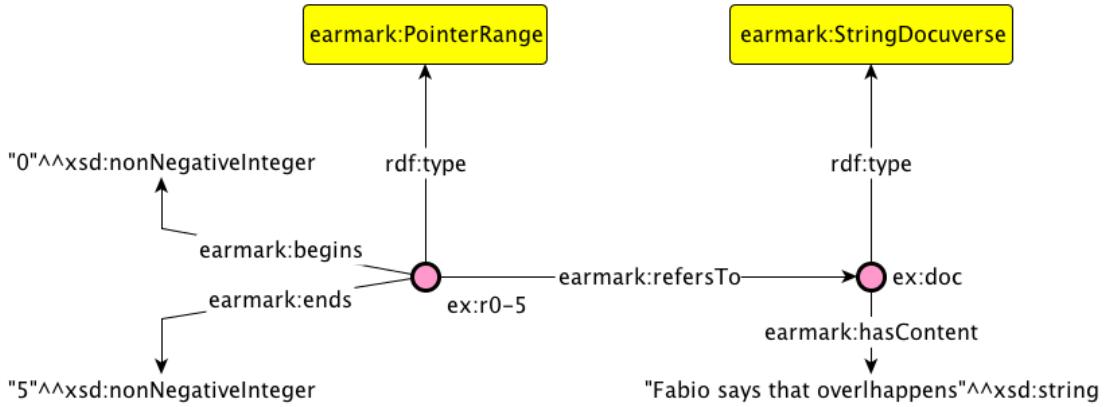


Figure 5.16: Widgets defining individuals and related assertions.

All the entities contained by this widget are formally defined within the related ontology. Moreover, like the other resources, ontologies can be linked to create assertions (e.g., *owl:imports*) through (labelled) black solid arrows.

For instance, the following excerpt is the Manchester Syntax linearisation of the diagram in Fig. 5.17 on the next page:

```

Ontology: <http://www.essepuntato.it/2008/12/earmark>
Import: <http://www.essepuntato.it/2010/05/ghost>

Class: earmark:PointerRange
SubClassOf:
  earmark:Range ,
  (earmark:begins only xsd:nonNegativeInteger) and
  (earmark:ends only xsd:nonNegativeInteger)
DisjointWith: earmark:XPathRange
HasKey: earmark:begins earmark:ends earmark:refersTo

Class: earmark:XPathRange
SubClassOf: earmark:Range

Class: earmark:Range
Annotations: rdfs:isDefinedBy
  <http://www.essepuntato.it/2010/05/ghost>

Class: earmark:Docuverse
Annotations: rdfs:isDefinedBy
  <http://www.essepuntato.it/2010/05/ghost>

ObjectProperty: earmark:refersTo
  
```

```

Annotations: rdfs:isDefinedBy
  <http://www.essepuntato.it/2010/05/ghost>

DataProperty: earmark:begins
Annotations: rdfs:isDefinedBy
  <http://www.essepuntato.it/2010/05/ghost>

DataProperty: earmark:ends
Annotations: rdfs:isDefinedBy
  <http://www.essepuntato.it/2010/05/ghost>

```

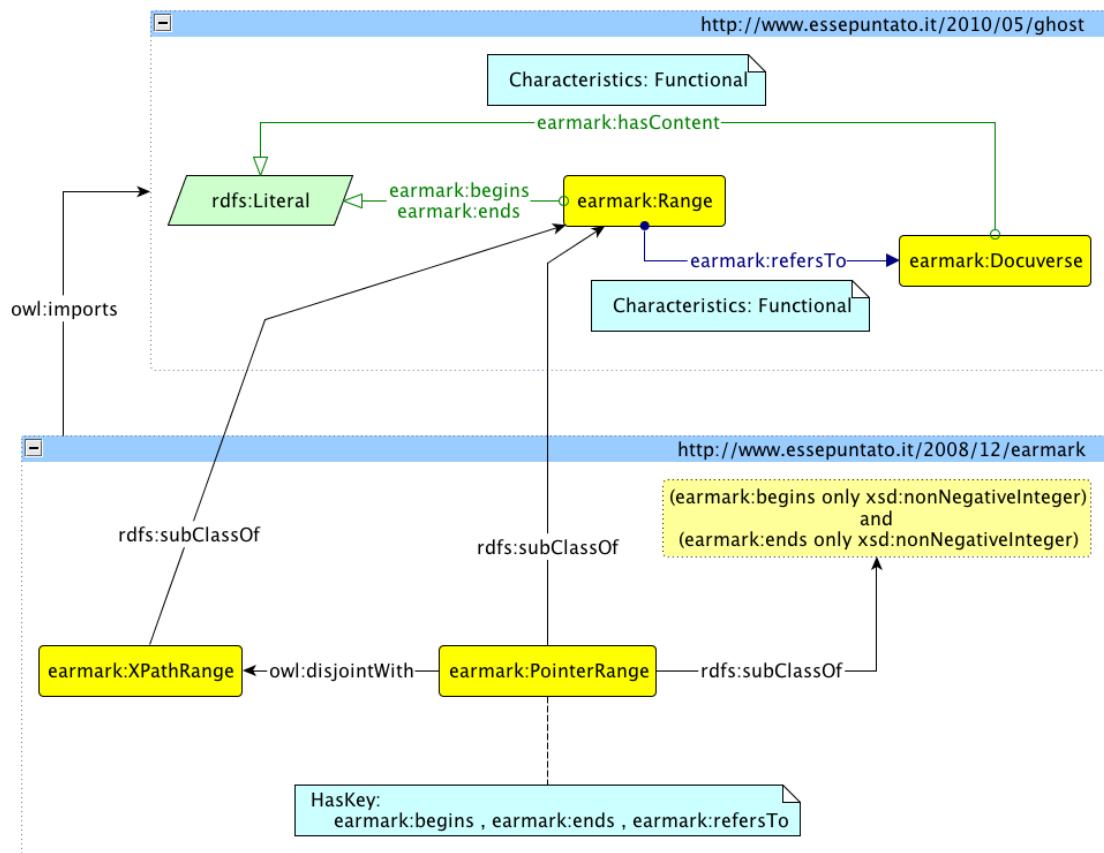


Figure 5.17: Widgets for defining ontologies and related assertions.

5.4 Gaffe, a flexible and user-friendly authoring tool for semantic data

The process of associating semantic data to resources, such as documents, is quite complex in principle. The first issue is that several and alternative models (e.g., metadata schemas, vocabularies, ontologies) can be used to describe the same resource within a particular domain (e.g., publishing). Some of them are almost equivalent, others are characterized by individual features. For instance, all metadata models for the description of bibliographic documents are expected to include information about the “author”, “publisher” and “year”. At the same time, a schema describing Ph.D. theses needs to include similar information (for instance, “author” and “year”) and more domain-specific data such as “id number”, “supervisor” and “discipline”. The choice of the most suitable model depends on two main factors:

- the nature of the resource (e.g., a document);
- the applications (and users) that the resource is meant to be processed by (e.g., Wikis and word processors).

Still, choosing the appropriate model is not enough. It is also important that the interface enables the creation of semantic data in an intuitive and usable way, thus hiding the complexity of the particular formalism or model used. A good model not supported by a good editing interface risks to be useless: authors would often decide not to insert data, considering it a pointless, time-consuming and postponable task.

I identify four main features that a flexible and user-friendly authoring tool for semantic data should have:

- *genericness*, the editor should support any model in a flexible way;
- *customizability*, instead of generating a static form – strictly dependent on a given model – the editor should be customizable for models, users’ needs and preferences;
- *proactivity*, the editor should provide users with facilities that simplify the authoring process, such as pre-filled form fields, suggestions, default values, access to environment variables, and so on;
- *validation*, the editor should apply validation mechanisms to check the correctness of the inserted values.

A solid approach to flexible interfaces consists of adopting the “Model-View-Controller” (MVC) pattern [69], as developed in the software engineering community. This pattern implements a clear separation between the *business logic* of an application and the *user interface* for visualizing/editing data: it allows designers

to generate applications whose interfaces can be easily modified without affecting the model and vice versa.

Discussing the benefits of MVC is out of the scope of this thesis, but what is important is exploring how this pattern can help to design a flexible and sophisticated metadata editor. In the context of metadata editors, the three components of MVC become:

- **Model.** The model corresponds to the actual semantic data as manipulated by the editor and associated to the document/resource. Changing the model describing semantic data means changing the model of MVC, and this should at all times be possible, in order to obtain a model-independent editor;
- **View.** The view is the way semantic data are shown to the users. Usually, the view can be classified according to two different types: the *edit interface* and the *visualisation interface*. The edit interface has to be a rich graphical interface, with a large number of graphical widgets to specify semantic data values according to their type and expected values. The more widgets are sophisticated and well-structured, the more easily users can create semantic data. The visualisation interface shows the current resource-related data only, without changing the internal model. The visualisation can happen through deactivated form fields, but also with plain textual or tabular visualisation. Since model and view are separated, it is possible to assign multiple views to the same semantic model, each tailored to roles, personal preferences and local policies of the intended users;
- **Controller.** The controller is the component in charge of managing the interaction between the users and the application. It has to store the values provided by users into the model. Moreover, it is expected to run a *pre-processing phase* to provide default values to relevant form fields and a *post-processing* one to validate metadata values as provided by the user. The controller thus handles all input events and notifies the model the users' actions that generate changes in the model itself.

Of course, in past solutions that combined the MVC pattern with other approaches were proposed so as to address the issue of associating semantic data to electronic documents. In particular, my research group has been working for years on an approach consisting of two steps. First, creating ontological descriptions of the domain and of the interfaces to manipulate semantic data. Second, transforming those descriptions into actual interfaces shown to the users. My personal contribution in this field is the development of the new Java implementation of *Gaffe* (*Generator of Advanced Forms and Friendly Editor*) [22]. Gaffe is a MVC-based API that makes it possible to build customizable editors for semantic data, so as to allow users to decorate a resource according to any scheme as expressed through an OWL ontology. More precisely, Gaffe uses two different ontologies:

- the *domain* ontology represents the conceptual model. Since this ontology is unconstrained, users can adopt any custom model without any restriction, as long as it is expressible in OWL;
- the *GUI* ontology specifies the classes and properties of widgets and form elements of a graphical user interfaces, as well as the mapping between interface widgets and properties of the domain ontology;

The *instance* document is an instantiation of the GUI ontology to describe an actual interface, as generated by associating domain elements to graphical widgets and by customizing the final appearance of each item.

In the following sections I introduce a Gaffe-based prototypical application called OWiki, that was developed expressively for demonstrating the capabilities of the Gaffe principles.

5.4.1 OWiki: ontology-driven generation of templates and forms for semantic wikis

OWiki [52] [54] [53] is Gaffe-based extension of MediaWiki that supports users in creating and editing semantic data. The basic idea of OWiki is to exploit ontologies and MediaWiki editing/viewing facilities to simplify the process of authoring semantic wiki content.

In particular, OWiki exploits MediaWiki templates, infoboxes and forms. A *template* is set of pair *key-value*, edited as a record and usually formatted as a table in the final wiki page. Templates are particularly useful to store structured information: very easy to edit, disconnected from the final formatting of a page, very easy to search, and so on. Templates are defined in special pages that can be referenced from other pages. These pages include fragments with the same structure of the template but filled with instance data. The template-based component of a page is also called *infobox*.

OWiki exploits ontologies to represent the (semantic) knowledge base of a wiki and templates to display and add ABox assertions of that ontology through the wiki itself. The integration and interaction between ontologies and templates can be summarized in two points:

- each class of the ontology is associated to a template-page. Each property is mapped into a key of the infobox;
- each instance of that class is represented by a page associated to that template. Each line in the infobox then contains the value of a property for that instance. Data properties are displayed as simple text while object properties are displayed as links to other pages (representing other instances of the ontology).

OWiki templates are actually transparent to users. Each template is associated to a form that allows users to create and edit the relative instances. Users do not modify the templates *directly* but they only access specialized form fields.

The crucial point is that even forms are generated automatically from ontological data. Obviously, OWiki includes a *GUI ontology* describing widgets and interface elements. The concepts and relations of the *domain ontology* – that is the ontology according to which semantic data are specified – are mapped into form elements that are delivered to the final user.

During the installation phase, OWiki creates a basic set of forms by merging the domain ontology with the GUI one. At the editing phase, the system shows a very basic form and saves it as a special page (template). This page can then be organized as a new form by adding dynamic behaviours, moving buttons, changing the field order and so on.

Before describing the internal architecture of the system, it is worth spending few more words about the way OWiki uses ontologies. The extensive usage of ontologies makes it possible (i) to make OWiki independent on the domain it is used for, (ii) to easily customize forms and templates, and (iii) to fully describe the evolution of a wiki page and its semantic content.

The architecture of OWiki

OWiki is an integrated framework composed of three modules, delivered with different technologies:

- a *MediaWiki extension*. It is a module integrated in MediaWiki written in PHP that adds OWiki facilities;
- an *Ontology manager*. It is a Java web service that processes OWL ontologies to produce forms for editing metadata. This manager uses internally both Jena API [31] and OWLAPI [94];
- an *Ajax-based interface*: a client-side module that allows users to actually insert data through the forms generated by the OWiki engine.

The PHP OWiki module follows the same architecture of any MediaWiki extension: some scripts and methods are overridden to provide new features. In particular, the module implements a revised editor that initializes OWiki environment variables, sets the communication with the client and sets data necessary to store forms in the MediaWiki database without interfering with existing data.

To manipulate ontologies, OWiki implements a web service that uses internally the Jena API. Jena is integrated with the Pellet reasoner [170], which is exploited to extract information about the instances in the ontology. Ranges of some properties, as well as their values, are in fact derived from subsumptions or other relations expressed in the ontology itself.

The web-service actually generates templates from the ontological data, that are later sent to the PHP module and stored in the right place of the MediaWiki installation.

The connection between the PHP and Java modules, and the core of the overall framework is the OWiki client. The client is a javascript application, based on Mootools²⁰, in charge of actually generating and delivering forms. It is strongly based on the Model-View-Controller (MVC) pattern and its internal architecture can be divided in four layers:

- *The Connection Layer* manages the overall environment, the initialisation phase and the communication between all other layers.
- *The Model Layer* (Model of MVC) manages the data to be displayed on the page. It is composed of a factory that creates wrappers for each type of data and instantiates data from the ontology.
- *The Look And Feel* (View of MVC) manages the final representation of the form, containing atomic and complex widgets, manipulators and decorators.
- *The Interaction Layer* (Controller of MVC) implements the logic of the application, the communication with the web-service, the generation of semantic data and the end-user interaction.

Using ontologies to model the domain

Since it implements the Gaffe architecture, in OWiki the entire *domain of discourse* – i.e., all the topics each page talks about – is handled by using a *domain ontology*. Two different kinds of classes exist in this ontology: those – *page-domain* classes – that strictly relate to articles and pages visualized by the wiki, and others – *data-domain* classes – that define additional data around the former ones.

Each *page-domain* individual results in a wiki page containing text content (the content is now stored in the MediaWiki internal database) and all semantic data directly related to that individual. Fig. 5.18 on the following page shows a page about a particular beer²¹ that contains a textual description of it in the central page area, while in the right box there are all metadata about the beer.

While some metadata such as “Beer Alcoholic content” or “Beer Brewed by” are proper to any beer directly, since they are defined by OWL data or object properties having the class *Beer* as domain, that is not true for other metadata, such as “Winner Award” and “Winner Awarded on”. In fact, those properties are handled using the *data-domain* class *Awarding* that represents an event concerning

²⁰Mootools: <http://mootools.net>.

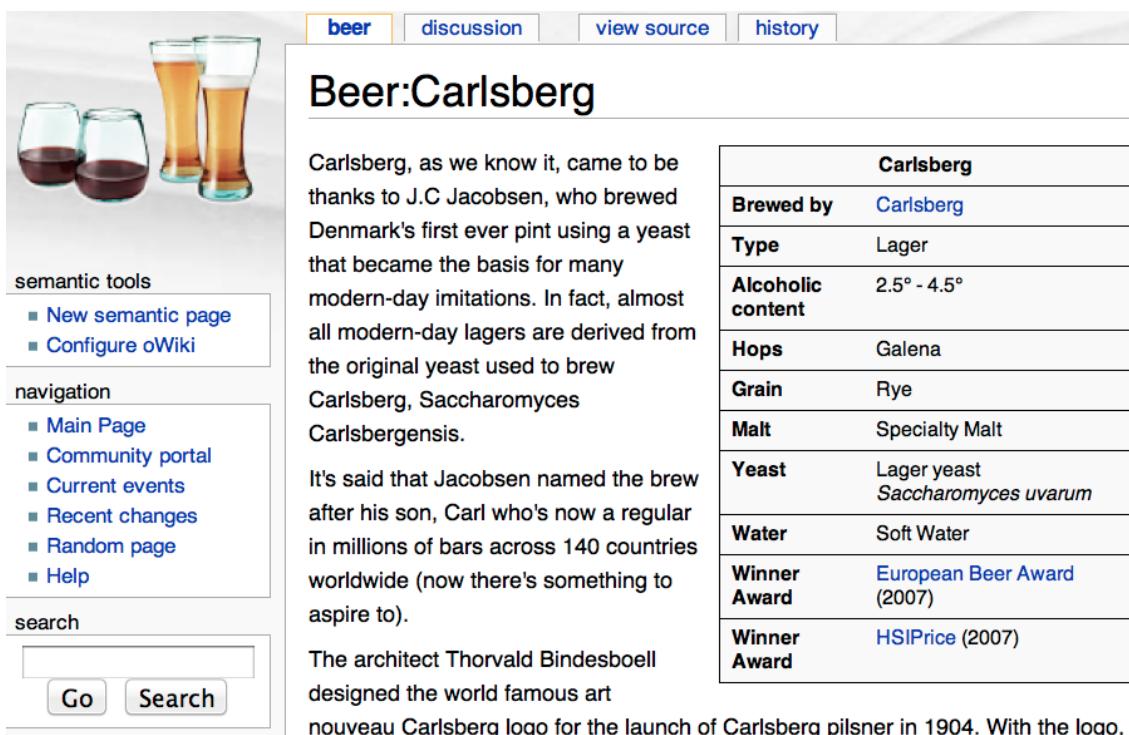
²¹The demo installation of OWiki I use in the following examples is available at <http://owiki.web.cs.unibo.it>.

a particular prize occurred at a specific time. The model of such properties for the beer in consideration, shown in Fig. 5.18, is explained in the following excerpt (in Turtle syntax):

```
:carlsberg a :Beer ; :hasAwarding :awardingEuropean2007 .

:awardingEuropean2007 a :Awarding ;
  :hasAward :europeanBeerAward ;
  :hasYear "2007" .
```

The values shown in the Carlsberg page are not directly extracted from the Carlsberg ontological individual: they are taken from the awarding event the Carlsberg beer participated to.



The screenshot shows a wiki page for 'Beer:Carlsberg'. At the top, there are tabs for 'beer', 'discussion', 'view source', and 'history'. Below the tabs, there's a header image of several glasses filled with beer. The main content area has a section titled 'Beer:Carlsberg' with a detailed description of the beer's history and brewing process. To the right of the description is a table listing various properties of Carlsberg beer. The table has two columns: 'Carlsberg' (property name) and its corresponding value. The properties listed are: Brewed by (Carlsberg), Type (Lager), Alcoholic content (2.5° - 4.5°), Hops (Galena), Grain (Rye), Malt (Specialty Malt), Yeast (Lager yeast *Saccharomyces uvarum*), Water (Soft Water), Winner Award (European Beer Award (2007)), and another entry for Winner Award (HSIPrice (2007)). On the left side of the page, there are sidebar sections for 'semantic tools' (New semantic page, Configure oWiki), 'navigation' (Main Page, Community portal, Current events, Recent changes, Random page, Help), and 'search' (with 'Go' and 'Search' buttons). There is also a small image of the Carlsberg logo.

Carlsberg	
Brewed by	Carlsberg
Type	Lager
Alcoholic content	2.5° - 4.5°
Hops	Galena
Grain	Rye
Malt	Specialty Malt
Yeast	Lager yeast <i>Saccharomyces uvarum</i>
Water	Soft Water
Winner Award	European Beer Award (2007)
Winner Award	HSIPrice (2007)

Figure 5.18: An example page of the Beer OWiki.

Even if they are not directly represented as wiki pages, OWiki uses *data-domain* individuals to enrich even more the metadata of *page-domain* individuals. This enrichment needs to retrieve related data from non-page-domain individuals making at least two steps on the RDF graph represented by the model. We call *property flattening* the visualization of those *data-domain* property values into a *page-domain* individual.

Using ontologies to model the interface

OWiki exploits ontologies to also model end-user interfaces. In particular, the system includes a preliminary version of the GUI ontology developed for Gaffe. This ontology is used for identifying all the components of web forms. The system instantiates and merges that ontology with the domain one in order to generate the final forms. The definitive version of the Gaffe GUI ontology is still under development but the core concepts and relations are stable and already tested in the current prototype.

Separating the GUI ontology from the domain one has a two-fold goal:

1. generating a declarative description of the interface widgets that can be reused across multiple domains not being bounded to specific data;
2. allowing users to customize final interfaces by only changing the association between content and interface widgets.

Note also that the GUI ontology can be designed once for all, while the domain one requires different expertises for different application scenarios. The GUI ontology defines two types of graphical elements: *controllers* and *panels*. Panels are containers for other elements (that can be panels, in turn) used to organize the overall interface, while controllers are single widgets allowing users to actually fill metadata.

The main class of the ontology is *OWikiForm*. Instances of this class will be used to generate each form associated to each wiki page. Each instance will in fact contain either graphical elements or property values from the domain ontology. *OWikiForms* can contain *simple* or *complex* types of controllers. Simple types will be associated to data properties in the domain ontology, while complex type will be associated to object properties.

Simple types model the basic form elements (*TextField*, *ComboBox*, *CheckBox* e *RadioButton*) while complex types model constructs useful for mapping the GUI ontology to the domain one. There are two complex types: *ConnectField* and *ObjectContainer*. *ConnectField* model links to another wiki document. This will be finally used to provide users with auto-completion operations on corresponding form fields: when the user will fill this field, the system will suggest a set of linked documents she/he can choose from (or create a link to a completely new resource). These links are in fact derived from the relations in the domain input ontology. *ObjectContainers* are widgets that include properties of a class non-directly linked to the one defining a particular page, including into documents data about other (related) subjects. This class implement what we illustrated previously as *property flattening*.

5.4.2 Studying OWiki through a use-case

The main goal of OWiki is to simplify the creation of semantic data *through and within* wikis. The complexity of such metadata authoring process, in fact, is hidden behind the application in order to not force users to learn new interfaces and tools. They can easily create semantic data by exploiting forms and templates that are automatically generated from ontological data.

In this section I explain with much details this generation process, clarifying how ontological data are converted into (customized) interfaces. Basically, the overall OWiki process consists of three steps:

1. ontology import and forms generation;
2. forms customization;
3. templates and data generation.

From ontologies to forms

The first step consists of importing the input domain ontology into the wiki. Let us consider a sample application we will discuss throughout the following sections: an OWiki demo installation describing beers, breweries, ingredients, etc. Fig. 5.19 on the facing page shows some classes and properties of a domain ontology suitable for such an application.

Classes and properties are mapped into wiki pages as follows: each concept is mapped into a page and properties are expressed through templates. In particular, data properties become lines of templates infoboxes and object properties become typed links.

Note that the overall status of the OWiki installation is consistent at this stage, assuming that domain input ontology was consistent. The process is in fact a straightforward translation of classes and relations into pages and links.

The OWiki conversion process also produces forms to edit the ontological content. Forms are dynamically built by analysing the class properties of the imported ontology and by mapping each property in the proper element of the GUI interface.

In the example, the class *Beer* defines three properties: *name*, *beerType* and *alcoholContent*. According to the type of these properties OWiki generates text fields or radio buttons. The default element is a text field that allows any type of value. Since in the input ontology the only possible values of the property *beerType* are “Ale”, “Lager” and “Pilsner”, the system add to the form a RadioButton element specifying those values.

For object properties OWiki chooses between two types of widgets according to their range: whether the range class is associated (through the property *hasOWikiPage*) to the class *oWiki*, the system adds a *ConnectField* to the form; otherwise it adds an *ObjectContainer*. Since the class *Beer* has the object property

brewedBy with the class *Brewery* specified as range, the system add to the form a widget that allows one to include a link to a corresponding brewery page. This widget will also provide auto-completion features built on top of the relations expressed in the input ontology.

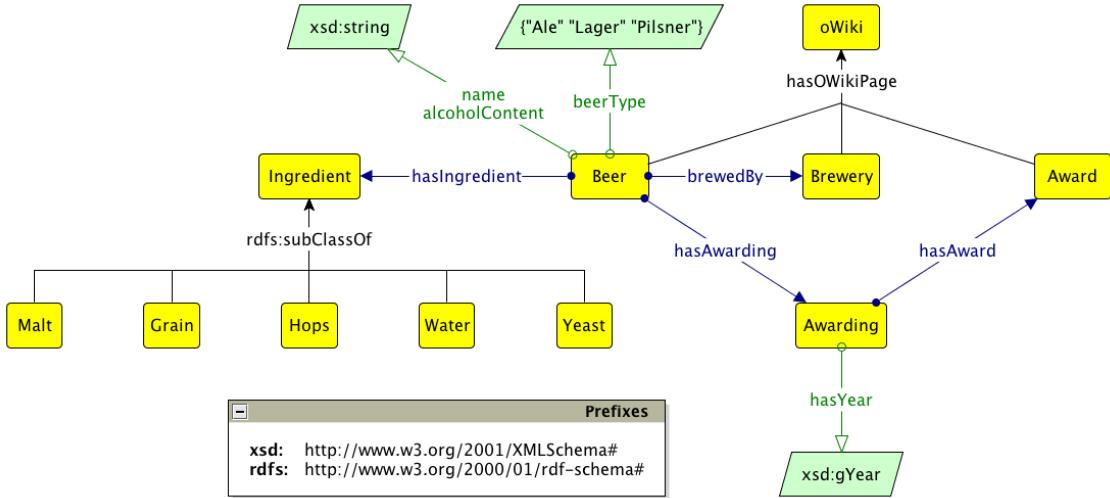


Figure 5.19: A graphical representation of the OWiki domain ontology about beers.

A point is very important at this stage: there is a default mapping between classes of the domain ontology and elements in the GUI ontology based on *the type of the properties*. The name of a property or its meaning in a specific domain is not relevant.

There is actually a configuration file that specifies, for each type, which widget to use and how to configure it. In the previous case, for instance, there was an association between enumerations and radio buttons. That mapping is deployed whenever a class has a property that may only have a finite set of values, regardless of the actual domain ontology. A change in the OWiki configuration file would be reflected in using a different widget for the same property.

Forms customization and filling

Furthermore OWiki includes a configuration interface that allows users to set a domain-specific mapping between the input (domain and GUI) ontologies, and to configure the overall organization of the form and its formatting properties.

OWiki shows a basic form the first time a user edits a page. The author can then organize a new form adding dynamic behaviours, moving buttons, changing fields order and so on. Fig. 5.20 on the next page shows a simple example of a customized forms: while the original form only listed a set of plain text-fields, this one is organized in panels and uses radio-buttons, images and dynamic widgets.

Customization can happen at different level. The user can change colour, font, background of the text to increase the appeal and impact of the form; she/he can change the position and the order of the elements to increase the importance of certain data; she/he can change the optionality of the elements, their default values, and so on.

The current implementation requires users to customize forms by editing an XML configuration file, through the wiki itself. Even if such an approach is not optimal, the internal architecture of the system relies on a strong distinction between the declarative description of the form (through the GUI ontology) and its actual delivery. That makes possible to implement a user-friendly and graphic environment to create and customize forms. One of our future activities is the implementation of such an editor within the OWiki framework.

Editing Carlsberg

Description

Name: Carlsberg

Brewed by: C (Carlsberg)

Characteristics

Type: Lager

Alcoholic content: 2.5° - 4.5°

Carlsberg, as we know it, came to be thanks to J.C Jacobsen, who brewed Denmark's first ever pint using a yeast that became the basis for many modern-day imitations. In fact, almost all modern-day lagers are derived from the original yeast used to brew Carlsberg, *Saccharomyces Carlsbergensis*.

It's said that Jacobsen named the brew after his son, Carl who's now a regular in millions of bars across 140 countries worldwide (now there's something to aspire to).

The architect Thorvald Bindesboell designed the world famous art nouveau Carlsberg logo for the launch of Carlsberg pilsner in 1904. With the logo, this pioneer in Danish design forever left his mark on Denmark's leading beer.

He was paid 500 kroner for the design – it was expensive at the time, but it turned out to be money

Figure 5.20: A customized form generated by OWiki.

From semantic data to templates and views

Automatically-generated forms are finally exploited by the wiki users to actually write the semantic data. As described in the previous section, data are stored as templates and templates are manipulated by forms *in a transparent manner*.

Let me consider again the *Beer* class of the example. OWiki generates a form to create instances of those classes showing three main components:

- a text field to insert the name of the beer;
- a radio-button to select the type of the beer. Values in the radio button are directly extracted from the domain ontology;

- a text field to insert the brewery, which suggests breweries by exploiting information in the domain ontology.

These components can even be organized in multiple panels. Once the user fills the form OWiki saves a template with the proper information. Infobox templates, in fact, are used to display metadata and to cluster information about the same document.

Each infobox line corresponds to a field in the form that, in turn, corresponds to a parameter and its value in the domain ontology. As expected, the data properties of a class are displayed as simple text while the object properties are displayed as links to other documents.

The page corresponding to the Carlsberg beer in the example, that is an instance of the class *Beer* and has been edited via the corresponding form, will contain the following (partial) infobox:

```
 {{Infobox Beer |
hasoWikiNamePage=Carlsberg |
Beer_brewedBy=[[Brewery:Carlsberg|Carlsberg]] |
Beer_beerType=Lager |
Beer_hasAlcoholicContent=2.5 - 4.5 |
Hops_hasName=Galena | ... }}
```

Notice that the property *Beer_brewedBy* contains a link to the page *Carlsberg* that is now an instance of the *Brewery* class. Relations in the input ontology are then mapped into the links between pages. And the *Carlsberg* instance follows the same approach, being it described by the infobox:

```
 {{Infobox Brewery |
hasoWikiNamePage=Carlsberg |
Brewery_hasAddress=Valby 11 DK - 2500, Copenhagen |
Brewery_brews=[[Beer:Carlsberg|Carlsberg]] }}
```

Some final considerations are worth about the consistency of OWiki. First of all, note that OWiki forms only work on the instances of the underlying ontology, without any impact on the classes and relations among them. The consequence is that, assuming that users do not corrupt infoboxes (that are anyway available in the source code of a wiki page), the overall ontology keeps being consistent. The OWiki instance is in fact consistent *by construction* with the domain and the GUI ontology and it is populated via forms in a controlled way.

Thus, we can conclude – going back to the distinction between “wikis for ontologies” and “ontologies for wikis” proposed in Section 2.3.4 – that OWiki currently belongs to the second group and does not properly use the wiki to build and update ontologies. In the future we also plan to investigate a further integration between the wiki and the ontology – and a further integration between the textual content of a wiki page and the relative infoboxes – in order to also use OWiki as a full-fledged simplified authoring environment for ontologies.

Chapter 6

Conclusions

In the early days of the Web, the intrinsic meaning of the content of a document such as a Web page was accessible only to human readers, using their capabilities to conceptualise the particular semantics starting from natural language descriptions. The Semantic Web was born from a desire to develop mechanisms for machine understanding of that same content that would be as effective as that of humans. Its final goal was to “bring structure to the meaningful content of Web pages” and to provide “a new class of tools by which we can live, work and learn together” [16]. In other words, it tried to link authored text (i.e., Web pages) to its formal semantics in a way that “intelligent” applications can be developed so as to significantly assist people in their everyday life.

To this end, the Semantic Web communities initially started to develop standards and technologies with the aim of giving a theoretical and practical background to enable the creation of intelligent applications and enhanced Web resources. Starting from these bases, recently some research and institutional domains are trying to make a further step towards the final aspirations of Semantic Web, putting people and documents back into the roles as first actors and supporting them with Semantic Web technologies and standards. This is the case for *Semantic Publishing*.

Semantic Publishing concerns “anything that enhances the meaning of a published journal article [more generally, a document], facilitates its automated discovery, enables its linking to semantically related articles, provides access to data within the article in actionable form, or facilitates integration of data between papers” [164]. The Semantic Publishing approach goes beyond the current interest of recognising relevant entities in the text and/or transforming natural language statements into formal assertions. In fact, Semantic Publishing aims at describing the entire discourse and argumentation of (bibliographic) documents through formal tools and semantic technologies. The final aim is to increase the *users’ comprehension* of documents through software and applications that work “intelligently” on the formal conceptualisation of the narrative of the documents themselves.

To realise this vision, the actors involved – i.e., publishers, authors, readers, technologists and developers – must be part of an organised cooperative community.

Given the intrinsic heterogeneity of the actors involved, Semantic Publishing must be addressed from different perspectives.

Rather than explore the social and economical aspects of Semantic Publishing, I have in this thesis focussed on its technological environment. In order to link a text to the formal representation of its meaning and thus to represent its argumentative discourse, Semantic Publishing needs at least two distinct resources: on the one hand, a powerful and expressive document markup language that allows semantic characterisations of its elements and content. On the other hand, shared models (ontologies) that allow the formal description of all the aspects of a document, from its structure to its argumentative discourse.

My contributions in this direction are shown in two projects: *EARMARK* and *SPAR*. As illustrated in Chapter 3, EARMARK is a markup metalanguage that allows one to create markup documents as sets of OWL assertions, without the structural and semantic limits imposed by meta-markup languages such as XML. EARMARK is a platform to link the content layer of a document with its intended formal semantics. Having EARMARK as a solid base for defining the content of documents and its syntactical organisation, I then developed the *Semantic Publishing And Referencing (SPAR)* ontologies (Chapter 4), a collection of formal models providing an upper semantic layer for describing the publishing domain. SPAR is a set of eight modular and interoperable ontologies that precisely describe the whole publishing domain using terms from publishers' vocabulary: ranging from bibliographic, structural and rhetorical descriptions of documents to specification of publishing workflows. Thus, using EARMARK as a foundation for SPAR descriptions opens up to a semantic characterisation of all the aspects of a document and of its parts.

Of course, these two aspects – the *markup* and the *semantics* – must be understood, discussed, developed and used within an heterogeneous community that includes people who do not care about the technologies, but who are extremely competent in their own specific domains. Being domain experts, they know the needs and constraints of their own communities. Thus, their contributions to the development of sophisticated Semantic Publishing technologies are crucial.

However, such people may have difficulties in interacting with the Semantic Publishing technologies. Thus, we need user-friendly interfaces that shield such users from the underlying formalisms and semantic models of such technologies.

This is the reason why a good half of my research has concerned the development of interfaces that hide the complexity of markup and ontology formalisms behind user-friendly views. The tools I presented in Chapter 5 – *LODE*, *KC-Viz*, *Graffoo* and *Gaffe* – will hopefully find extensively use for presenting ontologies to publishers, for developing new ontologies to meet particular needs, and for allowing authors to add semantic data to their own documents. These tools have had a crucial role in the development of the SPAR ontologies themselves. Without doubt, they have facilitated the frequent and productive interactions I had with publishers and

domain experts, and have provided one of the main reasons for the early adoption of SPAR in the publishing domain, as described in Section 4.7. LODE and KC-Viz are currently being used in even broader domains and they have been flagged as important contributions in the Semantic Web community¹.

My future research will cover further aspects of Semantic Publishing. In the following sections I introduce the planned future works for all the languages, models and tools presented in my thesis.

6.1 EARMARK: future works

The main and urgent future development of my work on EARMARK, concerns a study of the applicability of this approach to markup in different research domains. In particular, my aim is to investigate real use-case scenarios that involve researchers of different disciplines, such as Humanities or Law. For instance, a relevant issue in Humanities is the use of overlapping markup structures to represent differences among different copies of the same manuscript as a unique digital document. This particular branch of Philology, called *textual criticism*, aims at reconstructing the original text of a manuscript starting from an analysis held on multiple copies of it written by different scribes. Although TEI [186] enables one to store all the overlapping fragments of a *critical edition* via XML workarounds (introduced in Section 2.1.1), my interest is to investigate how different approaches to overlapping markup such as EARMARK can address this problem. Since an interaction with humanists and other researchers that may not be expert in markup technologies is needed, I plan to develop a user interface that facilitates the specification of overlapping markup in EARMARK.

Although I have already carried out a first comparison between XML approaches to overlap and EARMARK (introduced in Section 3.2.3), it may be interesting to develop a complexity-based comparison as well, using a richer and more heterogeneous set of input documents. Moreover, this set of documents will be useful for the evaluation of a conversion framework, called the *EARMARK framework*, I am currently developing with my research group. The main aim of the EARMARK framework is to enable the automatic conversion of XML documents with overlapping markup from a format (e.g., ODT) into another (e.g., OpenXML). The framework has been developed so as to use EARMARK as intermediate format to apply the conversion. Part of this work has been already done and introduced in [11].

¹For instance, LODE is listed in the W3C wiki page about tools for semantic data, available at <http://www.w3.org/2001/sw/wiki/LLDtools>. Moreover, KC-Viz is now part of the core components of the NeOn Toolkit.

6.2 SPAR: future works

Although the SPAR ontologies are already being used within different communities (Section 4.7), my prior interest is to empirically evaluate the goodness of all the eight ontological modules to assess the quality of their vocabulary and their ease of use. I also plan to carry out other formal evaluations to understand the quality of those ontologies according to their logical organisation (e.g., through OntoClean [85] and similar frameworks).

Moreover, I am currently working on the release of triplestores of bibliographic information compliant with SPAR. In particular, in addition to the work already done with the JISC OpenCitation project (Section 2.4.1), my research group and I are collaborating with the publishing house *Società Editrice il Mulino*. Our aim is to study a way to enhance its bibliographic objects through SPAR-based semantic assertions and then to publish them as open linked data. Along the lines of the work with the above publishing house, David Shotton (University of Oxford) and I are now discussing with Mulberry Technologies Inc.² to study an alignment strategy between their *Journal Article Tag Sets (JATS)*³ – i.e., a set of XML DTDs to store journal articles – and SPAR entities.

Another interesting aspect of my proposed research will be the study and development of algorithms for the automatic or semi-automatic identification of structural and rhetoric characteristics of document parts. Starting from a pattern-based description of a markup document (as introduced in Section 4.4.1), it should be possible to deduce the structural roles of its components (sections, chapters, paragraphs, figures, etc.) as well as their rhetoric functions (introduction, background, experiment, results, etc.) without having an *a priori* knowledge of the intended meaning of such markup elements. My aim is to develop automatic mechanisms that assign structural patterns and DoCO characterisations (Section 4.4) to markup elements of XML and EARMARK documents.

6.3 LODE: future works

I plan to evaluate the documentation produced by LODE in both quantitative and qualitative terms. The idea is to develop a comparative user-testing that involves LODE and all the tools introduced in Section 2.3.1.

In addition to the evaluation, I plan to extend the functionalities of the tool with new features. In particular, future versions of LODE will support full multi-language documentation and the explicit handling of the OWL 2 DL meta-modelling capabilities (i.e., OWL punning) in entity descriptions. Moreover I plan to use the KC-Viz abstraction capabilities, introduced in Section 5.2.1, to highlight the most

²Mulberry Technologies Inc.: <http://www.mulberrytech.com>.

³Journal Article Tag Sets: <http://www.mulberrytech.com/JATS/>.

important classes of an ontology in its HTML documentation as rendered through LODE and to develop two plugins, one for the NeOn Toolkit and the other for Protégé, to use LODE within ontology development applications.

6.4 KC-Viz: future works

From the study of the user questionaries discussed in Section 5.2.3 interesting ideas for future works arose. Although KC-Viz is already integrated within the NeOn Toolkit, some criticisms came up with its integration with other plugins, in particular with those supporting the reasoning and query infrastructure of the NeOn Toolkit. I plan to work on extending KC-Viz in order to enable the key-concept extraction mechanism and navigation according to both the declared and inferred ontological axioms.

Moreover, I plan to increase the interface behaviours of KC-Viz by adding layout mechanisms for coarse-grained views (or sky views) to show the entire ontology, by extending the snapshot feature to handle multiple loaded snapshots at the same time, and by highlighting connected links for a class when it is selected.

6.5 Graffoo: future works

There are several planned future developments of Graffoo, but the main priority will be given to its empirical evaluation. In particular, I am interested in understanding whether and how much Graffoo diagrams enable users to understand and develop ontologies. To this end, I plan to carry out a user-testing session that involves people of different fields (Semantic Web practitioners, computer scientists, humanists, etc.) interested in ontologies. The aim is to understand whether Graffoo widget are enough to make sense of a first informal presentation of an ontology.

Besides that, I also plan to work on possible prototypical applications based on the Graffoo widget. First of all, I want to develop a set of XSLT stylesheets to enable the automatic conversion of a set of *yEd* documents specifying Graffoo diagrams into OWL 2 DL ontologies. The next step will be to develop and implementing a pure Web-based editors for the development and publication of Graffoo diagrams as OWL 2 DL ontologies.

6.6 Gaffe: future works

I plan to carry out several evaluation studies to assess the advantages of using Gaffe as authoring tool and form editor according to different kinds of users (ontologists, publishers, semantic data publishers, etc.). In particular, I am now developing a first user-testing session that aims at investigating the benefits introduced by Gaffe

when ontologists use it to develop Web forms through the specification of instance documents that link domain ontologies to ontological descriptions of the forms. My aim is to demonstrate how experts in ontology development can make real and usable Web forms despite their inexperience in the development of Web interfaces.

Moreover, I plan to extend the Gaffe API in order to use the systems in different environment such as word processor. In particular I am now designing an integrated system to support users when enriching documents through SPAR. The idea is to use Gaffe as the main interface by which users can associate semantic metadata to markup documents defined through EARMARK, keeping track of provenance information (e.g., through the W3C Provenance Ontology [154]) of both the author(s) of the formal semantic statements about the text and the author(s) of the text itself.

References

- [1] Accomazzi, A., Dave, R. (2011). Semantic Interlinking of Resources in the Virtual Observatory Era. ArXiv:1103.5958. <http://arxiv.org/pdf/1103.5958> (last visited March 12, 2012).
- [2] Adida, B., Birbeck, M., McCarron, S., Pemberton, S. (2008). RDFa in XHTML: Syntax and processing. W3C Recommendation, 14 October 2008. World Wide Web Consortium. <http://www.w3.org/TR/rdfa-syntax/> (last visited March 12, 2012).
- [3] Alexander, C. (1979). The Timeless Way of Building. New York, New York, USA: Oxford University Press. ISBN: 0195024029.
- [4] Allsopp, J. (2007). Microformats: Empowering Your Markup for Web 2.0. New York, New York, USA: Friends of ED Press. ISBN: 1590598146.
- [5] Attwood, T .K., Kell, D. B., McDermott, P., Marsh, J., Pettifer, S. R., Thorne, D. (2010). Utopia documents: linking scholarly literature with research data. Bioinformatics, 26 (18): 568-574. DOI: 10.1093/bioinformatics/btq383.
- [6] Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., Ives, Z. (2007). DBpedia: A Nucleus for aWeb of Open Data. In Aberer, K., Choi, K., Noy, N. F., Allemang, D., Lee, K., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (Eds.), Proceedings of 6th International Semantic Web Conference and of the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007). Berlin, Germany: Springer.
- [7] Auer, S., Dietzold, S., Riechert, T. (2006). OntOWiki – A Tool for Social, Semantic Collaboration. In Cruz, I. F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (Eds.), Proceedings of the 5th International Semantic Web Conference (ISWC 2006). Berlin, Germany: Springer.
- [8] Bao, J., Smart, P. R., Shadbolt, N., Braines, D., Jones, G. (2009). A Controlled Natural Language Interface for Semantic Media Wiki. In Proceedings of the 3rd Annual Conference of the International Technology Alliance (ACITA2009). September 23-24, 2009, Maryland, USA.

- <http://www.usukita.org/papers/4551/SemanticWikiv7.pdf> (last visited March 12, 2012).
- [9] Barabucci, G., Cervone, L., Di Iorio, A., Palmirani, M., Peroni, S., Vitali, F. (2010). Managing semantics in XML vocabularies: an experience in the legal and legislative domain. In Proceedings of Balisage: The Markup Conference 2009. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://www.balisage.net/Proceedings/vol5/html/Barabucci01/BalisageVol5-Barabucci01.html> (last visited March 12, 2012).
- [10] Barabucci, G., Cervone, L., Palmirani, M., Peroni, S., Vitali, F. (2009). Multi-layer markup and ontological structures in Akoma Ntoso. In Casanovas, P., Paganelli, U., Sartor, G., Ajani, G. (Eds.), Proceeding of the International Workshop on AI approaches to the complexity of legal systems II (AICOL-II). Berlin, Germany: Springer.
- [11] Barabucci, G., Peroni, S., Poggi, F., Vitali, F. (2012). Embedding semantic annotations within texts: the FRETTA approach. To be published in Proceedings of the 27th Symposium On Applied Computing (SAC 2012). New York, New York, USA: ACM.
- [12] Basca, C., Corlosquet, S., Cyganiak, R., Fernández, S., Schandl, T. (2008). Neologism: Easy Vocabulary Publishing. In Bizer, C., Auer, S., Grimnes, G. A., Heath, T. (Eds.), Proceedings of the 4th Workshop on Scripting for the Semantic Web. Aachen, Germany: SunSITE Central Europe. <http://CEUR-WS.org/Vol-368/paper10.pdf> (last visited March 12, 2012).
- [13] Bauman, S. (2010). The 4 “Levels” of XML Rectitude. Presented as poster in Balisage: The Markup Conference 2010. August 3-6, 2010, Montréal, Canada. http://bauman.zapto.org/~syd/temp/XML_rectitude.pdf (last visited March 12, 2012).
- [14] Bański, P. (2010). Why TEI stand-off annotation doesn’t quite work: and why you might want to use it nevertheless. In Proceedings of Balisage: The Markup Conference 2010. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://www.balisage.net/Proceedings/vol5/html/Banski01/BalisageVol5-Banski01.html> (last visited March 12, 2012).
- [15] Berglund, A., Boag, S., Chamberlin, D., Fernández, M. F., Kay, M., Robie, J., Siméon, J. (2007). XML Path Language (XPath) 2.0. W3C Recommendation 23 January 2007. World Wide Web Consortium. <http://www.w3.org/TR/xpath20/> (last visited March 12, 2012).
- [16] Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. In Scientific American, May 17, 2001.

- [17] Berrueta, D., Phipps, J. (2008). Best Practice Recipes for Publishing RDF Vocabularies. W3C Working Group Note 28 August 2008. World Wide Web Consortium. <http://www.w3.org/TR/swbp-vocab-pub/> (last visited March 12, 2012).
- [18] Birks, M., Mills, J. (2011). Grounded Theory: A Practical Guide. SAGE Publications Ltd. ISBN 978-1848609938.
- [19] Bjork, B., Hedlund, T. (2009). Two Scenarios for How Scholarly Publishers Could Change Their Business Model to Open Access. In *Journal of Electronic Publishing*, 12 (1). DOI: 10.3998/3336451.0012.102.
- [20] Bojars, U., Breslin, J. G. (2010). SIOC Core Ontology Specification. <http://rdfs.org/sioc/spec/> (last visited March 12, 2012).
- [21] Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (2010). RIF Core Dialect. W3C Recommendation 22 June 2010. World Wide Web Consortium. <http://www.w3.org/TR/rif-core/> (last visited March 12, 2012).
- [22] Bolognini, V., Di Iorio, A., Duca, S., Musetti, A., Peroni, S., Vitali, F. (2009). Exploiting Ontologies To Deploy User-Friendly and Customized Metadata Editors. In White, B., Isaías, P., Nunes, M. B. (Eds.), *Proceedings of the IADIS Internet/WWW 2009 conference*. Lisbon, Portugal: IADIS.
- [23] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., Cowan, J. (2006). Extensible Markup Language (XML) 1.1 (Second Edition). W3C Recommendation 16 August 2006. World Wide Web Consortium. <http://www.w3.org/TR/xml11/> (last visited March 12, 2012).
- [24] Brickley, D., Guha, R.V. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. World Wide Web Consortium. <http://www.w3.org/TR/rdf-schema/> (last visited March 12, 2012).
- [25] Brickley, D., Miller, L. (2010). FOAF Vocabulary Specification 0.98. Namespace Document, 9 August 2010 - Marco Polo Edition. <http://xmlns.com/foaf/spec/> (last visited March 12, 2012).
- [26] Brockmans, S., Haase, P., Hitzler, P., Studer, R. (2006): A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies. In Sure, Y, Domingue, J. (Eds.), *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*. Berlin, Germany: Springer.
- [27] Brockmans, S., Volz, R., Eberhart, A., Löffler, P. (2004). Visual Modeling of OWL DL Ontologies Using UML. McIlraith, S. A., Plexousakis, D., van Harmelen, F. (Eds.), *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*. Berlin, Germany: Springer.

- [28] Bromley, A. (1991). Policy Statements on Data Management for Global Change Research. <http://www.gcrio.org/USGCRP/DataPolicy.html> (last visited March 12, 2012).
- [29] Brooke, J. (1996). SUS: a “quick and dirty” usability scale. In Jordan, P. W., Thomas, B., Weerdmeester, B. A., McClelland, A. L. (Eds.), *Usability Evaluation in Industry*: 189-194. London, UK: Taylor and Francis. ISBN: 0748404600.
- [30] Buffa, M., Gandon, F., Ereteo, G., Sander, P., Faron, C. (2008). Sweet-Wiki: A semantic wiki. In *Journal of Web Semantics* 6 (1): 84-97. DOI: 10.1016/j.websem.2007.11.003.
- [31] Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K. (2004). Jena: implementing the semantic web recommendations. In Feldman, S. I., Uretsky, M., Najork, M., Wills, C. E. (Eds.), *Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters (WWW 2004)*. New York, New York, USA: ACM.
- [32] Carroll, J., Klyne, G. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 10 February 2004. World Wide Web Consortium. <http://www.w3.org/TR/rdf-concepts/> (last visited March 12, 2012).
- [33] Ciccarese, P., Groza, T. (2011). Ontology of Rhetorical Blocks (ORB). Editor’s Draft, 5 June 2011. World Wide Web Consortium. <http://www.w3.org/2001/sw/hcls/notes/orb/> (last visited March 12, 2012).
- [34] Ciccarese, P., Shotton, D., Peroni, S., Clark, T. (2011). CiTO + SWAN: The Web Semantics of Bibliographic Records, Citations, Evidence and Discourse Relationships. Submitted for publication in *Semantic Web – Interoperability, Usability, Applicability*.
- [35] Ciccarese, P., Wu, E., Kinoshita, J., Wong, G., Ocana, M., Ruttenberg, A., Clark, T. (2008). The SWAN Biomedical Discourse Ontology. *Journal of Biomedical Informatics*, 41 (5), 739-751. DOI: 10.1016/j.jbi.2008.04.010.
- [36] Cimiano, P., Volker, J. (2005). Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery. In Montoyo, A., Munoz, R., Metais, E. (Eds.), *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB05)*: 227-238. Berlin, Germany: Springer.
- [37] Clark, J. (2001). RELAX NG Specification. Committee Specification. Organization for the Advancement of Structured Information Standards. <http://relaxng.org/spec-20011203.html> (last visited March 12, 2012).

- [38] Clark, J. (2002). RELAX NG Compact Syntax. Committee Specification. Organization for the Advancement of Structured Information Standards. <http://relaxng.org/compact-20021121.html> (last visited March 12, 2012).
- [39] Connolly, D. (2007). Gleaning Resource Descriptions from Dialects of Languages (GRDDL). W3C Recommendation, 11 September 2007. World Wide Web Consortium. <http://www.w3.org/TR/grddl/> (last visited March 12, 2012).
- [40] Coombs, J. H., Renear A. H., DeRose, S. J. (1987). Markup Systems and the Future of Scholarly Text Processing. Communications of the ACM 30 (11): 933-947. DOI: 10.1145/32206.32209.
- [41] d'Aquin, M., Motta, E. (2011). Watson, more than a Semantic Web search engine. In Semantic Web – Interoperability, Usability, Applicability, 2 (1): 55-63. DOI: 10.3233/SW-2011-0031.
- [42] D'Arcus, B., Giasson, F. (2009). Bibliographic Ontology Specification. Specification Document, 4 November 2009. <http://bibliontology.com/specification> (last visited March 12, 2012).
- [43] Dattolo, A., Di Iorio, A., Duca, S., Feliziani, A.A., Vitali, F. (2007). Structural patterns for descriptive documents. In Baresi, L., Fraternali, P., Houben, G. (Eds.), Proceedings of the 7th International Conference on Web Engineering 2007 (ICWE 2007). Berlin, Germany: Springer.
- [44] De Coi, J. L., Fuchs, N. E., Kaljurand, K. Kuhn, T. (2009). Controlled English for Reasoning on the Semantic Web. In Bry, F., Maluszynski, J., (Eds.), Semantic Techniques for the Web – The REWERSE Perspective: 276-308. ISBN: 3642045806.
- [45] De Waard, A. (2010). From Proteins to Fairytales: Directions in Semantic Publishing. In IEEE Intelligent Systems, 25 (2): 83-88. DOI: 10.1109/MIS.2010.49.
- [46] De Waard, A. (2010). Medium-Grained Document Structure. <http://www.w3.org/wiki/HCLSIG/SWANSIOC/Actions/RhetoricalStructure/models/medium> (last visited March 12, 2012).
- [47] Dello, K., Paslaru, E. B. S., Tolksdorf, R. (2006). Creating and using semantic web information with makna. In Völkel, M., Schaffert, S. (Eds.), Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics. Aachen, Germany: SunSITE Central Europe. <http://www.ceur-ws.org/Vol-206/paper4.pdf> (last visited March 12, 2012).

- [48] DeRose, S. (2004). Markup Overlap: A Review and a Horse. In Proceedings of the Extreme Markup Languages 2004. Rockville, MD, USA: Mulberry Technologies, Inc. <http://conferences.idealliance.org/extreme/html/2004/DeRose01/EML2004DeRose01.html> (last visited March 12, 2012).
- [49] DeRose, S., Maler, E., Daniel, R. (2001). XPointer xp-pointer() Scheme. W3C Working Draft, 19 December 2002. World Wide Web Consortium. <http://www.w3.org/TR/xptr-xpointer/> (last visited March 12, 2012).
- [50] Di Iorio, A., Gubellini, D., Vitali, F. (2005). Design patterns for document substructures. In Proceedings of the Extreme Markup Languages 2005. Rockville, MD, USA: Mulberry Technologies, Inc. <http://conferences.idealliance.org/extreme/html/2005/Vitali01/EML2005Vitali01.html> (last visited March 12, 2012).
- [51] Di Iorio, A., Marchetti, C., Schirinzi, M., Vitali, F. (2009). Natural and Multi-layered Approach to Detect Changes in Tree-based Textual Documents. In Cordeiro, J., Filipe, J. (Eds.), Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS 2009). Berlin, Germany: Springer.
- [52] Di Iorio, A., Musetti, A., Peroni, S., Vitali, F. (2010). Crowdsourcing semantic content: a model and two applications. In Pardela, T. (Eds.), Proceedings of the 3rd International Conference on Human System Interaction (HSI10). Washington, District Columbia, USA: IEEE Computer Society.
- [53] Di Iorio, A., Musetti, A., Peroni, S., Vitali, F. (2010). Ontology-driven generation of wiki content and interfaces. In New Review of Hypermedia and Multimedia, 16 (1): 9-31. DOI: 10.1080/13614568.2010.497194.
- [54] Di Iorio, A., Musetti, A., Peroni, S., Vitali, F. (2011). OWiki: enabling an ontology-led creation of semantic data. In Hippe, Z. S., Kulikowski, J. L., Mroczek, T. (Eds.), Human-Computer Systems Interaction: Backgrounds and Applications 2. Berlin, Germany: Springer. ISBN: 3642231711.
- [55] Di Iorio, A., Peroni, S., Vitali, F. (2009). Towards markup support for full GODDAGs and beyond: the EARMARK approach. In Proceedings of Balisage: The Markup Conference 2009. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://balisage.net/Proceedings/vol3/html/Peroni01/BalisageVol3-Peroni01.html> (last visited March 12, 2012).
- [56] Di Iorio, A., Peroni, S., Vitali, F. (2010). Handling markup overlaps using OWL. In Cimiano, P., Pinto, H. S. (Eds.), Proceedings of the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010). Berlin, Germany: Springer.

- [57] Di Iorio, A., Peroni, S., Vitali, F. (2011). A Semantic Web Approach To Every-day Overlapping Markup. In *Journal of the American Society for Information Science and Technology*, 62 (9): 1696-1716. DOI: 10.1002/asi.21591.
- [58] Di Iorio, A., Peroni, S., Vitali, F. (2011). Using Semantic Web technologies for analysis and validation of structural markup. In *International Journal of Web Engineering and Technologies*, 6 (4): 375-398. DOI: 10.1504/IJWET.2011.043439.
- [59] Di Iorio, A., Peroni, S., Vitali, F., Lumley, J., Wiley, T. (2009). Towards XML Transclusions. In Vitali, F., Di Iorio, A., Blustein, J. (Eds.), *Proceedings of the 1st Workshop on New Forms of Xanalogical Storage and Function*. Aachen, Germany: Sun SITE Central Europe. <http://ceur-ws.org/Vol-508/paper5.pdf> (last visited March 12, 2012).
- [60] Dragan, L, Handschuh, S., Decker, S. (2011). The semantic desktop at work: interlinking notes. In Ghidini, C., Ngonga Ngomo, A., Lindstaedt, S. N., Pellegiani, T. (Eds.), *Proceedings the 7th International Conference on Semantic Systems (I-SEMANTICS 2011)*. New York, New York, USA: ACM.
- [61] Drummond, N., Rector, A., Stevens, R., Moulton, G., Horridge, M., Wang, H. H., Seidenberg, J. (2006). Putting OWL in Order: Patterns for Sequences in OWL. In B. C. Grau, P. Hitzler, C. Shankey, Evan Wallace (Eds.), *Proceedings of the Workshop on OWL: Experiences and Directions (OWLED 2006)*. Aachen, Germany: Sun SITE Central Europe. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-216/submission_9.pdf (last visited March 12, 2012).
- [62] Dubin, D. (2003). Object mapping for markup semantics. In *Proceedings of the Extreme Markup Languages 2003*. Rockville, MD, USA: Mulberry Technologies, Inc. <http://conferences.idealliance.org/extreme/html/2003/Dubin01/EML2003Dubin01.html> (last visited March 12, 2012).
- [63] Dublin Core Metadata Initiative (2010). DCMI Metadata Terms. DCMI Recommendation. <http://dublincore.org/documents/dcmi-terms/> (last visited March 12, 2012).
- [64] Dublin Core Metadata Initiative (2010). Dublin Core Metadata Element Set, Version 1.1. DCMI Recommendation. <http://dublincore.org/documents/dces/> (last visited March 12, 2012).
- [65] Dublin Core Metadata Initiative (2010). Expressing Dublin Core metadata using HTML/XHTML meta and link elements. DCMI Recommendation. <http://dublincore.org/documents/dcq-html/> (last visited March 12, 2012).

- [66] Durand, D. G. (1994). Palimpsest, a Data Model for Revision Control. Paper presented at the Workshop on Collaborative Editing Systems, co-located with the Computer Supported Cooperative Work Conference (CSCW94). October 22-26, 1994, Chapel Hill, North Carolina, USA.
- [67] Durand, D. G. (2008). Palimpsest: Change-Oriented Concurrency Control for the Support of Collaborative Applications. Charleston, SC, USA: CreateSpace.
- [68] Ferdinand, M., Zirpins, C., Trastour, D. (2004). Lifting Xml Schema to Owl. In Koch, N., Fraternali, P., Wirsing, M. (Eds.), Proceedings of the 4th International Conference on Web Engineering 2004 (ICWE 2004). Berlin, Germany: Springer.
- [69] Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Boston, Massachusetts, USA: Addison-Wesley. ISBN: 0201633610.
- [70] Gangemi, A. (2010). Submission: Participation. <http://ontologydesignpatterns.org/wiki/Submissions:Participation> (last visited March 12, 2012).
- [71] Gangemi, A. (2010). Submission: Region. <http://ontologydesignpatterns.org/wiki/Submissions:Region> (last visited March 12, 2012).
- [72] Gangemi, A. (2010). Submission: Sequence. <http://ontologydesignpatterns.org/wiki/Submissions:Sequence> (last visited March 12, 2012).
- [73] Gangemi, A. (2010). Submission: TimeIndexedSituation. <http://ontologydesignpatterns.org/wiki/Submissions:TimeIndexedSituation> (last visited March 12, 2012).
- [74] Gangemi, A., Peroni, S., Vitali, F. (2010). Literal Reification. In Proceedings of the Workshop on Ontology Pattern 2010 (WOP 2010). Aachen, Germany: Sun SITE Central Europe. <http://CEUR-WS.org/Vol-671/pat04.pdf> (last visited March 12, 2012).
- [75] Gao, S., Sperberg-McQueen, C. M., Thompson, H. S. (2011). W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. W3C Candidate Recommendation 21 July 2011. World Wide Web Consortium. <http://www.w3.org/TR/xmlschema11-1/> (last visited March 12, 2012).
- [76] Garcia, R., Celma, O. (2005) Semantic Integration and Retrieval of Multimedia Metadata. In Proceedings of the 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005).

- Aachen, Germany: Sun SITE Central Europe. <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-185/semAnnot05-07.pdf> (last visited March 12, 2012).
- [77] Gardenfors, P. (2000). Conceptual Spaces: The Geometry of Thought. Cambridge, Massachusetts: USA. ISBN: 0262071991.
- [78] Garlik, S. H., Seaborne, A. (2011). SPARQL 1.1 Query Language. W3C Working Draft 12 May 2011. World Wide Web Consortium. <http://www.w3.org/TR/sparql11-query/> (last visited March 12, 2012).
- [79] Gasevic, D., Djuric, D., Devedzic, V., Damjanovic, V. (2004). Converting UML to OWL Ontologies. In Aßmann, U., Aksit, M., Rensink, A. (Eds.), Model Driven Architecture, European MDA Workshops: Foundations and Applications (MDAFA 2003 and MDAFA 2004). Berlin, Germany: Springer.
- [80] Georg, R., Schonefeld, O., Trippel, T., Witt, A. (2010). Sustainability of Linguistic Resources Revisited. In Proceedings of the International Symposium on XML for the Long Haul: Issues in the Long-term Preservation of XML. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://www.balisage.net/Proceedings/vol6/html/Witt01/BalisageVol6-Witt01.html> (last visited March 12, 2012).
- [81] Goldfarb, C. F. (1990). The SGML Handbook. New York, New York, USA: Oxford University Press. ISBN: 0198537373.
- [82] Groza, T., Handschuh, S., Decker, S. (2011). Capturing Rhetoric and Argumentation Aspects within Scientific Publications. In Journal on Data Semantics 15: 1-36. DOI: 10.1007/978-3-642-22630-4_1.
- [83] Groza, T., Handschuh, S., Möller, K., Decker, S. (2007). SALT – Semantically Annotated LaTeX for Scientific Publications. In Franconi, E., Kifer, M., May, W. (Eds.), Proceedings of the fourth European Semantic Web Conference (ESWC 2007). Berlin, Germany: Springer.
- [84] Groza, T., Möller, K., Handschuh, S., Trif, D., Decker, S. (2007). SALT: Weaving the claim web. In Aberer, K., Choi, K., Noy, N. F., Allemang, D., Lee, K., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (Eds.), Proceedings of 6th International Semantic Web Conference and of the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007). Berlin, Germany: Springer.
- [85] Guarino, N., Welty, C. (2002). Evaluating ontological decisions with OntoClean. In Communications of the ACM, 45 (2): 61-65. DOI: 10.1145/503124.503150.

- [86] Guthrie, L., Pustejovsky, J., Wilks, Y., Slator, B. M. (1996). The role of lexicons in Natural Language Processing. In Communications of the ACM, 39 (1): 63-72. DOI: 10.1145/234173.234204.
- [87] Hammond, T. (2008). RDF Site Summary 1.0 Modules: PRISM. http://nurture.nature.com/rss/modules/mod_prism.html (last visited March 12, 2012).
- [88] Harnad, S., Brody, T. (2004). Comparing the Impact of Open Access (OA) vs. Non-OA Articles in the Same Journals. In D-Lib Magazine, 10 (6). DOI: 10.1045/june2004-harnad.
- [89] Harnad, S., Brody, T., Vallieres, F., Carr, L., Hitchcock, S., Gingras, Y., Oppenheim, C., Stamerjohanns, H. and Hilf, E. R. (2004). The Access/Impact Problem and the Green and Gold Roads to Open Access. In Serials Review, 30 (4): 310-314. DOI: 10.1016/j.serrev.2004.09.013.
- [90] Harth, C., Gassert, H., O'Murchu, I., Breslin, J. G., Decker S., 2005. WikiOnt: An Ontology for Describing and Exchanging Wikipedia Articles. In Voss, J., Lih, A., Klein, S., Ma, C. (Eds.), Proceedings of Wikimania 2005. <http://meta.wikimedia.org/wiki/Wikimania05/Paper-IM1> (last visited March 12, 2012).
- [91] Hayes, P., Welty, C. (2006). Defining N-ary Relations on the Semantic Web. W3C Working Group Note, 12 April 2006. World Wide Web Consortium. <http://www.w3.org/TR/swbp-n-aryRelations/> (last visited March 12, 2012).
- [92] Hickson, I. (2011). HTML5: A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft 25 May 2011. World Wide Web Consortium. <http://www.w3.org/TR/html5/> (last visited March 12, 2012).
- [93] Hobbs, J .R., Pan, F. (2006). Time Ontology in OWL. W3C Working Draft, 27 September 2006. World Wide Web Consortium. <http://www.w3.org/TR/owl-time/> (last visited March 12, 2012).
- [94] Horridge, M., Bechhofer, S. (2011). The OWL API: A Java API for OWL ontologies. In Semantic Web – Interoperability, Usability, Applicability, 2 (1): 11-21. DOI: 10.3233/SW-2011-0025.
- [95] Horridge, M., Patel-Schneider, P. (2009). OWL 2 Web Ontology Language: Manchester Syntax. W3C Working Group Note October 27, 2009, World Wide Web Consortium. <http://www.w3.org/TR/owl2-manchester-syntax/> (last visited March 12, 2012).

- [96] Horrocks, I., Patel-Schneider, P. F., Boley, H. Tabet, S., Grosof, B., Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004. World Wide Web Consortium. <http://www.w3.org/Submission/SWRL/> (last visited March 12, 2012).
- [97] Huitfeldt, C., Sperberg-McQueen, C. M. (2003). TexMECS: An experimental markup metalanguage for complex documents. <http://decentius.aksis.uib.no/mlcd/2003/Papers/texmechs.html> (last visited March 12, 2012).
- [98] Iannella, R. (2010). Representing vCard Objects in RDF. W3C Member Submission, 20 January 2010. World Wide Web Consortium. <http://www.w3.org/TR/vcard-rdf/> (last visited March 12, 2012).
- [99] International Digital Enterprise Alliance (2009). Publishing Requirements for Industry Standard Metadata Specification Version 2.0. Alexandria, VA, USA: IDEAlliance. <http://www.idealliance.org/specifications/prism> (last visited March 12, 2012).
- [100] International Federation of Library Associations and Institutions Study Group on the Functional Requirements for Bibliographic Records (2009). Functional Requirements for Bibliographic Records Final Report. International Federation of Library Associations and Institutions. http://www.ifla.org/files/cataloguing/frbr/frbr_2008.pdf (last visited March 12, 2012).
- [101] International STM publishing community (2007). Brussels Declaration on STM Publishing. <http://www.stm-assoc.org/brussels-declaration/> (last visited March 12, 2012).
- [102] Jakobson, R. (1960). Closing statements: Linguistics and Poetics. In Sebeok, T. A. (Eds.), *Style in language: 351-377*. Cambridge, Massachusetts, USA: The MIT Press. ISBN: 0262690101.
- [103] JTC1/SC34 WG 4. (2011). ISO/IEC 29500-1:2011 – Information technology – Document description and processing languages – Office Open XML File Formats – Part 1: Fundamentals and Markup Language Reference. Geneva, Switzerland: International Organization for Standardization. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=59575 (last visited March 12, 2012).
- [104] JTC1/SC34 WG 6. (2006). ISO/IEC 26300:2006 – Information technology – Open Document Format for Office Applications (OpenDocument)

- v1.0. Geneva, Switzerland: International Organization for Standardization. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485 (last visited March 12, 2012).
- [105] Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E. (2007). Ontology Visualization Methods – a Survey. *ACM Computing Surveys*, 39 (4). DOI: 10.1145/1287620.1287621.
- [106] Katifori, A., Torou, E., Halatsis, C., Lepouras, G., Vassilakis, C. (2006). A Comparative Study of Four Ontology Visualization Techniques in Protege: Experiment Setup and Preliminary Results. In Proceedings of the 10th International Conference on Information Visualisation (IV 2006). Washington, District Columbia, USA: IEEE Computer Society.
- [107] Kay, J. Lum, A. (2003). An Ontologically Enhanced Metadata Editor. University of Sidney, School of Information Technologies, Technical Report 541. <http://www.it.usyd.edu.au/research/tr/tr541.pdf> (last visited March 12, 2012).
- [108] Kircz, J. G. (1991). Rhetorical structure of scientific articles: the case for argumentational analysis in information retrieval. In *Journal of Documentation*, 47 (4): 354-372. DOI: 10.1108/eb026884.
- [109] Klein, B., Höcht, C., Decker, B. (2005). Beyond Capturing and Maintaining Software Engineering Knowledge – “Wikitology” as Shared Semantics, Workshop on Knowledge Engineering and Software Engineering. Presented during the Workshop on Knowledge Engineering and Software Engineering at the German Conference on Artificial Intelligence (KI 2005). Koblenz, Germany. <http://www.dfki.uni-kl.de/~klein/papers/finalKESE05.pdf> (last visited March 12, 2012).
- [110] Knublauch, H., Horridge, M., Musen, M. A., Rector, A. L., Stevens, R., Drummond, N., Lord, P. W., Noy, N. F., Seidenberg, J., Wang, H. (2005). The Protege OWL Experience. In Grau, B. C., Horrocks, I., Parsia, B., Patel-Schneider, P. F. (Eds.), *Proceedings of the OWLED 05 Workshop on OWL: Experiences and Directions*. Aachen, Germany: SunSITE Central Europe. <http://ceur-ws.org/Vol-188/sub14.pdf> (last visited March 12, 2012).
- [111] Koutsomitropoulos, D.A., Solomou, G.D., Papatheodorou, T.S. (2008). Semantic interoperability of dublin core metadata in digital repositories. In Proceedings of the 5th International Conference on Innovations in Information Technology. Washington, District Columbia, USA: IEEE Computer Society.
- [112] Kriglstein, S. and Motschnig-Pitrik, R. (2008). Knoocks: A New Visualization Approach for Ontologies. In Proceedings of the 12th International Conference

- on Information Visualisation (IV 2008). Washington, District Columbia, USA: IEEE Computer Society.
- [113] Krotzsch, M., Simancik, F., Horrocks, I. (2011). A Description Logic Primer. Ithaca, New York, New York: Cornell University Library. <http://arxiv.org/pdf/1201.4089v1> (last visited March 12, 2012).
- [114] Lakoff, G. (1987). Women, Fire, and Dangerous Things. Chicago, Illinois: USA. University Of Chicago Press. ISBN: 0226468046.
- [115] Lakoff, G., Johnson M. (1980). Metaphors we live by. Chicago, Illinois: USA. University Of Chicago Press. ISBN: 0226468011.
- [116] Lawrence, S. (2001). Free online availability substantially increases a paper's impact. In *Nature*, 411 (6837): 521. DOI: 10.1038/35079151.
- [117] Li, N., Motta, E. d'Aquin, M. (2010). Ontology summarization: an analysis and an evaluation. In Gomez-Perez, A., Ciravegna, F., van Harmelen, F., Hefflin, J. (Eds.), *Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010)*. Aachen, Germany: SunSITE Central Europe. <http://CEUR-WS.org/Vol-666/paper8.pdf> (last visited March 12, 2012).
- [118] Library of Congress - Network Development and MARK Standard Office (2010). MARK 21 format for bibliographic data. 1999 edition, further updates October 2001 and October 2010. <http://www.loc.gov/marc/bibliographic/> (last visited March 12, 2012).
- [119] Marcoux, Y. (2006). A natural-language approach to modeling: Why is some XML so difficult to write? In *Proceedings of the Extreme Markup Languages 2006*. Rockville, MD, USA: Mulberry Technologies, Inc. <http://conferences.idealliance.org/extreme/html/2006/Marcoux01/EML2006Marcoux01.html> (last visited March 12, 2012).
- [120] Marcoux, Y. (2008). Graph characterization of overlap-only TexMECS and other overlapping markup formalisms. In *Proceedings of Balisage: The Markup Conference 2008*. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://www.balisage.net/Proceedings/vol1/html/Marcoux01/BalisageVol1-Marcoux01.html> (last visited March 12, 2012).
- [121] Marcoux, Y., Rizkallah, E. (2009). Intertextual semantics: A semantics for information design. *Journal of the American Society for Information Science and Technology*, 60 (9): 1895-1906. DOI: 10.1002/asi.21134.
- [122] Marinelli, P., Vitali, F., Zacchiroli, S. (2008). Towards the unification of formats for overlapping markup. In *New Review of Hypermedia and Multimedia*, 14 (1): 57-94. DOI: 10.1080/13614560802316145.

- [123] Miles, A., Bechhofer, S. (2009). SKOS Simple Knowledge Organization System Reference. W3C Recommendation, 18 August 2009. World Wide Web Consortium. <http://www.w3.org/TR/skos-reference/> (last visited March 12, 2012).
- [124] Moller, K., Bechhofer, S., Heath, T. (2009). Semantic Web Conference Ontology. http://data.semanticweb.org/ns/swc/swc_2009-05-09.html (last visited March 12, 2012).
- [125] Montoya, E., Ruiz, M., Giraldo, J. (2005). BDNG: A Dublin Core-based architecture for digital libraries. In Proceedings of the International Conference on Dublin Core and Metadata Applications 2005. Singapore, Singapore: Dublin Core Metadata Initiative.
- [126] Motik, B., Patel-Schneider, P. F., Grau, B.C. (2009). OWL 2 Web Ontology Language: Direct Semantics. W3C Recommendation 27 October 2009. World Wide Web Consortium. <http://www.w3.org/TR/owl2-direct-semantics/> (last visited March 12, 2012).
- [127] Motik, B., Patel-Schneider, P. F., Parsia, B. (2009). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation, 27 October 2009. World Wide Web Consortium. <http://www.w3.org/TR/owl2-syntax/> (last visited March 12, 2012).
- [128] Motta, E., Mulholland, P., Peroni, S., D'Aquin, M., Gomez-Perez, J. M., Mendez, V., Zablith, F. (2011). A Novel Approach to Visualizing and Navigating Ontologies. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N. F., Blomqvist, E. (Eds.), Proceedings of the 10th International Semantic Web Conference (ISWC 2011). Berlin, Germany: Springer.
- [129] Motta, E., Peroni, S., d'Aquin, M. (2011). Latest Developments in KC-Viz. To be presented during the demo session of the 10th International Semantic Web Conference (ISWC 2011). Bonn, Germany.
- [130] Motta, E., Peroni, S., Gómez-Pérez, J. M., d'Aquin, M., Ning Li, N. (2012). Visualizing and Navigating Ontologies with KC-Viz. In Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (Eds.), Ontology Engineering in a Networked World. Berlin, Germany: Springer. ISBN: 3642247934.
- [131] Motta, E., Peroni, S., Li, N., D'Aquin, M. (2010). KC-Viz: A Novel Approach to Visualizing and Navigating Ontologies. In a supplementary book of the demo session of the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010). Lisbon, Portugal.

- [132] Nelson, T. (1980). Literary Machines: The report on, and of, Project Xanadu concerning word processing, electronic publishing, hypertext, thinkertoys, tomorrow's intellectual... including knowledge, education and freedom. Sausalito, CA, USA: Mindful
- [133] Nuzzolese, A., Gangemi, A., Presutti, V. (2010). Gathering Lexical Linked Data and Knowledge Patterns from FrameNet. In Mark A. Musen, Óscar Corcho (Eds.), Proceedings of the 6th International Conference on Knowledge Capture (K-CAP 2011). New York, New York, USA: ACM.
- [134] Object Management Group (2009). Ontology Definition Metamodel (ODM) Version 1.0. <http://www.omg.org/spec/ODM/1.0/PDF> (last visited March 12, 2012).
- [135] Object Management Group (2011). Unified Modeling LanguageTM (UML®). <http://www.omg.org/spec/UML/2.4/> (last visited March 12, 2012).
- [136] Odlyzko, A. (2002). The rapid evolution of scholarly communication. In Learned Publishing, 15 (1): 7-19. DOI: 10.1087/095315102753303634.
- [137] Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G. (2008). Sindice.com: a document-oriented lookup index for open linked data. In International Journal of Metadata, Semantics and Ontologies, 3 (1): 37-52. DOI: 10.1504/IJMSO.2008.021204.
- [138] Passant, A., Laublet, P. (2008). Towards an Interlinked Semantic Wiki Farm. In Lange, C., Schaffert, S., Skaf-Molli, H., Völkel M. (Eds.), Proceedings of the 3rd Semantic Wiki Workshop (SemWiki 2008). Aachen, Germany: SunSITE Central Europe. <http://ceur-ws.org/Vol-360/paper-19.pdf> (last visited March 12, 2012).
- [139] Peirce, C. S. (1958). Collected Papers of Charles Sanders Peirce. Hartshorne, C., Weiss, P. (Eds.). Cambridge, Massachusetts, USA: Harvard University Press. ISBN: 0674138001.
- [140] Peroni, S., Gangemi, A., Vitali, F. (2011). Dealing with Markup Semantics. In Ghidini, C., Ngonga Ngomo, A., Lindstaedt, S. N., Pellegrini, T. (Eds.), Proceedings the 7th International Conference on Semantic Systems (I-SEMANTICS 2011). New York, New York, USA: ACM.
- [141] Peroni, S., Motta, E., d'Aquin, M. (2008). Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In Domingue, J., Anutariya, C. (Eds.), Proceedings of the 3rd Asian Semantic Web Conference (ASWC 2008). Berlin, Germany: Springer.

- [142] Peroni, S., Vitali, F. (2009). Annotations with EARMARK for arbitrary, overlapping and out-of order markup. In Borghoff, U. M., Chidlovskii, B. (Eds.), Proceedings of the 2009 ACM Symposium on Document Engineering (DocEng 2009). New York, New York, USA: ACM.
- [143] Picca, D., Gliozzo, A., Gangemi, A. (2008). LMM: an OWL-DL MetaModel to Represent Heterogeneous Lexical Knowledge. In Proceedings of the 6th Language Resource and Evaluation Conference (LREC 2008). Luxembourg, Luxembourg: European Language Resources Association.
- [144] Plaisant, C., Grosjean, J., and Bederson, B. B. (2002). Spacetree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2002). Washington, District Columbia, USA: IEEE Computer Society.
- [145] Portier, P., Calabretto, S. (2009). Methodology for the construction of multi-structured documents. In Proceedings of Balisage: The Markup Conference 2009. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://balisage.net/Proceedings/vol3/html/Portier01/BalisageVol3-Portier01.html> (last visited March 12, 2012).
- [146] Presutti, V., Gangemi, A. (2008). Content Ontology Design Patterns as practical building blocks for web ontologies. In Li, Q., Spaccapietra, S., Yu, E. S. K., Olivé, A. (Eds.), Proceedings of the 27th International Conference on Conceptual Modeling (ER 2008). Berlin, Germany: Springer.
- [147] Prud'hommeaux, E., Carothers G. (2011). Turtle, Terse RDF Triple Language. W3C Working Draft 09 August 2011, World Wide Web Consortium. <http://www.w3.org/TR/turtle/> (last visited March 12, 2012).
- [148] Rector, A. (2003). Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL. Gennari, J. H., Porter, B. W., Gil, Y., (Eds.), Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP 2003). New York, New York, USA: ACM.
- [149] Renear, A., Dubin, D., Sperberg-McQueen, C. M. (2002). Towards a Semantics for XML Markup. In the Proceedings of the 2002 ACM Symposium on Document Engineering. New York, New York, USA: ACM.
- [150] Renear, A., Dubin, D., Sperberg-McQueen, C. M., Huitfeldt, C. (2003). XML Semantics and Digital Libraries. In the Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries. Washington, District Columbia, USA: IEEE Computer Society.

- [151] Riggs, K.R. (2002). XML and Free Text. *Journal of the American Society for Information Science and Technology*, 53 (6): 526-528. DOI: 10.1002/asi.10063.
- [152] Rodrigues, T., Rosa, P., Cardoso, J. (2006). Mapping Xml to Existing Owl Ontologies. In Nunes, M. B., Isaías, P., Martínez, I. J. (Eds.), *Proceedings of the IADIS International conference on WWW/Internet 2006*. Lisbon, Portugal: IADIS.
- [153] Rosch, E. (1978). Principles of categorization. In Rosch, E., Lloyd, B. (Eds.), *Cognition and Categorisation*. New Jersey, USA: Lawrence Erlbaum. ISBN: 0470263778.
- [154] Sahoo, S., McGuinness, D. (2011). The PROV Ontology: Model and Formal Semantics. W3C Working Draft, 13 December 2011. World Wide Web Consortium. <http://www.w3.org/TR/prov-o/> (last visited March 12, 2012).
- [155] Saussure, F. (2006). *Writings in General Linguistics*. New York, New York, USA: Oxford University Press. ISBN: 019926144X.
- [156] Schaffert, S. (2006). IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In *Proceedings of 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006)*. Washington, District Columbia, USA: IEEE Computer Society.
- [157] Schmidt, D. (2009). Merging Multi-Version Texts: a Generic Solution to the Overlap Problem. In *Proceedings of Balisage: The Markup Conference 2009*. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://balisage.net/Proceedings/vol3/html/Schmidt01/BalisageVol3-Schmidt01.html> (last visited March 12, 2012).
- [158] Schmidt, D., Colomb, R. (2009). A data structure for representing multi-version texts online. In *International Journal of Human-Computer Studies*, 67 (6): 497-514. DOI: 10.1016/j.ijhcs.2009.02.001.
- [159] Schneider, J., Groza, T., Passant, A. (2011). A Review of Argumentation for the Social Semantic Web. Accepted for publication with Major Revision in *Semantic Web – Interoperability, Usability, Applicability*. <http://www.semantic-web-journal.net/sites/default/files/swj138.pdf> (last visited March 12, 2012).
- [160] Schonefeld, O., Witt, A. (2006). Towards validation of concurrent markup. In *Proceedings of the Extreme Markup Languages 2006*. Rockville, MD, USA: Mulberry Technologies, Inc. <http://conferences.idealliance.org/extreme/html/2006/Schonefeld01/EML2006Schonefeld01.html> (last visited March 12, 2012).

- [161] Searle, J. (1970). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge, UK: Cambridge University Press. ISBN: 052109626X.
- [162] Shneiderman, B. (1992). Tree Visualization with Tree-Maps: A 2d Space-Filling Approach. In *ACM Transactions on Graphics*, 11 (1): 92-99. DOI: 10.1145/102377.115768.
- [163] Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages (VL '96)*. Washington, District Columbia, USA: IEEE Computer Society.
- [164] Shotton, D. (2009). Semantic Publishing: the coming revolution in scientific journal publishing. *Learned Publishing*, 22 (2): 85-94. DOI: 10.1087/2009202.
- [165] Shotton, D. (2010). CiTO, the Citation Typing Ontology. In *Journal of Biomedical Semantics*, 1 (1): S6. DOI: 10.1186/2041-1480-1-S1-S6.
- [166] Shotton, D., Caton, C., Klyne, G. (2010). Ontologies for sharing, ontologies for use. <http://ontogenesis.knowledgeblog.org/2010/01/22/ontologies-for-sharing/> (last visited 12 March, 2012).
- [167] Shotton, D., Portwin, K., Klyne, G., Miles, A. (2009). Adventures in Semantic Publishing: Exemplar Semantic Enhancements of a Research Article. *PLoS Computational Biology*, 5 (4): e1000361. DOI: 10.1371/journal.pcbi.1000361.
- [168] Simon, J., Birukou, A., Casati, F., Casati, R., Marchese, M. (2011). Liquid Publications Green Paper. http://peerevaluation.org/data/ca75910166da03ff9d4655a0338e6b09/PE_doc_28223.pdf (last visited March 12, 2012)
- [169] Simons, G. F., Lewis, W. D., Farrar, S. O., Langendoen, D. T., Fitzsimons, B., Gonzalez, H. (2004). The semantics of markup: mapping legacy markup schemas to a common semantics. In *Proceedings of the Workshop on NLP and XML (NLPXML-2004)*. Stroudsburg, Pennsylvania, USA: Association for Computational Linguistics. <http://acl.ldc.upenn.edu/acl2004/nlpxml/pdf/simons-et-al.pdf> (last visited March 12, 2012).
- [170] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. In *Journal of Web Semantics*, 5 (2): 51-53. DOI: 10.1016/j.websem.2007.03.004.
- [171] Solomon, J. S. (2008). *Developing Open Access Journals: A Practical Guide*. Chandos Publishing (Oxford) Limited, Oxford, UK. ISBN: 1843343394.

- [172] Souza, K., Dos Santos, A., Evangelista, S. (2003). Visualization of Ontologies through Hypertrees. In Proceedings of the Latin American Conference on Human-Computer Interaction. New York, New York: ACM.
- [173] Souzis, A. (2005). Building a Semantic Wiki. In IEEE Intelligent Systems, 20 (5): 87-91. DOI: 10.1109/MIS.2005.83.
- [174] Sowa, J. F. (1987). Semantic Networks. In Shapiro, S. C. (Eds.), Encyclopedia of Artificial Intelligence. New York, New York, USA: John Wiley & Sons. ISBN: 0471503053.
- [175] Sperberg-McQueen, C. M. (2006). Rabbit/duck grammars: a validation method for overlapping structures. In Proceedings of Extreme Markup Languages Conference 2006. Rockville, Maryland, USA: Mulberry Technologies, Inc. <http://conferences.idealliance.org/extreme/html/2006/SperbergMcQueen01/EML2006SperbergMcQueen01.html> (last visited March 12, 2012).
- [176] Sperberg-McQueen, C. M., Huitfeldt, C. (2004). GODDAG: A Data Structure for Overlapping Hierarchies. In P. R. King, E. V. Munson (Eds.), Proceeding of the 5th International Workshop on the Principles of Digital Document Processing (PODDP 2000). Berlin, Germany: Springer.
- [177] Sperberg-McQueen, C. M., Huitfeldt, C., Renear, A.. (2000). Meaning and interpretation of markup. In Markup Languages: Theory & Practice, 2 (3): 215-234. DOI: 10.1162/109966200750363599.
- [178] Sperberg-McQueen, C. M., Marcoux, Y., Huitfeldt, C. (2009). Two representations of the semantics of TEI Lite. In Proceedings of Digital Humanities 2010. 7-10 July 2010, London, UK. <http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/html/ab-663.html> (last visited March 12, 2012).
- [179] Storey, M. A., Musen, M.A., Silva, J., Best, C., Ernst, N., Fergerson, R. Noy, N.F. (2001). Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege. Presented during the K-CAP 2001 Workshop on Interactive Tools for Knowledge Capture. Victoria, Canada. <http://www.isi.edu/~blythe/kcap-interaction/papers/storey.pdf> (last visited March 12, 2012).
- [180] Styles, R., Ayers, D., Shabir, N. (2008). Semantic Marc, MARC21 and The Semantic Web. In C. Bizer, T. Heath, K. Idehen & Berners-Lee, T. (eds.), Proceedings of the Workshop on Linked Data on the Web (LDOW2008). Aachen, Germany: SunSITE Central Europe. <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-369/paper02.pdf> (last visited March 12, 2012).

- [181] Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (2012). Ontology Engineering in a Networked World. Berlin, Germany: Springer. ISBN: 3642247934.
- [182] Swan, A. (2009). The Open Access citation advantages: Studies and results to date. Technical Report, School of Electronics & Computer Science, University of Southampton. <http://eprints.ecs.soton.ac.uk/18516/> (last visited March 12, 2012).
- [183] Tartir, S., Arpinar, I.B., Moore, M., Sheth, A.P., Aleman-Meza, B. (2006). OntoQA: Metric-based ontology quality analysis. In Proceedings of the IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources. November 27, 2005, Huston, Texas, USA. <http://lsdis.cs.uga.edu/library/download/OntoQA.pdf> (last visited March 12, 2012).
- [184] Tejo-Alonso, C., Berrueta, D., Polo, L., Fernandez, S. (2011). Metadata for Web Ontologies and Rules: Current Practices and Perspectives. In García-Barriocanal, E., Cebeci, Z., Okur, M. C., Öztürk, A. (Eds.), Proceeding of the 5th International Conference on Metadata and Semantic Research (MTSR 2011). Berlin, Germany: Springer.
- [185] Tennison, J., Piez, W. (2002). The Layered Markup and Annotation Language (LMNL). Presented at the Extreme Markup Languages Conference 2002. 4-9 August 2002, Montreal, Canada.
- [186] Text Encoding Initiative Consortium (2005). TEI P5: Guidelines for Electronic Text Encoding and Interchange. Charlottesville, Virginia, USA: TEI Consortium. <http://www.tei-c.org/Guidelines/P5> (last visited March 12, 2012).
- [187] Toulmin, S. (1959). The uses of argument. Cambridge, UK: Cambridge University Press. ISBN: 0521827485.
- [188] Tummarello, G., Morbidoni, C., Pierazzo, E. (2005). Toward Textual Encoding Based on RDF. In Milena Dobreva, Jan Engelen (Eds.), Proceedings of the 9th ICCC International Conference on Electronic Publishing (ELPUB2005). Leuven, Belgium: Peeters Publishing Leuven.
- [189] Van Deursen, D., Poppe, C., Martens, G., Mannens, E., Van de Walle, R. (2008). XML to RDF Conversion: a Generic Approach. In Nesi, P., Delgado, J., Ng, K. (Eds.), Proceedings of the 4th International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution (AXMEDIS 08). Florence, Italy: Firenze University Press.

- [190] Van Rijsbergen, C. J. (1986). A new theoretical framework for information retrieval. In Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR86). New York, New York, USA: ACM.
- [191] Varma, P. (2010). Project Documents Ontology. <http://vocab.deri.ie/pdo> (last visited March 12, 2012).
- [192] Vrandecic, D., Krötzsch, M. (2006). Reusing Ontological Background Knowledge in Semantic Wikis. In Völkel, M., Schaffert, S. (Eds.), Proceedings of the 1st Semantic Wiki Workshop –From Wiki To Semantics (SemWiki 2008). Aachen, Germany: SunSITE Central Europe. <http://www.ceur-ws.org/Vol-206/paper2.pdf> (last visited March 12, 2012).
- [193] Všlkel, M., Krštžsch, M., Vrandecic, D., Haller, H., Studer, R. (2006). Semantic wikipedia. In Carr, L., De Roure, D., Iyengar, A., Goble, C. A., Dahlin, M. (Eds.), Proceedings of the 15th international conference on World Wide Web (WWW 2006). New York, New York, USA: ACM.
- [194] Walsh, N. (2010). DocBook 5: The Definitive Guide. Sebastopol, CA, USA: O'Reilly Media. Version 1.0.3. ISBN: 0596805029.
- [195] Wan, S., Paris, C., Dale, R. (2010). Supporting browsing-specific information needs: Introducing the Citation-Sensitive In-Browser Summariser. In Journal of Web Semantics, 8 (2-3): 196-202. DOI: 10.1016/j.websem.2010.03.002.
- [196] Wang, T. D., Parsia, B. (2006). Cropecircles: Topology Sensitive Visualization of Owl Class Hierarchies. In Cruz, I. F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M, Aroyo, L. (Eds.), Proceedings of the 5th International Semantic Web Conference (ISWC 2006). Berlin, Germany: Springer.
- [197] Woods, W. A. (1975). What's in a Link: Foundations for Semantic Networks. In Bobrow, D., Collins, A. (Eds.), Representation and Understanding: Studies in Cognitive Science. New York, New York, USA: Academic Press. ISBN: 0121085503.
- [198] Yang, K., Steele, R., Lo, A. (2007). An Ontology for Xml Schema to Ontology Mapping Representation. In Kotsis, G., Taniar, D., Pardede, E., Ibrahim, I. K. (Eds.), Proceedings of the 9th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2007). Vienna, Austria: Austrian Computer Society.